G57

SISTEMA DE EVALUACIÓN DE RIESGO CREDITICIO

FASE 2 - 02/11/2025

Agenda

- Estructuración del Proyecto
- Estructuración y Refactorización del Código
- Mejores Prácticas de Codificación en el Pipeline de Modelado
- Seguimiento de Experimentos, Visualización de Resultados y Gestión de Modelos
- Conclusiones

Agencia G57

ROLES

DATA ENGINEER

Maximiliano Zapater Cornejo
A01840258@tec.mx

CIENTÍFICO DE DATOS

José Luis Parada Guitiérrez
A00939669@tec.mx

SOFTWARE ENGINEER

Julián Jesús Moreno Ovando.
A01795915@tec.mx

ML ENGINEER

Oscar Luis Guadarrama Jiménez.
A01796245@tec.mx

DEVOPS

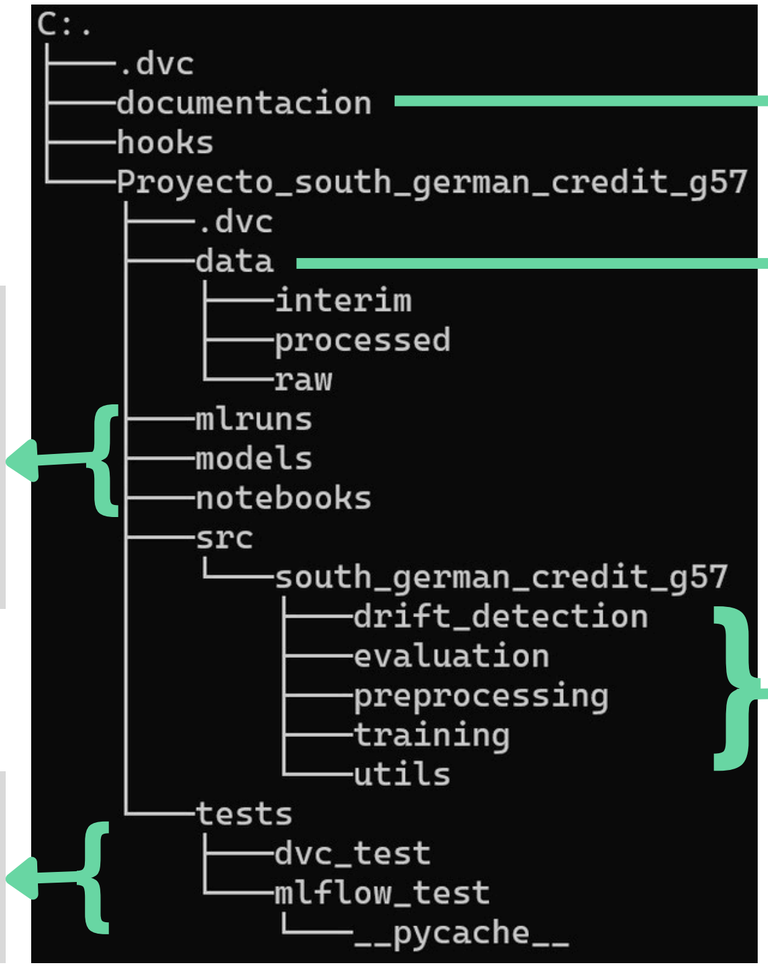
Ronald Sandí Quesada
A01794620@tec.mx

Estructura proyecto

Cookiecutter-data-science

Roles:

- Data Engineer
- Data Scientist
- MLOps Engineer
- DevOps
- Software Engineer



Documentación técnica, manuales, resultados y diagramas del proyecto.

Gestión y almacenamiento de datasets.

- raw/: datos originales
- interim/: transformaciones intermedias
- processed/: datos finales para modelado

Módulo principal del proyecto; **integra** todos los subprocesos. Limpieza, encoding, imputación y feature engineering.

- Historial de **experimentos** (métricas, artefactos, parámetros).
- **Modelos** entrenados, checkpoints, entre otros.
- **Notebooks** exploratorios para **EDA**, prototipado de features y evaluación visual.

- **Tests** de dependencias, pipelines y versiones de datos.
- **Validación** del tracking y registro de experimentos.

Una buena estructura de proyecto facilita la **colaboración, el mantenimiento y la escalabilidad** de los desarrollos en ML.

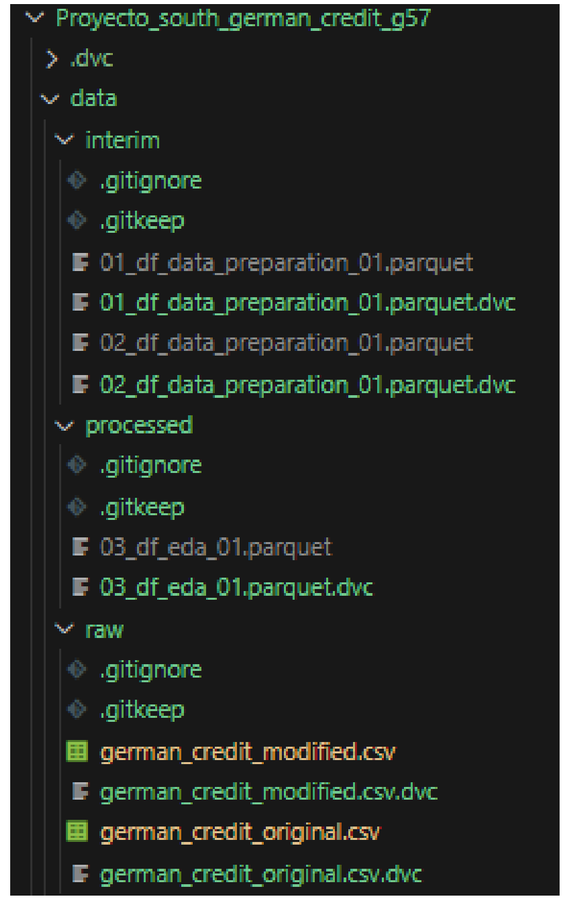
Estructura datos

En el proyecto, el contenido del **directorio data/ se encuentra versionado con DVC**, lo que permite mantener **trazabilidad** sobre cada versión del dataset y reproducir experimentos de manera controlada.

Google Drive (equipo) ⇄ DVC Remote

↑ dvc pull **↓ dvc push**

- Los archivos de datos (raw, processed, interim) **no se suben directamente al repositorio Git**.
- En su lugar, DVC crea **archivos de control (.dvc)** que guardan los hashes de versión.
- Los datos reales se almacenan en un remoto compartido en **Google Drive del equipo.**



Se implementó una **clase que ejecuta sincronizaciones automáticas de datos** para garantizar la coherencia del entorno de ejecución.

```
# Al inicio del script(main.py)
DVCManager.pull()  # descarga los datasets actualizados
# ... flujo de trabjo del main.py ...
# Al finalizar el script (main.py)
DVCManager.push()  # sube las nuevas versiones de datos/modelos
```

Con DVC logramos un flujo de datos coherente, reproducible y colaborativo, integrado directamente al pipeline del proyecto

Arquitectura

PROGRAMA CONSOLA DE PRUEBA
App en Python que permite probar la conexión con el servidor MLFlow en forma segura y sencilla <mlflow_test.py>

CUADERNO JUPYTER
Área de trabajo de los experimentos ML
<CreditRisk_Prototype_SouthGermanDataset.ipynb>

A — Clientes

API DE GCP <GOOGLE CLOUD PLATFORM>
Punto de entrada entre las Apps clientes y la solución arquitectónica

DNS EN GCP
Medio para resolver los nombres de los elementos en Internet

IP EXTERNA
Componente de red para publicar y relacionar en el DNS versus los nombres de domino. Localizada en la Zona central US <34.143.77.2>

CAPA DE AUTENTICACIÓN IAM
Enlace de seguridad entre los usuarios autorizados a acceder la solución arquitectónica y las VPN. Los usuarios (principales) están en el dominio <tec.mx>

VPN PÚBLICA
Enlace entre la red de área amplia <WAN> y la red privada de los elementos de la solución <34.143.77.0/16>

VPN PRIVADA
Enlace seguro entre el área pública y los elementos embebidos en la zona altamente protegida <10.84.112.0/16>

B — Internet y DNS

CONTENEDOR DE APLICACIÓN
Contenedor Cloud-Run <mlflow-super-g5> generado a raíz de una imagen en el registro de contenedores <mlflow_super_postgresql>

END-POINT
Si bien el contenedor es el mismo, el API del Servidor MLFlow debe ser expuesto para uso de los clientes en el puerto <5000> vía end-point en GCP

SERVIDOR MLFLOW
La instalación de servidor MLFlow que intrínsecamente contiene el sistema de gestión de versiones de los modelos y el backend para explorar los modelos almacenados en forma persistente <mlflow-super-g57-137680020436>

C — GCP Cómputo

BUCKET PARA ARTEFACTOS
Es un Bucket de almacenamiento de nivel y persistencia estándares. En él se aloja la meta-data de los artefactos relativos a las diversas corridas <laboratorio1-447417-mlflow-artifacts>

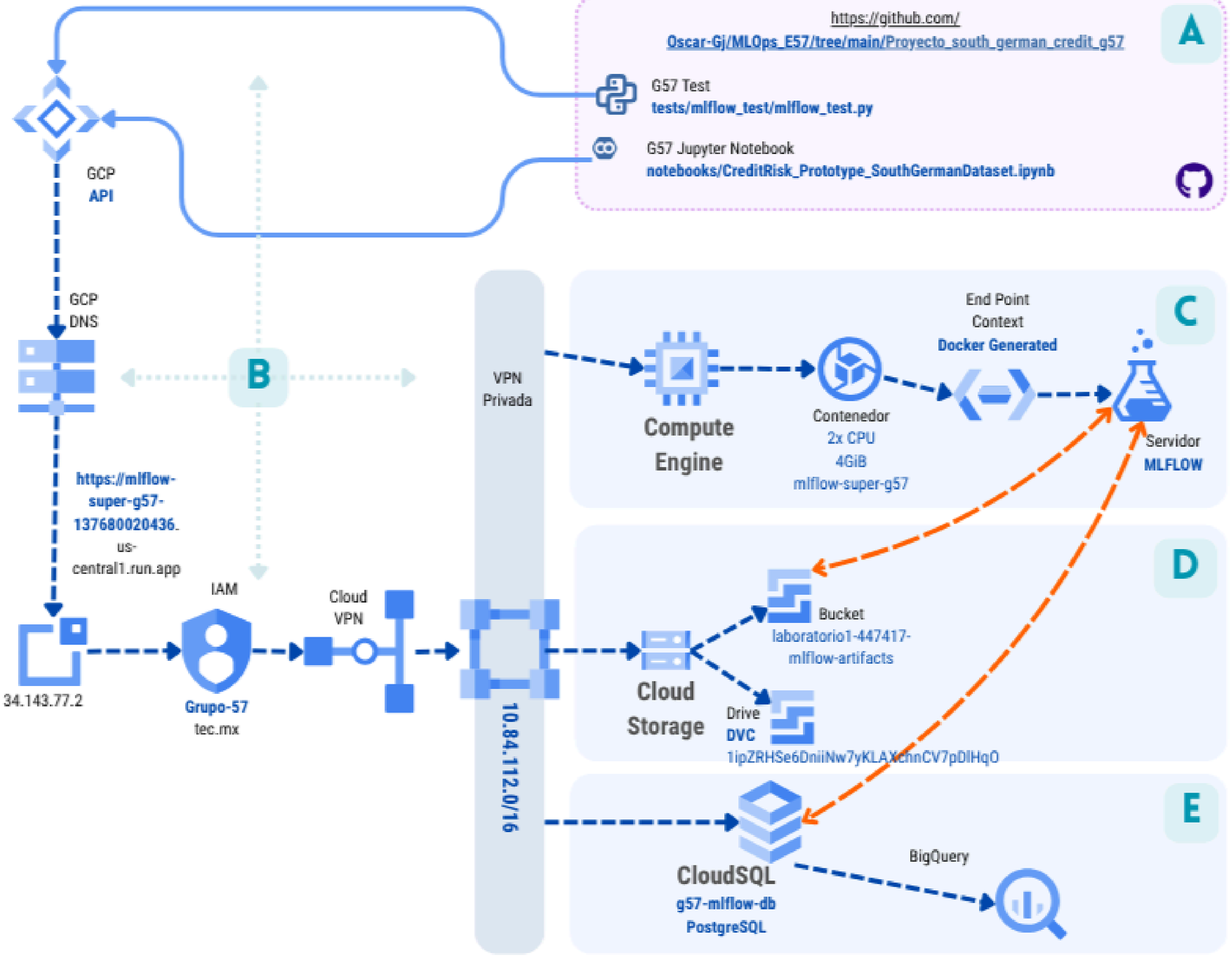
DISCO PARA DVC
Es un disco de GDrive, pero gestionado con seguridad IAM para poder controlar la integridad de los archivos emitidos desde DVC
<1ipZRHSe6DniiNw7yKLAXchnCV7pDlHqO>

D — GCP Almacenamiento

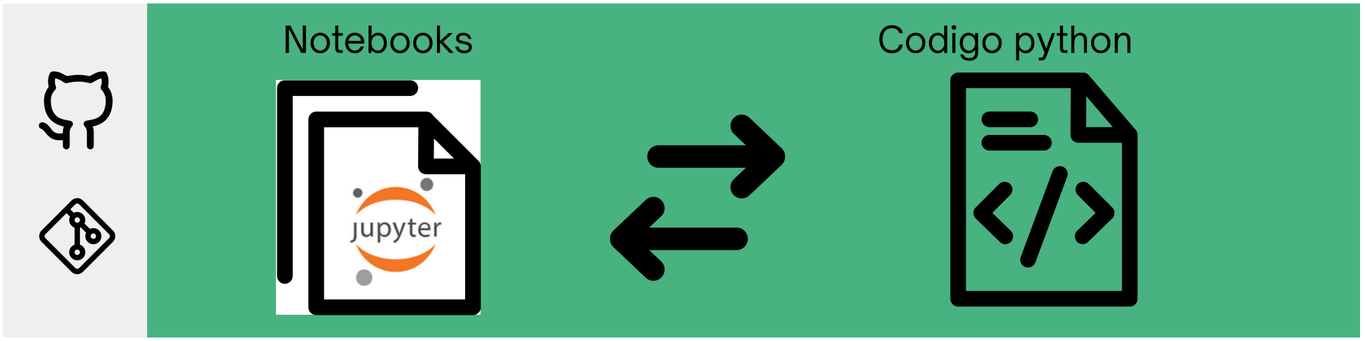
SERVIDOR DE BASE DE DATOS <POSTGRESQL>
Medio estático para persistir los datos relativos a las corridas en el MLFlow. Al ser un DBMS los registros están disponibles también vía consultas en Big-Query <g57-mlflow-db>

BIGQUERY
Si bien no es parte per se de la solución MLOPS, es un plus ofrecido por GCP para realizar consultas de alto nivel en la información de los modelos entrenados

E — GCP SQL



Estructuración y refactorización de código



Data Scientist

El Científico de datos realiza revisiones de la data y posibles modelos sobre un Notebook

Software Eng

El ingeniero de software refactoriza código y aplica POO para dejar el trabajo del DS en un script de python.

Trabajo **colaborativo** y **coordinado** para cada iteración de cambios.

Importante:

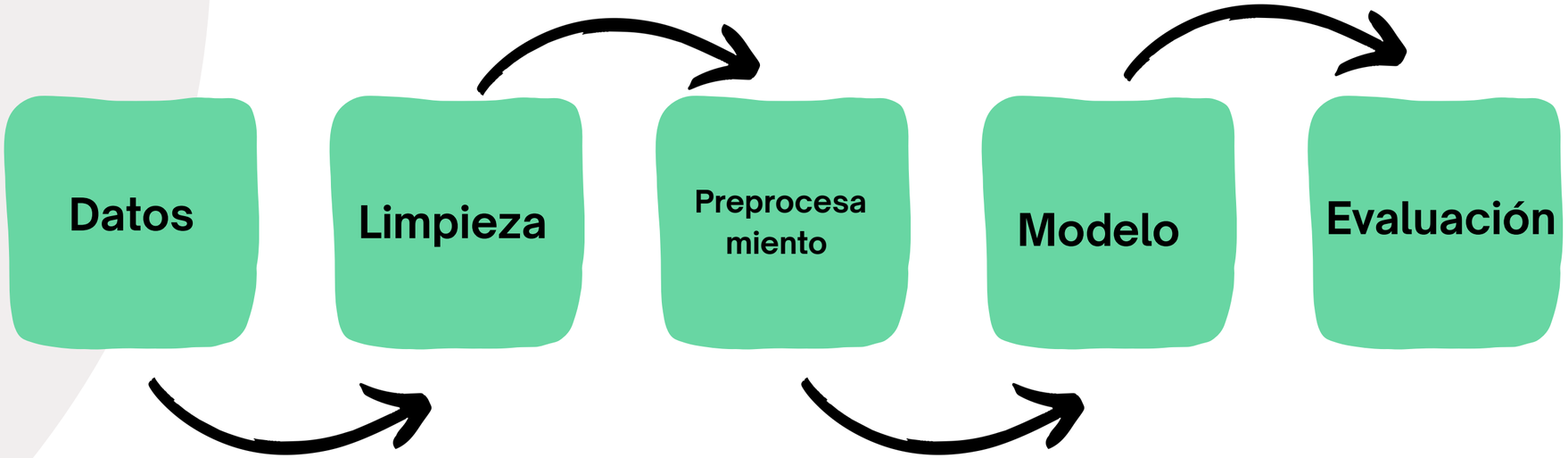
- Organizar el código en **módulos** y funciones con **responsabilidades** bien definidas.
- Aplicar principios de programación orientada a objetos (**POO**).
- **Refactorizar** el código existente para mejorar su **eficiencia, legibilidad, escalabiliad y mantenimiento a largo plazo.**

Ventajas:

- Facilita la **mantenibilidad** y **escalabilidad** de los proyectos de ML.
- Mejora la **colaboración** y **reproducibilidad**, haciendo más eficiente el trabajo en equipo y el control de versiones.

Mejores Prácticas de Codificación en el Pipeline de Modelado

- Implementación de un pipeline modular con Scikit-Learn.
- Automatización del flujo completo:
- preprocesamiento → entrenamiento → evaluación.
- Integración de transformaciones
  - Escalado de variables numéricas.
  - Codificación de variables categóricas.
  - Entrenamiento con Random Forest Classifier.
- Código limpio, reproducible y alineado con buenas prácticas de MLOps.



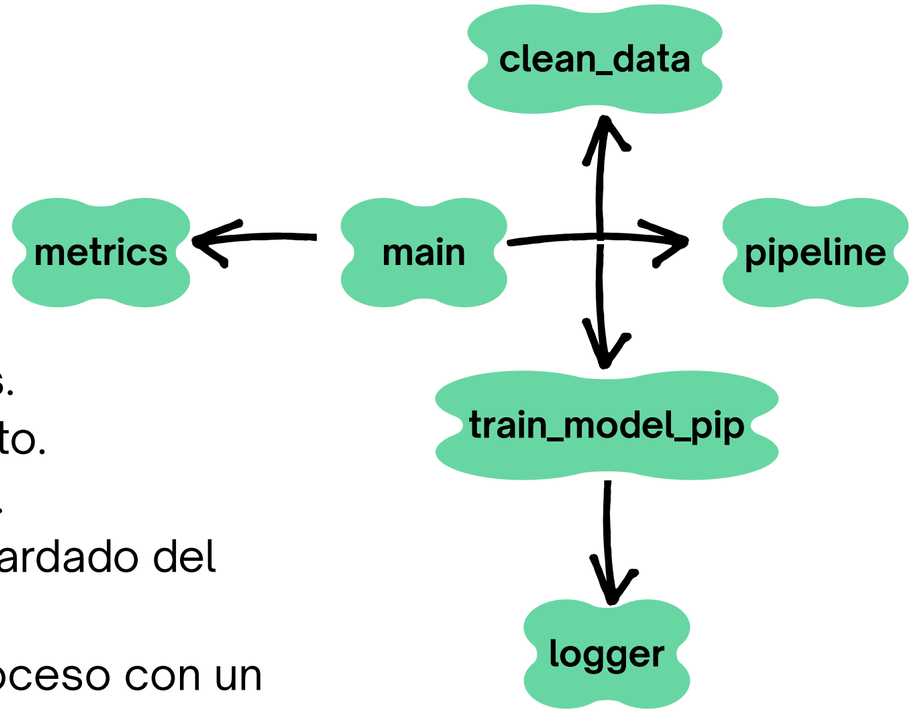
Datos → Limpieza → Preprocesamiento → Modelo → Evaluación

Orquestación del Proceso

main.py actúa como el cerebro del proyecto.

Orquesta la ejecución de los módulos principales:

- clean_data.py → Limpieza y validación de datos.
- pipeline.py → Construcción del pipeline (preprocesamiento + modelo).
- train_model.py → Entrenamiento, validación y evaluación del modelo.
- metrics.py → Generación y análisis de métricas.
- logger.py → Registro de logs, trazas y seguimiento del flujo.



Controla las fases clave:

1. Carga de datos procesados.
2. Limpieza y preprocesamiento.
3. Entrenamiento y evaluación.
4. Registro de resultados y guardado del modelo.
5. Permite ejecutar todo el proceso con un solo comando, de forma automatizada y reproducible.

Registro y Seguimiento de Experimentos con MLflow

Se configuró una conexión remota con el servidor de MLflow:

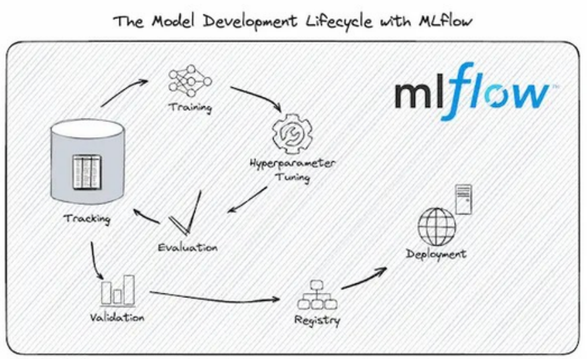
mlflow.set_tracking_uri("https://mlflow-super-g57-137680020436.us-central1.run.app")
mlflow.set_experiment("Experimento-Conexión-MLFlow-Grupo57")

Se registran automáticamente:

- Parámetros del modelo (model_params)
- Métricas de rendimiento (accuracy, precision, recall, f1)
- El modelo entrenado (pipeline_credit.pkl)

El pipeline se entrena y guarda dentro de un run remoto, quedando disponible en el panel de control de MLflow.

Esto permite comparar versiones de modelos, visualizar métricas y reproducir resultados fácilmente.



Estructura de operacion

1 - INGESTA DE DATOS

2 - PREPARACIÓN DE DATOS

3 - ANÁLISIS EXPLORATORIO DE DATOS

4 - MODELADO

5 - EVALUACIÓN Y REPORTES

1 - Ingesta de Datos

La i**ngesta de datos** consiste en tomar la información desde su fuente y cargarla al sistema para su análisis.

En nuestro caso, los datos **provienen de un archivo CSV**, el cual se importó y se transformó en un **DataFrame de Pandas**. Una vez obtenido el DataFrame en crudo, se **guardó en formato Parquet**, que es más rápido y eficiente para trabajar en las siguientes etapas del proyecto.

2 - Preparación de Datos

La **preparación de datos** consiste en **limpiar, transformar y organizar la información** para que pueda ser utilizada correctamente por los modelos.
En esta etapa se **trataron valores nulos**, se **codificaron variables categóricas** y se **escalaron las variables numéricas** para mantenerlas en un mismo rango.

En nuestro caso, se **importó el DataFrame crudo** en formato Parquet y se **realizó la limpieza** correspondiente hasta obtener un conjunto de datos listo para ser explorado.
Finalmente, el **DataFrame limpio se guardó en formato Parquet** para ser utilizado en las siguientes etapas del análisis.

3 - Análisis Exploratorio de Datos (EDA)

El **EDA** tiene como objetivo **comprender y visualizar la información** antes de aplicar cualquier modelo.
En esta etapa **se analizó la distribución de las variables**, se **detectaron valores atípicos**, se **identificaron correlaciones** y se **observaron patrones importantes** dentro del conjunto de datos.

Se **importa el dataframe limpio** y se generaron **gráficas y estadísticas descriptivas** que permitieron obtener una visión general del comportamiento de los datos, realizando **ajustes adicionales al DataFrame** cuando fue necesario.
El **DataFrame final se guardó en formato Parquet** para ser utilizado en la etapa de modelado.

Las **modificaciones realizadas durante el EDA** se **incorporaron a la estructura del pipeline de preprocesamiento**, asegurando coherencia y consistencia en el flujo antes de pasar al modelado.

4 - Modelado

Tipo de variable	Pipelines
Numerica	imputar (imputer) escalar (scalar) distribucion gaussiana (power_transform)
Nominales	Imputar (imputer) codificador binario (binary_encoder)
Ordinales	Imputar (imputer) codificador ordinario (ordinal_encoder)

La **etapa de modelado** consiste en **entrenar y evaluar diferentes algoritmos de machine learning** para encontrar el que mejor se ajuste a los datos.

En esta fase se **probaron varios modelos** (LR, RF, MLP, SVC, DT, XGV y KNN); se **ajustaron sus hiperparámetros y se compararon sus métricas de rendimiento**. Todo el proceso se **registró y gestionó con MLflow**, lo que permitió **guardar los experimentos, parámetros, métricas y versiones de los modelos** de forma organizada y reproducible. Finalmente, el **mejor modelo fue seleccionado y guardado** para su posterior análisis y uso en las siguientes etapas del proyecto.

5 - Evaluación y Reportes

La **etapa de evaluación y reportes** tiene como objetivo **medir el desempeño del modelo y comunicar los resultados obtenidos.** En esta fase se **evaluó el modelo seleccionado con datos de prueba**, analizando métricas como **precisión, recall, F1-score y AUC**, entre otras. Los resultados se **compararon con los obtenidos durante el entrenamiento** para verificar que el modelo no presentara sobreajuste.

De todos los **modelos registrados en MLflow**, se **ejecutó un programa que identifica el mejor modelo de cada tipo, se graficaron las métricas de rendimiento y finalmente se eligió el mejor modelo entre todos** para utilizarlo en las pruebas finales. El **modelo seleccionado se guardó en formato .joblib** para poder ser **evaluado posteriormente con nuevos datos**, asegurando así su **reutilización y trazabilidad** dentro del proyecto.

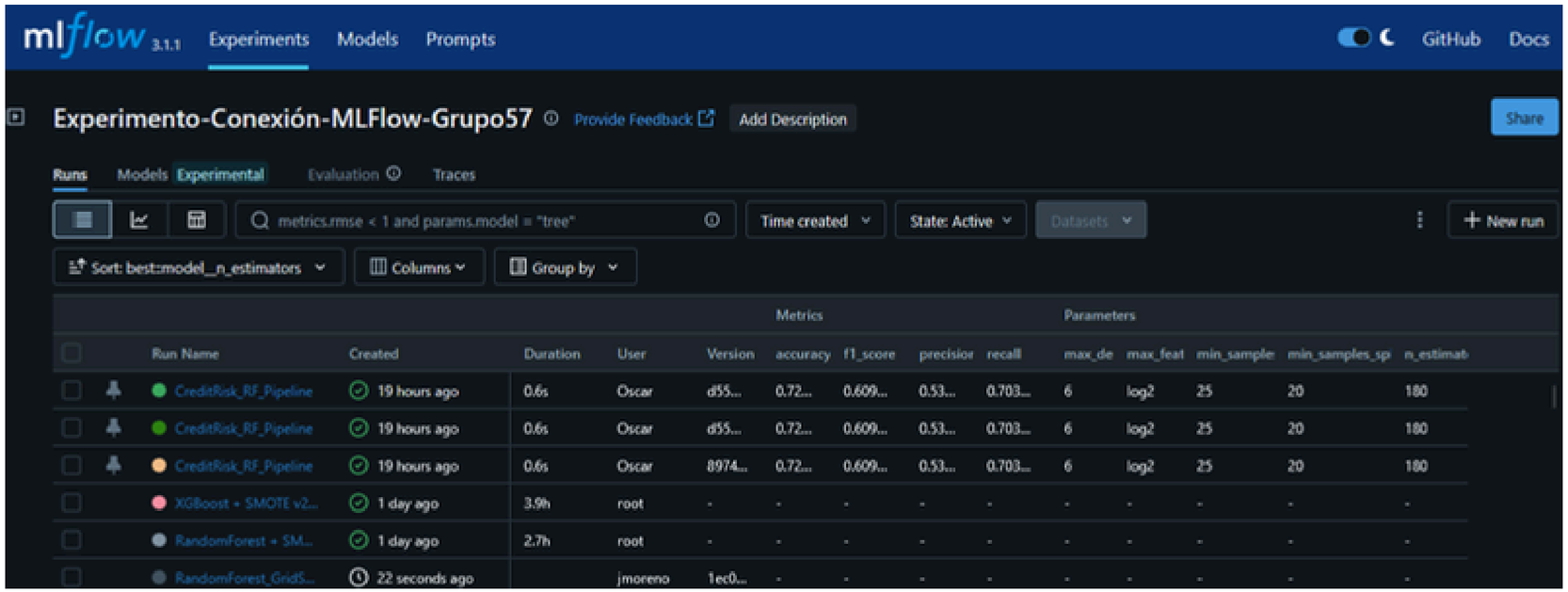
Seguimiento de Experimentos, Visualización de Resultados y Gestión de Modelos

Ventajas: un **registro** sistemático de los experimentos y modelos, acompañado de una **visualización** clara, permite el análisis comparativo, la **toma de decisiones informadas y una gestión profesional de los proyectos.**

Aplicación:
Se utilizan herramientas como MLflow, DVC para dar seguimiento a los experimentos.
Se documentan y comparan configuraciones, parámetros y resultados de cada ejecución.
Se mantiene un registro actualizado de los modelos generados, incluyendo:

- Versión
- Hiperparámetros
- Métricas de evaluación
- Resultados relevantes

Ejemplo visualización y control de runs de los modelos:



Agencia G57

CONCLUSIONES DEVOPS

Si bien el Equipo57 logra una arquitectura robusta y escalable mediante la implementación en **GCP**, es posible reconocer estas áreas de mejora:

1. Continuar ante los retos de CI/CD, a través de procesos automáticos que aseguren un empaquetado del modelo y registro en el gestor de imágenes de GPC (*Container Registry*).
2. Implementar el modelo en un servicio REST (u homólogo), para ser invocado desde cualquier código cliente mediante las API de GCP.
3. El Bucket de registro usa un drive en Google Storage, no obstante es posible probar un medio de persistencia alternativo en la nube AWS (tipo S3), cómo una medida de contingencia (este segundo almacenamiento ha sido proporcionado por la cátedra recientemente).

Estrategias usadas en análisis/implementación de la solución:

Los investigadores basan las estrategias de la solución DevOps en el material aprendido en:

1. El magnífico curso sobre GCP en Cloud Skills Boost (2025), el cuál es una base metodológica sobre buenas prácticas de uso de la nube de Google.
2. En cuanto a la praxis de las secciones de almacenamiento, cómputo y gestión de contenedores en GCP, la referencia mayor ha sido la consola oficial de la plataforma en Google (2025 a) y Google (2025 b).
3. De Dev Genius (2022) se toma un buen ejemplo para arrancar con la instalación/ configuración del contenedor de despliegue de MLFlow.
4. Finalmente, la creación, etiquetación y registro de contenedores, ha sido posible solo con el uso de la aplicación por antonomasia en este tipo de tecnologías, a saber, Docker (2025).

Agencia G57

CONCLUSIONES DATA ENGINEER

GENERAL	• Desde el rol de **Data Engineer**, el trabajo permitió **estructurar y versionar** los datos del proyecto de forma **trazable y reproducible**, facilitando la colaboración con los demás roles técnicos del equipo.
DONE	**Estrategias aplicadas** • Implementación de **DVC para control de versiones y trazabilidad** de datasets. • Organización modular del flujo de datos bajo la **estructura Cookiecutter-Data-Science.** • **Automatización de la sincronización** con el remoto mediante la clase DVCManager. • Aseguramiento de **consistencia y reproducibilidad** en las distintas etapas del pipeline.

NEXT STEPS	**<u>Oportunidades de mejora</u>**
	
	• Actualmente trabajamos con **bases estáticas**, pero en un entorno productivo los datos deberían actualizarse de forma continua o diaria.
	• Como mejora futura, se propone:
	  ◦ Incorporar **mecanismos de monitoreo y detección de cambios** en los datos (data drift, anomalías, formatos).
	  ◦ Desarrollar **alertas automáticas** frente a variaciones o fallos en el flujo de ingesta.
	  ◦ Avanzar hacia un **esquema de data streaming y control en tiempo real,** integrando nuevas herramientas

Agencia G57

CONCLUSIONES DATA SCIENTIST

En el proyecto de Machine Learning Operations del equipo G57, trabajamos en todo el ciclo de vida de un modelo de aprendizaje automático. Como Data Scientist, me enfoqué en el análisis de datos, el desarrollo y la evaluación de modelos, colaborando con el equipo para conectar los resultados del modelado con las demás etapas del proyecto.

En esta fase me concentré en la etapa de modelado, probando distintos algoritmos —como Logistic Regression, Decision Tree, Random Forest, XGBoost, KNN, MLP y SVC— y aplicando GridSearchCV para optimizar sus hiperparámetros y obtener el mejor desempeño posible. También incorporé técnicas de manejo del desbalance de clases, como SMOTE, con el objetivo de mejorar la capacidad predictiva y la equidad del modelo, integrando todo el proceso de modelado con MLflow.

La integración con MLflow me permitió registrar y dar seguimiento a todos los experimentos realizados, incluyendo parámetros, métricas y artefactos de cada ejecución. Este registro facilitó la comparación entre modelos y permitió al equipo seleccionar de forma objetiva el modelo con mejor desempeño global, considerando métricas como Accuracy, Precision, Recall, F1-score, AUC...

Finalmente, el mejor modelo fue registrado y exportado en formato .joblib para su futura implementación en producción. Esta experiencia me permitió fortalecer mis habilidades como Data Scientist, trabajando en un entorno colaborativo y con un enfoque orientado a la trazabilidad, automatización y mejora continua de modelos de machine learning.

Como oportunidades de mejora, considero que podríamos optimizar el proceso de experimentación automatizando aún más la selección y evaluación de modelos dentro de MLflow, e integrar herramientas de monitoring y drift detection para un seguimiento continuo del rendimiento en producción. Además, sería valioso ampliar el dataset y probar nuevas estrategias de feature engineering que podrían mejorar la capacidad de generalización del modelo.

Agencia G57

CONCLUSIONES SOFTWARE ENGINEER

Mi función fue servir como el puente de ingeniería entre la exploración de datos y la producción. Mientras el Data Scientist se enfocó en qué modelo construir (XGBoost, SMOTE) y el Data Engineer en qué datos versionar (DVC), mi rol fue definir cómo construir el código de ese modelo para que fuera robusto, mantenible y escalable.

Desafío

El punto de partida fueron los notebooks de experimentación (01 al 05.ipynb). Estos son indispensables para la investigación, pero representan un activo de alto riesgo: son frágiles, difíciles de versionar e imposibles de automatizar de forma fiable.

Solución

Mi labor consistió en refactorizar esta lógica de prototipo y convertirla en activos de ingeniería de software:

- Estructura Profesional: Implementé la arquitectura de código Cookiecutter, creando el esqueleto estándar sobre el cual el Data Engineer versionó sus flujos y el ML Engineer construyó sus pipelines.
- Código Modular: Traduje la lógica de los notebooks a scripts modulares y parametrizables:
  - **clean_data**.py: Asegura una limpieza de datos determinista.
  - **train_model**.py: Contiene el pipeline de entrenamiento.
  - **metrics**.py: Estandariza la evaluación del modelo.
- Estandarización (Pipelines SK-Learn): Encapsulé el preprocesamiento y el modelado en Pipelines de Scikit-Learn. Esto garantiza que el modelo se ejecute exactamente igual en entrenamiento y en producción, eliminando inconsistencias.

Impacto

Este trabajo de ingeniería de software fue la base que habilitó al resto del equipo:

- **ML Engineer**: Le proporcionó los componentes de código estables para ser orquestados en un pipeline automatizado (main.py).
- **Data Scientist**: Le permitió ejecutar sus experimentos de forma sistemática (MLflow) desde un script, facilitando la comparación objetiva de modelos.
- **DevOps**: Generó activos de código versionables, listos para ser empaquetados en contenedores y desplegados en GCP como un servicio.

Agencia G57

CONCLUSIONES ML ENGINEER

Como Machine Learning Engineer, mi función en esta fase fue garantizar que el flujo completo —desde los datos procesados hasta el modelo desplegable— fuera totalmente automatizado, reproducible y trazable.

Implementé un pipeline modular que une el preprocesamiento, el entrenamiento y la evaluación en un solo flujo controlado por main.py, asegurando consistencia entre entornos y ejecuciones.

Además, la integración con MLflow remoto permitió llevar el registro profesional de todos los experimentos, métricas, parámetros y versiones de modelos, siguiendo las mejores prácticas de MLOps.

En resumen, esta fase consolida la base técnica del proyecto, dejando preparado el entorno para la siguiente etapa: el despliegue y monitoreo del modelo en producción.

Enlace del vídeo del proyecto:

LINK VIDEO

Enlace del Github del proyecto:

https://github.com/Oscar-Gj/MLOps_E57

Agencia G57

REFERENCIAS BIBLIOGRÁFICAS

- Cloud Skills Boost (2025). *Google Cloud Career Launchpad - LatAm*. [Plataforma de lanzamiento profesional de Google Cloud - Latinoamérica]. Cloud Skills Boost. Recuperado en febrero 11, 2025 de https://www.cloudskillsboost.google/my_account/programs
- Dev Genius (2022). *Install MLFlow on GCP for Your Team: The Simplest Way*. [Instala MLFlow en GCP para tu equipo: La forma más sencilla]. Recuperado en octubre 7, 2025 de https://blog.devgenius.io/install-mlflow-on-gcp-for-your-team-the-simplest-way-d208b4ee3cd3
- Docker (2025). *Develop faster. Run anywhere*. [Desarrolle más rápido y ejecute desde cualquier lugar] Docker. Recuperado en octubre 10, 2025 de https://www.docker.com
- Google (2025 a) *Google Cloud*. [Nube de Google]. Recuperado en octubre 15 de 2025 de https://cloud.google.com
- Google (2025 b) *Google Cloud BigQuery*. Recuperado en marzo 11 de 2025 de https://console.cloud.google.com/bigquery?d=ml_datasets&p=bigquery-public-data&page=table&t=ulb_fraud_detection&project=stone-arcade-452811-f2&ws=!1m5!1m4!4m3!1sbigquery-public-data!2sml_datasets!3sulb_fraud_detection

G57

Gracias