
LABORATORIO 1

GESTIÓN DE EXPERIMENTACIÓN Y MODELOS CON DVC AUTOMATED MACHINE LEARNING (AUTOML)

Objetivo:

Diseñar un pipeline automatizado inspirado en los principios de **AutoML**, utilizando **DVC** (**Data Version Control**) y **Git** para gestionar datasets, modelos y experimentos. El propósito es construir un flujo reproducible y flexible que permita:

- Entrenar automáticamente distintos modelos y configuraciones.
- Versionar los resultados y las métricas de cada experimento.
- Comparar el rendimiento entre distintas versiones de datos y modelos.

La complejidad principal del laboratorio radica en la **gestión de la experimentación con DVC** y el **control de versiones con Git**, no en la programación del modelo en sí.

Descripción del Problema:

El laboratorio consiste en construir un pipeline de **AutoML reproducible** y controlado con DVC, capaz de:

1. Preprocesar los datos automáticamente según su tipo (numéricos o categóricos).
2. Entrenar distintos modelos (por ejemplo, Regresión Lineal, Random Forest y Gradient Boosting).
3. Evaluar los resultados y seleccionar el mejor modelo por rendimiento.

El objetivo no es crear un framework de AutoML completo, sino **gestionar los experimentos de AutoML como un producto reproducible**. Se deben versionar datasets, configuraciones y resultados con DVC, y usar Git para documentar y etiquetar cada versión del pipeline.

A lo largo del laboratorio, el dataset evolucionará en múltiples versiones (v1, v2, v3...), simulando un entorno real donde los datos cambian. El pipeline deberá mantenerse reproducible ante cada cambio.

Beneficios del Enfoque AutoML con DVC:

- **Ahorro de tiempo:** automatiza la reejecución del pipeline ante cualquier cambio en datos o parámetros.
- **Reproducibilidad:** cada experimento puede reconstruirse exactamente mediante `dvc repro`.

- **Colaboración eficiente:** Git y DVC permiten comparar versiones, resultados y ramas experimentales.
- **Toma de decisiones informada:** el registro histórico de métricas y configuraciones facilita seleccionar la mejor estrategia.

Requerimientos:

1. Inicialización y Pipeline Base

- Crear un repositorio Git y un proyecto DVC.
- El archivo `dataset_v1.csv` se encuentra disponible en el GES y debe ubicarse dentro del directorio `data/`.
- Versionar el dataset inicial con `dvc add data/dataset_v1.csv`.
- Definir un pipeline de DVC con las siguientes etapas:
 1. **preprocess:** limpieza y codificación de datos.
 2. **train:** entrenamiento del modelo según parámetros definidos en `params.yaml`.
 3. **evaluate:** cálculo y almacenamiento de métricas en `metrics.json`.
- Garantizar que el pipeline sea completamente reproducible mediante `dvc repro`.

2. Evolución de Datos y Versionado

- Incorporar nuevas versiones del dataset (`dataset_v2.csv`, `dataset_v3.csv`, `dataset_v4.csv`).
- Las versiones posteriores deben crearse aplicando modificaciones progresivas a los datos, como por ejemplo:
 - **Limpieza:** eliminar filas duplicadas o con valores nulos.
 - **Ampliación:** agregar nuevas observaciones o registros al dataset.
 - **Transformación:** modificar el formato de las variables (por ejemplo, normalización, codificación de categorías, creación o eliminación de columnas).
- Cada nueva versión debe guardarse con un nombre distinto y versionarse con DVC mediante:

```
dvc add data/dataset_v2.csv
git add data/dataset_v2.csv.dvc
git commit -m "Dataset v2: datos limpios"
```

- Actualizar la referencia del dataset en `params.yaml` y ejecutar nuevamente el pipeline con `dvc repro`.
- Comparar resultados entre versiones usando `dvc metrics diff` o `dvc plots diff`.
- Documentar cada iteración con commits descriptivos, tags de Git y una breve explicación de los cambios en el `README.md`.

3. Implementación del Componente AutoML

- Configurar en `params.yaml` una lista de modelos y sus hiperparámetros.
- El pipeline debe iterar sobre las configuraciones, entrenar y evaluar cada modelo automáticamente.
- Los resultados se registran en `metrics.json`, indicando el mejor modelo.
- No se requiere usar librerías externas de AutoML; basta con automatizar la comparación de modelos.

4. Comparación y Análisis de Resultados

- Usar `dvc metrics diff` o `dvc plots diff` para comparar métricas entre versiones.
- Generar un reporte final (`report.md` o `results.csv`) con las métricas principales, el modelo ganador y los parámetros finales.
- Analizar cómo los cambios en los datos y configuraciones afectaron el rendimiento del modelo.

5. Entrega y Reproducibilidad

- La entrega debe realizarse a partir de un **release o tag en GitHub**.
- El proyecto debe ser totalmente reproducible mediante:

```
dvc repro
dvc metrics show
dvc metrics diff
```

- Incluir un archivo `README.md` con instrucciones detalladas de ejecución, dependencias y explicación de los cambios realizados en cada versión del dataset.

Entregables:

- Un archivo `.zip` descargado desde el **release/tag de GitHub**, que incluya:
 - Código completo del proyecto.
 - Archivos `.dvc`, `params.yaml` y `metrics.json`.
 - Archivo `README.md` con pasos de ejecución y análisis de resultados.
 - Reporte con comparación de métricas y conclusiones sobre la evolución de los datos.
- En el GES, además del `.zip`, incluir el **link al release o tag en GitHub**.

Resultado Esperado:

Un pipeline de AutoML reproducible en el que todas las versiones de datasets, configuraciones de modelos y resultados estén gestionadas con Git y DVC. Debe demostrarse la capacidad de reproducir cualquier experimento pasado y analizar cómo los cambios en los datos y modelos impactan en el rendimiento.