# Trabajo Práctico
# MLOps con Databricks y MLflow

**Asignatura:** CD.05 - Herramientas de Procesamiento para Grandes Volúmenes de Datos.

**Alumnos:**

- Daniel Emilio Fernandez
- Leandro Hermann Sauer
- Nicolás Moccagatta
- Pedro Ugarte Larraín

---

## 0. Repositorio Github

- https://github.com/LhermannSauer/itba-bigdata

## 1. Introducción

El objetivo de este trabajo práctico es diseñar, implementar y documentar un pipeline completo de **análisis de sentimientos** sobre reseñas textuales, integrando buenas prácticas de **MLOps** en un entorno de datos moderno.

En particular se busca:

- Construir una **arquitectura de datos en capas (Medallion)** para separar datos crudos, limpios y listos para modelado.
- Implementar un flujo de **experimentación reproducible** con MLflow, incluyendo registro de parámetros, métricas y artefactos.
- Comparar varias familias de modelos clásicos de Machine Learning para texto vectorizado y seleccionar el mejor según **F1 macro**.
- Registrar el modelo final en un **Model Registry**, dejando la base para futuros despliegues y monitoreo.

### Herramientas utilizadas

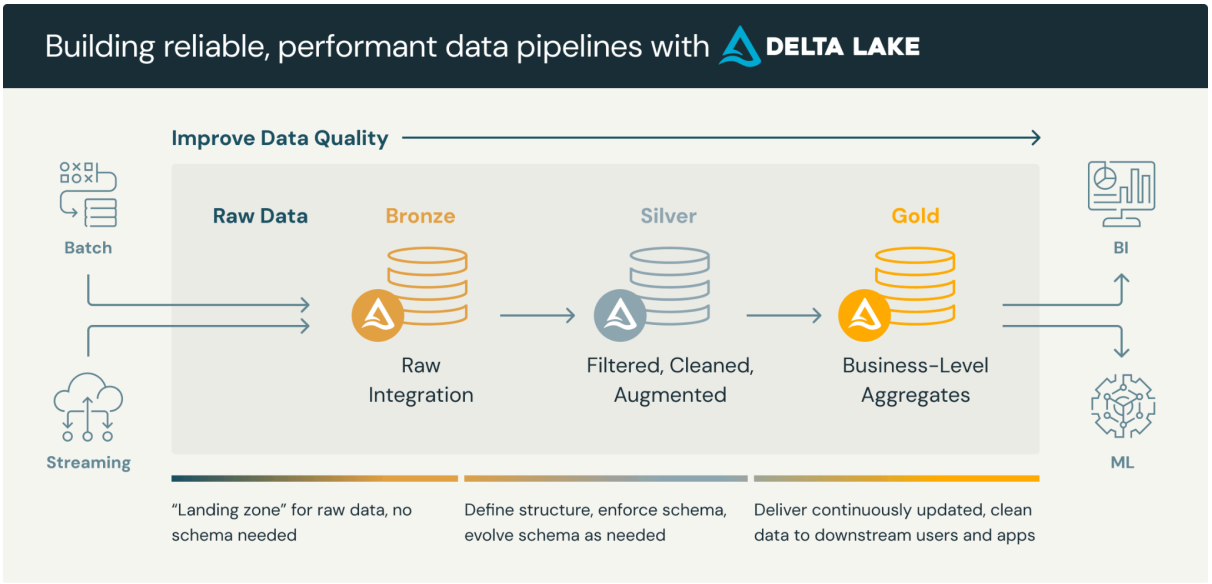
- **Databricks**: entorno unificado para notebooks, ejecución distribuida y orquestación de pipelines.
- **PySpark**: procesamiento distribuido de datos y construcción de las tablas Bronze, Silver y Gold.
- **MLflow**: tracking de experimentos, registro de modelos y almacenamiento de artefactos.
- **Unity Catalog**: catalogación, versionado y gobernanza de datos y modelos en el lakehouse.

El dataset utilizado contiene aproximadamente **15.000 reseñas** de texto con tres clases de sentimiento: **negativo, neutral y positivo**, preprocesadas y vectorizadas mediante TF-IDF.

---

# 2. Arquitectura

El proyecto implementa la **Medallion Architecture** con tres capas principales: **Bronze** → **Silver** → **Gold**.




https://www.databricks.com/glossary/medallion-architecture

## 2.1 Capa Bronze – Datos crudos

- **Origen**: archivos TSV crudos con reseñas y metadatos.
- **Ubicación**: `/Volumes/workspace/sentiment_analysis/raw` → `/bronze`.

- **Objetivo**: almacenar los datos **tal como llegan**, preservando el histórico y permitiendo reprocesar ante cambios futuros.

- No se realizan filtros agresivos, solo se añaden metadatos mínimos (timestamps, origen de archivo).

## 2.2 Capa Silver – Datos limpios y validados

- **Entrada**: tablas Bronze.

- **Procesos principales**:

    - Normalización de columnas, tipos de datos y formato de textos.
    - Manejo de nulos, textos vacíos y registros inconsistentes.
    - Validaciones básicas de calidad (rangos válidos, longitud mínima de reseña, etc.).

- **Salida**: tabla Silver con datos limpios y listos para análisis exploratorio.

- **Objetivo**: contar con un dataset **consistente y confiable**, reduciendo ruido y errores.

## 2.3 Capa Gold – Datos listos para ML (feature store ligero)

- **Entrada**: tabla Silver.

- **Procesos principales**:

    - Limpieza avanzada de texto (lowercase, remoción de HTML, regex).
    - Mapeo de estrellas a sentimiento:
        - 1–2 estrellas → **negativo**
        - 3 estrellas → **neutral**
        - 4–5 estrellas → **positivo**
    - **Feature engineering**:
        - TF-IDF con **~50.000 features** (unigrams + bigrams).
        - Métricas simples de texto (longitud, conteo de palabras, etc.).

    - Selección de columnas relevantes para entrenamiento.

- **Salida**: tabla Gold con features numéricas y etiqueta de sentimiento.

- **Objetivo**: tener un dataset **directamente consumible por los modelos** de Machine Learning.

---

# 3. Pipeline de Datos

## 3.1 Capa Bronze: ingesta de datos crudos

- Ingesta de archivos TSV a formato **Parquet** en Databricks.
- Se loguean en MLflow:
    - Cantidad total de filas y columnas.
    - Fecha/hora de ingesta.
    - Tasa de registros con campos críticos nulos.

## 3.2 Capa Silver: limpieza y validación

- Se eliminan registros con texto vacío o etiquetas inválidas.
- Se unifican formatos de fecha, tipos numéricos y codificación de texto.
- **Métricas de calidad registradas en MLflow**:
    - Porcentaje de filas retenidas vs. Bronze.
    - Distribución de clases (negativo / neutral / positivo).
    - Ratio de valores nulos por columna y “completeness score”.

En términos generales, la capa Silver retiene **la gran mayoría de los registros**, descartando principalmente filas con texto ausente o corrupto.

## 3.3 Capa Gold: feature engineering

- Se aplica el pipeline de **TF-IDF** para convertir el texto en un vector de alta dimensionalidad (~50k features).
- Se incluyen algunas features adicionales de texto simple (longitud, conteo de tokens), aunque el foco está en la representación TF-IDF.
- Se realiza **split train/test estratificado** (por ejemplo 80/20) manteniendo la proporción de clases.

También en esta etapa se loguean en MLflow:

- Cantidad de filas de entrenamiento y prueba.
- Distribución de clases en cada partición.
- Resumen de dimensionalidad de las features.

---

# 4. Experimentación con Modelos

Sobre la tabla Gold se probaron **múltiples variantes de modelos clásicos** para clasificación de texto:

- **Regresión Logística (One-vs-Rest)** con distintos valores de C.
- **SVM lineal (One-vs-Rest)** con distintos C.
- **Naive Bayes Multinomial** con distintos valores de `alpha`.

En total se entrenaron **12 variantes**, registradas en la tabla `model_comparison.csv`.

## 4.1 Hiperparámetros evaluados

Algunos de los hiperparámetros explorados fueron:

- **Regresión Logística**:
  - `C` ∈ {1, 5, 10}
  - `max_iter = 5000`
  - `solver = "liblinear"`
  - `class_weight = "balanced"`
- **SVM Lineal (One-vs-Rest)**:
  - `C` ∈ {0.1, 1, 10, 50}
  - Kernel lineal implícito.
  - Estrategia One-vs-Rest para las 3 clases.
- **Naive Bayes Multinomial**:
  - `alpha` ∈ {0.2, 0.5, 1.0, 2.0}

Todos los experimentos se registraron en MLflow con:

- Parámetros (`C`, `alpha`, `max_iter`, etc.).
- Métricas de evaluación.
- Artefactos del modelo (objetos serializados).

## 4.2 Métricas utilizadas

Las métricas principales utilizadas fueron:

- **F1 Score** (macro) → métrica principal para comparar modelos.
- **Precision** (macro).

- **Recall** (macro).

Estas métricas se calcularon sobre el **conjunto de test**, garantizando una comparación justa entre todas las variantes.

---

# 5. Resultados

## 5.1 Tabla comparativa de modelos

A partir del archivo `model_comparison.csv`, se obtuvo la siguiente tabla de comparación:

| Modelo | F1 Score | Precision | Recall |
|---|---|---|---|
| LR_C1 | 0.7179 | 0.7050 | 0.7326 |
| LR_C5 | 0.7148 | 0.7019 | 0.7298 |
| LR_C10 | 0.7132 | 0.7001 | 0.7284 |
| SVM_OVR_C10 | 0.6752 | 0.7191 | 0.6648 |
| SVM_OVR_C50 | 0.6750 | 0.7183 | 0.6647 |
| SVM_OVR_C1 | 0.6744 | 0.7225 | 0.6641 |
| SVM_OVR_C01 | 0.6631 | 0.7283 | 0.6538 |
| NB_alpha02 | 0.6566 | 0.6971 | 0.6349 |
| NB_alpha05 | 0.6554 | 0.6967 | 0.6339 |
| NB_alpha1 | 0.6529 | 0.6960 | 0.6315 |
| NB_alpha2 | 0.6473 | 0.6955 | 0.6261 |

## 5.2 Gráficos

**Figura 1 – Comparación de F1 por modelo** (`f1_comparison.png`).

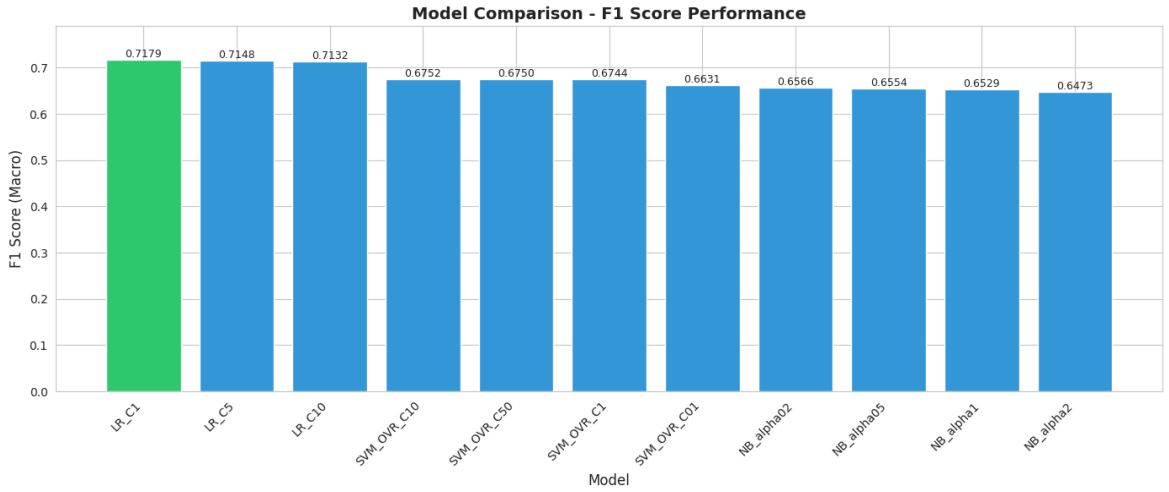
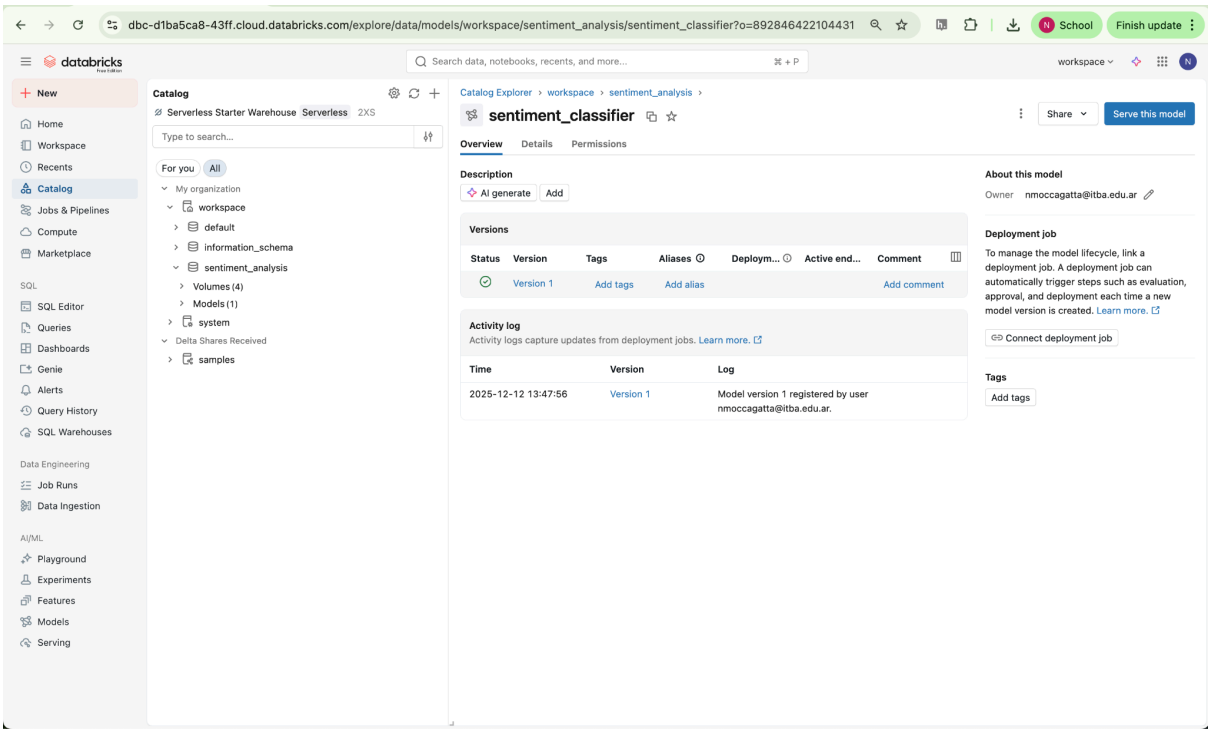



**Figura 2 – Vista del modelo registrado en MLflow**




## 5.3 Mejor modelo

El mejor modelo según el F1 Score es:

- **Modelo ganador:** `LR_C1`
- **F1 Score:** 0.7179
- **Precisión:** 0.7050
- **Recall:** 0.7326

### 5.4 Análisis: ¿por qué ganó este modelo?

Algunas razones que justifican la elección de `LR_C1`:

- **Buen equilibrio entre Precisión y Recall**: mantiene valores altos y relativamente balanceados, lo que es importante en un problema con 3 clases y cierto desbalance.

- **Capacidad de generalización**: la regularización con `C = 1` parece suficiente para evitar overfitting en el espacio de 50k features, manteniendo buena performance en test.

- **Simplicidad y estabilidad**: la Regresión Logística es más simple de interpretar y suele ser robusta para TF-IDF, lo que facilita el debugging y mantenimiento del pipeline.

- Frente a las SVM lineales, el modelo de LR muestra un F1 superior, y frente a Naive Bayes, mejora tanto en F1 como en Recall, especialmente en clases minoritarias.

---

# 6. MLOps Implementado

En este trabajo se implementaron varios componentes clave de MLOps:

- **Experiment tracking con MLflow**
  - Cada corrida registra parámetros, métricas y artefactos.
  - Se utilizó un experimento dedicado para el entrenamiento de modelos de sentimiento.

- **Model registry con Unity Catalog / MLflow Model Registry**
  - El mejor modelo (`LR_C1`) se registró en un **Model Registry**, con versión, tags y descripción.
  - Esto permite promover el modelo a “Staging” o “Production” en futuros desarrollos.

- **Data versioning con Arquitectura Medallion**
  - Las tablas Bronze, Silver y Gold permiten re-entrenar modelos sobre versiones específicas de los datos.
  - La separación en capas facilita auditoría y reproducibilidad.

- **Reproducibilidad**
  - Todos los parámetros relevantes se loguean en MLflow.
  - El código de preprocesamiento y entrenamiento está centralizado en notebooks versionados (y potencialmente en un repo de GitHub).
  - Las splits train/test son estratificadas y determinísticas (semilla fija).

- **Pipelines automatizados en Databricks**
    - Cada etapa (Bronze, Silver, Gold, entrenamiento) puede ser orquestada mediante **jobs** en Databricks.
    - Esto permite ejecutar el flujo completo de forma programada o bajo demanda.

---

# 7. Conclusiones

## 7.1 Qué aprendimos

- Diseñar y operar una **arquitectura de datos escalable** (Medallion) en Databricks.

- A utilizar **MLflow** como herramienta central de experiment tracking y gestión de modelos.

- Que modelos relativamente simples como **Regresión Logística** pueden ser muy competitivos en tareas de texto vectorizado mediante TF-IDF.

- La importancia de separar claramente:
    - **Datos crudos**,
    - **Datos limpios**,
    - **Datos listos para ML**
    - y también la importancia de loguear métricas de calidad en cada paso.

## 7.2 Desafíos encontrados

- Limitaciones de recursos del entorno Databricks Serverless (memoria, tiempo de ejecución).

- Manejo de la alta dimensionalidad (50k features) y el impacto en tiempos de entrenamiento.

- Necesidad de coordinar la experimentación entre varios notebooks y mantener trazabilidad clara.

## 7.3 Mejoras futuras

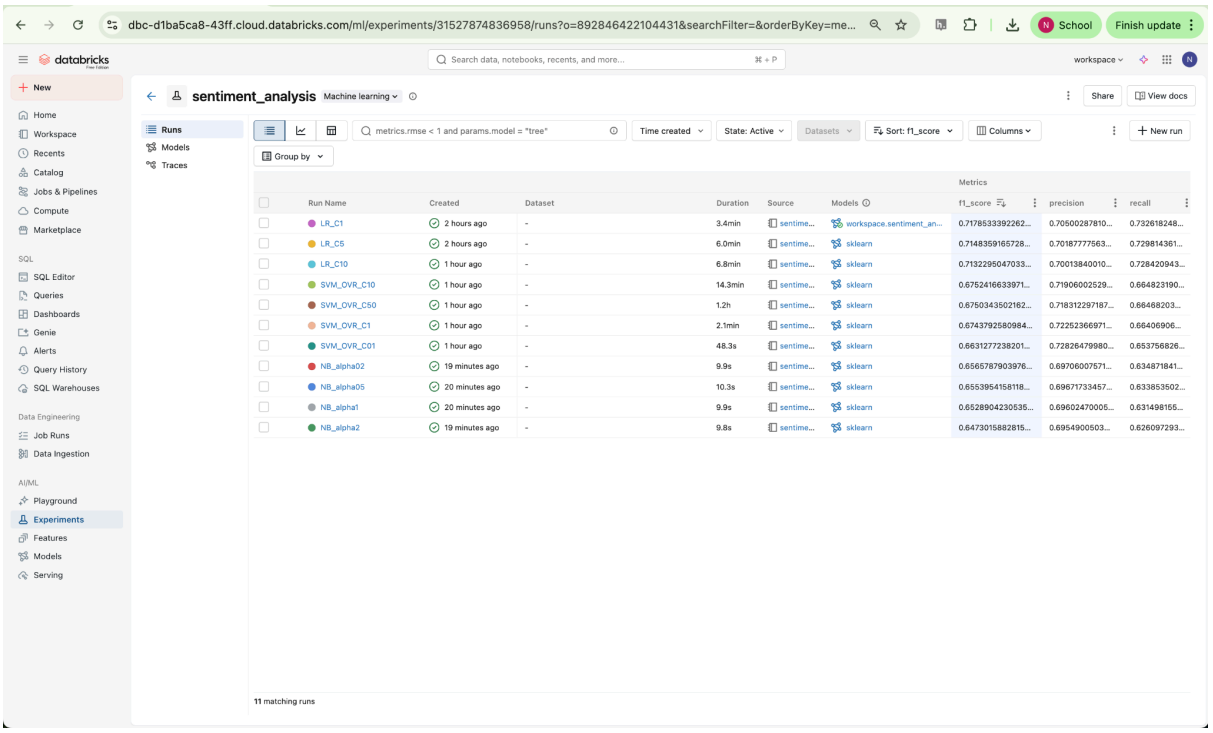
- Incorporar **modelos más avanzados** (por ejemplo, embeddings o modelos pre-entrenados tipo BERT) y compararlos en el mismo entorno de MLflow.

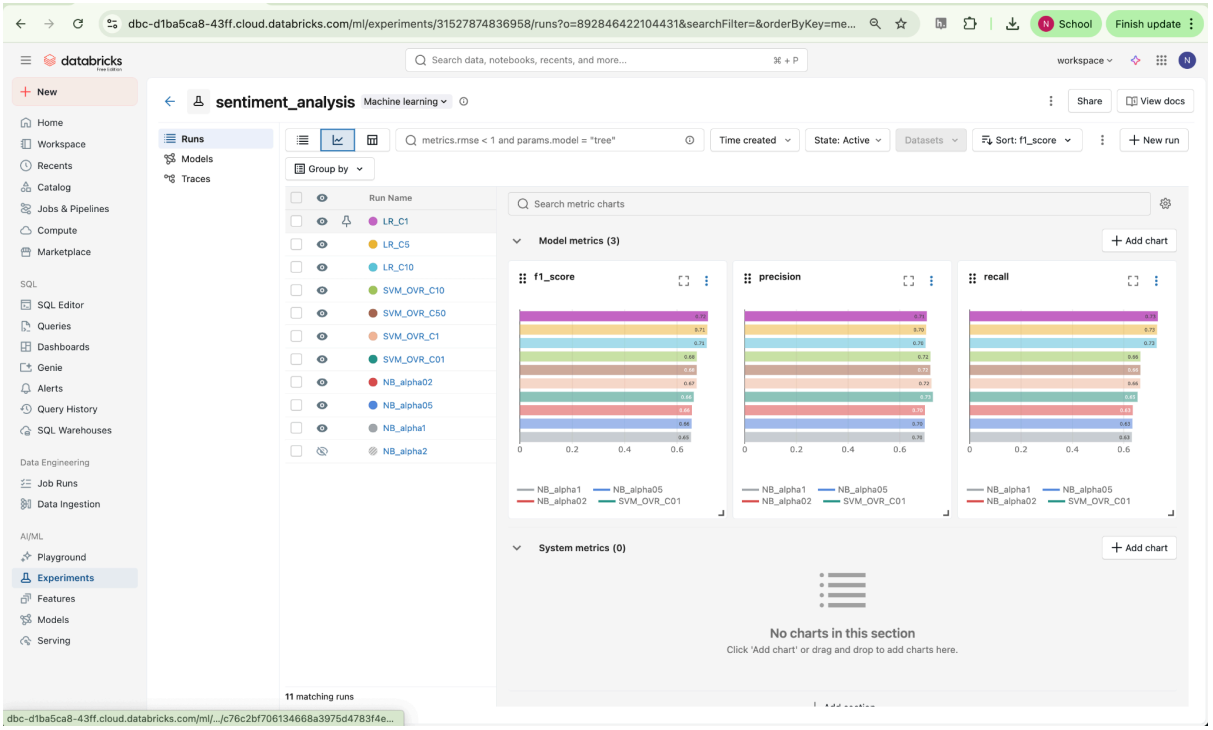
- Extender el pipeline con **monitoreo en producción** (drift de datos, degradación de performance).
- Automatizar completamente la orquestación con **GitHub Actions + Databricks Jobs**, desde la ingesta hasta el registro del modelo.
- Agregar **tests automatizados** para validar esquemas y calidad en cada capa.

---

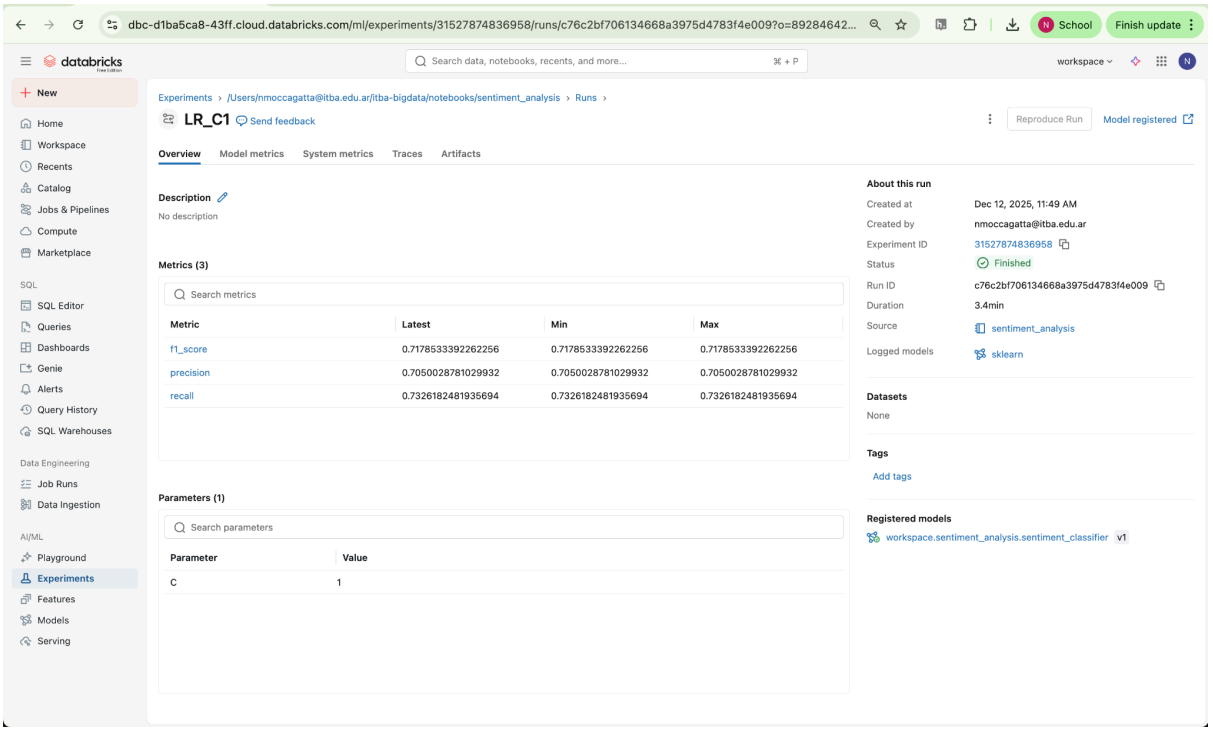
# 8. Screenshots

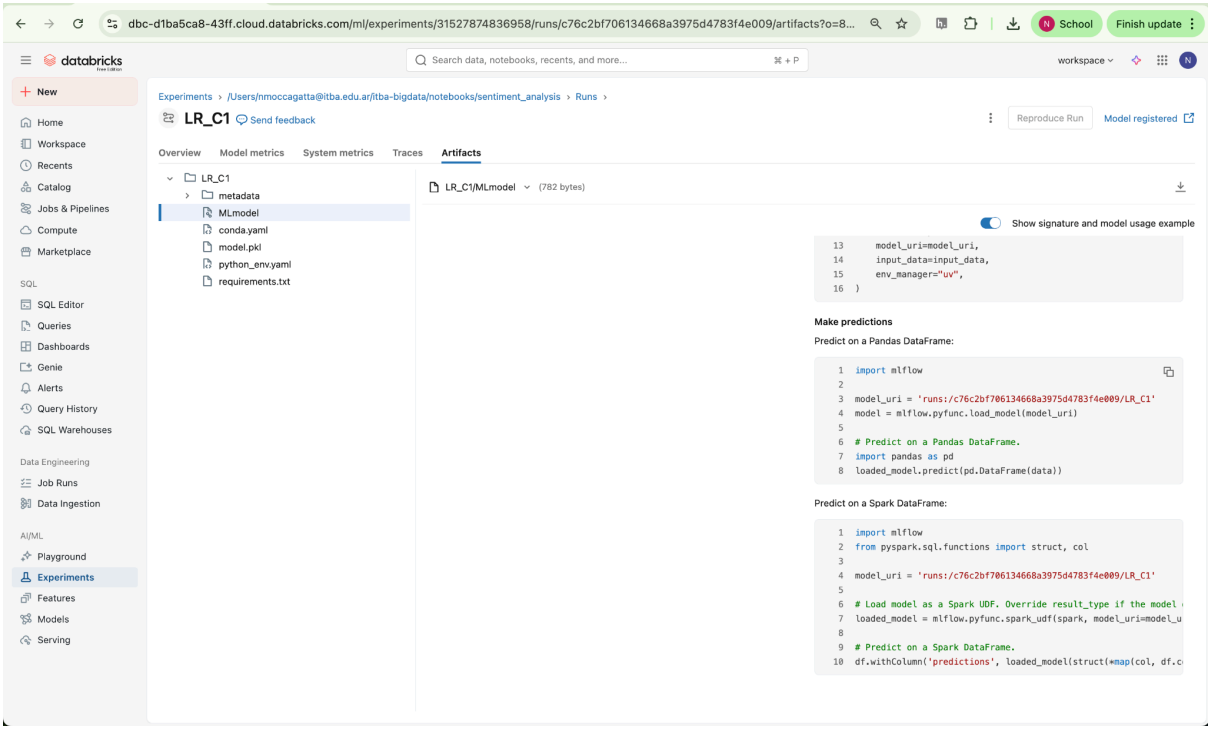
**Captura 1:** Vista del experimento en MLflow con la lista de las corridas correspondientes:

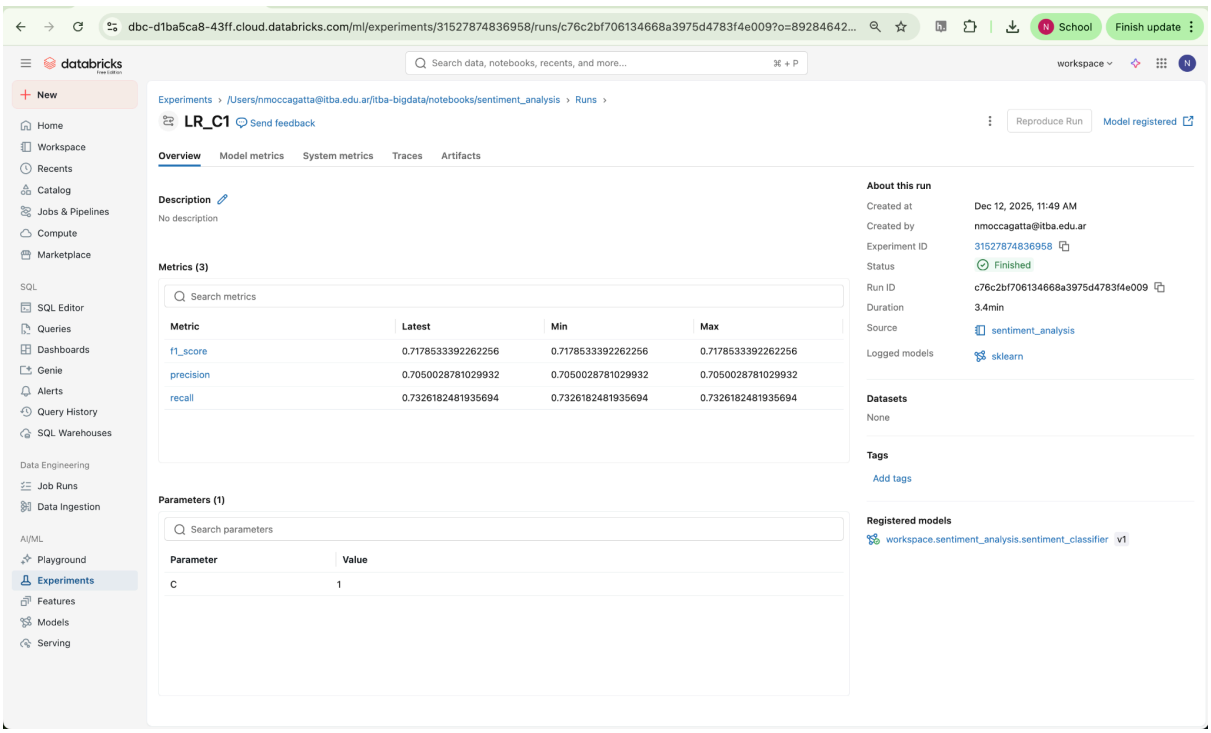




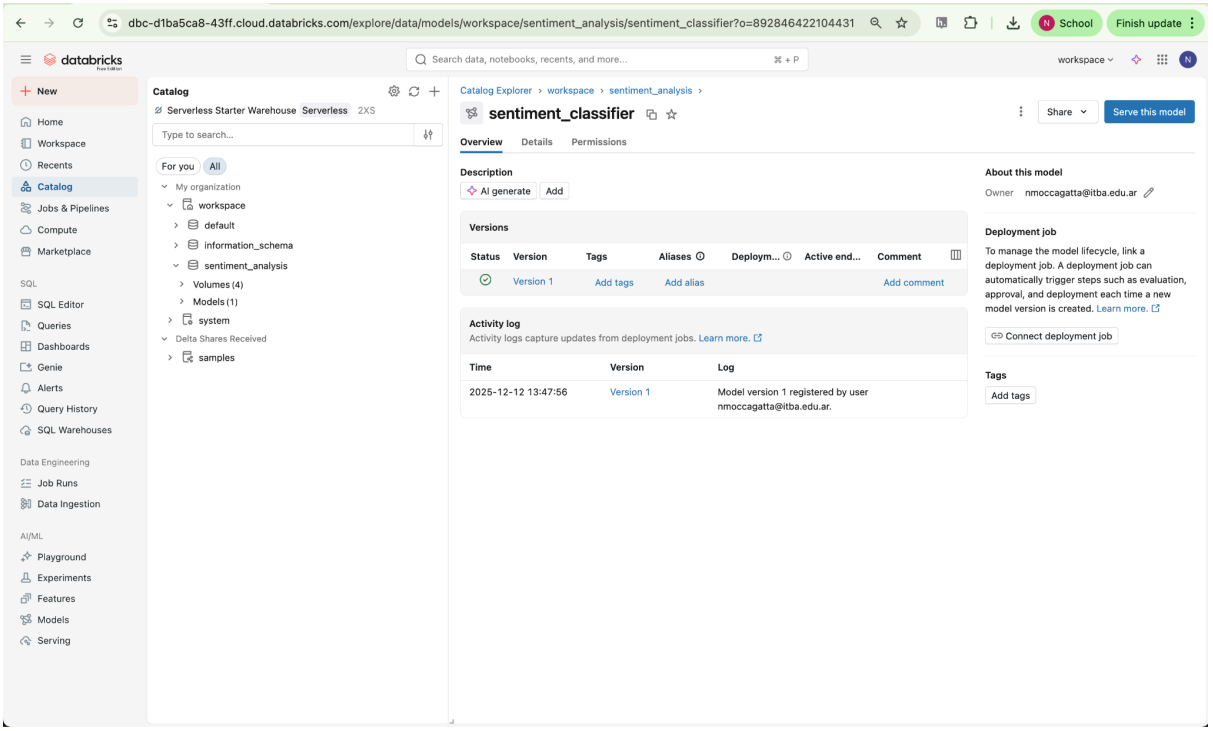

**Captura 2:** Detalle de una run (parámetros, métricas, artefactos).



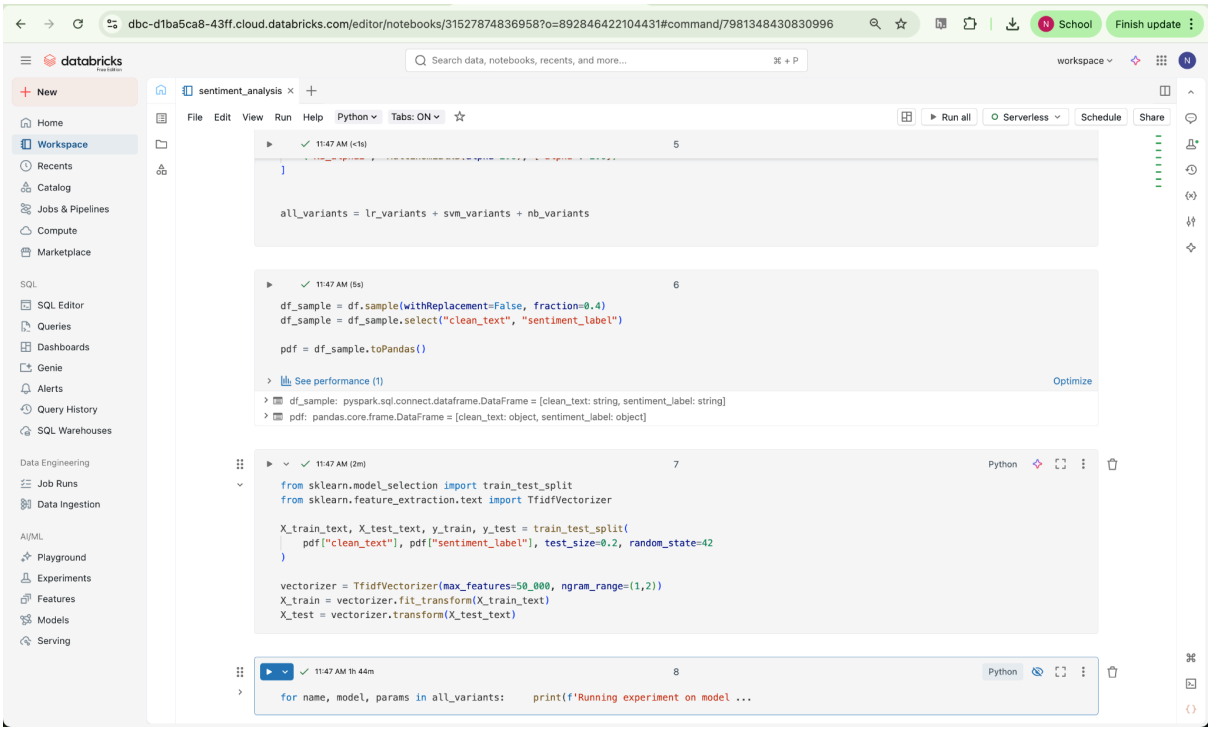

dbc-d1ba5ca8-43ff.cloud.databricks.com/ml/experiments/31527874836958/runs/c76c2bf706134668a3975d4783f4e009/artifacts?o=8...
School
Finish update
databricks
Search data, notebooks, recents, and more...
workspace
New
Home
Workspace
Recents
Catalog
Jobs & Pipelines
Compute
Marketplace
SQL
SQL Editor
Queries
Dashboards
Genie
Alerts
Query History
SQL Warehouses
Data Engineering
Job Runs
Data Ingestion
AI/ML
Playground
Experiments
Features
Models
Serving
Experiments › /Users/nmoccagatta@itba.edu.ar/itba-bigdata/notebooks/sentiment_analysis › Runs ›
LR_C1
Send feedback
Reproduce Run
Model registered
Overview
Model metrics
System metrics
Traces
Artifacts
LR_C1
metadata
MLmodel
conda.yaml
model.pkl
python_env.yaml
requirements.txt
LR_C1/MLmodel (782 bytes)
Show signature and model usage example
model_uri=model_uri,
input_data=input_data,
env_manager="uv",
)
Make predictions
Predict on a Pandas DataFrame:
import mlflow
model_uri = 'runs:/c76c2bf706134668a3975d4783f4e009/LR_C1'
model = mlflow.pyfunc.load_model(model_uri)
# Predict on a Pandas DataFrame.
import pandas as pd
loaded_model.predict(pd.DataFrame(data))
Predict on a Spark DataFrame:
import mlflow
from pyspark.sql.functions import struct, col
model_uri = 'runs:/c76c2bf706134668a3975d4783f4e009/LR_C1'
# Load model as a Spark UDF. Override result_type if the model
loaded_model = mlflow.pyfunc.spark_udf(spark, model_uri=model_u
# Predict on a Spark DataFrame.
df.withColumn('predictions', loaded_model(struct(*map(col, df.c


dbc-d1ba5ca8-43ff.cloud.databricks.com/ml/experiments/31527874836958/runs/c76c2bf706134668a3975d4783f4e009?o=89284642...
School
Finish update
databricks
Search data, notebooks, recents, and more...
workspace
New
Home
Workspace
Recents
Catalog
Jobs & Pipelines
Compute
Marketplace
SQL
SQL Editor
Queries
Dashboards
Genie
Alerts
Query History
SQL Warehouses
Data Engineering
Job Runs
Data Ingestion
AI/ML
Playground
Experiments
Features
Models
Serving
Experiments › /Users/nmoccagatta@itba.edu.ar/itba-bigdata/notebooks/sentiment_analysis › Runs ›
LR_C1
Send feedback
Reproduce Run
Model registered
Overview
Model metrics
System metrics
Traces
Artifacts
Description
No description
Metrics (3)
Search metrics
Metric
Latest
Min
Max
f1_score
0.7178533392262256
0.7178533392262256
0.7178533392262256
precision
0.7050028781029932
0.7050028781029932
0.7050028781029932
recall
0.7326182481935694
0.7326182481935694
0.7326182481935694
Parameters (1)
Search parameters
Parameter
Value
C
1
About this run
Created at
Dec 12, 2025, 11:49 AM
Created by
nmoccagatta@itba.edu.ar
Experiment ID
31527874836958
Status
Finished
Run ID
c76c2bf706134668a3975d4783f4e009
Duration
3.4min
Source
sentiment_analysis
Logged models
sklearn
Datasets
None
Tags
Add tags
Registered models
workspace.sentiment_analysis.sentiment_classifier
v1

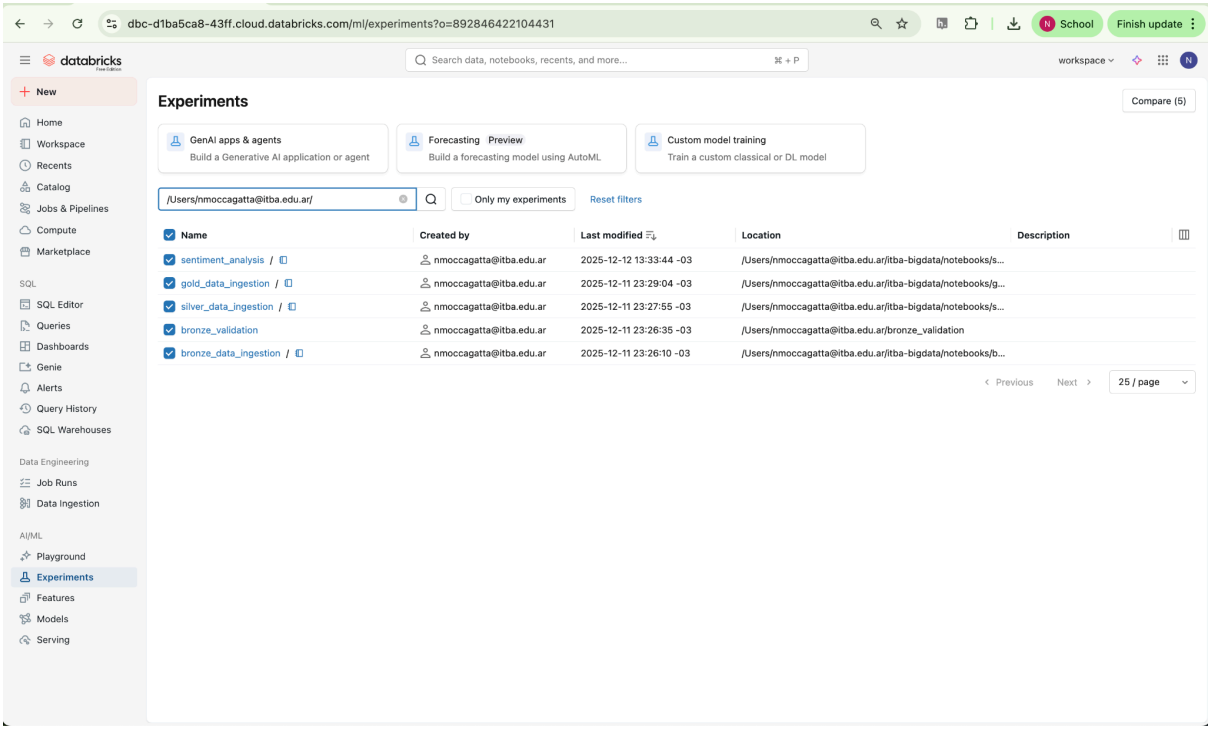
**Captura 3:** Vista del modelo registrado en Model Registry.




**Captura 4:** Vista de la notebook final sentiment_analysis




**Captura 5:** Vista de experimentos relacionados:


dbc-d1ba5ca8-43ff.cloud.databricks.com/ml/experiments?o=892846422104431
School
Finish update
databricks
Search data, notebooks, recents, and more...
⌘ + P
workspace
New
Home
Workspace
Recents
Catalog
Jobs & Pipelines
Compute
Marketplace
SQL
SQL Editor
Queries
Dashboards
Genie
Alerts
Query History
SQL Warehouses
Data Engineering
Job Runs
Data Ingestion
AI/ML
Playground
Experiments
Features
Models
Serving
Experiments
Compare (5)
GenAI apps & agents
Build a Generative AI application or agent
Forecasting Preview
Build a forecasting model using AutoML
Custom model training
Train a custom classical or DL model
/Users/nmoccagatta@itba.edu.ar/
Only my experiments
Reset filters
Name
Created by
Last modified
Location
Description
sentiment_analysis
nmoccagatta@itba.edu.ar
2025-12-12 13:33:44 -03
/Users/nmoccagatta@itba.edu.ar/itba-bigdata/notebooks/s...
gold_data_ingestion
nmoccagatta@itba.edu.ar
2025-12-11 23:29:04 -03
/Users/nmoccagatta@itba.edu.ar/itba-bigdata/notebooks/g...
silver_data_ingestion
nmoccagatta@itba.edu.ar
2025-12-11 23:27:55 -03
/Users/nmoccagatta@itba.edu.ar/itba-bigdata/notebooks/s...
bronze_validation
nmoccagatta@itba.edu.ar
2025-12-11 23:26:35 -03
/Users/nmoccagatta@itba.edu.ar/bronze_validation
bronze_data_ingestion
nmoccagatta@itba.edu.ar
2025-12-11 23:26:10 -03
/Users/nmoccagatta@itba.edu.ar/itba-bigdata/notebooks/b...
Previous
Next
25 / page