



Introducción

Aprendizaje de Máquina II - CEIA - FIUBA

Dr. Ing. Facundo Adrián Lucianna

Dr. Ing. Álvaro Gabriel Pizá

Introducción

- Materia de 8 clases teórico-prácticas
- Clases con diapositivas y desarrollo
- Algunas clases desarrollaremos algún *hands-on* para familiarizarnos con las herramientas.
- Estructuras de las clases:
 - 10 minutos repaso de clase anterior
 - 3 bloques de 50 minutos de clases teórico-prácticas
 - 2 recreos de 10 minutos
 - Ejercicio de práctica entre las clases. Sin entrega y evaluación.

Introducción

- Repositorio de la materia:

https://github.com/FIUBA-Posgrado-Inteligencia-Artificial/aprendizaje_maquina_ll

Consultas

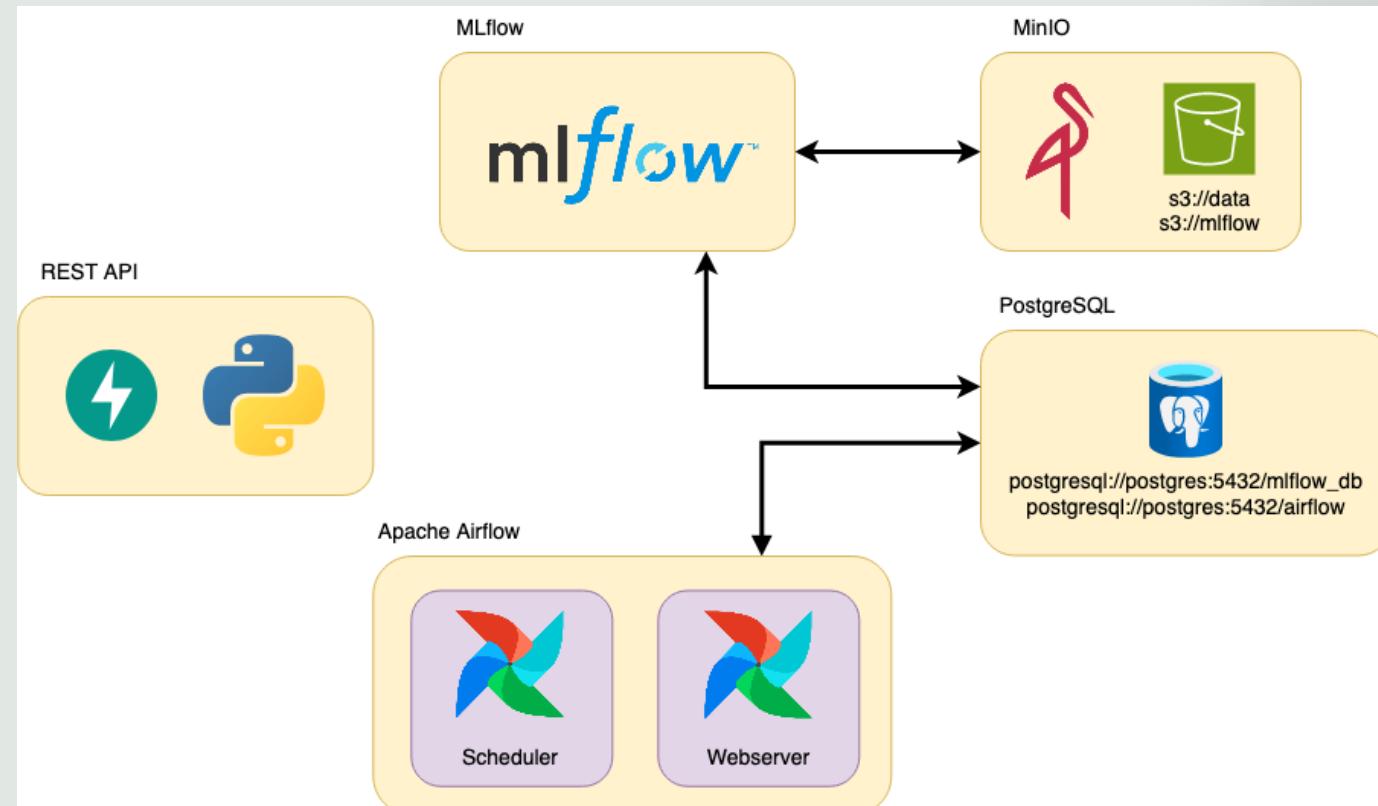
- Servidor en Discord: <https://discord.gg/5cdQt7FD8c>
- Correo
 - Facundo Adrián Lucianna: facundolucianna@gmail.com
 - Álvaro Gabriel Pizá: piza.ag@gmail.com

Evaluación

- La evaluación de los conocimientos impartidos durante las clases será a modo de entrega de un trabajo práctico final. El trabajo es obligatoriamente grupal (máximo 6 personas).
- La fecha de entrega máxima es 7 días después de la última clase.
- La idea de este trabajo es suponer que trabajamos para **ML Models and something more Inc.**, la cual ofrece un servicio que proporciona modelos mediante una REST API o predicción por lote.
- Internamente, tanto para realizar tareas de DataOps como de MLOps, la empresa cuenta con Apache Airflow y MLflow. También dispone de un Data Lake en S3.
- Tenemos tres diferentes tipos de tareas (nivel fácil, medio y alto) que lo vemos más adelante.

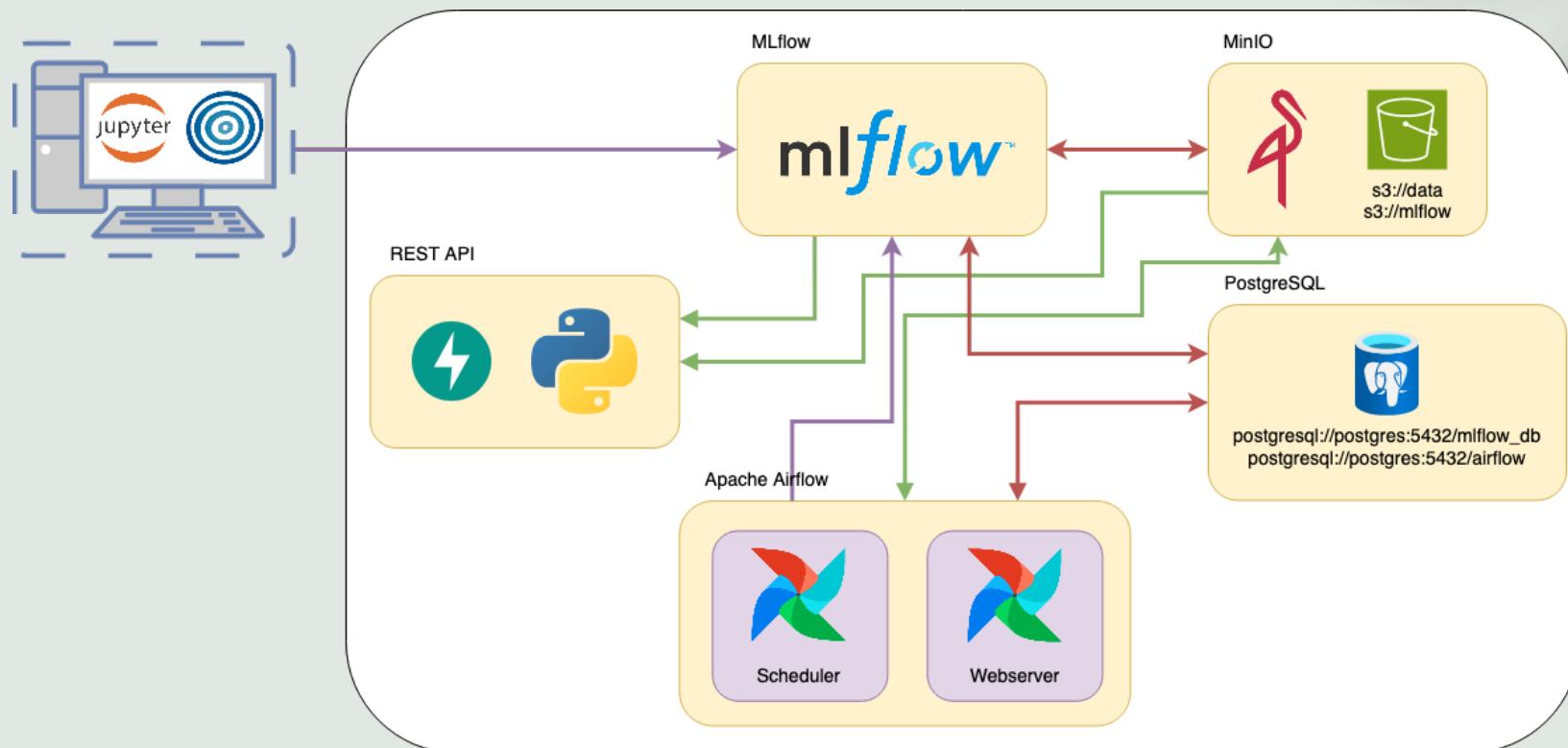
Evaluación

El sistema está desarrollado en Docker y la base se encuentra en: <https://github.com/facundolucianna/amq2-service-ml>



Evaluación

Para ayudarlos, pueden encontrar un ejemplo en: https://github.com/facundolucianna/amq2-service-ml/tree/example_implementation



Evaluación

Ofrecemos tres tipos de evaluaciones:

- **Nivel fácil** (nota entre 4 y 5): Hacer funcionar el ejemplo de aplicación.
Filmar un video del sistema funcionando:
 - Ejecutar en Airflow el DAG llamado *process_etl_heart_data*
 - Ejecuta la notebook (ubicada en *notebook_example*) para realizar la búsqueda de hiper-parámetros y entrenar el mejor modelo.
 - Utilizar la REST-API del modelo.
 - Ejecutar en Airflow el DAG llamado *process_etl_heart_data* y luego *retrain_the_model*.
 - En todos estos pasos verificar lo que muestra MLFlow.
- **Nivel medio** (nota entre 6 y 8): Implementar en local usando Metaflow el ciclo de desarrollo del modelo que desarrollaron en Aprendizaje de Máquina I y generar un archivo para predicción en bache (un csv o un archivo de SQLite). La nota puede llegar a 10 si implementan una base de datos (ya sea KVS u otro tipo) con los datos de la predicción en bache.
- **Nivel alto** (nota entre 8 y 10): Implementar el modelo que desarrollaron en Aprendizaje de Máquina I en este ambiente productivo. Para ello, pueden usar los recursos que consideren apropiado. Los servicios disponibles de base son Apache Airflow, MLflow, PostgresSQL, MinIO, FastAPI. Todo está montado en Docker, por lo que además deben instalado Docker.



python™

```
101
102     if city != "all":
103         df_operation_table = df_operation_table[df_operation_table["city_id"] == city]
104     if trunk != "all":
105         df_operation_table = df_operation_table[df_operation_table["trunk_id"] == trunk]
106
107     if df_operation_table.empty:
108         logg.warning("No data to be read", status_code=404, reason="No data to be read",
109                     country=country, city=city, trunk=trunk)
110         return 404, "No data to be read", None
111
112     # Force the datatype to avoid problems
113     df_operation_table['app_store_id'] = df_operation_table['app_store_id'].astype(str, errors='ignore')
114     df_operation_table['brand_id'] = df_operation_table['brand_id'].astype(np.int64, errors='ignore')
115     df_operation_table['branch_id'] = df_operation_table['branch_id'].astype(np.int64, errors='ignore')
116     df_operation_table['internal_store_id'] = df_operation_table['internal_store_id'].astype(str,
117                                              errors='ignore')
118
119     df_operation_table['lat'] = df_operation_table['lat'].astype(np.float64, errors='ignore')
120     df_operation_table['lng'] = df_operation_table['lng'].astype(np.float64, errors='ignore')
121
122     # Add the currency information
123     df_operation_table['base_currency'] = dfx.get_base_currency(currency)
124
125     # Rename columns
126     df_operation_table.rename(columns={'internal_store_id': "store_id"}, inplace=True)
127
128     # Make compatibility between changes in backoffice
129     if 'company_id' not in df_operation_table.columns:
130         df_operation_table['operator_id'] = df_operation_table['operator_id'].astype(np.int64, errors='ignore')
131         df_operation_table['company_id'] = df_operation_table['operator_id']
132         df_operation_table['company_name'] = df_operation_table['operator_name']
133     else:
134         df_operation_table['company_id'] = df_operation_table['company_id'].astype(np.int64, errors='ignore')
135
136     # Drop all this columns
137     drop_columns_list = ['operator_id', 'operator_name']
138     df_operation_table = df_operation_table.drop(drop_columns_list, errors='ignore', axis=1)
139
140
141     except Exception as e:
142         logg.error("Problem loading the operation table", status_code=500, reason=e,
143                    path=unquote(path_snapshot_operation.as_uri()))
144         return 500, ("Problem loading the operation table: " + str(e) + " - path: " +
145                     unquote(path_snapshot_operation.as_uri())), None
146
147
148     return 200, None, df_operation_table
```

Herramientas

- Lenguaje de Programación
 - Python >=3.10
 - Poetry / Pip / Conda para instalar librerías
- Librerías
 - MLflow
 - Librerías de manejo de datos y de modelos de aprendizaje automático.
 - Jupiter Notebook
- Herramientas
 - GitHub para repositorios
 - Docker
 - Apache Airflow
- IDE Recomendados
 - Visual Studio Code
 - PyCharm Community Edition

Programa

- 1 Ciclo de vida de un proyecto de Aprendizaje Automático. Machine Learning Operations (MLOps). Niveles de MLOps. Entorno productivo. Buenas prácticas de programación.
- 2 Desarrollo de modelos. Selección del tipo de modelo. Las 4 fases del desarrollo de modelos. Ensambles. Depurando modelos. Entrenamiento distribuido. Métodos de evaluación. Desplegado de modelos. Contenedores y Docker.
- 3 Infraestructura. Capa de almacenamiento. Capa de cómputo. Plataforma de ML. MLFlow.
- 4 Administración de recursos. Orquestadores y sincronizadores. Gestión del flujo de trabajo de ciencia de datos. Apache Airflow

Programa

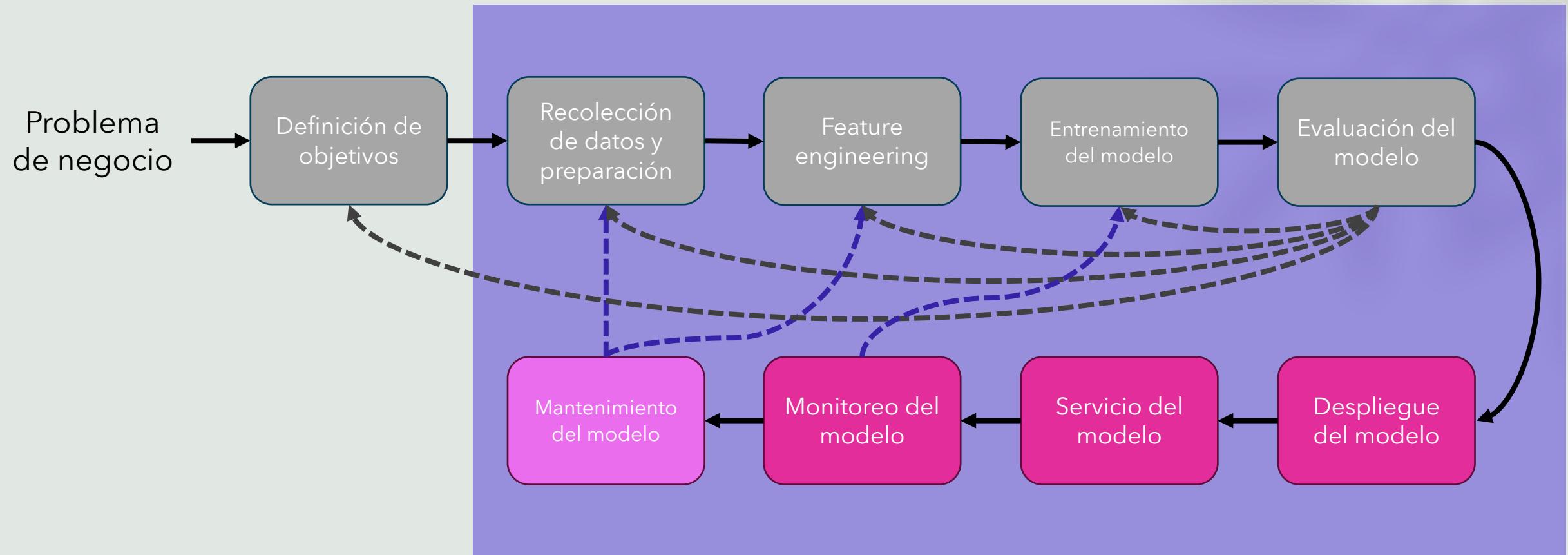
- 5 Despliegue de modelos. Estrategias de despliegue. Sirviendo modelos. Propiedades del entorno de ejecución de un modelo. Predicción en lotes
- 6 Despliegue de modelos. Desplegado on-line. APIs y Microservicios. REST API. Implementación de REST APIs en Python
- 7 Sirviendo modelos en el mundo real. Estrategias de implementación. Ejemplo de servicios de modelos

Bibliografía

- Designing Machine Learning Systems. An Iterative Process for Production-Ready Applications - Chip Huyen (Ed. O'Reilly)
- Machine Learning Engineering with Python: Manage the production life cycle of machine learning models using MLOps with practical examplesv - Andrew P. McMahon (Ed. Packt Publishing)
- Engineering MLOps: Rapidly build, test, and manage production-ready machine learning life cycles at scale - Emmanuel Raj (Ed. Packt Publishing)
- Introducing MLOps: How to Scale Machine Learning in the Enterprise - Mark Treveil, Nicolas Omont, Clément Stenac, Kenji Lefevre, Du Phan, Joachim Zentici, Adrien Lavoillotte, Makoto Miyazaki, Lynn Heidmann (Ed. O'Reilly)
- Practical MLOps: Operationalizing Machine Learning Models - Noah Gift, Alfredo Deza (Ed. O'Reilly)
- Machine Learning Engineering - Andriy Burkov (Ed. True Positive Inc.)
- Machine Learning Engineering in Action - Ben Wilson (Manning)

Ciclo de vida de un proyecto de Aprendizaje Automático

Ciclo de vida de un proyecto de Aprendizaje Automático



Ciclo de vida de un proyecto de Aprendizaje Automático

A lo largo del ciclo de vida de un proyecto de ML deben intervenir varios participantes para que el desarrollo se lleve a cabo de la mejor manera posible. La distribución de tareas en los distintos roles puede variar según cada una de las organizaciones, pero de manera general podemos definir las siguientes:

- Data Engineer
- Data Scientist
- Data Analyst
- Machine Learning Engineer

Ciclo de vida de un proyecto de Aprendizaje Automático

Data Engineer

El Data Engineer es responsable de la **preparación y limpieza de los datos**, la creación de **pipelines de datos** y la integración de diferentes fuentes de datos. Su trabajo también incluye la selección de las herramientas y tecnologías adecuadas para la gestión de datos y la implementación de soluciones de **almacenamiento y procesamiento de datos escalables**.

Data Scientist

El Data Scientist se encarga de **definir y crear modelos de machine learning** que permitan hacer predicciones a partir de los datos. Su trabajo implica seleccionar los algoritmos adecuados, entrenar los modelos y optimizar su rendimiento. Los Data Scientists también pueden participar en la identificación de variables relevantes y en la **exploración de los datos para encontrar patrones y tendencias**.

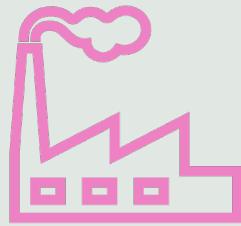
Data Analyst

El Data Analyst trabaja con datos para descubrir patrones y tendencias que puedan ser útiles para la **toma de decisiones empresariales**. Su trabajo implica realizar análisis estadísticos y visualizaciones de datos para **entenderlos mejor** y hacer recomendaciones sobre cómo pueden utilizarse **para mejorar el negocio**.

Machine Learning Engineer

El Machine Learning Engineer es responsable de llevar los modelos de machine learning a **producción y asegurarse de que estén funcionando correctamente**. Su trabajo implica seleccionar la infraestructura adecuada para el despliegue de los modelos, integrar los modelos con otras aplicaciones y sistemas, y **supervisar el rendimiento de los modelos**.

Consideraciones para aplicaciones en industria



Producción

Para que el modelo pueda entregar valor al negocio debe estar productivo.



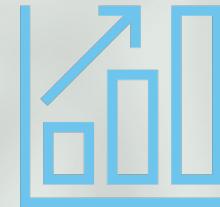
Usabilidad

Un modelo con 70% de exactitud en producción produce mucho más valor que uno con 100% de exactitud que no se puede usar.



Dependencia

Los modelos en producción requieren mantenimiento para prevenir el desvío en los datos o en el target.

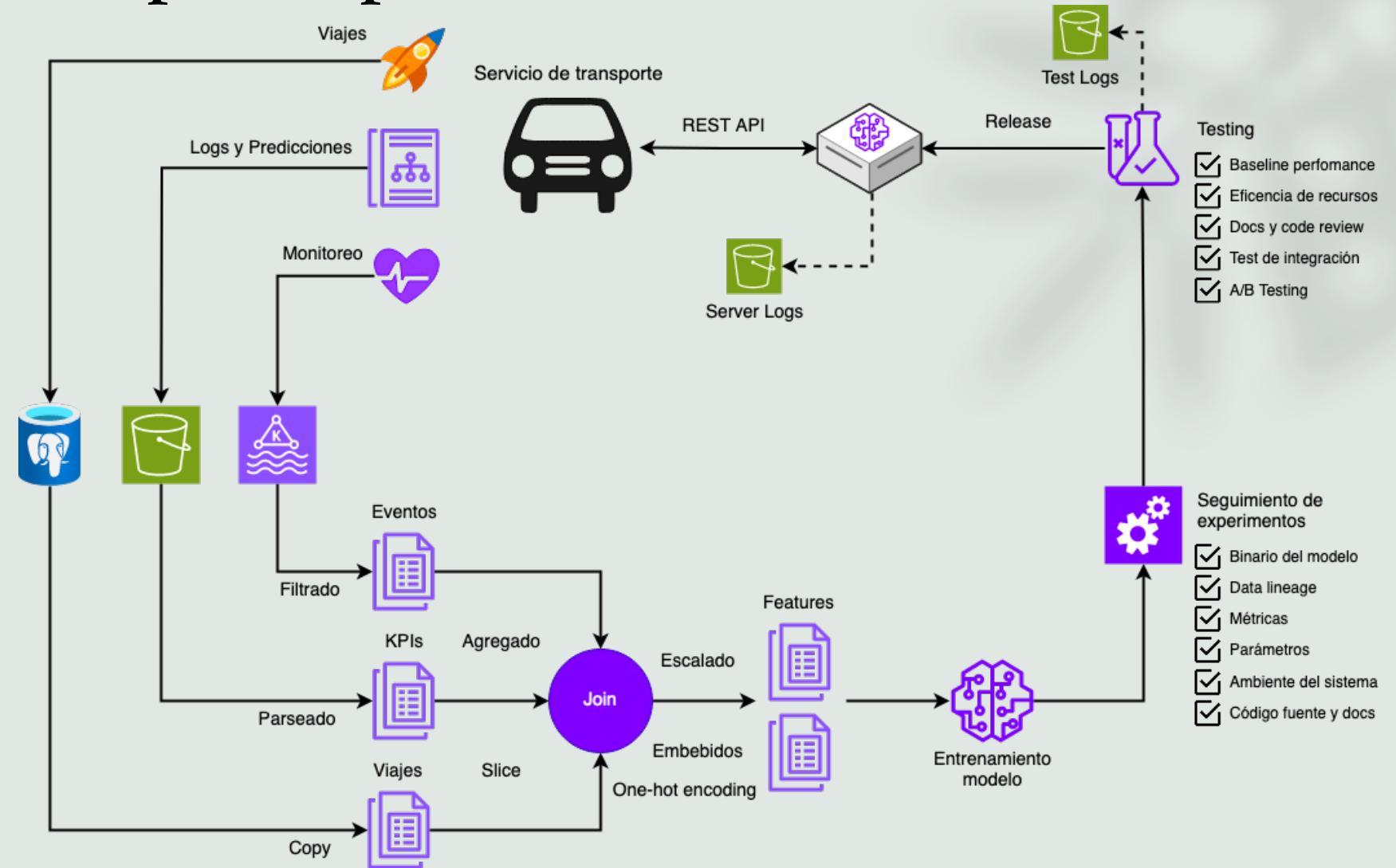


Escalabilidad

El proceso debe ser implementado para que otras personas del equipo lo entiendan, debe ser transparente y replicable.

Consideraciones para aplicaciones en industria

Veamos un sistema ejemplo de una implementación de un servicio de transporte tipo Uber y el modelo de ML que determina el precio del viaje.



Pipelines/flujos de trabajo reproducibles dentro de ML

¿Qué es un pipeline?

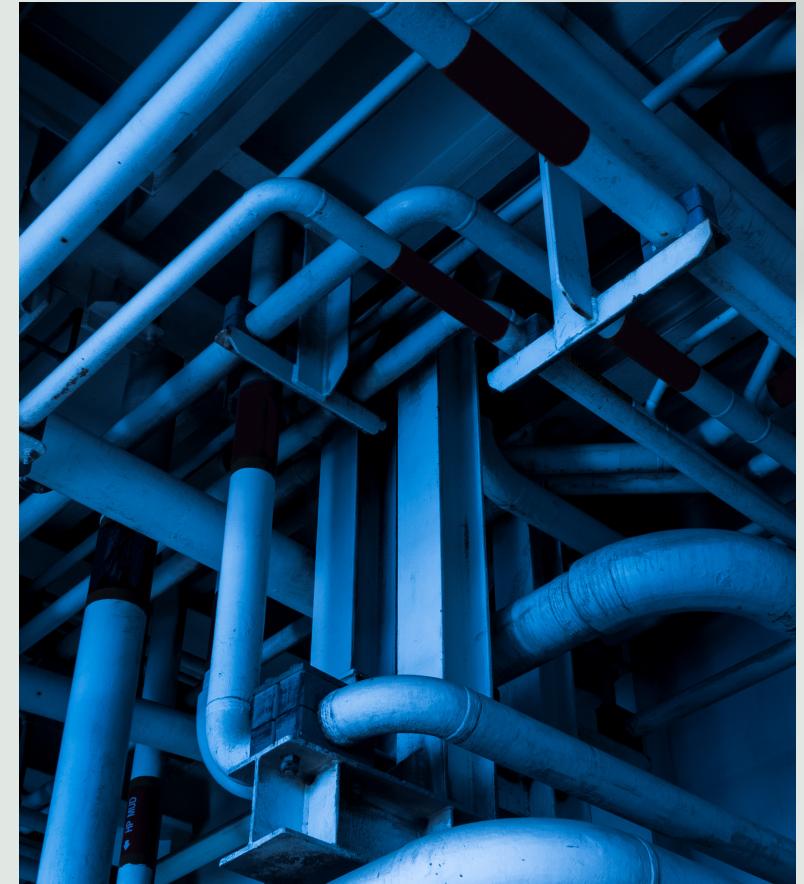
Un pipeline de datos es una construcción lógica que representa un proceso dividido en fases.

Los pipelines de datos se caracterizan por definir el conjunto de pasos o fases y las tecnologías involucradas en un proceso de movimiento o procesamiento de datos.

Esto nos permite encapsular el código, hacerlo más legible, más ordenado, estandarizar y automatizar los procesos, entre otros beneficios.

Los pipelines de Machine Learning permiten a los equipos de datos iterar rápidamente sobre diferentes modelos y ajustes y mejorar continuamente el rendimiento del modelo.

Los pipelines están conformados por **componentes** y por **artefactos** (artifacts).



Pipelines/flujos de trabajo reproducibles dentro de ML

Componentes

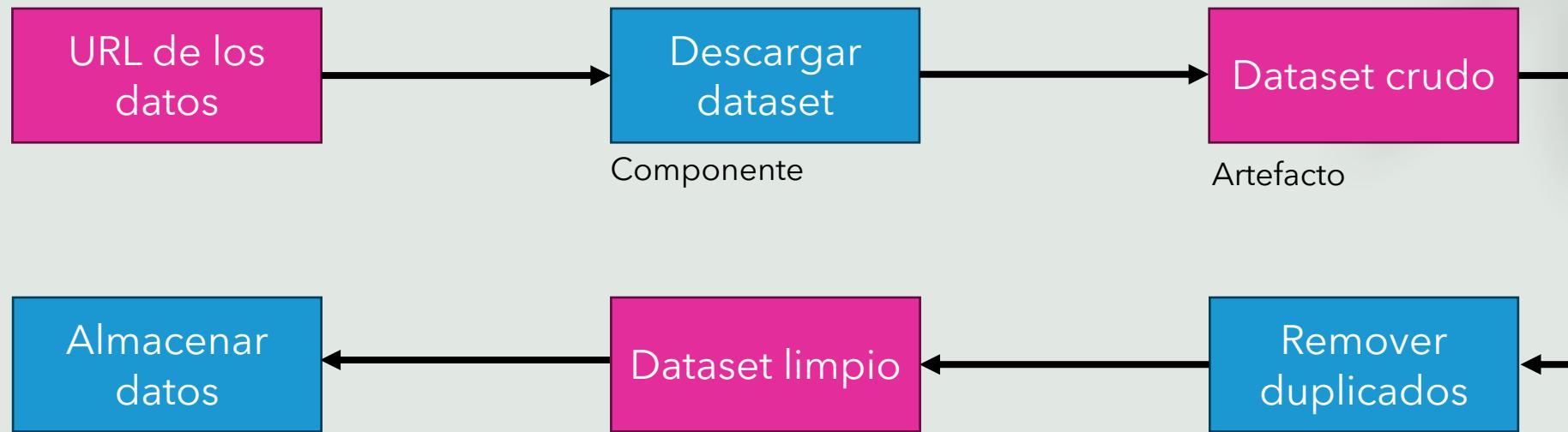
Los componentes o pasos de un pipeline son piezas de código reutilizables y modulares que reciben una o varias entradas y producen una o varias salidas. Pueden ser scripts, notebooks u otro ejecutable.

Artifact

Los artifacts son el resultado de los componentes, su salida. Estos pueden convertirse en la entrada de uno o más componentes para unir los distintos pasos de un pipeline. Los artifacts deben ser trackeados y versionados.

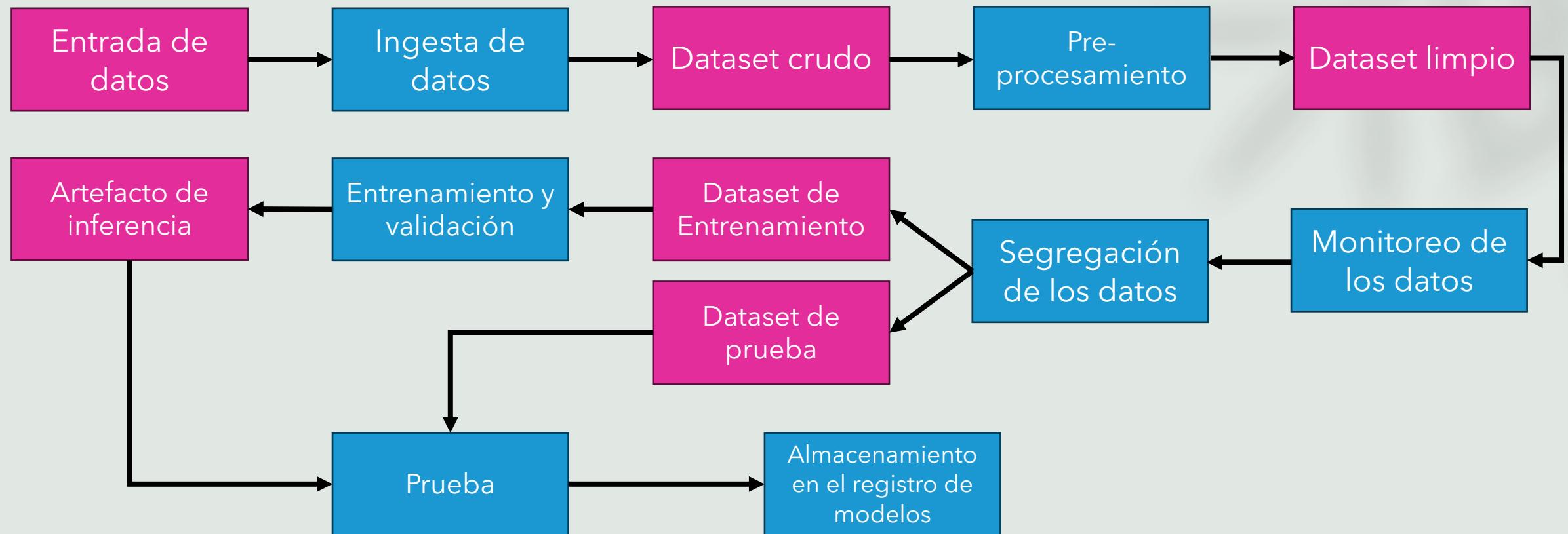
Pipelines/flujos de trabajo reproducibles dentro de ML

Ejemplo de pipeline de Extract Transform Load



Pipelines/flujos de trabajo reproducibles dentro de ML

Ejemplo de pipeline de entrenamiento



Machine Learning Operations (MLOps)

Machine Learning Operations (MLOps)

MLOps, o **Machine Learning Operations**, es un término que se refiere a las prácticas y herramientas utilizadas para gestionar y desplegar modelos de aprendizaje automático a gran escala en producción de manera efectiva y eficiente.

MLOps es una **disciplina emergente** que se enfoca en la gestión de los modelos de machine learning en producción y busca establecer procesos y herramientas para garantizar que los modelos de machine learning sean precisos, escalables y adaptables a diferentes situaciones.

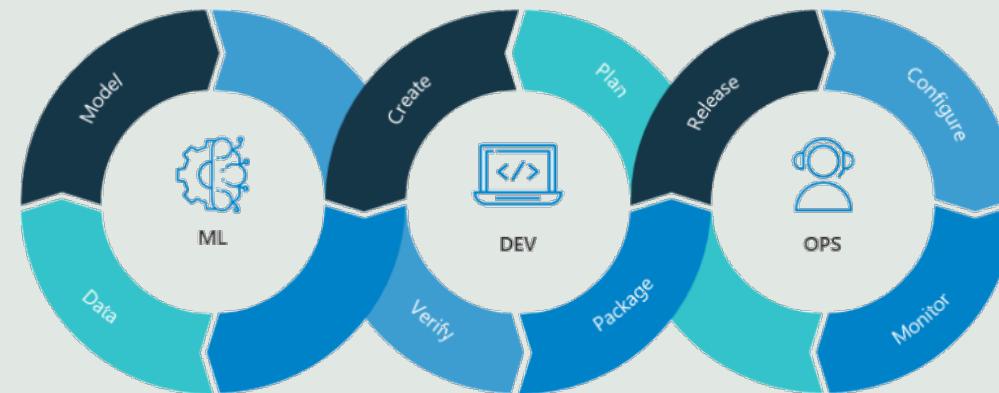


Imagen obtenida de Neural Analytics

Niveles de MLOps

Los 3 niveles de MLOps

Frecuentemente dentro de la industria se pueden encontrar diferenciados tres niveles de MLOps. Estos niveles se diferencian en cuanto a la cantidad de herramientas/prácticas de MLOps que incluyen dentro de su funcionamiento.

- Nivel 0
- Nivel 1
- Nivel 2

Niveles de MLOps

Nivel 0 de MLOps

En este nivel no hay prácticas de MLOps en el proceso. Es adecuado para proyectos personales, cuando se está probando algún concepto/arquitectura nueva, para POCs, etc.

En estos casos las ventajas de MLOps se dejan de lado debido a los tiempos de entrega o presupuestos destinados para esas etapas de desarrollo.

Características de esta etapa:

- **El código es monolítico**: se compone de uno o pocos scripts/notebooks que tienen una reusabilidad muy limitada.
- **El objetivo del desarrollo es el modelo y sus métricas**, no un pipeline de ML.
- **El foco del equipo no es la puesta en producción** del modelo, si se decide llevarlo a producción tal vez sea tarea de otro equipo de trabajo.
- **No hay conocimiento de la necesidad de monitoreo y reentrenamiento** del modelo.

Niveles de MLOps

Nivel 1 de MLOps

Este nivel de MLOps es importante cuando ya pasamos por la etapa de POCs y el equipo empieza a pensar en pasar a producción, por lo que se deben considerar procesos más maduros para un desarrollo de software.

Características de esta etapa:

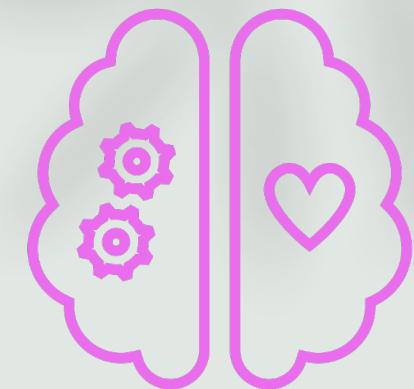
- **El objetivo de esta etapa es un pipeline de ML** que sea reproducible y que, por ejemplo, facilite el re-entrenamiento sobre nuevos datos.
- El pipeline es desarrollado con **componentes reutilizables**.
- El código, los artefactos y experimentos se comienzan a seguir (**tracking**) para generar **reproducibilidad y transparencia**.
- **La salida del pipeline de ML es un artefacto de inferencia** que contiene los pasos de preprocessamiento.
- Se incorpora el **seguimiento/monitoreo** del modelo en producción.

Niveles de MLOps

Ventajas de implementar el Nivel 1

Con respecto a la implementación manual del nivel 0, al implementar el nivel 1 de MLOps podemos obtener las siguientes ventajas:

- Estandarización de los procesos
- Desarrollo más rápido de prototipos: reutilización de código
- Rapidez en llevar al mercado nuevos productos de datos
- Evitar **model drift**, la efectividad de predicción que va perdiendo un modelo debido a los cambios de donde se originan los datos.



Niveles de MLOps

Nivel 2 de MLOps

Este nivel de MLOps está pensado para compañías o proyectos de ML de gran escala, largo alcance y con un nivel de madurez muy avanzado. En esta etapa se cambia el foco del trabajo de desarrollar el pipeline de ML a mejorar sus componentes.

El nivel 2 de MLOps asume que ya se cuenta con múltiples pipelines de ML productivos y continúa aumentando el nivel de automatización aún más.

Características de esta etapa:

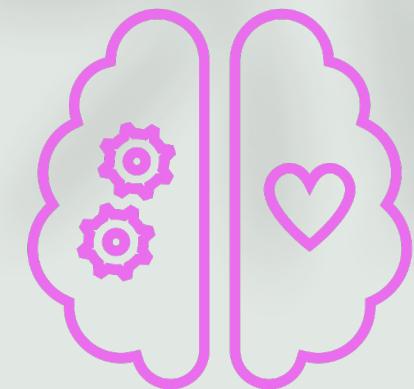
- **Integración continua (CI):** cada vez que un componente es modificado se ejecutan pruebas de integración para verificar que el componente funciona de la forma esperada.
- **Despliegue continuo (CD):** cada componente que pasa satisfactoriamente las pruebas es desplegado de manera automática y comienza a ejecutarse en producción como parte de los pipelines de ML.
- **Entrenamiento continuo:** cuando un componente cambia o cuando ingresan datos con nuevas distribuciones, se disparan las ejecuciones de los pipelines de entrenamiento y el proceso de CI/CD es ejecutado nuevamente.

Niveles de MLOps

Ventajas de implementar el Nivel 2

Con respecto a la implementación del nivel 1, al implementar el nivel 2 de MLOps podemos obtener las siguientes ventajas:

- Iteración más rápida para llevar pipelines a producción
- Es más sencillo implementar A/B testing sobre los cambios
- El trabajo cooperativo entre grandes equipos de personas se facilita
- Se comienza a trabajar en la mejora continua del proceso productivo



Niveles de MLOps

Comparación entre los distintos niveles de MLOps

	Objetivo	Re-entrenamiento	Equipo de trabajo	Aplicación	Producción	Reutilización	Infraestructura	Dificultad
Nivel 0	Modelo	Difícil, manual	1-5	POCs	No	No	Poca	Fácil
Nivel 1	Pipeline	Fácil, manual o disparador	1-20	Pequeña/ mediana escala	Si	Si	Intermedia	Media
Nivel 2	Pipeline	Fácil, automático	+10	Mediana/ gran escala	Si	Si	Mucha	Difícil

¿Qué es producción?

¿Qué es producción?

Entorno de desarrollo

Entorno donde comienzan a gestarse los proyectos, se realizan los primeros análisis exploratorios de datos y POCs.

Es un entorno donde podemos hacer pruebas sin miedo a que, si nos equivocamos, afectemos un proceso crítico.

Debe ser lo más parecido posible al entorno productivo.

Entorno productivo

Entorno donde se ejecutan los procesos que ya fueron validados por el negocio.

Hay más tareas que se ejecutan de manera automática, por ejemplo: predicciones, tests unitarios sobre funciones, etc.

Es un entorno más estable que el de desarrollo.

¿Qué es producción?

Propósito

Un **entorno de desarrollo** se utiliza para desarrollar y probar nuevas aplicaciones y funcionalidades.

Un **entorno de producción** se utiliza para alojar aplicaciones y servicios que están siendo utilizados por los usuarios finales.

Escala

Un **entorno de desarrollo** se suele ejecutar en una sola máquina o en un pequeño grupo de máquinas, mientras que un **entorno de producción** suele tener múltiples máquinas y una mayor capacidad para manejar grandes volúmenes de datos.

Configuración

En un **entorno de desarrollo**, la configuración es más flexible y menos rigurosa, y los desarrolladores pueden hacer cambios y ajustes según sea necesario. En un **entorno de producción**, la configuración es más rígida y estándar para garantizar la estabilidad y seguridad de los sistemas.

¿Qué es producción?

Acceso

En un **entorno de desarrollo**, los desarrolladores tienen un acceso completo y libre para modificar y probar el sistema. En cambio, en un **entorno de producción**, el acceso se limita solo a aquellos usuarios que necesitan interactuar con el sistema para cumplir con sus roles y responsabilidades.

Mantenimiento

En un **entorno de desarrollo**, los desarrolladores son responsables de mantener el sistema y corregir los errores que se encuentran durante el proceso de desarrollo y pruebas. En cambio, en un **entorno de producción**, el equipo de operaciones y soporte son responsables de mantener el sistema y corregir los errores en un ambiente de producción en vivo.

¿Qué es producción?

Estandarización del entorno de desarrollo

De lo que vimos, un entorno de producción es un entorno muy estandarizado, pero en general el entorno de desarrollo se deja de lado, el cual debe ser estandarizado idealmente a nivel empresa, pero al menos equipo.

Cuando se trabaja en equipo todos deben usar las mismas herramientas, por ejemplo, un desarrollo de Python,

- Se debe usar la misma versión de Python, y principalmente la misma versión que el usado en el entorno de producción.
- Las librerías usadas, registrando con detalle la versión usada.

Esto se puede hacer mediante scripting, un script usando pyenv-virtualenv o conda, creando el entorno virtual e instalando la versión correcta de Python y las librerías. Esto se puede subir a un repositorio, y cada vez que hay un nuevo miembro, que ejecute ese script.

¿Qué es producción?

Estandarización del entorno de desarrollo

Una forma de estandarizar fácilmente el entorno de desarrollo es pasar a una solución de la nube, el cual permite estandarizar a todo miembro del equipo a tener las mismas herramientas.

Al hacer esto siempre se asegura que todos tengan el mismo stack de tecnología, corriendo sobre el mismo hardware y además es fácil de administrar por el equipo de IT.

Algunas soluciones ofrecen hasta el IDE en la nube, como [Amazon SageMaker Studio](#).

Otras soluciones permiten usar el IDE que el usuario desee, pero corriendo en una máquina de la nube, tal como la solución propuesta por [GitHub Codespaces](#).

Buenas prácticas de programación

Buenas prácticas de programación

Para comenzar a trabajar pensando en un modelo de aprendizaje automático que será productivo, y visto por otras personas, el código debe cumplir con ciertos estándares de buenas prácticas de programación.

Cuando nuestro código va a ser potencialmente usado en producción, debe cumplir con ser **legible, simple** y **conciso**.

Buenas prácticas de programación

Realicemos un *Hands-on* de cómo llevar nuestro desarrollo de una notebook a archivos fuentes usando buenas prácticas...