Programa de formación

MACHINE LEARNING AND DATA SCIENCE MLDS

Facultad de INGENIERÍA

UNIVERSIDAD NACIONAL DE COLOMBIA

Módulo 6

Metodologías de Desarrollo de Aplicaciones con Machine Learning

Unidad 2

Estructura Estandarizada, Herramientas y Utilidades

Facultad de INGENIERÍA

UNIVERSIDAD NACIONAL DE COLOMBIA

Bienvenida

Jorge Eliécer Camargo Mendoza, PhD.

UNIVERSIDAD NACIONAL DE COLOMBIA

https://dis.unal.edu.co/~jecamargom/

jecamargom@unal.edu.co

Departamento de Ingeniería de Sistemas e Industrial
Facultad de Ingeniería
Universidad Nacional de Colombia
Sede Bogotá

Tabla de contenidos

Objetivos de aprendizaje

Unidad 2 – Estructura Estandarizada, Herramientas y Utilidades

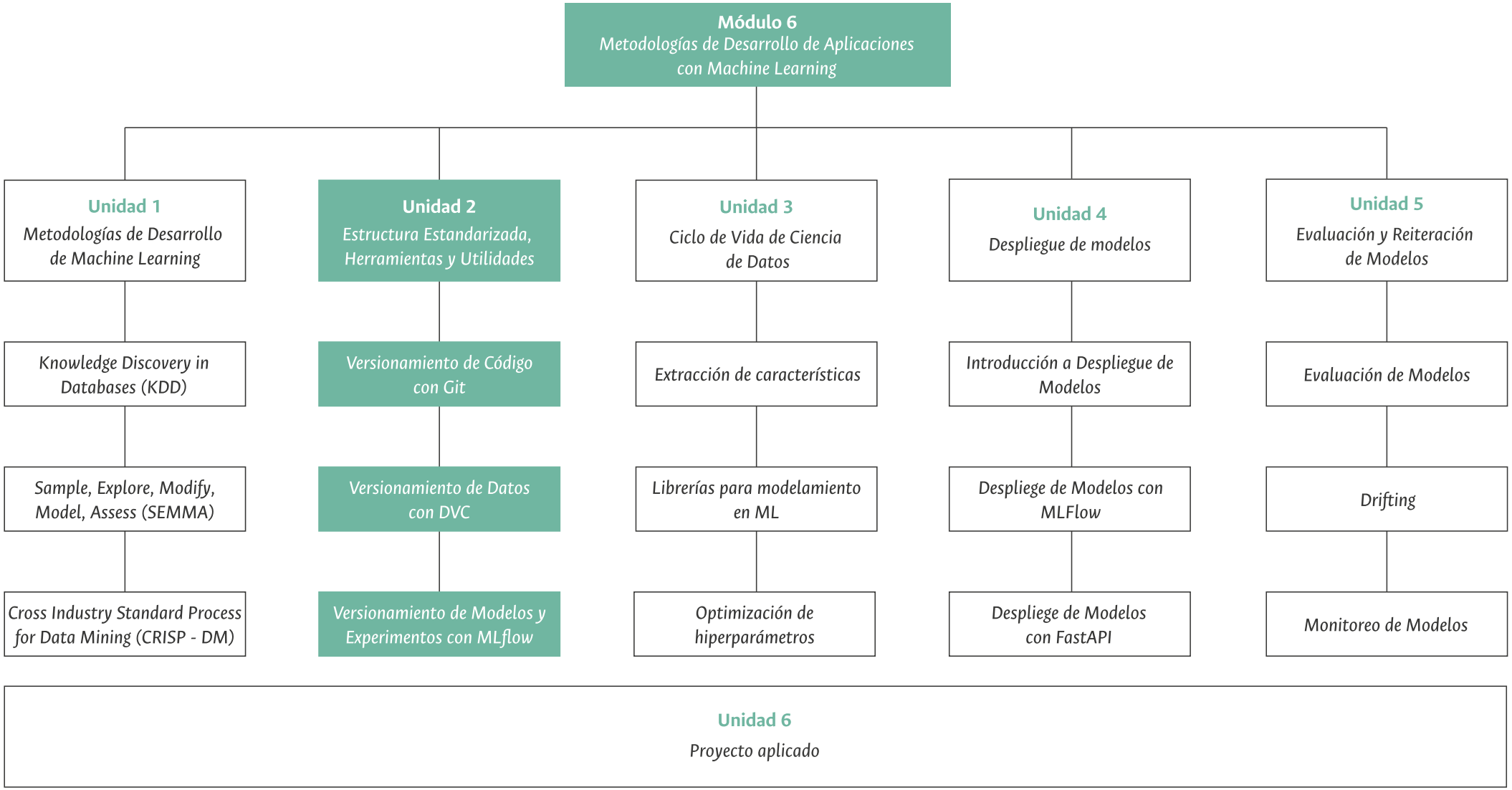
Al finalizar la unidad usted deberá ser capaz de:

Entender cómo estructurar un proyecto de ciencia de datos.

Seleccionar una herramienta de versionado apropiada para distintos componentes del proyecto.

Integrar herramientas de versionado modernas con herramientas clásicas de *Machine Learning*.

Mapa de contenidos de la unidad



Herramientas y utilidades para Machine Learning

En esta unidad hablaremos solamente de versionamiento de Código , Datos y Modelos

Versionamiento de Código

Posibilita compartir el código fuente del desarrollo y a la vez mantener un registro de los cambios realizados.

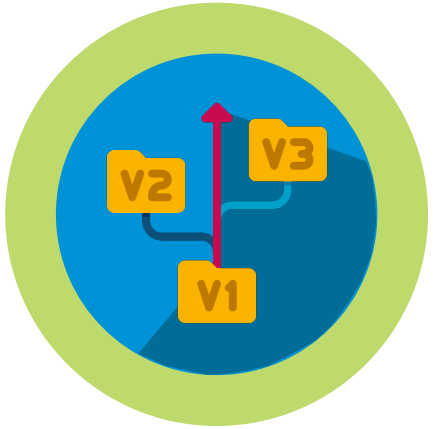
Versionamiento de Datos

Permite capturar las versiones de los datos y modelos en commits de Git, mientras se almacenan en la nube.

Versionamiento de Modelos

Ayuda a seguir y controlar los cambios los modelos, lo que permite recuperar fácilmente una versión anterior cuando sea necesario.

Versionamiento en *Machine Learning*

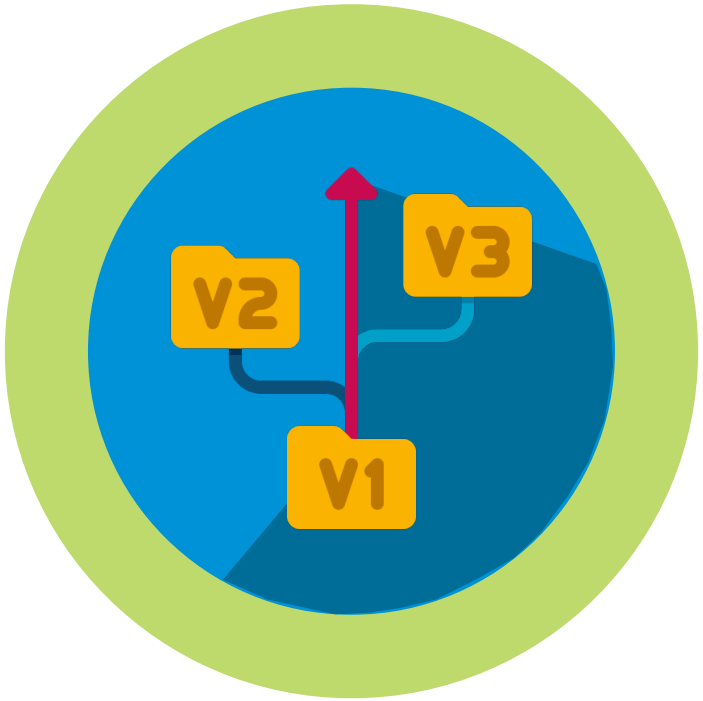


Versionamiento de Código

Versionamiento de Datos

Versionamiento de Modelos y Experimentos

Versionamiento en *Machine Learning*



Versionamiento de Código

Herramientas y utilidades para ML

Versionamiento de código

Existen distintos sistemas de gestión de paquetes. Estos se encargan de manejar las distintas dependencias que existen dentro de un proyecto especifico.

Los sistemas de gestión de paquetes permiten:

1. Diferencias entre versiones locales y oficiales
2. Instalación, actualización y eliminación simple de paquetes
3. Resolución de dependencias para garantizar que el software funcione correctamente
4. Búsqueda de actualización de los paquetes


```
- title: "Programa de Formación en
  Machine Learning and Data Science"
+ title: "Programa de Formación en
  Machine Learning and Data Science -
  Metodologías Ágiles para el Desarrollo
  de Aplicaciones con Machine Learning"
```

Versionamiento en ML - Código

¿Qué es Git?

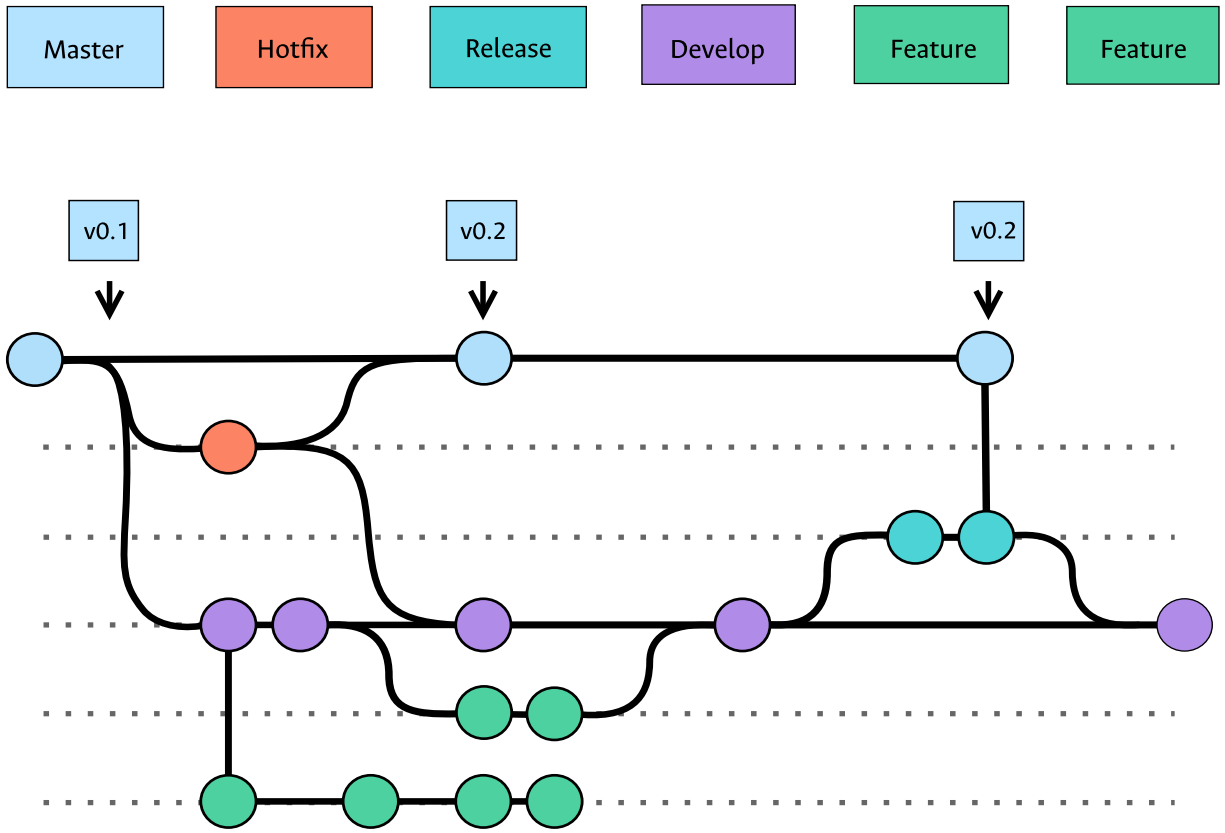
`git` es una herramienta donde las versiones representan una especie de grafo dirigido acíclico (DAG), posee nodos (commits) y versiones paralelas como ramas.

- Puede ser usado con distintos repositorios cloud como *github*, *gitlab*, *bitbucket*, *azure repositories*, entre otros.
- El sistema de `git` lleva un registro de los archivos nuevos y cambios realizados sobre los archivos previamente almacenados.

git status

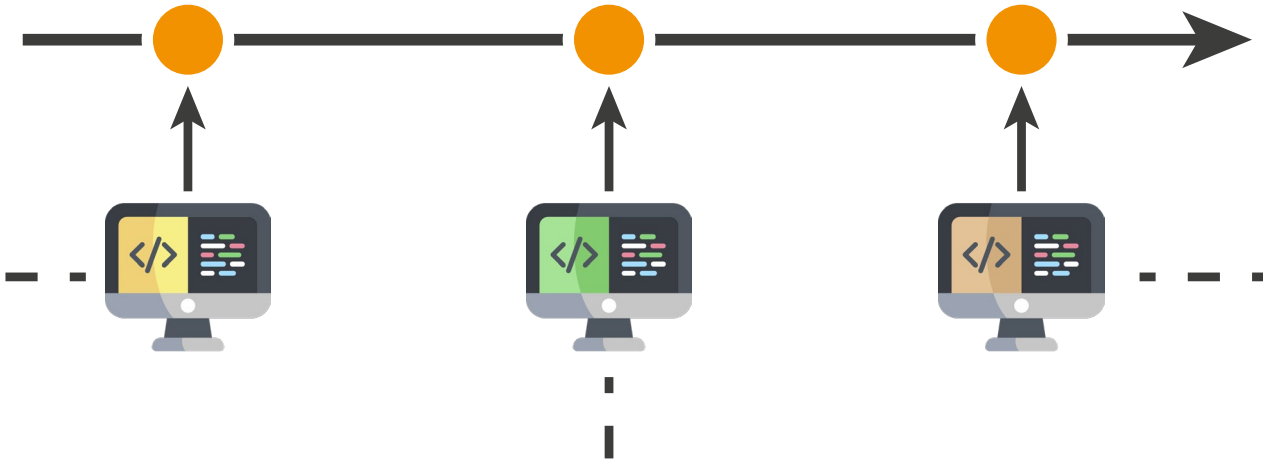
git checkout

git commit



Versionamiento en ML - Código

Commit



```
git commit
```

Abre el editor de código de texto para que se escriba el mensaje más detallado.

```
git commit -m "mensaje"
```

Permite escribir el mensaje en el mismo comando entre comillas.

```
git commit -am "mensaje"
```

Agrega todos los archivos modificados y te deja escribibr el mensaje en un sólo paso.

Versionamiento en ML - Código

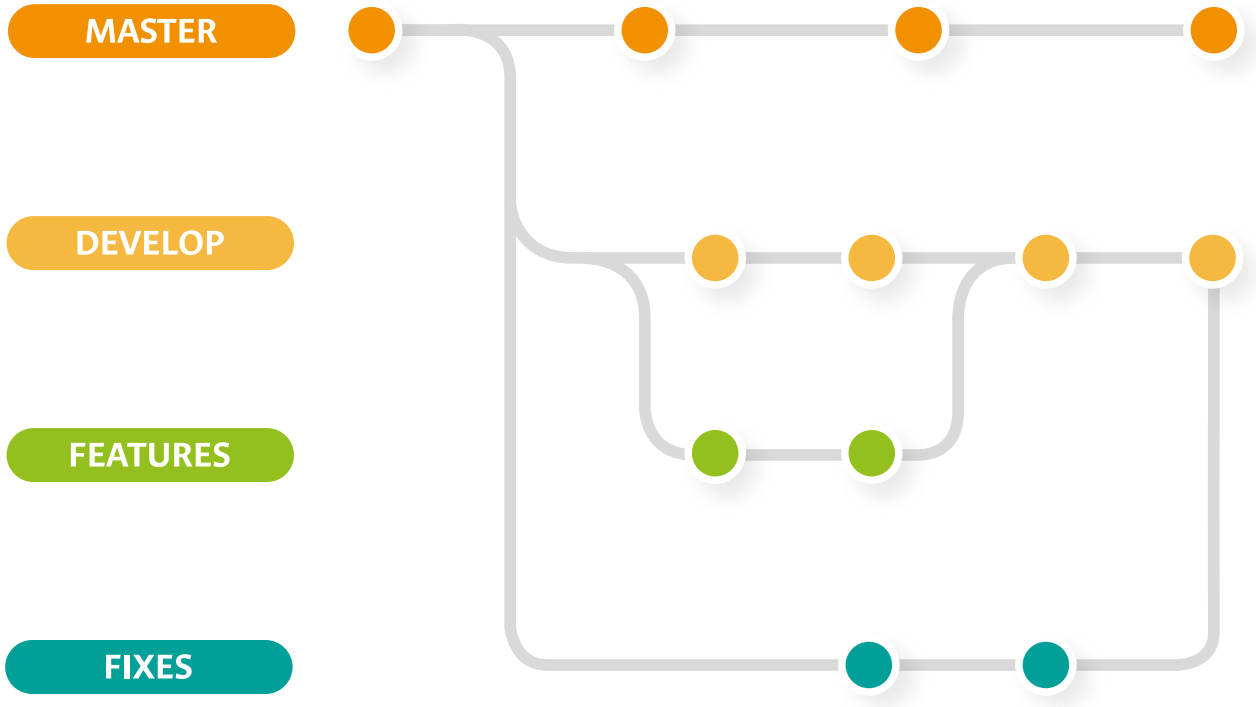
Branches – Ramas

Crear una rama

```
git branch "nombre_rama"
```

Cambiar entre ramas

```
git switch "nombre_rama"
```



Versionamiento en ML - Código

Tags – Etiquetas

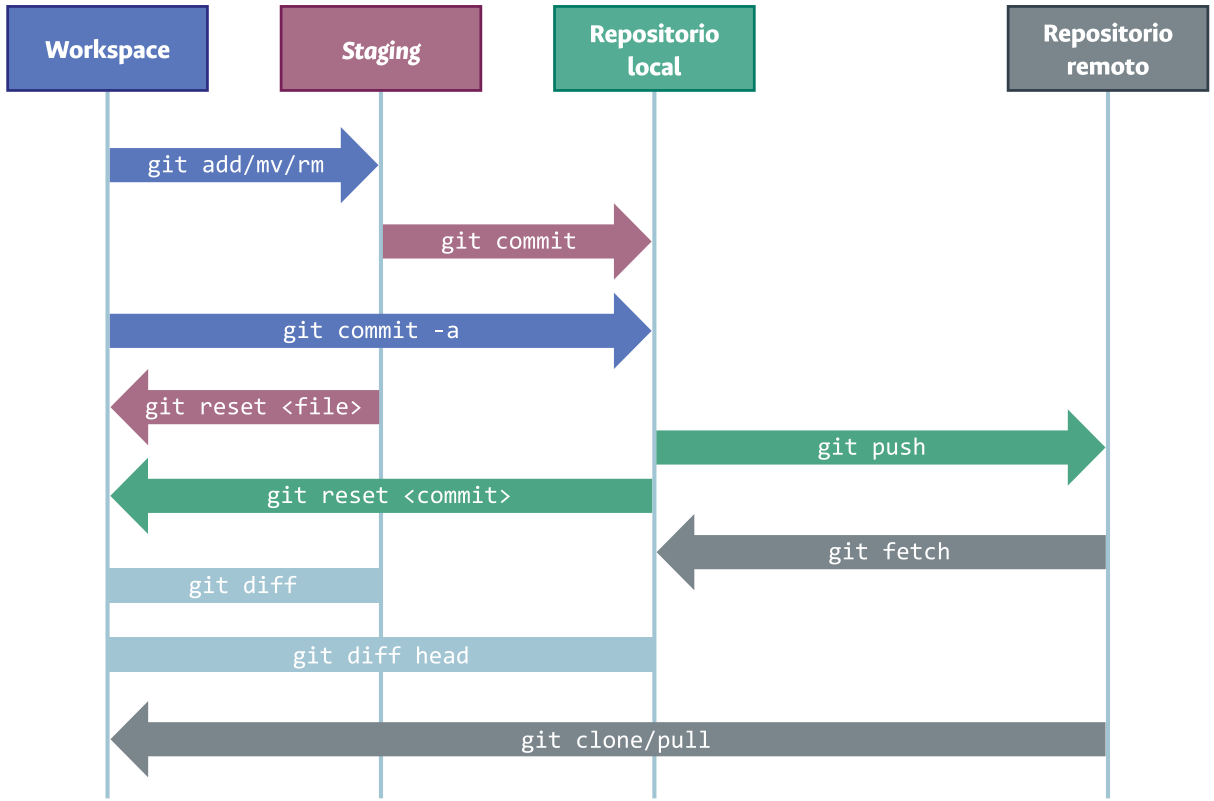
Un tag es una referencia a un *commit* específico en el historial, lo que significa que puede ser utilizado para marcar versiones específicas de un proyecto.

```
git tag "nombre_del_tag" <commit_id>
```

Platforma	angular, node
Problemas	bug, seguridad, desempeño
Mindless	tarea, legal
Experiencia	copia, diseño, ux
Entorno	desarrollo, testing, producción
Feedback	discusión, pregunta
Mejoras	perfeccionamiento, optimización
Adición	funcionalidad
Pendiente	en progreso, bajo supervisión
Inactivo	inválido, duplicado

Versionamiento en ML - Código

GitHub



GitHub es una plataforma de alojamiento, propiedad de Microsoft, que ofrece a los desarrolladores la posibilidad de crear repositorios de código y guardarlos en la nube de forma segura al igual que llevar un registro y control de cualquier cambio sobre este código, usando Git.

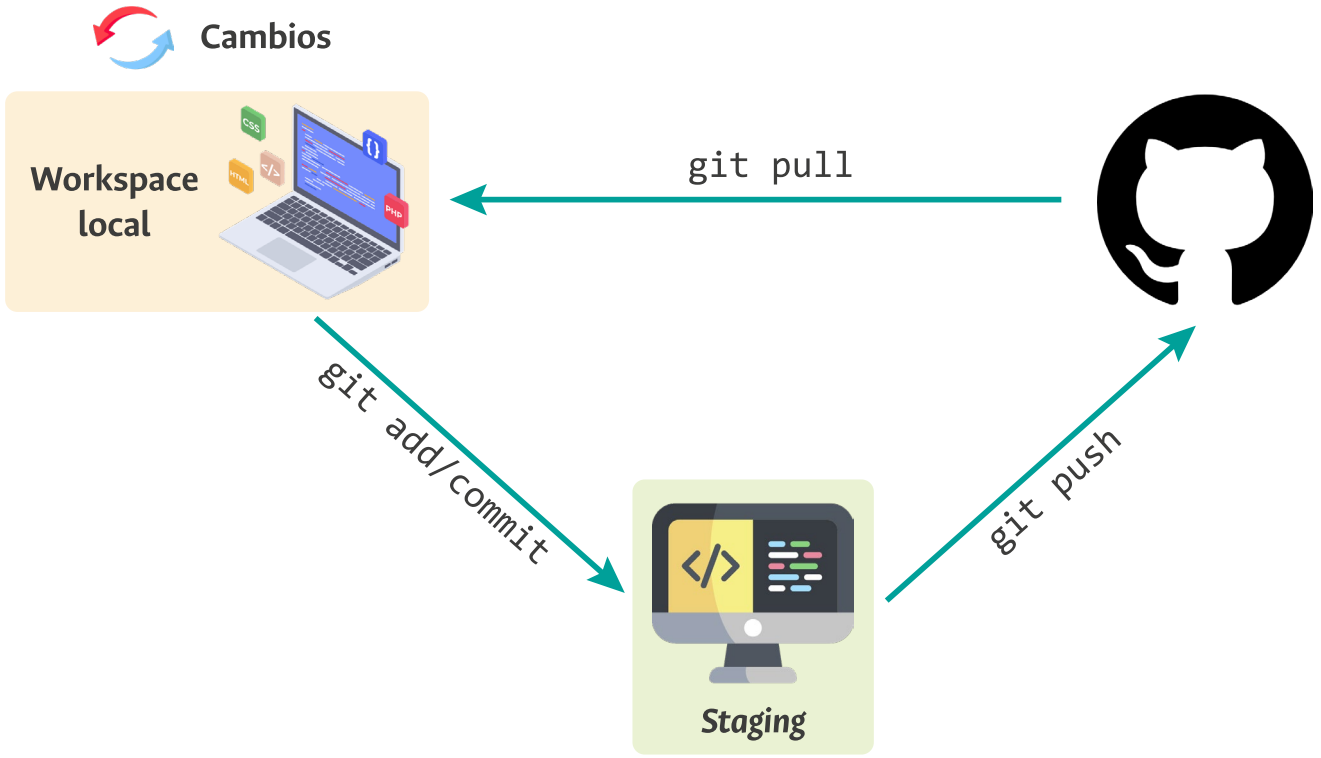
Git es un sistema que permite establecer un control de versiones, mientras que GitHub es una plataforma que ofrece un grupo de funciones que facilitan el uso de Git y la colaboración en tiempo real, así como el almacenamiento en la nube.

Versionamiento en ML - Código

Push y Pull

1. ***git pull***: actualiza el repositorio local con los cambios más recientes del repositorio remoto y fusionarlos en la rama local actual.

2. ***git push***: se utiliza para enviar los cambios locales a un repositorio remoto actualizando las referencias de este último para que coincidan con los cambios locales.

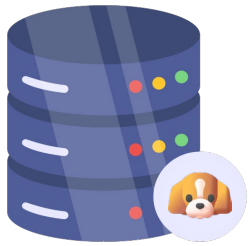


Versionamiento en *Machine Learning*

Versionamiento de Datos

Herramientas y utilidades para ML

Datos



Raza	Color	Tamaño
Caniche	Blanco	Mediano
Pekinés	Marrón	Pequeño
Dálmata	Blanco	Grande
Axolote	Rosa	Pequeño

	Raza	Color	Tamaño
	Caniche	Blanco	Mediano
	Pekinés	Marrón	Pequeño
	Dálmata	Blanco	Grande
−	Axolote	Rosa	Pequeño
+	Doberman	Negro	Grande
+	Beagle	Mixto	Mediano

Versión final

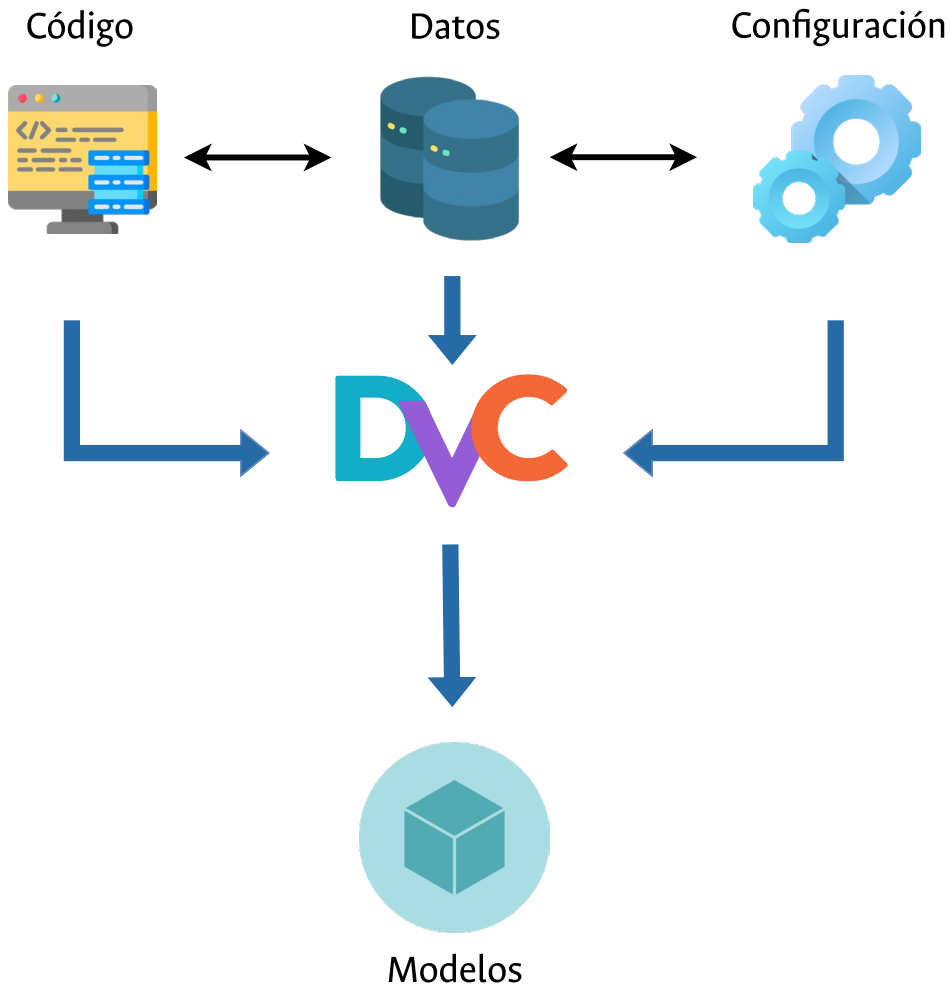
Raza	Color	Tamaño
Caniche	Blanco	Mediano
Pekinés	Marrón	Pequeño
Dálmata	Blanco	Grande
Doberman	Negro	Grande
Beagle	Mixto	Mediano

Versionamiento en ML - Datos

¿Qué es DVC?

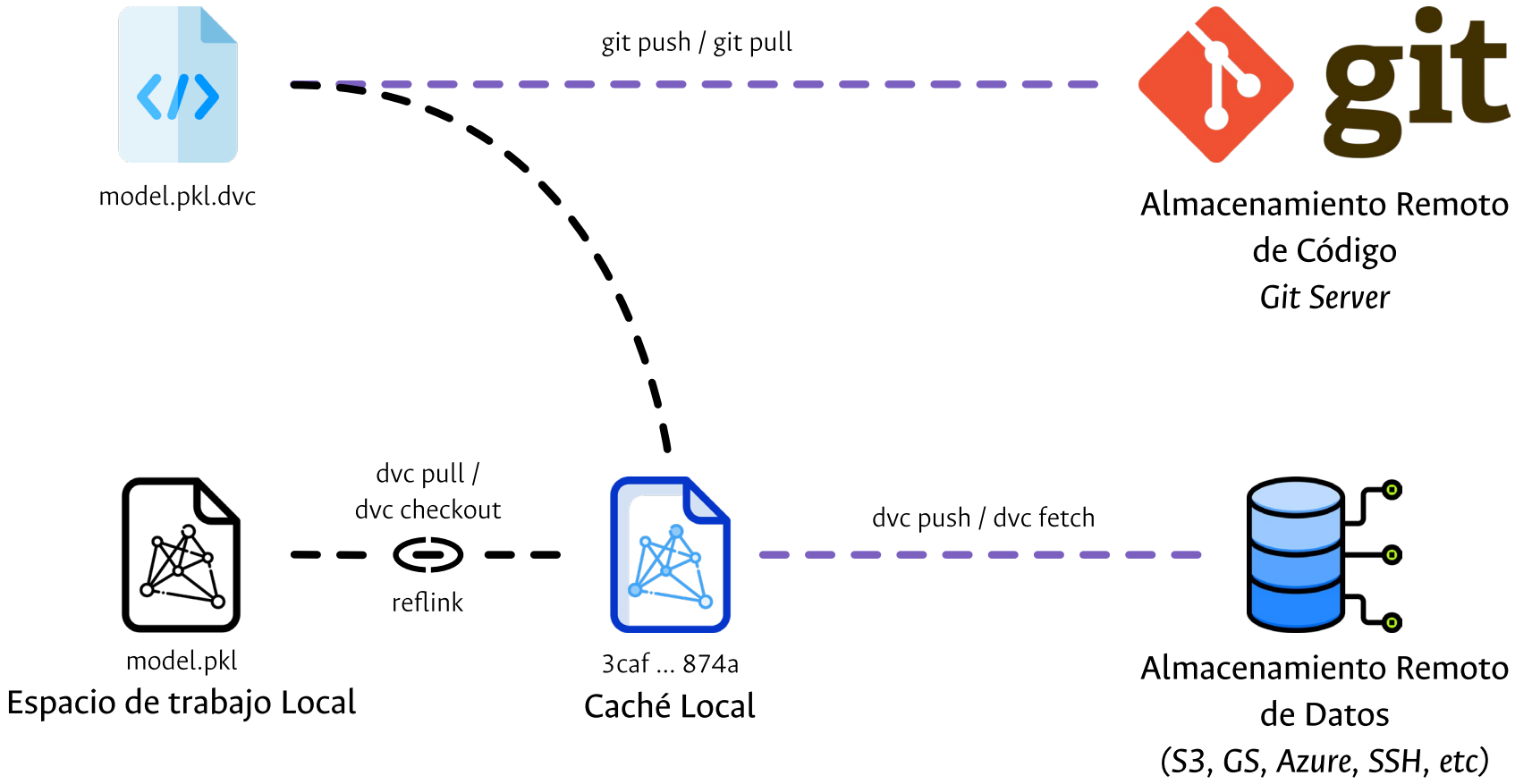
Data Version Control (DVC) es una herramienta que permite gestionar grandes conjuntos de datos, modelos y métricas y la cual se integra para complementar a git

DVC permite almacenar la información en algún sistema de almacenamiento como S3, azure storage, cloud storage, google drive, HDFS, entre otros.



Versionamiento en ML - Datos

¿Qué es DVC?

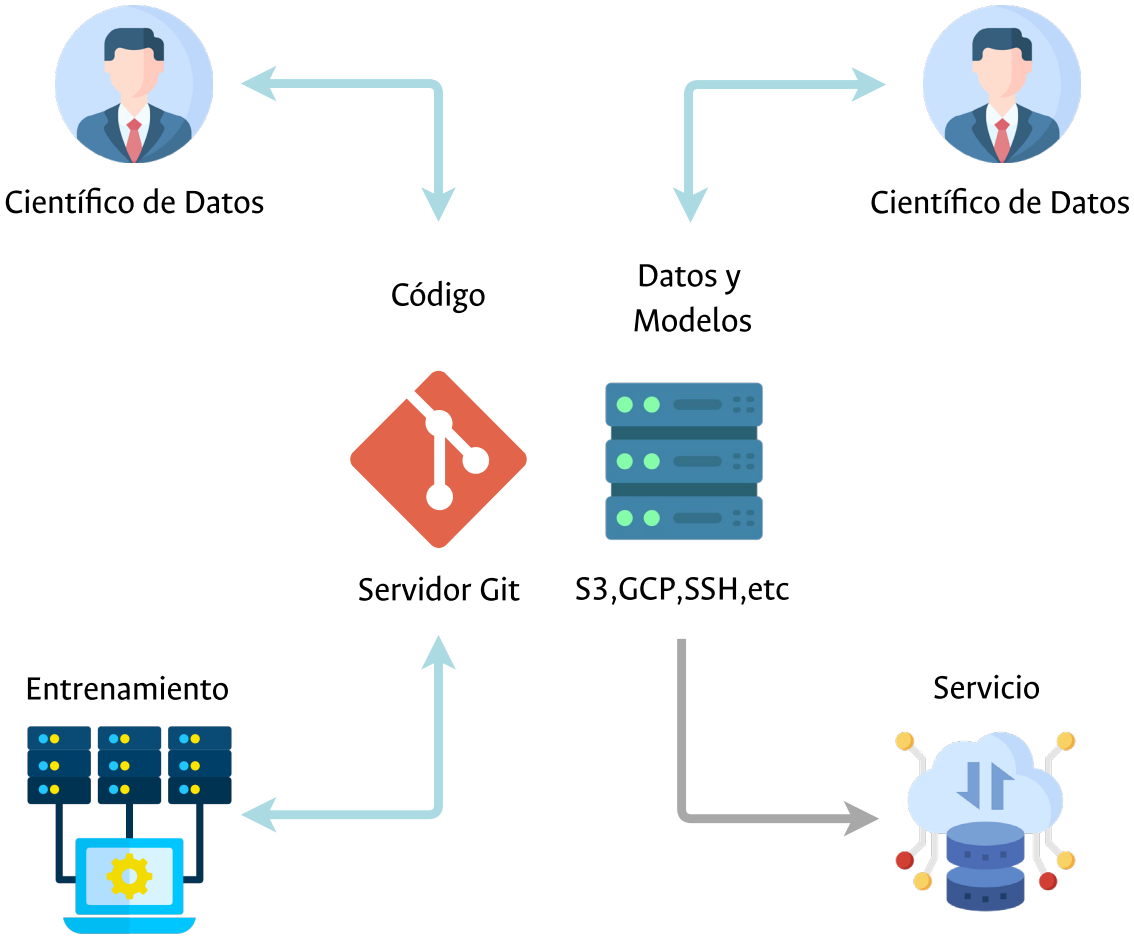


Propiedades DVC

Reproducibilidad	Los resultados verificables son cruciales para la investigación. Si los resultados se han obtenido utilizando ML en cualquier parte del proceso, esta parte también debe ser reproducible para investigadores independientes.
Integración del flujo de trabajo	Dado que la herramienta puede ser utilizada tanto por profesionales de la informática como por científicos, debe ser lo menos invasiva posible.
***Backend* intercambiable**	Generalmente se tendrán centros propios de datos e infraestructura en la nube, por lo que las herramientas deben ser capaces de trabajar con soluciones de almacenamiento personalizadas.
Agnóstico	No debería importar si usamos *pytorch*, *tensorflow* o una librería personalizada. Sería preferible si también es agnóstico al lenguaje.
Código abierto	Creemos que es lo correcto y lo que hay que apoyar. Además, por lo general es mucho más fácil de adaptar a las necesidades personalizadas.
Otros	Preferimos soluciones que podamos ajustar a nuestras necesidades. Además, el presupuesto es importante.

Versionamiento en ML - Datos

Integración con GIT



Se puede pensar a DVC como un control de versiones ampliado para guardar datos y poder hacer seguimiento a ellos.

Este proceso no es sencillo hacerlo en GIT debido al tamaño normal de los datasets

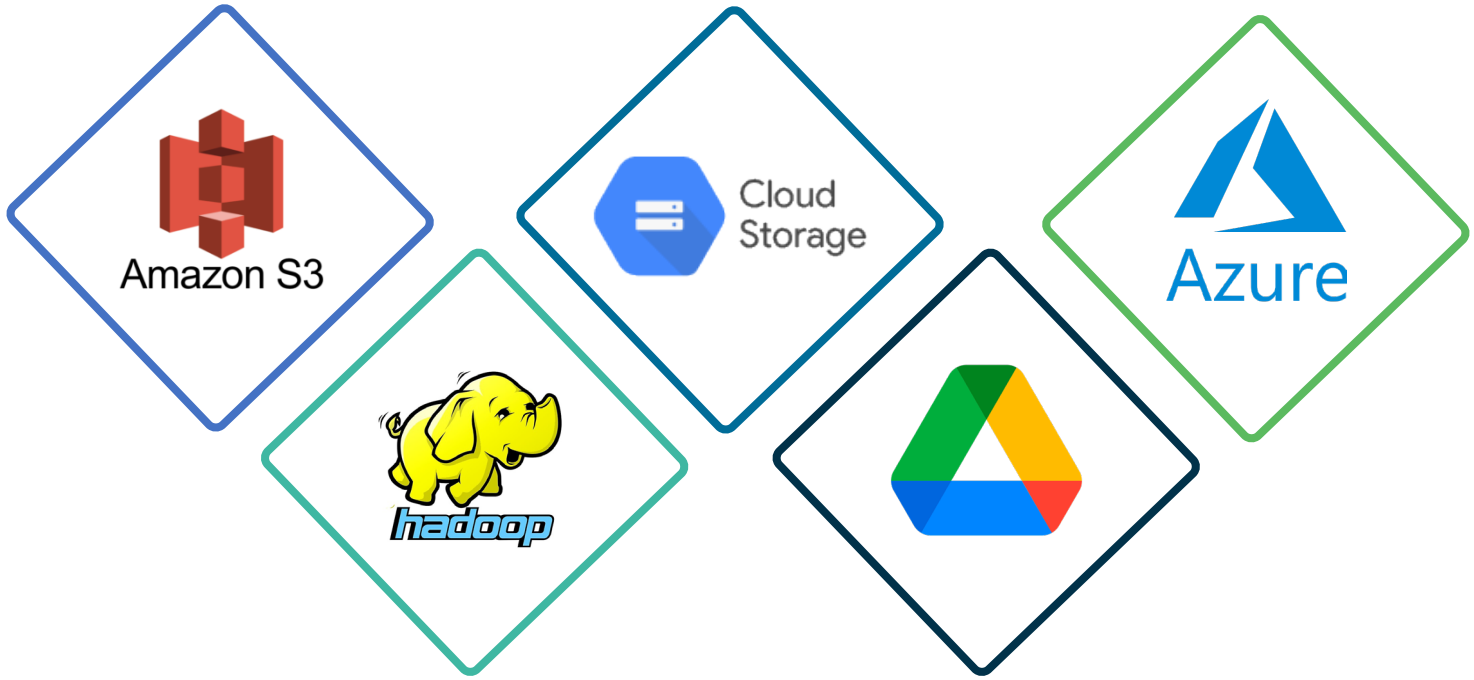
GIT no trabaja bien con archivos muy grandes y en general no trabaja bien con archivos binarios que son comunes en las tareas de aprendizaje automático.

En este caso nos encontramos con una herramienta que nos sirve para aprendizaje de maquina y para la construcción de pipelines

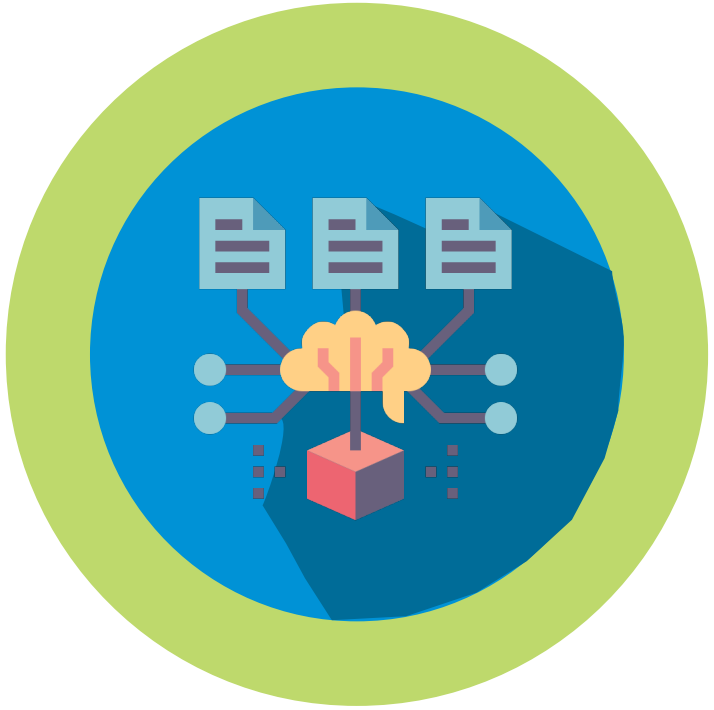
Versionamiento en ML - Datos

Remotos

DVC admite varios sistemas de almacenamiento para almacenar los datos. También es posible utilizar sistemas de almacenamiento locales o en red para guardarlos.



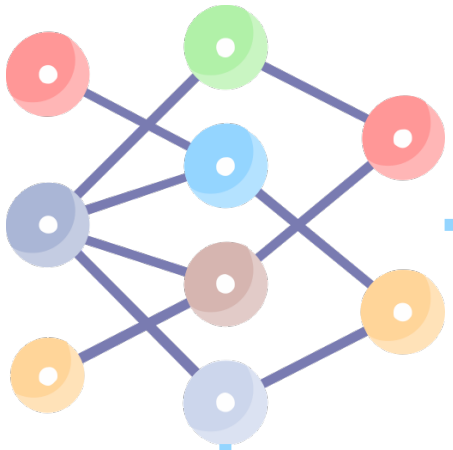
Versionamiento en *Machine Learning*



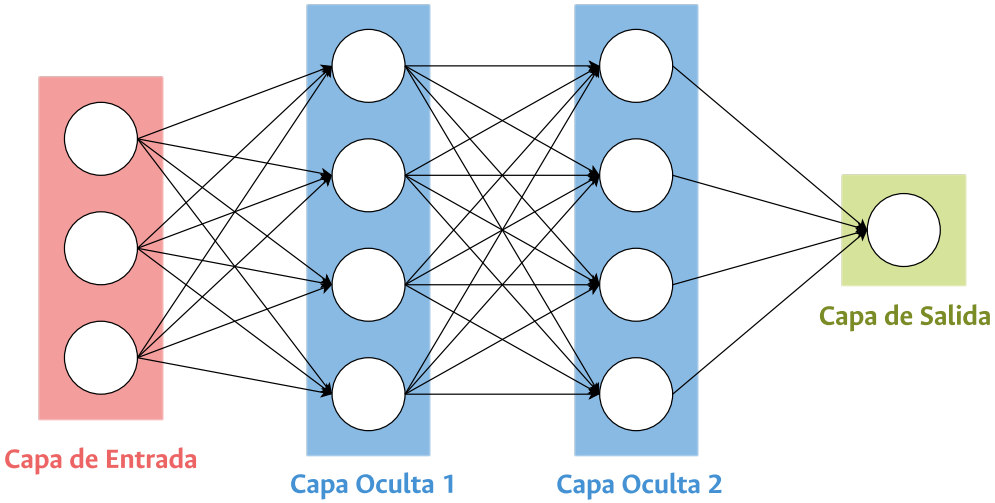
Versionamiento de Modelos y Experimentos

Herramientas y utilidades para ML

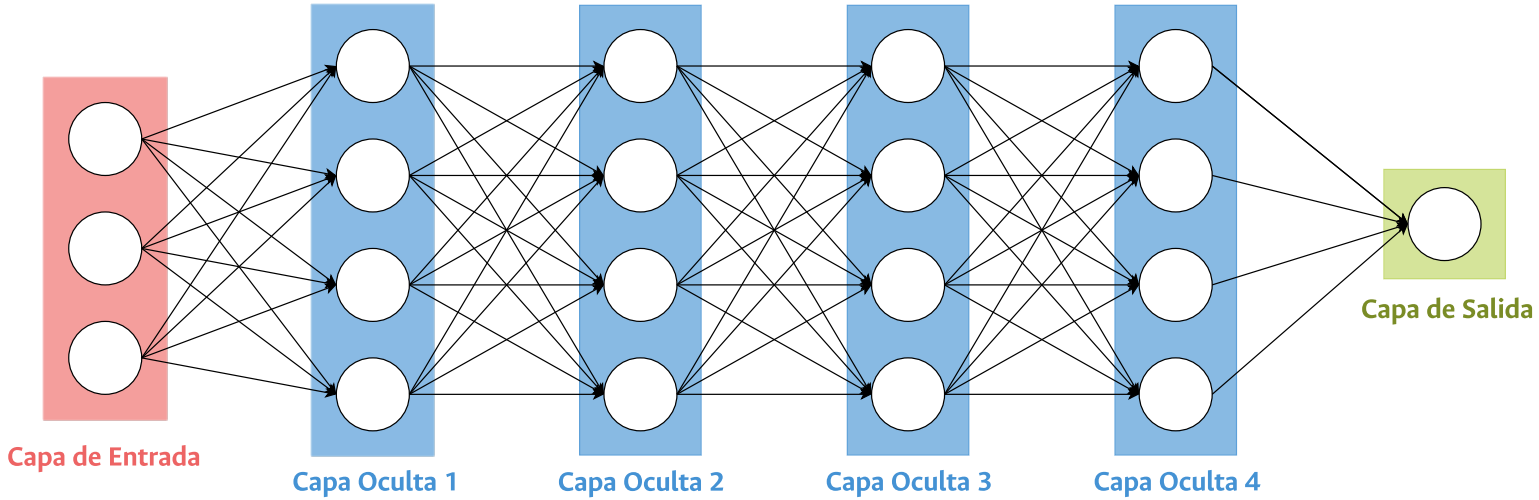
Modelos



Modelo 1



Modelo 2



Versionamiento en ML – Modelos y Exp.

¿Qué es MLFlow?

MLflow es una plataforma de código abierto para la gestión de flujos de trabajo de aprendizaje automático. La utilizan los equipos de MLOps y los científicos de datos.

Proporciona un conjunto de herramientas para facilitar el desarrollo, el seguimiento y la implementación de proyectos de aprendizaje automático.

Versionamiento en ML – Modelos y Exp.

¿Qué es MLFlow? – diagrama componentes

mlflow™

Seguimiento

Registro y consulta de experimentos: código, datos, configuración, resultados

Posibilita el seguimiento de ejecuciones y experimentos en proyectos de Machine Learning.

Proyectos

Formato de empaquetado para *reproducible runs* en cualquier plataforma

Permite definir proyectos de ciencia de datos de forma más estructurada.

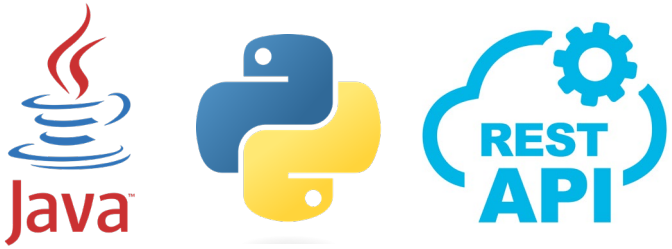
Modelos

Formato general para enviar modelos a diversas herramientas de implementación

Proporciona una forma estructurada y unificada de manejar modelos de Machine Learning sin importar la librería que estemos usando.

Versionamiento en ML – Modelos y Exp.

MLflow Tracking

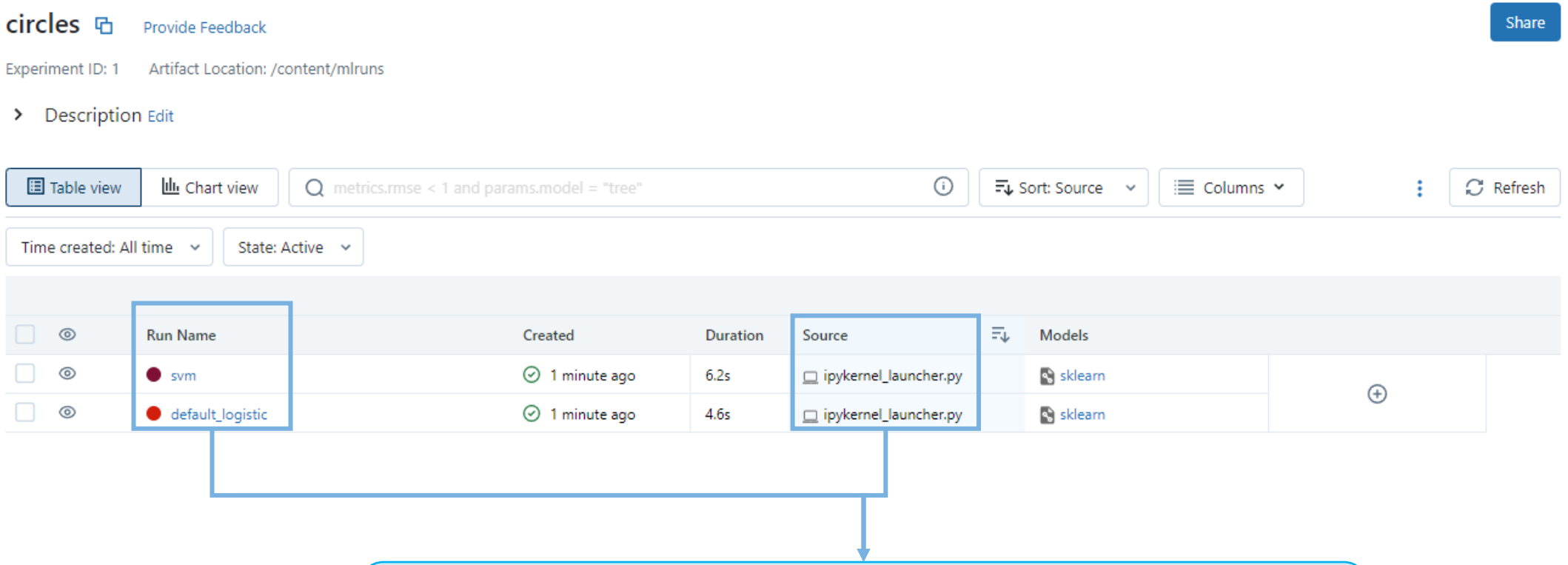


El componente de seguimiento (The Tracking Component) permite registrar sesiones de entrenamiento de modelos de máquina (llamadas ejecuciones) y ejecutar consultas utilizando Java, Python, R y API REST.

Puede utilizar este componente para registrar varios aspectos de sus ejecuciones.

Versionamiento en ML – Modelos y Exp. Tracking

Los principales componentes que puede registrar para cada una de sus ejecuciones son:



Fuente (Source)

Puede ser el nombre del archivo que lanza la ejecución. Alternativamente, si está utilizando un proyecto MLflow, puede ser el nombre del proyecto y el punto de entrada de la ejecución

Versionamiento en ML – Modelos y Exp.

Tracking

Versión del código (Code Version)

svm

Run ID: d3e3267a662c43999294b9cfd6790b6d

Duration: 6.2s

Date: 2023-03-23 23:50:29

Status: FINISHED

Source: ipykernel_launcher.py

User: root

Lifecycle Stage: active

Description Edit

Parameters (3)

Name	Value
C	1.0
gamma	0.1
kernel	rbf

Metrics (4)

Name	Value
accuracy	0.99
f1	0.99
precision	0.98
recall	1

Fecha de Inicio y Estado final(Start and End time)

Permite registrar métricas de valor clave que contienen valores numéricos. Es posible actualizar cada métrica a lo largo de la duración de una ejecución.

Parámetros (Parameters)

Puede ser cualquier parámetro de entrada clave-valor que elija, siempre y cuando los valores y las claves sean ambos cadenas.

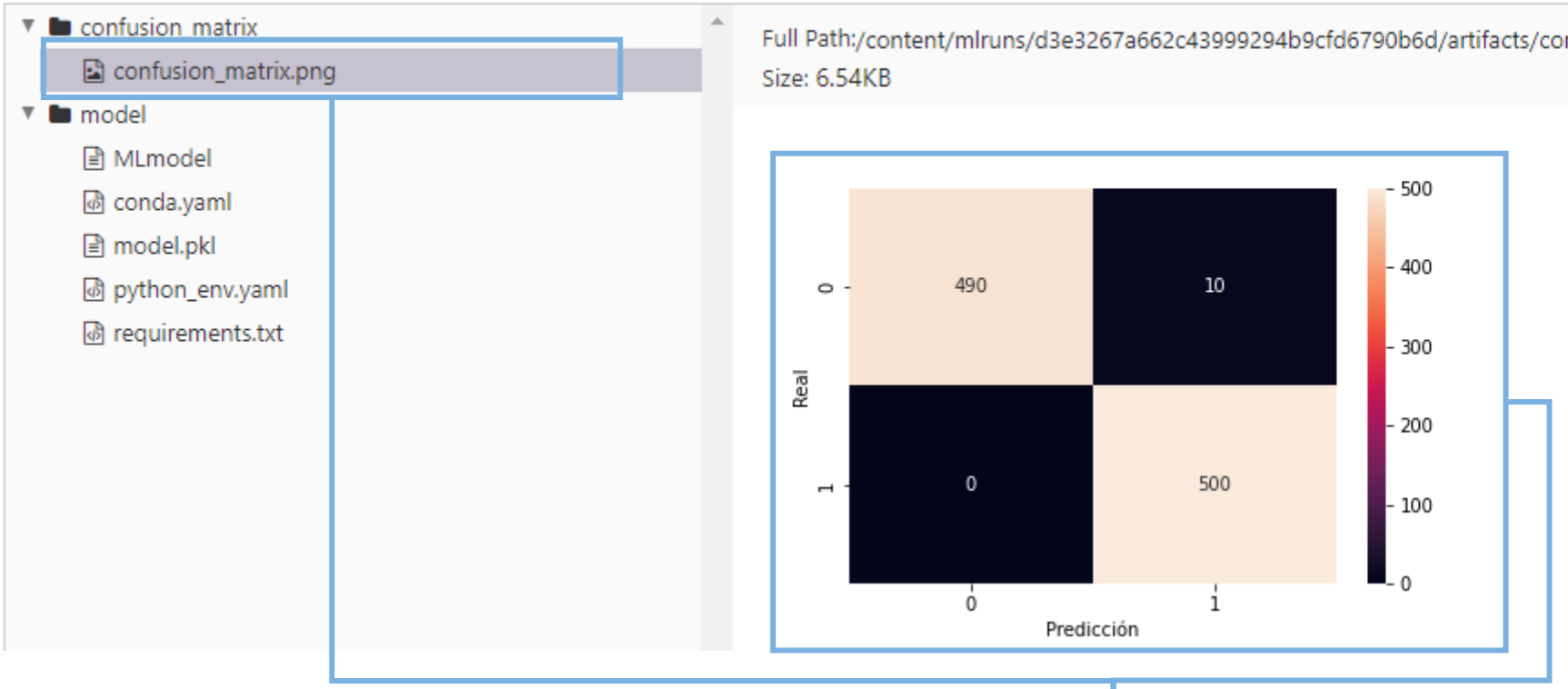
Métricas (Metrics):

Permite registrar métricas de valor clave que contienen valores numéricos. Es posible actualizar cada métrica a lo largo de la duración de una ejecución.

Versionamiento en ML – Modelos y Exp.

Tracking

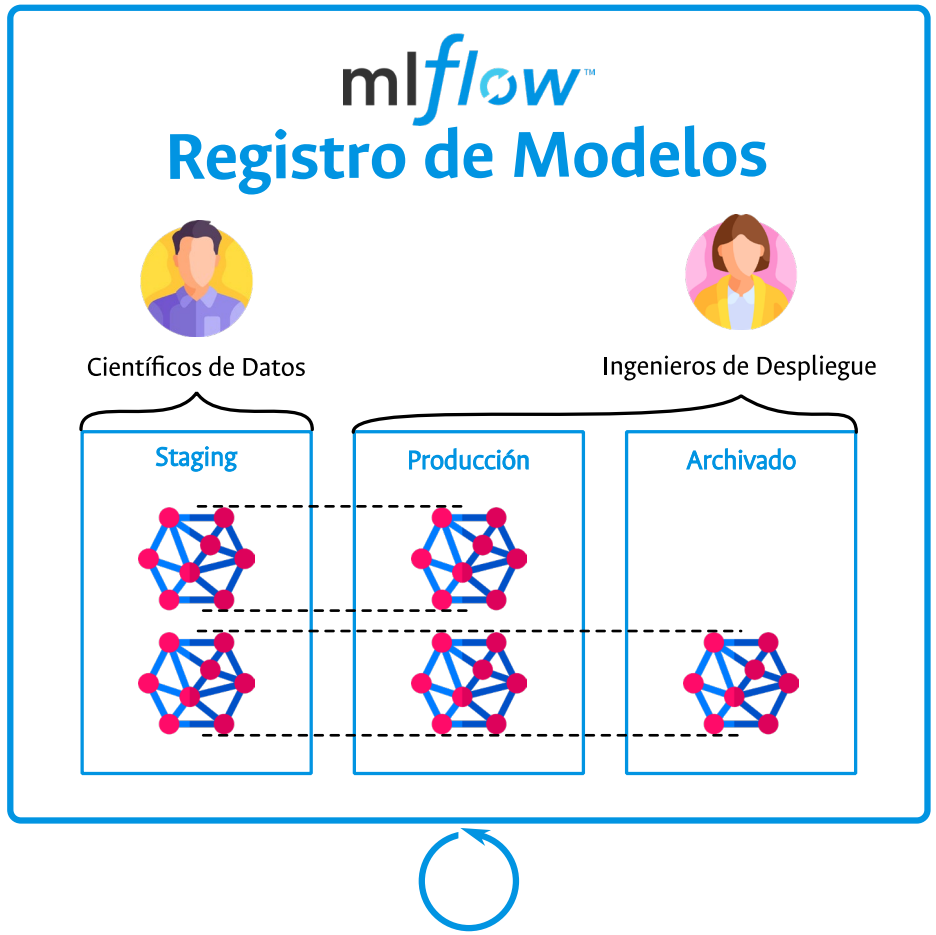
Artifacts



Artefactos (Artifacts)

Son archivos de salida (en todos los formatos). Los artefactos le permiten registrar imágenes, modelos (como modelos de scikit-learn decapados), y archivos de datos como archivos Parquet.

MLflow Proyectos

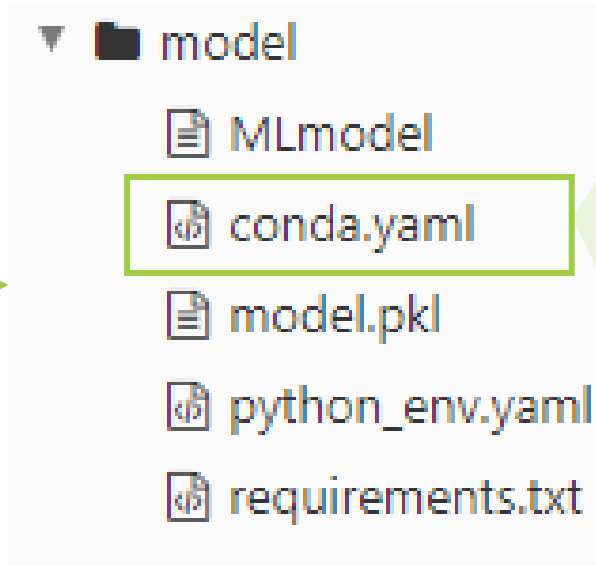


Los proyectos de MLflow le permiten empaquetar el código de ciencia de datos de una manera reproducible y reutilizable, principalmente de acuerdo con las convenciones.

Un proyecto de MLFlow se define a partir del archivo *MLproject*, se trata de un archivo en formato *yaml* que define los posibles parámetros que podemos probar, sus valores de defecto, el nombre del proyecto y el comando que se debe usar para correr el script

Proyectos en MLflow – Estructuración en carpetas

Para cada proyecto, puede especificar las siguientes propiedades:




```yaml
channels:
- conda-forge
dependencies:
- python=3.9.16
- pip<=22.0.4
- pip:
  - mlflow<3,>=2.2
  - cloudpickle==2.2.1
  - pathlib==1.0.1
  - psutil==5.9.4
  - scikit-learn==1.2.2
  - typing-extensions==4.5.0
name: mlflow-env
```

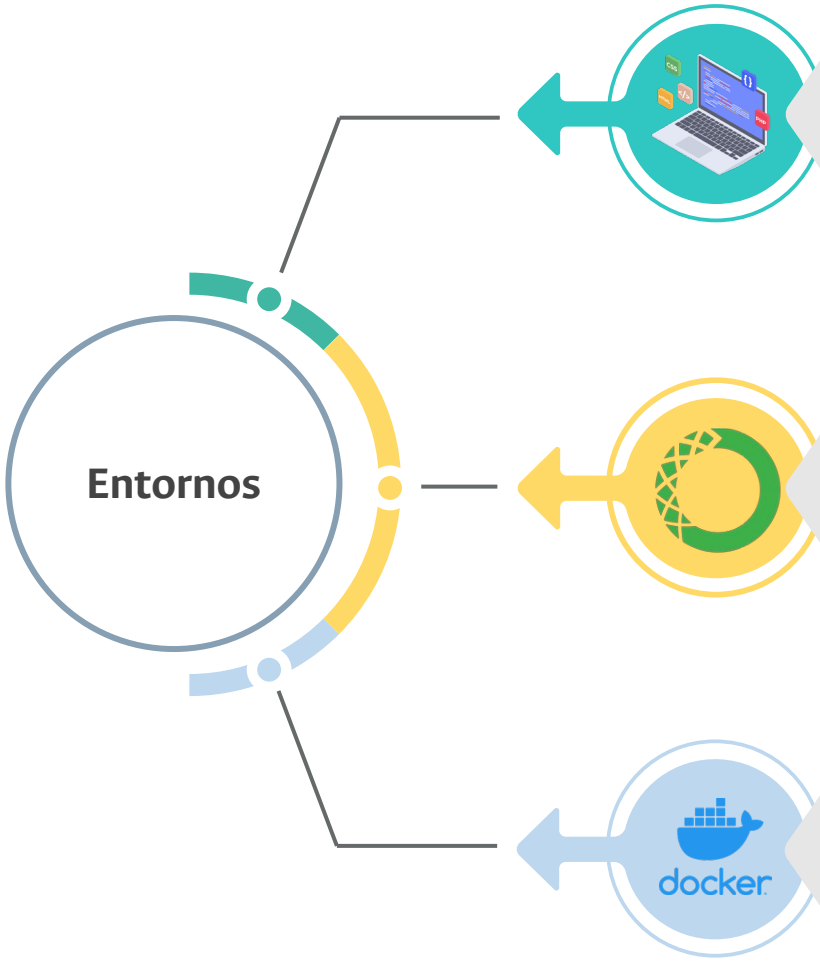
Entorno

Permite definir el entorno de software utilizado para ejecutar los puntos de entrada del proyecto, incluidas todas las dependencias de bibliotecas requeridas por el código del proyecto.

Puntos de entrada

Permite especificar los comandos que desea ejecutar dentro del proyecto, así como información sobre los parámetros. Un proyecto suele contener al menos un punto de entrada que los usuarios pueden llamar.

Proyectos en MLflow - Entornos



Permite ejecutar los proyectos directamente en su entorno de sistema existente. Para ello, se instalan todas las dependencias del proyecto en el sistema antes de ejecutar el proyecto. El entorno del sistema no forma parte del directorio y se suministra en tiempo de ejecución.

Ofrece soporte para bibliotecas nativas como Intel MKL o CuDNN, así como para paquetes Python. Si especifica un entorno Conda para su proyecto, se activará antes de que el código del proyecto comience a ejecutarse.

Los proyectos de MLflow que utilizan imágenes Docker obtienen una capa Docker añadida que copia el contenido del proyecto en un directorio llamado */mlflow/projects/code*. Una vez que esto sucede, se produce una nueva imagen del contenedor. A continuación, MLflow ejecuta la imagen e invoca el punto de entrada del proyecto en el contenedor resultante.

Modelos en MLflow - Carga

Los modelos de MLflow le permiten empaquetar modelos de aprendizaje automático en un formato compatible con muchas herramientas posteriores. Puede añadir metadatos a sus modelos MLflow

```
model = mlflow.pyfunc.load_model("nombre_del_modelo")
```

scikit learn

Transformers

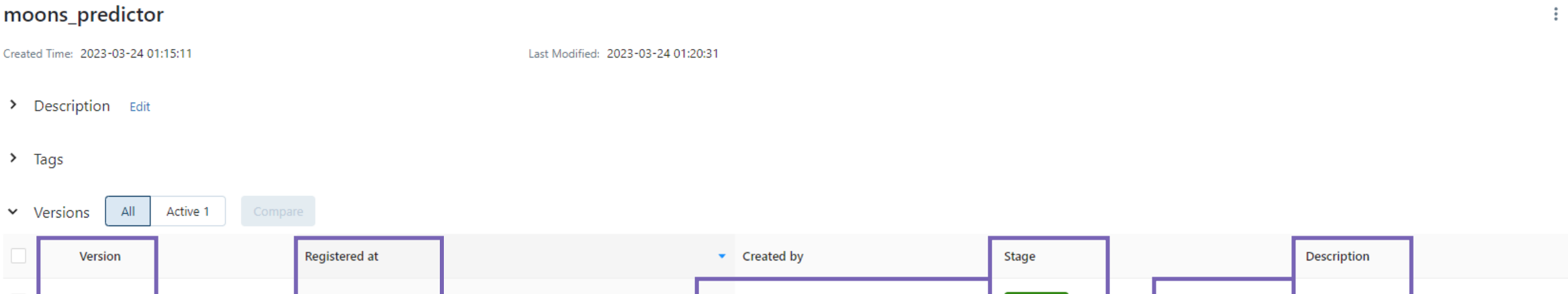
TensorFlow

Keras

Versionamiento en ML – Modelos y Exp.

Modelos en MLflow - Registro

Proporciona una API y una interfaz de usuario para gestionar de forma centralizada sus modelos y su ciclo de vida. El registro proporciona el linaje del modelo, la versión del modelo, las anotaciones y las transiciones de etapa.



Versión

Modelo registrado

Etapa del modelo

- *Staging*
- Producción
- Archivado

Anotaciones y descripciones

Puede añadir descripciones y cualquier información relevante, como descripciones de algoritmos, metodología y conjunto de datos empleado.

Despedida

¡Gracias por su atención!

Jorge Eliécer Camargo Mendoza, PhD.

https://dis.unal.edu.co/~jecamargom/

jecamargom@unal.edu.co

Departamento de Ingeniería de Sistemas e Industrial
Facultad de Ingeniería
Universidad Nacional de Colombia
Sede Bogotá

UNIVERSIDAD NACIONAL DE COLOMBIA

Referencias

- Git. (s. f.). https://git-scm.com/
- Get Started: Data and Model Access. (s. f.). Data Version Control · DVC. https://dvc.org/doc/start/data-management/data-and-model-access
- Get Started: Data Versioning. (s. f.). Data Version Control · DVC. https://dvc.org/doc/start/data-management/data-versioning
- MLflow Tracking – MLflow 2.2.2 documentation. (s. f.-b). https://mlflow.org/docs/latest/tracking.html
- MLflow Models – MLflow 2.2.2 documentation. (s. f.-b). https://mlflow.org/docs/latest/models.html
- MLflow Projects – MLflow 2.2.2 documentation. (s. f.). https://mlflow.org/docs/latest/projects.html
- MLflow Documentation – MLflow 2.2.2 documentation. (s. f.). https://mlflow.org/docs/latest/index.html

Derechos de imágenes

- Flaticon. (s.f.). Version control icon. [Icono]. https://www.flaticon.com/free-icon/version-control_4661413
- Flaticon. (s.f.). Predictive models icon. [Icono]. https://www.flaticon.com/free-icon/predictive-models_2103652
- Flaticon. (s.f.). Data collection icon. [Icono]. https://www.flaticon.com/free-icon/data-collection_2103552
- Flaticon. (s.f.). Git icon. [Icono]. https://www.flaticon.com/free-icon/git_6022862
- Flaticon. (s.f.). Branch icon. [Icono]. https://www.flaticon.com/free-icon/branch_6577257
- Permitted GitHub logos. Invertocat [PNG]. https://github.com/logos
- Flaticon. (s.f.). Data server icon. [Icono]. https://www.flaticon.com/free-icon/data-server_10150842
- Flaticon. (s.f.). Beagle icon. [Icono]. https://www.flaticon.com/free-icon/beagle_10118937
- Flaticon. (s.f.). Data icon. [Icono]. https://www.flaticon.com/free-icon/data_1197511
- Flaticon. (s.f.). Database icon. [Icono]. https://www.flaticon.com/free-icon/database_1664316
- Flaticon. (s.f.). Databases icon. [Icono]. https://www.flaticon.com/free-icon/databases_5134601
- Flaticon. (s.f.). Settings icon. [Icono]. https://www.flaticon.com/free-icon/settings_3039498
- (s.a.). (s.f.). DVC Logo. [PNG]. https://iconduck.com/icons/101909/file-type-dvc
- (s.a.). (s.f.). Amazon S3 Logo. [PNG]. https://trackvia.com/blog/integrations/amazon-s3/
- (s.a.). (s.f.). Hadoop Logo. [PNG]. https://aitor-medrano.github.io/iabd2223/hadoop/images/03hadoop-logo.jpg
- (s.a.). (s.f.). Google Cloud Storage Logo. [PNG]. https://www.cdata.com/ui/img/logo-googlecloudstorage.png
- Microsoft Corporation. (s.f.). Azure Logo. [PNG]. https://es.m.wikipedia.org/wiki/Archivo:Microsoft_Azure_Logo.svg
- Google. (2015). Google Drive Logo. [PNG]. https://es.wikipedia.org/wiki/Archivo:Google_Drive_logo.png

Derechos de imágenes

- Scikit learn. (8 de agosto de 2018). Scikit learn logo [Vector]. https://es.wikipedia.org/wiki/Archivo:Scikit_learn_logo_small.svg
- (s.a.). (s.f.). Hugging Face logo [PNG]. https://www.stickpng.com/img/icons-logos-emojis/tech-companies/hugging-face-full-logo
- (s.a.). (s.f.). Keras logo [PNG]. https://keras.io/img/logo.png
- (s.a.). (s.f.). TensorFlow logo [PNG]. https://www.vectorlogo.zone/logos/tensorflow/index.html
- (s.a.). (s.f.). Anaconda logo [PNG]. https://anaconda.org/
- Docker Company. (s.f.). Docker Logo Blue [PNG]. https://www.docker.com/company/newsroom/media-resources/

Créditos

Facultad de
INGENIERÍA

Profesor
Jorge Eliécer Camargo Mendoza, PhD

Asistente docente
Juan Sebastián Lara Ramírez

Coordinador de virtualización
Edder Hernández Forero

Diagramadores PPT
Mario Andrés Rodríguez Triana
Rosa Alejandra Superlano Esquibel

Diseño gráfico
Clara Valeria Suárez Caballero
Milton R. Pachón Pinzón

2025