



Módulo 1: Introduction to MLFlow

What is MLFlow

MLFlow es una plataforma open-source que busca ayudar a los usuarios en el ciclo de vida del ML; simplifica y automatiza el seguimiento de modelos y métricas, la reproducibilidad de experimentos y el despliegue.

Components of MLFlow

MLFlow Tracking

- Almacenar métricas y parámetros de ejecuciones de entrenamiento
- Consultar datos de experimentos
- Almacenar modelos, artefactos y código

MLFlow Models

- Estandarizar los modelos para su posterior despliegue
- Construir modelos personalizados

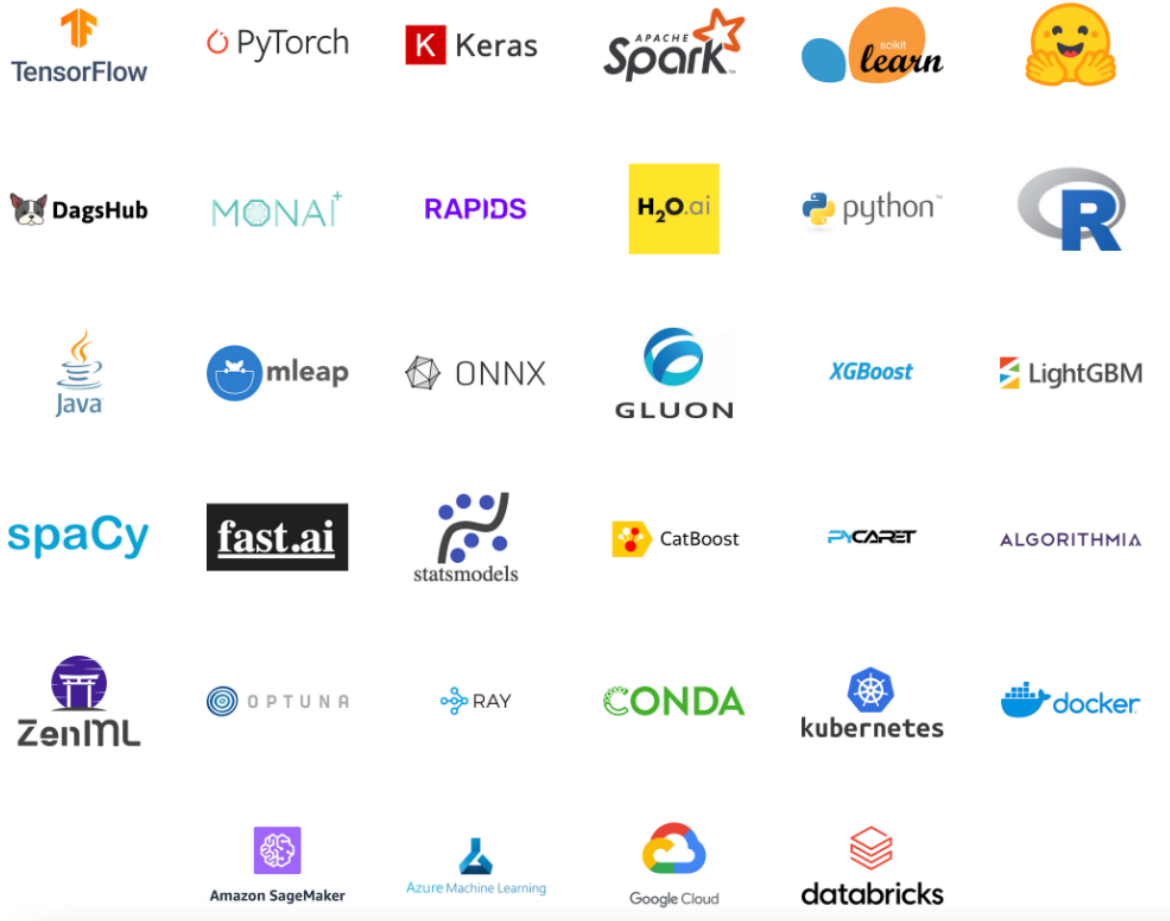
Model Registry

- Guardar y versionar modelos
- Cargar y desplegar modelos

MLFlow Projects

- Empaquetar código de ML para reproducibilidad y repetibilidad

Integrations with:





























MLFlow Experiments

El concepto más básico en MLFlow es el experimento, el cual es una manera en la que MLFlow organiza el seguimiento de las ejecuciones del entrenamiento del modelo.

Cuando empezamos a hacer el seguimiento, debemos especificar cuál experimento usaremos, así sabemos dónde encontrar los datos más tarde.

Experiments

- ☒ Default  
- ☐ Scores Experiment  
- ☐ Scores  
- ☐ Unicorn Experiment  
- ☐ Unicorn  
- ☐ Unicorn Model  
- ☐ 5  
- ☐ Test  
- ☐ 7  
- ☐ Test 2  
- ☐ 9  
- ☐ Test 3  
- ☐ 11  

Default

Track machine learning training runs in experiments. [Learn more](#)

Experiment ID: 0 Artifact Location: ./mlruns/0

> Description [Edit](#)

Sort: Created Columns

Time created: All time State: Active

Showing 3 matching runs

	Run Name	Created	Duration	metric_1	metric_2
<input type="checkbox"/>	rumbling-deer-742	3 months ago	2.0s	0.872	1.824
<input type="checkbox"/>	receptive-kit-255	3 months ago	2.0s	0.86	1.356
<input type="checkbox"/>	bright-gnu-469	3 months ago	2.0s	0.242	1.263

Working with experiments

MLFlow proporciona un par de maneras de trabajar con los experimentos

Una es a través del **MLFlow Client**, que es una API para interactuar directamente con diferentes aspectos de MLFlow. La otra es a través del **MLFlow Module**, que es una API de alto nivel usada para comenzar y gestionar directamente las ejecuciones de entrenamiento.

MLflow Client

- Create Experiments

```
client.create_experiment("Name")
```

- Tag Experiments

```
client.set_experiment_tag("Name",
k, v)
```

- Delete Experiments

```
client.delete_experiment("Name")
```

MLflow module

- Create Experiments

```
mlflow.create_experiment("Name")
```

- Tag Experiments

```
mlflow.set_experiment_tag(k, v)
```

- Delete Experiments

```
mlflow.delete_experiment("Name")
```

- Set Experiment

```
mlflow.set_experiment("Name")
```

```
import mlflow
# Create new Experiment
mlflow.create_experiment("My Experiment")
# Tag new experiment
mlflow.set_experiment_tag("scikit-learn", "lr")
# Set the experiment
mlflow.set_experiment("My Experiment")
```

MLFlow Tracking

Cada vez que entrenamos un modelo, ¿Cómo se supone que podamos llevar un seguimiento de las métricas de rendimiento, el código, los parámetros u otros artefactos usados durante el entrenamiento?

MLFlow Tracking nos permite llevar un seguimiento de métricas y parámetros a través de una API, guardar artefactos como código u otro tipo de archivos.

MLFlow usa el término “logging” cuando los datos o artefactos se guardan en MLFlow Tracking.

Training runs

MLFlow Tracking está organizado alrededor de un concepto llamado **“Runs”**

Una nueva run equivale a al entrenamiento de un nuevo modelo y su información es registrada en MLFlow.

Una run es colocada dentro de un experimento.

Starting a training run

```
import mlflow
```

```
# Start a run
```

```
mlflow.start_run()
```

```
<ActiveRun: >
```

```
# End a run
```

```
mlflow.end_run()
```

Cuando se inicia una run, el módulo de MLFlow configura la run como activa, registrando todas las métricas, artefactos y parámetros en ella. El módulo continuará haciendo el registro en la run activa hasta que el código termine o se llame la función para terminar la run.

```
import mlflow
# Set experiment
mlflow.set_experiment("My Experiment")
# Start a run
run = mlflow.start_run()
# Print run info
run.info
```

```
<RunInfo: artifact_uri='./mlruns/0/9de5df4d19994546b03dce09aefb58af/artifacts',
end_time=None, experiment_id='31', lifecycle_stage='active',
run_id='9de5df4d19994546b03dce09aefb58af', run_name='big-owl-145',
run_uuid='9de5df4d19994546b03dce09aefb58af', start_time=1676838126924,
status='RUNNING', user_id='user'>
```

URI: Identificador universal para describir una localización física o lógica de recursos.

Logging to MLFlow Tracking

Logging es el proceso de guardar métricas, parámetros y artefactos en MLFlow Tracking para una run activa.

- **Metrics**

- `log_metric("accuracy", 0.90)`
- `log_metrics({"accuracy": 0.90, "loss": 0.50})`

- **Parameters**

- `log_param("n_jobs", 1)`
- `log_params({"n_jobs": 1, "fit_intercept": False})`

- **Artifacts**

- `log_artifact("file.py")`
- `log_artifacts("./directory/")`

Logging a run

Para registrar una run en MLFlow Tracking, comenzamos configurando el experimento en el que deseamos registrar la run. Hacemos la run una activa. Entrenamos nuestro modelo y registramos los resultados.

```
import mlflow
# Set Experiment
mlflow.set_experiment("LR Experiment")

# Start a run
mlflow.start_run()

# Model Training Code here
lr = LogisticRegression(n_jobs=1)

# Model evaluation Code here
lr.fit(X, y)
score = lr.score(X, y)
```

```
# Log a metric
mlflow.log_metric("score", score)

# Log a parameter
mlflow.log_param("n_jobs", 1)

# Log an artifact
mlflow.log_artifact("train_code.py")
```

Open MLFlow UI

Se puede abrir la interfaz de usuario de MLFlow Tracking.

```
# Open MLflow Tracking UI
mlflow ui
```


Go to: <http://localhost:5000>

LR Experiment


[Share](#)

Experiment ID: 37 Artifact Location: ./mlruns/37

> Description [Edit](#)




Sort: Created ▾

 Columns ▾

Time created: All time ▾ State: Active ▾

Showing 1 matching run

	Run Name	Created	Models	Metrics	Parameters
				score	n_jobs
<input type="checkbox"/>	silent-slug-662	 1 minute ago	-	0.951	1

[LR Experiment](#) >

silent-slug-662

Run ID: a410480d4ccc4601904085b5651483b4

Date: 2023-02-20 08:14:05

Source: 

User: [weston](#)

Duration: 1.5min

Status: FINISHED


Lifecycle Stage: [active](#)

> Description [Edit](#)

Parameters (1)

Name	Value
n_jobs	1

Metrics (1)

Name	Value
score 	0.951

> Tags

Artifacts

 train_code.py





Querying runs

La interfaz gráfica de MLFlows ofrece una vista de de las runs que pertenecen al mismo experimentos pero no ofrece la manera de comparar fácilmente los resultados

Insurnace Experiment

Experiment ID: 27 Artifact Location: ./mlruns/27

> Description [Edit](#)

<div><div><div>Q</div><div>metrics.rmse < 1 and params.model = "tree"</div><div>ⓘ</div></div><div><div>Sort: accuracy_score</div><div>▼</div></div><div><div>Columns</div><div>▼</div></div></div>									
Time created: All time		State: Active							
Metrics									
<input type="checkbox"/>	Run Name	Cr Models	accuracy_score	example_count	f1_score	false_negative	false_positives	precision_score	recall_score
<input type="checkbox"/>	wise-mole-318	 shap, 1 mo	0.621	335	0.623	65	62	0.629	0.618
<input type="checkbox"/>	invincible-lark-929	 shap, 1 mo	0.621	335	0.623	65	62	0.629	0.618
<input type="checkbox"/>	sedate-fawn-631	 shap, 1 mo	0.621	335	0.623	65	62	0.629	0.618
<input type="checkbox"/>	awesome-moth-446	 shap, 1 mo	0.621	335	0.623	65	62	0.629	0.618

Search runs

Nos ofrece acceso a los datos de las runs y nos permite hacer consultas, devolviendo los datos como salida, para un análisis a futuro.

```
mlflow.search_runs()
```

#	Column	Non-Null Count	Dtype
0	run_id	6 non-null	object
1	experiment_id	6 non-null	object
2	status	6 non-null	object
3	artifact_uri	6 non-null	object
4	start_time	6 non-null	datetime64[ns, UTC]
5	end_time	5 non-null	datetime64[ns, UTC]
6	metrics.test	1 non-null	float64
7	metrics.metric_2	3 non-null	float64
8	metrics.metric_1	3 non-null	float64
9	params.param_1	3 non-null	object
10	params.random_state	3 non-null	object
11	params.n_estimators	3 non-null	object
12	tags.mlflow.user	6 non-null	object
13	tags.mlflow.runName	6 non-null	object
14	tags.mlflow.source.type	6 non-null	object

Esta función es flexible y puede aceptar diferentes argumentos para devolver los datos que se ajusten a nuestras necesidades.

- `max_results` - maximum number of results to return.
- `order_by` - column(s) to sort in `ASC` ending or `DESC` ending order.
- `filter_string` - string based query.
- `experiment_names` - name(s) of experiments to query.

Insurance Experiment

Experiment ID: 27 Artifact Location: ./mlruns/27

> Description [Edit](#)

Sort: accuracy_score ▾

Columns ▾

Time created: All time ▾

State: Active ▾

S

	Run Name	Created	Duration	Metrics				
				accuracy_score ▾	f1_score	false_negative	false_positives	precision_score
<input type="checkbox"/>	wise-mole-318	✔ 3 months ago	10.4s	0.621	0.623	65	62	0.629
<input type="checkbox"/>	powerful-shoot-853	✔ 3 months ago	5.0s	0.621	0.623	65	62	0.629
<input type="checkbox"/>	amazing-penguin-22	✔ 3 months ago	4.6s	0.537	0.485	97	58	0.557
<input type="checkbox"/>	traveling-snipe-808	✔ 3 months ago	5.6s	0.537	0.485	97	58	0.557

```
# Search runs from Insurance Experiment
mlflow.search_runs(experiment_names=["Insurance Experiment"],
    filter_string=f1_score_filter,
    order_by=["metrics.precision_score DESC"])
```

	run_id	experiment_id	...	tags.mlflow.source.type	tags.mlflow.user
0	90407e29a5aa4a31954bed874c7d4337	27	...	LOCAL	user
1	c335c0b16a5d4cf398aaa7189362b577	27	...	LOCAL	user