

Proyecto MLOps

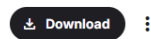
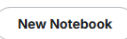
Integrantes:

Christian Andrés Delgado Cod. 2244244

Luis Felipe Campo Cod. 2243804

Repositorio proyecto y dataset: [felipkmpo/proyectomlops](https://github.com/felipkmpo/proyectomlops)

Dataset:



Skin Cancer (ISIC Images)

Classify skin images as benign or malignant



Se utilizó conjunto de datos de imágenes de cáncer de piel, el cual cuenta con dos clases, “maligno” y “benigno”, conjunto de datos disponible en kaggle.

Link dataset kaggle: <https://www.kaggle.com/datasets/rm1000/skin-cancer-isic-images>

Después del preprocesamiento de los datos, se dividió el dataset de la siguiente manera:

```
Shape of x_train: (2307, 100, 100, 3)
Shape of x_test: (891, 100, 100, 3)
Shape of x_val: (99, 100, 100, 3)
```

Observación: Se desarrollaron 3 experimentos, cada uno con parámetros diferentes que permitieron obtener métricas independientes y únicas.

DESARROLLO EXPERIMENTOS

EXPERIMENTO 1

WANDB

Parámetros

- Épocas 10

▼ **Config parameters:** {} 3 keys

architecture: "CNN"

dataset: "imagenes cancer de piel"

epochs: 10

Resumen de métricas

▼ **Summary metrics:** {} 6 keys

epoch/accuracy: 0.9008129835128784

epoch/epoch: 9

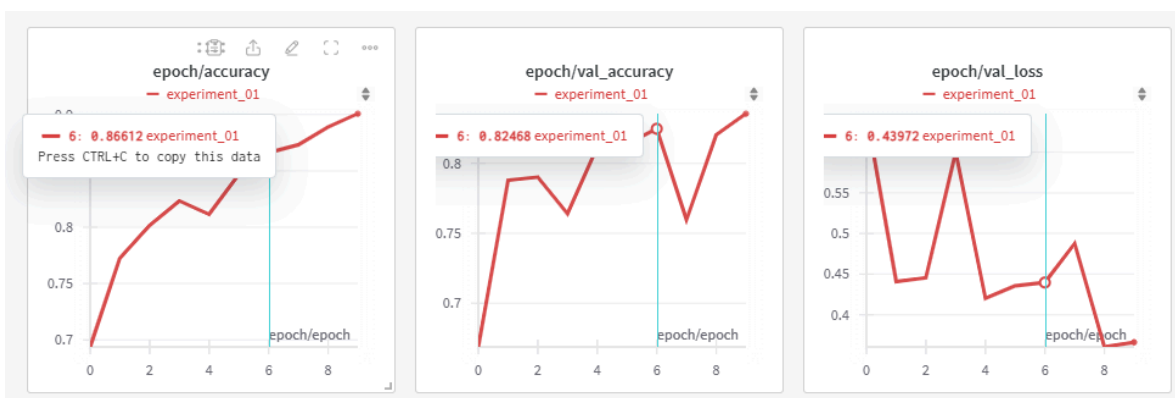
epoch/learning_rate: 0.0010000000474974513

epoch/loss: 0.22330164909362793

epoch/val_accuracy: 0.8354978561401367

epoch/val_loss: 0.36624276638031006

Gráficas Wandb



El primer gráfico, representa la **precisión del modelo** en el conjunto de entrenamiento a medida que avanza cada época, este experimento tiene 10 épocas parametrizadas, la precisión comienza alrededor del 0.7 en la primera época y se incrementa gradualmente

hasta alcanzar aproximadamente 0.87 en la última época; en conclusión el modelo está aprendiendo a clasificar mejor los datos de entrenamiento.

En el segundo gráfico identificamos la **precisión en el conjunto de validación**, aquí se mide el desempeño del modelo en datos que no ha visto durante el entrenamiento, en este gráfico la precisión fluctúa entre épocas, alcanzando un valor de aproximadamente 0.82 en la sexta época, estas fluctuaciones pueden ser indicativas de sobreajuste.

El tercer gráfico nos muestra la **pérdida en el conjunto de validación**, esto nos indica que tan bien o mal se están realizando las predicciones; en este experimento la pérdida comienza alrededor del 0.55 y disminuye con las épocas, en general una pérdida más baja indica mejor rendimiento.

MLFLOW

Resumen métricas

Metrics (6)

Search metrics	
Metric	Value
val_accuracy	0.8354978561401367
validation_accuracy	0.8354978561401367
validation_loss	0.36624276638031006
val_loss	0.36624276638031006
loss	0.22330164909362793
accuracy	0.9008129835128784

Gráficas MLFLOW



El primer gráfico muestra la **pérdida en el conjunto de validación**, la tendencia a disminuir en las últimas épocas indica que el modelo está generalizando mejor en el conjunto de validación.

El segundo gráfico muestra la **pérdida en el conjunto de entrenamiento**, la pérdida en entrenamiento comienza alta, desciende de manera constante hasta alcanzar 0.2 en la última época, esta disminución constante sugiere que el modelo está aprendiendo bien los patrones en los datos de entrenamiento.

La tercera gráfica muestra la **precisión en el conjunto de entrenamiento**, la precisión empieza alrededor de 0.7 y aumenta hasta superar el 0.9 en la última época, esto es una señal que el modelo está aprendiendo y ajustándose a los datos de entrenamiento.

EXPERIMENTO 2

WANDB

Parámetros

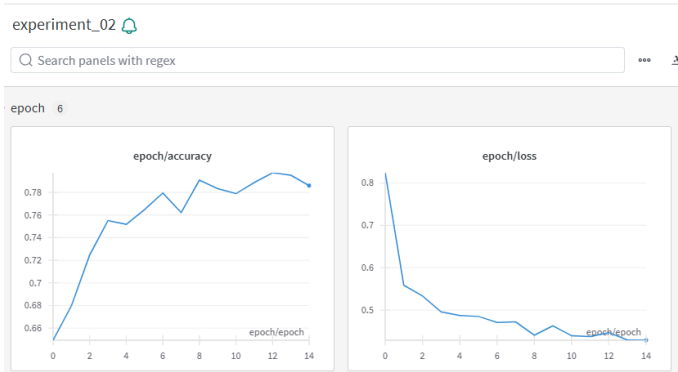
- Se aumentó épocas a 15
- Se implementó dropout 0.5

```
▼ Config parameters: {} 3 keys
architecture: "CNN"
dataset: "imagenes cancer de piel"
epochs: 15
```

Resumen de métricas

```
▼ Summary metrics: {} 6 keys
epoch/accuracy: 0.7859078645706177
epoch/epoch: 14
epoch/learning_rate: 0.0010000000474974513
epoch/loss: 0.4289230108261109
epoch/val_accuracy: 0.7965368032455444
epoch/val_loss: 0.4167916178703308
```

Gráficas Wandb



La **primera gráfica** nos muestra la relación entre épocas y precisión, esta gráfica sugiere que el modelo está aprendiendo a clasificar los datos correctamente a medida que se expone a más ejemplos.

La **segunda gráfica** relacionada con la pérdida y las épocas, identificamos una disminución en la pérdida signo de que el modelo está aprendiendo, sin embargo el aumento en la pérdida alrededor de la época 10 podría significar un ajuste inadecuado de hiperparametros.

MLFLOW

Resumen métricas

Metrics (6)

Search metrics

Metric	Value
val_accuracy	0.7965368032455444
validation_accuracy	0.7965368032455444
validation_loss	0.4167916178703308
val_loss	0.4167916178703308
loss	0.42892301082611084
accuracy	0.7859078645706177

Gráficas MLFLOW

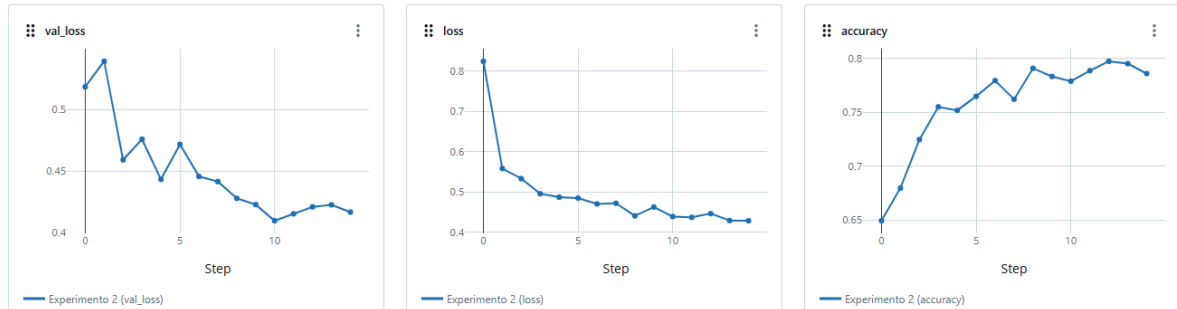
Experimento 2

Model registered

Overview **Model metrics** System metrics Artifacts

Search metric charts

Refresh



El primer gráfico muestra la **pérdida en el conjunto de validación**, la tendencia a disminuir en las últimas épocas indica que el modelo está generalizando mejor en el conjunto de validación.

El segundo gráfico muestra la **pérdida en el conjunto de entrenamiento**, esta mide que tan bien el modelo se ajusta a los datos de entrenamiento, al disminuir indica que el modelo está aprendiendo a representar los patrones presentes en los datos de entrenamiento.

La tercera gráfica muestra la **precisión en el conjunto de entrenamiento**, la precisión mide el porcentaje de predicciones correcta que realiza el modelo, si esta aumenta quiere decir que el modelo está cada vez más seguro de sus predicciones y está clasificando correctamente una mayor proporción de los datos.

EXPERIMENTO 3

WANDB

Parámetros

- Se aumentó épocas a 20
- Se eliminó dropout

▼ Config parameters: {} 3 keys

architecture: "CNN"

dataset: "imagenes cancer de piel"

epochs: 20

Resumen métricas

▼ Summary metrics: {} 6 keys

epoch/accuracy: 0.9647696614265442

epoch/epoch: 19

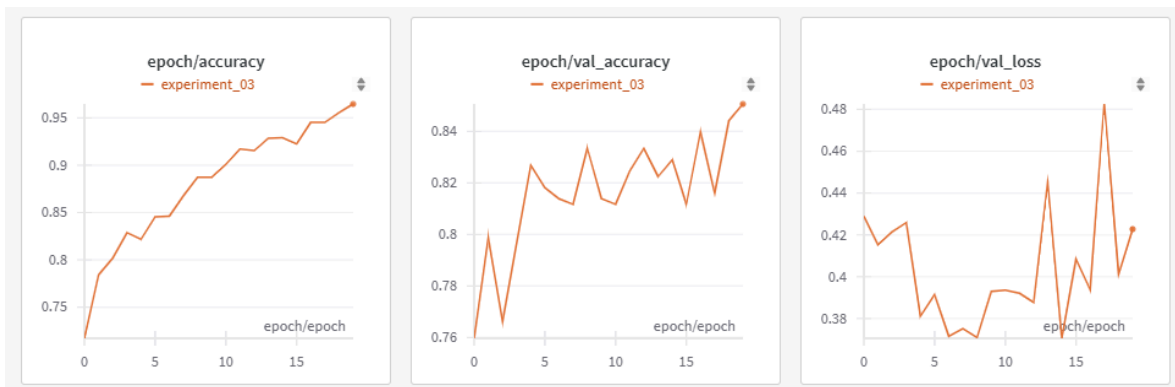
epoch/learning_rate: 0.0010000000474974513

epoch/loss: 0.12069535255432128

epoch/val_accuracy: 0.850649356842041

epoch/val_loss: 0.4227392077445984

GRÁFICAS WANDB



El primer gráfico representa la **precisión del modelo** en el conjunto de entrenamiento a medida que avanza cada época, este experimento tiene 20 épocas parametrizadas, la precisión comienza alrededor del 0.7 en la primera época y se incrementa gradualmente hasta alcanzar aproximadamente 0.96 en la última época; en conclusión, el modelo está aprendiendo a clasificar mejor los datos de entrenamiento, este ha sido la mejor precisión.

En el segundo gráfico identificamos la **precisión en el conjunto de validación**, se identifica que la precisión aumenta pero con algunas fluctuaciones, esto sugiere que el modelo está generalizando bien a nuevos datos, aunque podría haber un ligero sobreajuste hacia el final del entrenamiento.

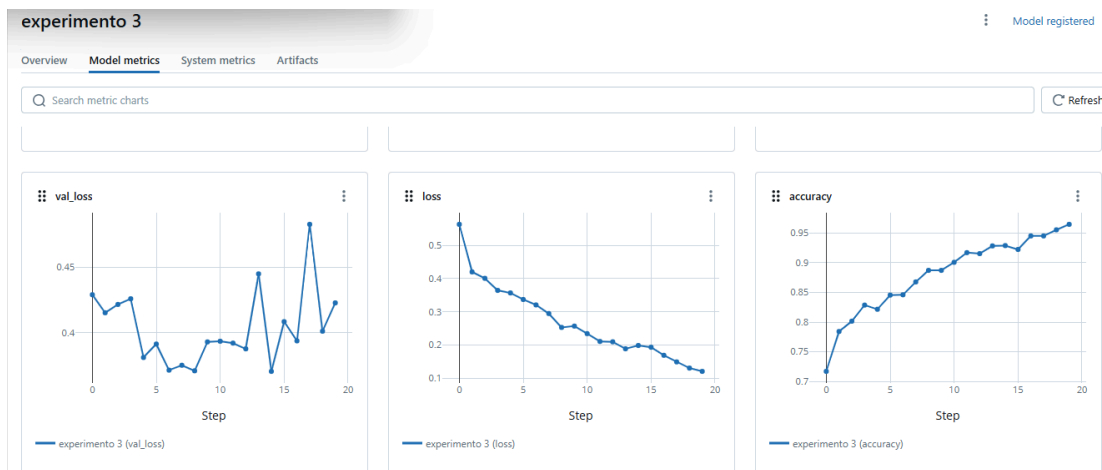
El tercer gráfico nos muestra la **pérdida en el conjunto de validación**, la pérdida disminuye al principio, lo que indica que el modelo está aprendiendo, luego comienza a aumentar ligeramente, esto podría ser un signo de sobreajuste, ya que se puede estar especializando en los datos de entrenamiento y no llegará a generalizar bien datos nuevos.

MLFLOW

Resumen métricas

Metric	Value
val_accuracy	0.850649356842041
validation_accuracy	0.850649356842041
validation_loss	0.4227392077445984
val_loss	0.4227392077445984
loss	0.12069535255432129
accuracy	0.9647696614265442

Gráficas MLFLOW



El primer gráfico muestra la **pérdida en el conjunto de validación**, la tendencia a disminuir en las últimas épocas indica que el modelo está generalizando mejor en el conjunto de validación.

El segundo gráfico muestra la **pérdida en el conjunto de entrenamiento**, esta disminuye de manera suave y constante a lo largo del entrenamiento, esto indica que el modelo está aprendiendo a representar los patrones presentes en los datos de entrenamiento.

La tercer gráfica muestra la **precisión en el conjunto de entrenamiento**, la precisión mide el porcentaje de predicciones correctas que realiza el modelo, en este experimento el modelo muestra el valor más alto de precisión, por encima del 90%, sin embargo no se debe descuidar el sobreajuste para validar que el modelo esté generalizando bien.

REGISTRO MODELOS PLATAFORMA MLFLOW

La implementación del registro de modelos se ejecuta directamente desde el código, antes del entrenamiento o compilación de modelo de aprendizaje mediante la instrucción “mlflow.tensorflow.autolog()” la cual está dejando guardadas todas la metricas, parámetros y el modelo directamente en la plataforma mlflow.

Registro modelo experimento 1

mlflow2.17.2

ExperimentsModels

Default

experimento 1

OverviewModel metricsSystem metricsArtifacts

checkpoints

model

data

MLmodel

conda.yaml

python_env.yaml

requirements.txt

tensorboard_logs

model_summary.txt

model

Path: file:///content/mlruns/0/1afcc42c971a4a619971ce1a871c7ed2/artifacts/model

MLflow Model

The code snippets below demonstrate how to make predictions using the logged model. This model is also registered to the model registry.

Model schema

Input and output schema for your model. Learn more

Name	Type
Inputs (1)	
-(required)	Tensor (dtype: float64, shape: [-1,100,100,3])
Outputs (1)	
-(required)	Tensor (dtype: float32, shape: [-1,1,1])

Validate the model before deployment

Run the following code to validate model inference works on the example payload, prior to deploying it to a serving endpoint

```
from mlflow.models import validate_serving_input

model_uri = 'runs:/1afcc42c971a4a619971ce1a871c7ed2/model'

# The logged model does not contain an input_example.
# Manually generate a serving payload to verify your model prior to deployment.
from mlflow.models import convert_input_example_to_serving_input

# Define INPUT_EXAMPLE via assignment with your own input example to the model
# A valid input example is a data instance suitable for python prediction
serving_payload = convert_input_example_to_serving_input(INPUT_EXAMPLE)

# Validate the serving payload works on the model
validate_serving_input(model_uri, serving_payload)
```

Make Predictions

Predict on a Pandas DataFrame:

```
import mlflow
logged_model = 'runs:/1afcc42c971a4a619971ce1a871c7ed2/model'

# Load model as a PyFuncModel.
model = mlflow.pyfunc.load_model(logged_model)
```

Model registered

experimento 1, v1

Registered on 2024/11/05

Registro modelo experimento 2

Experimento 2

OverviewModel metricsSystem metricsArtifacts

checkpoints

model

data

MLmodel

conda.yaml

python_env.yaml

requirements.txt

tensorboard_logs

model_summary.txt

model

Path: file:///content/mlruns/0/6ec4d258c3f74f0c98a7fae3cd670d12/artifacts/model

MLflow Model

The code snippets below demonstrate how to make predictions using the logged model. This model is also registered to the model registry.

Model schema

Input and output schema for your model. Learn more

Name	Type
Inputs (1)	
-(required)	Tensor (dtype: float64, shape: [-1,100,100,3])

Validate the model before deployment

Run the following code to validate model inference works on the example payload, prior to deploying it to a serving endpoint

```
from mlflow.models import validate_serving_input

model_uri = 'runs:/6ec4d258c3f74f0c98a7fae3cd670d12/model'

# The logged model does not contain an input_example.
# Manually generate a serving payload to verify your model prior to deployment.
from mlflow.models import convert_input_example_to_serving_input

# Define INPUT_EXAMPLE via assignment with your own input example to the model
```

Model registered

experimento 2, v1

Registered on 2024/11/05

Registro modelo experimento 3

experimento 3 Model registered

Overview Model metrics System metrics Artifacts

checkpoints

model

data

MLmodel

conda.yaml

python_env.yaml

requirements.txt

tensorboard_logs

model_summary.txt

model

Path: file:///content/mlruns/0/8f25475336fa4e8b8ef9525f56c7a04b/artifacts/model

experimento 3, v1
Registered on 2024/11/06

MLflow Model

The code snippets below demonstrate how to make predictions using the logged model. This model is also registered to the [model registry](#).

Model schema

Input and output schema for your model. [Learn more](#)

Name	Type
Inputs (1)	

Validate the model before deployment

Run the following code to validate model inference works on the example payload, prior to deploying it to a serving endpoint

```
from mlflow.models import validate_serving_input

model_uri = 'runs:/8f25475336fa4e8b8ef9525f56c7a04b/model'

# The logged model does not contain an input example.
```

Observación: Se registra modelo del experimento en plataforma mlflow en el apartado “artifacts”, si se requiere utilizar se pueden utilizar para algún despliegue o implementar predicción se utilizarán los comandos indicados o se puede descargar el archivo del modelo “model.keras”.

Overview Model metrics System metrics Artifacts

checkpoints

model

data

keras_module.txt

model.keras

save_format.txt

MLmodel

conda.yaml

python_env.yaml

requirements.txt

tensorboard_logs

model_summary.txt

model

Path: file:///content/mlruns/0/8f25475336fa4e8b8ef9525f56c7a04b/artifacts/model

experimento 3, v1
Registered on 2024/11/06

MLflow Model

The code snippets below demonstrate how to make predictions using the logged model. This model is also registered to the [model registry](#).

Model schema

Input and output schema for your model. [Learn more](#)

Name	Type
Inputs (1)	

Validate the model before deployment

Run the following code to validate model inference works on the example payload, prior to deploying it to a serving endpoint

```
from mlflow.models import validate_serving_input

model_uri = 'runs:/8f25475336fa4e8b8ef9525f56c7a04b/model'

# The logged model does not contain an input example.
```

REGISTRO MODELOS PLATAFORMA WANDB

El registro de los modelos en la plataforma wandb se realiza directamente en el desarrollo del código, se implementa después del entrenamiento del modelo, mediante la instrucción “model.save(“my_model.h5”)”; este queda almacenado en el menú “artifacts” de la plataforma wandb.

Modelo Experimento 1

my_model1 Version 0

Version Metadata Usage Files Lineage

Version overview [Link to registry](#)

Full Name	elvin-torres-universidad-aut-noma-de-...	Created At	November 6th, 2024 00:43:43
Aliases	@ latest @ v0 +	Num Consumers	0
Tags	+	Num Files	1
Digest	28555663f0335f0adf05ebc28308f450	Size	59.1MB
Created By	experiment_01	TTL Remaining	Inactive
Description	What changed in this version?		

Modelo Experimento 2

my_model2 Version 0

Version Metadata Usage Files Lineage

Version overview [Link to registry](#)

Full Name	elvin-torres-universidad-aut-noma-de-...	Created At	November 6th, 2024 00:52:09
Aliases	@ latest @ v0 +	Num Consumers	0
Tags	+	Num Files	1
Digest	66b1f9839d21e8e76ab89b41f405a76f	Size	59.1MB
Created By	experiment_02	TTL Remaining	Inactive
Description	What changed in this version?		

Modelo Experimento 3

my_model3 Version 0

Version Metadata Usage Files Lineage

Version overview [Link to registry](#)

Full Name	elvin-torres-universidad-aut-noma-de-...	Created At	November 6th, 2024 01:04:52
Aliases	@ latest @ v0 +	Num Consumers	0
Tags	+	Num Files	1
Digest	395998bf6e175d85775a64a2fbad98b5	Size	59.1MB
Created By	experiment_03	TTL Remaining	Inactive
Description	What changed in this version?		

Observación: en la pestaña “usage” encontraremos los comandos para utilizar nuestro modelo en el proyecto que necesitemos mediante api o si se requiere desde la pestaña “file” podremos descargar el modelo formato h5.

my_model3

Version 0 ▾

Version

Metadata

Usage

Files

Lineage

> root

Directory ▾



my_model.h5

59.1MB

REGISTRO DATASET ENTRENAMIENTO Y PRUEBAS PLATAFORMA WANDB

Para el registro del conjunto de datos de pruebas y entrenamiento utilizaremos el comando `dataset_artifact = wandb.Artifact("my_dataset", type="dataset")`, el cual se implementará dentro del código desarrollado, y se ubicará en justo después del `wandb.init` con el objetivo que todo el conjunto de datos quede registrado antes de compilación o entrenamiento del modelo requerido.

El dataset lo podremos identificar dentro de la opción artifacts, donde se nos listaran los datasets y modelos disponibles.

The screenshot displays the Weights & Biases (wandb) interface. The top section shows the 'my_dataset' artifact details, including its version (Version 1), full name, aliases (@ latest, @ v1), tags, digest, size (767.5MB), and description. A 'Link to registry' button is visible. Below this, the 'Artifacts' section shows a list of artifacts, with 'my_dataset' selected. The 'Files' tab for 'my_dataset' is active, showing a list of files: X_test.npy (213.8MB), X_train.npy (553.7MB), y_test.npy (7.3KB), and y_train.npy (18.6KB). Each file has a download icon.

File	Size	Action
X_test.npy	213.8MB	Download
X_train.npy	553.7MB	Download
y_test.npy	7.3KB	Download
y_train.npy	18.6KB	Download

REGISTRO DATASET ENTRENAMIENTO Y PRUEBAS PLATAFORMA MLFLOW

Para el registro del conjunto de datos de pruebas y entrenamiento en mlflow implementaremos el registro del artefacto en mlflow, para esto implementaremos el código necesario dentro del bloque `with mlflow.start_run()`.

Para el uso de artefacto tipo dataset, crearemos una carpeta temporal y en esta guardaremos arreglos tipo numpy con nuestros conjuntos de datos.

Cuando se termine de guardar los conjuntos de datos dentro de los arreglos se implementará la instrucción `mlflow.log_artifacts(temp_dir, artifact_path="dataset")` con

el fin de registrar el o los conjuntos de datos.
experimento_3

Overview

Model metrics

System metrics

Artifacts

▼ dataset

X_test.npy

X_train.npy

y_test.npy

y_train.npy

dataset

Path: file:///content/mlruns/0/77989c9a03a54a97a2d7c1405fcec427/artifacts/dataset [🔗](#)