
Kedro
+
mlflow™

Antes de começar...

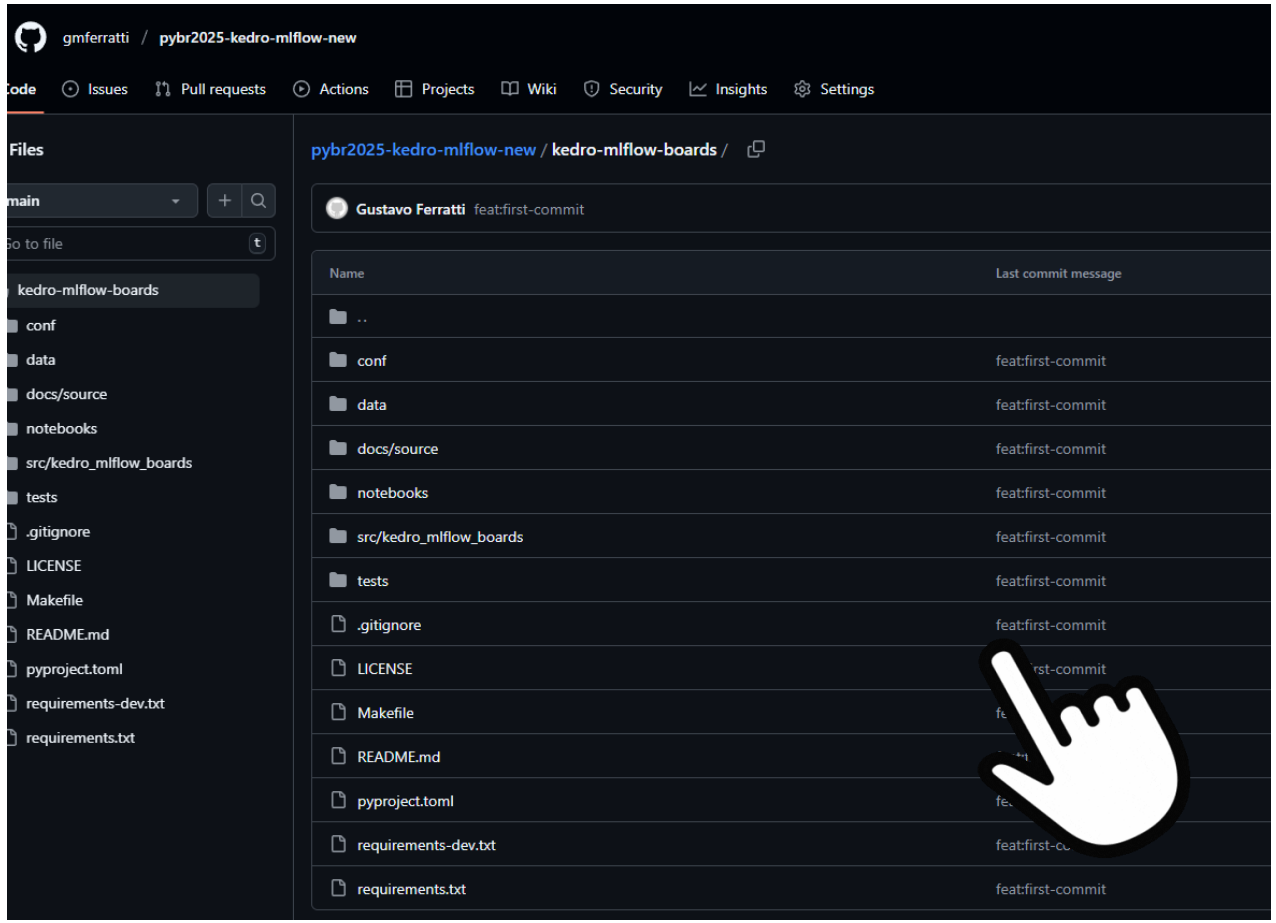
ML Engineering com Kedro e MLFlow: Primeiros Passos

Gustavo Ferratti 20 min read · Apr 14, 2025

Neste tutorial, vamos explorar como integrar duas ferramentas poderosas ao seu projeto de Machine Learning: Kedro e MLFlow. Elas permitem a construção de pipelines de dados robustos, unindo a organização e modularidade do bom desenvolvimento de software (Kedro), com a rastreabilidade e parametrização de modelos produtivos (MLFlow).

Nosso objetivo é que você, ao final deste tutorial, tenha construído **do zero** um projeto de Machine Learning escalável, monitorável e modularizado. No entanto, se você já tem um projeto em andamento e quer apenas adaptá-lo, não se preocupe — este artigo também é para você! 😉

Basta ignorar as etapas marcadas como **[Opcional]** e seguir diretamente para as próximas seções. Além do bloco **Prático**, nós teremos também um bloco



Agenda da Manhã

1. Apresentação Pessoal
2. Kedro (Teoria e Exploração)
3. MLFlow (Teoria e Exploração)
4. Kedro & MLFlow: Baby Steps
5. Hands-on: Projeto de DS com Kedro

Agenda da Tarde

1. Recap: Kedro MLFlow
2. Substituindo o nosso Modelo
3. Docker Config
4. Empacotamento e Deploy com Docker
5. Hands-on: Finalizando o Projeto e Deployando Localmente

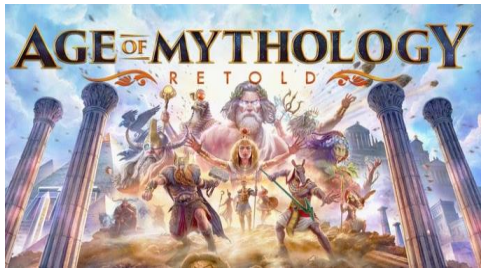
Gus

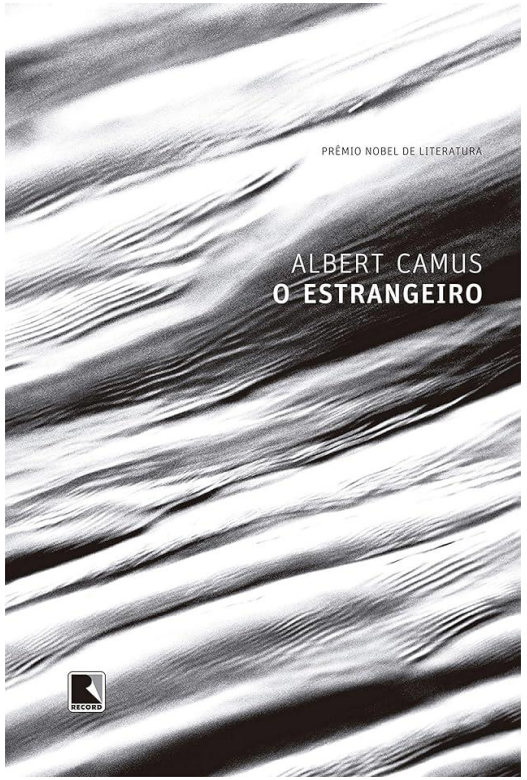
NTT DATA

ESCAFANDRO

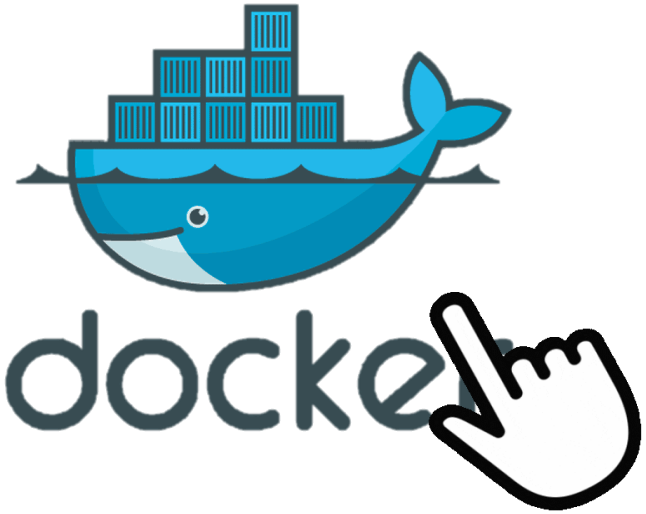
Carol

Creditas





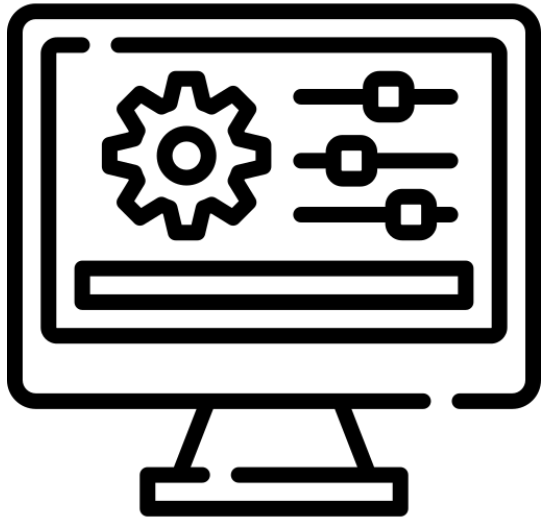
Setup Inicial



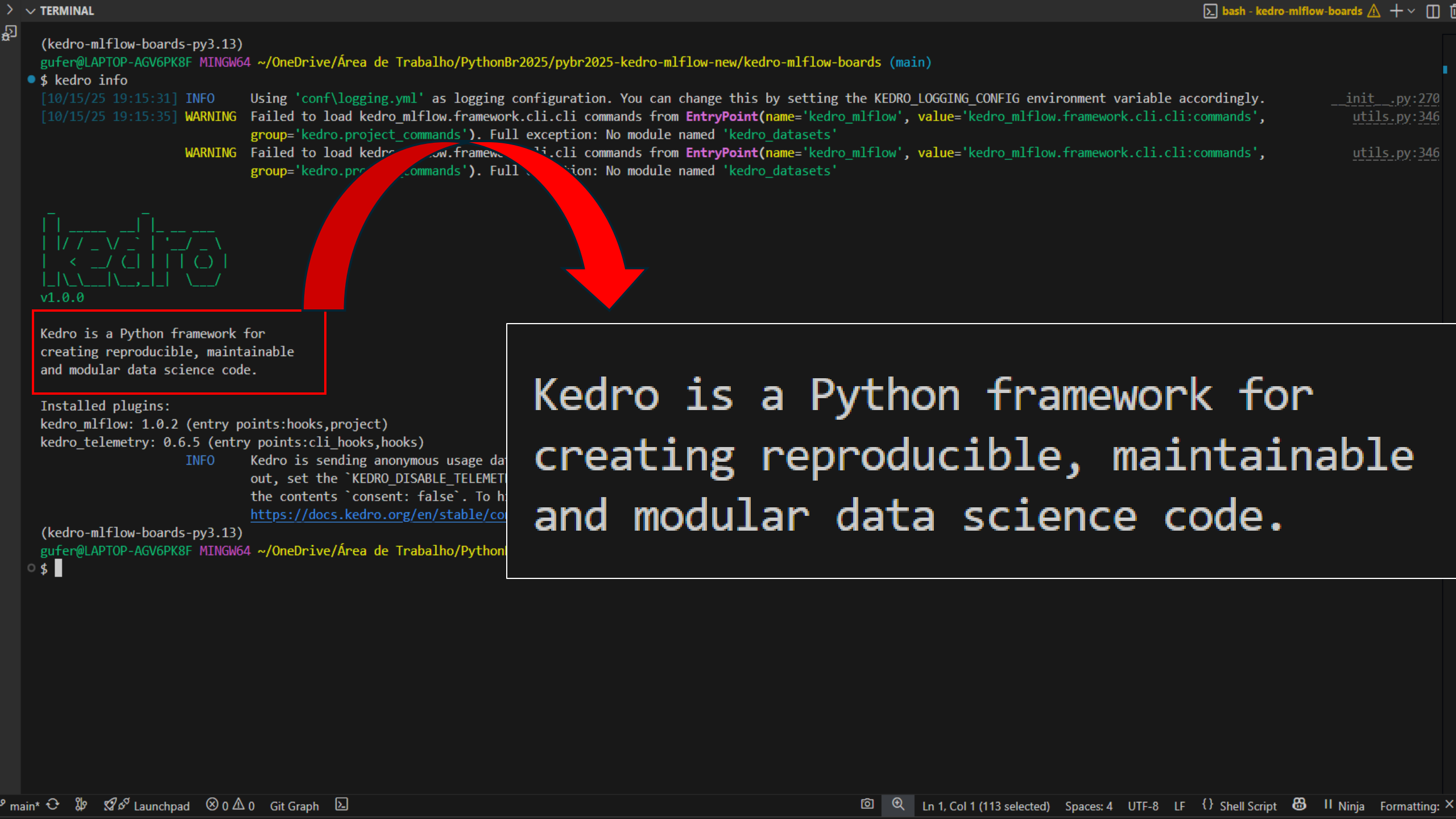

kedro

Setup kedro

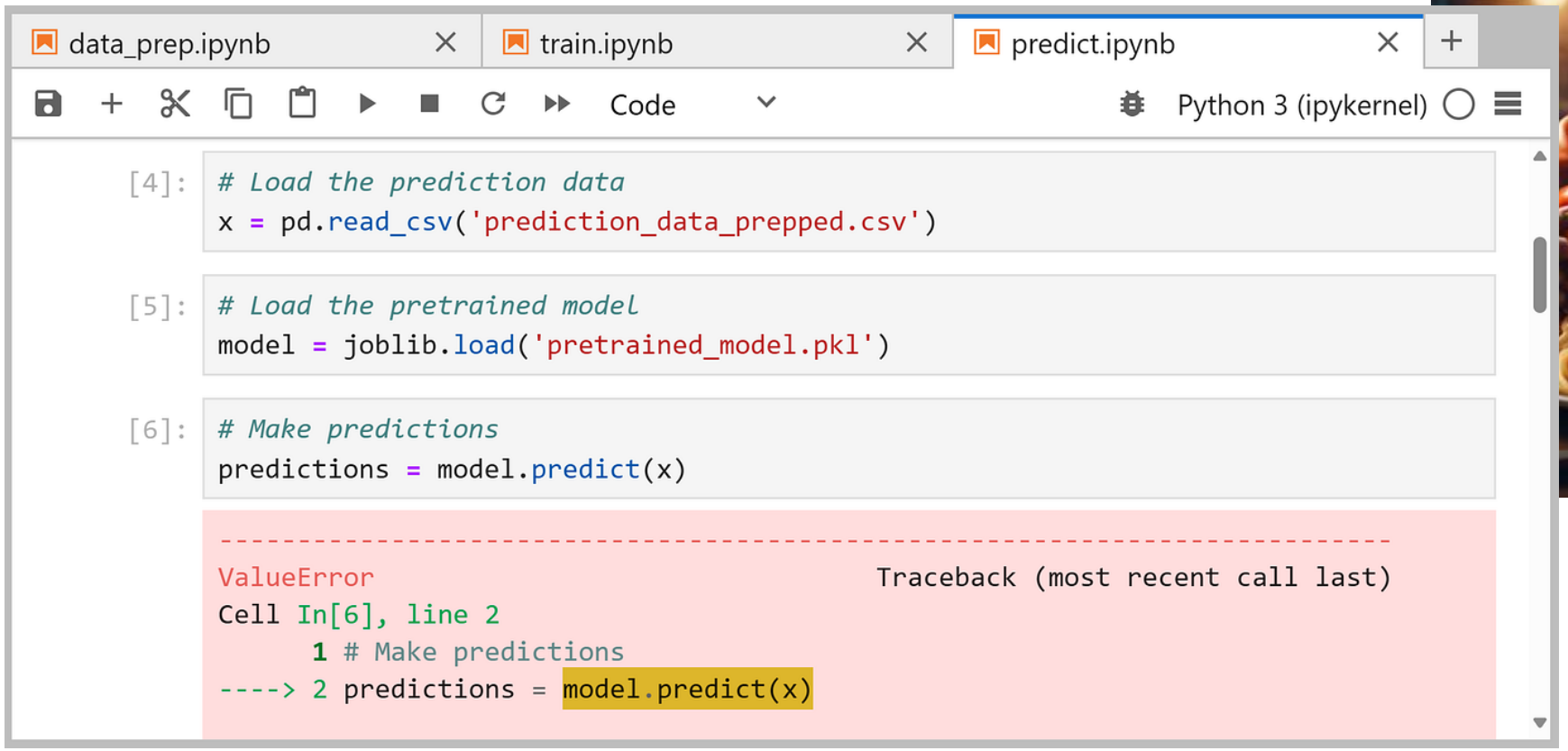
- Abra um terminal na IDE de sua preferência (recomendamos VS Code)
- Crie um novo ambiente com o Python 3.13 (recomendamos conda + uv)
- Com o ambiente ativado, instale o kedro

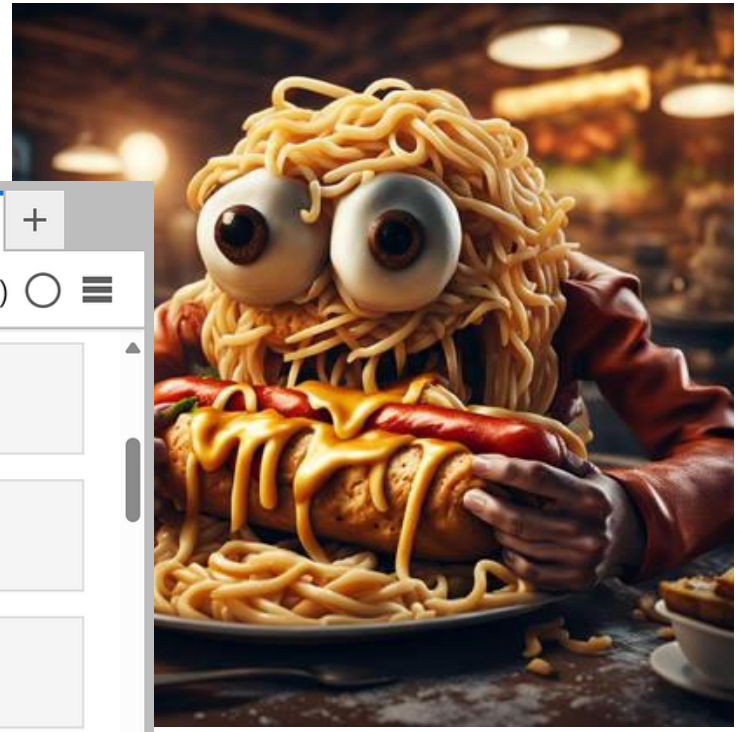


```
conda create -n <env_name> python=3.13 -y
conda activate <env_name>
pip install uv
uv pip install kedro
Kedro info
```

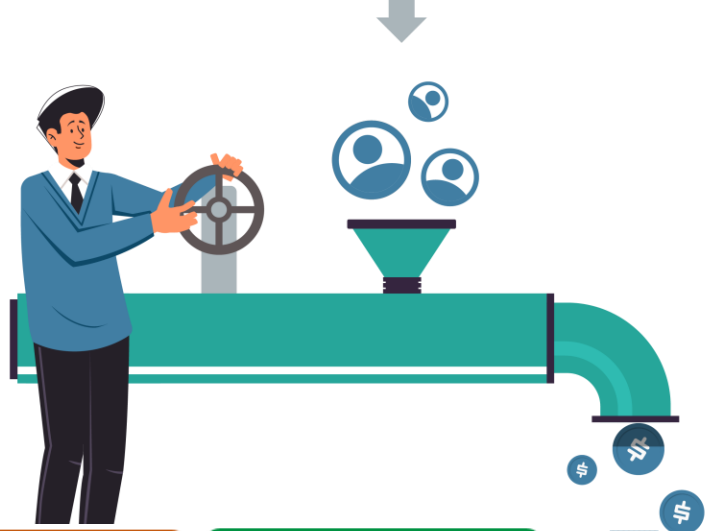

TERMINAL
bash - kedro-mlflow-boards
(kedro-mlflow-boards-py3.13)
gufer@LAPTOP-AGV6PK8F MINGW64 ~/OneDrive/Área de Trabalho/PythonBr2025/pybr2025-kedro-mlflow-new/kedro-mlflow-boards (main)
$ kedro info
[10/15/25 19:15:31] INFO Using 'conf\logging.yml' as logging configuration. You can change this by setting the KEDRO_LOGGING_CONFIG environment variable accordingly. __init__.py:270
[10/15/25 19:15:35] WARNING Failed to load kedro_mlflow.framework.cli.cli commands from EntryPoint(name='kedro_mlflow', value='kedro_mlflow.framework.cli.cli:commands', group='kedro.project_commands'). Full exception: No module named 'kedro_datasets' utils.py:346
WARNING Failed to load kedro utils.py:346
v1.0.0
Kedro is a Python framework for creating reproducible, maintainable and modular data science code.
Installed plugins:
kedro_mlflow: 1.0.2 (entry points:hooks,project)
kedro_telemetry: 0.6.5 (entry points:cli_hooks,hooks)
INFO Kedro is sending anonymous usage da
out, set the `KEDRO_DISABLE_TELEMET
the contents `consent: false`. To h
https://docs.kedro.org/en/stable/co
(kedro-mlflow-boards-py3.13)
gufer@LAPTOP-AGV6PK8F MINGW64 ~/OneDrive/Área de Trabalho/Python
$
Kedro is a Python framework for creating reproducible, maintainable and modular data science code.
main* Launchpad 0 0 Git Graph
Ln 1, Col 1 (113 selected) Spaces: 4 UTF-8 LF Shell Script Ninja Formatting:

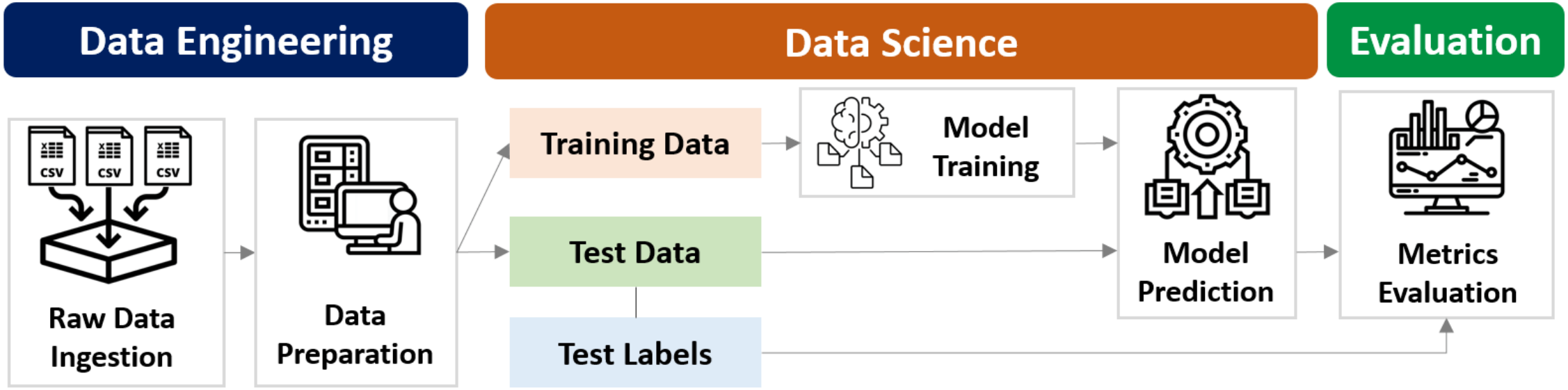
DS Code usually looks like this...





But should look like this!





— 01

ression

Random Forest

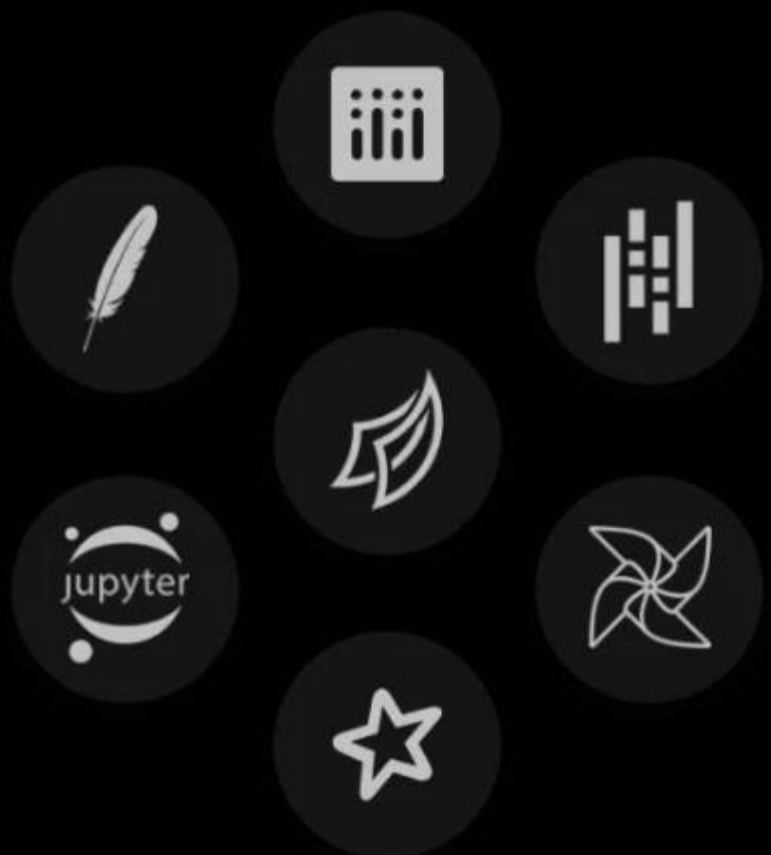
Data Catalog

A series of lightweight data connectors used to save and load data across many different file formats and file systems. The Data Catalog supports S3, GCP, Azure, sFTP, DBFS, and local filesystems. Supported file formats include Pandas, Spark, Dask, NetworkX, Pickle, Plotly, Matplotlib, and many more. The Data Catalog also includes data and model snapshots for file-based systems.

Explore the data catalog

```
# Load a Spark DataFrame on S3

flight_patterns:
  type: spark.SparkDataset
  filepath: s3a://your_bucket/data/01_raw/flight_patterns*
  credentials: dev_s3
  file_format: csv
```

```
# Save an image created with Matplotlib on Google Cloud Storage

results_plot:
  type: matplotlib.MatplotlibWriter
  filepath: gcs://your_bucket/data/08_results/plots/output_1.jpeg
  fs_args:
  project: my-project
  credentials: my_gcp_credentials
```



— 02

Integrations

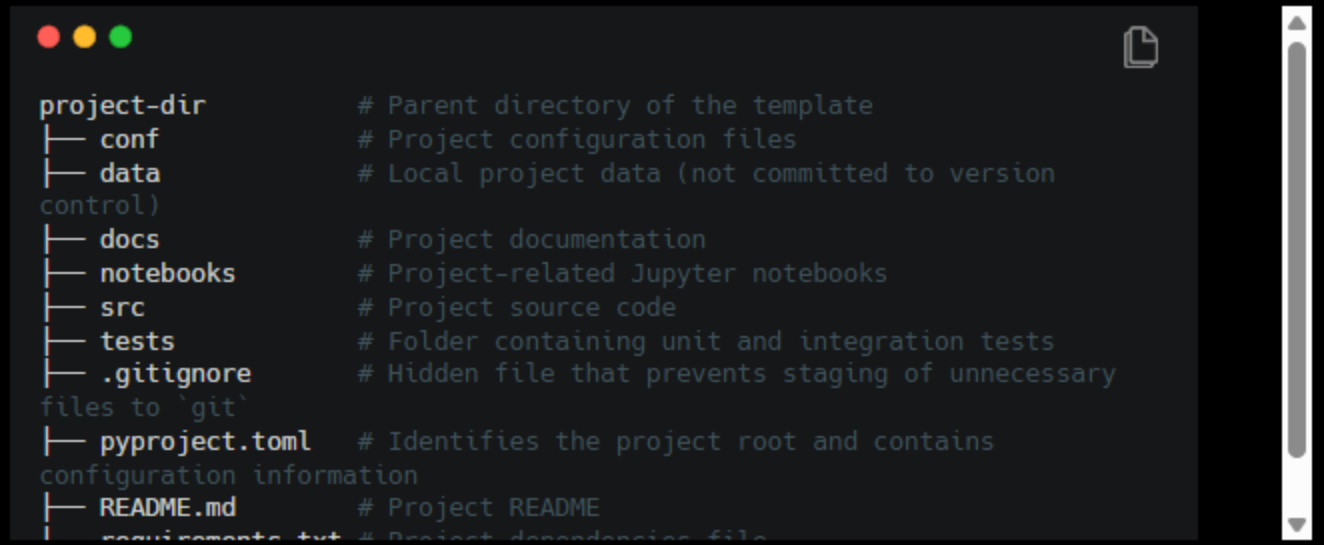
Amazon SageMaker, Apache Airflow, Apache Spark, Azure ML, Dask, Databricks, Docker, fsspec, Jupyter Notebook, Kubeflow, Matplotlib, MLflow, Plotly, Pandas, VertexAI, and more.

— 03

Project Template

You can standardise how configuration, source code, tests, documentation, and notebooks are organised with an adaptable, easy-to-use project template. Create your cookie cutter project templates with Starters.

View the project template



STARTER COOKIECUTTER

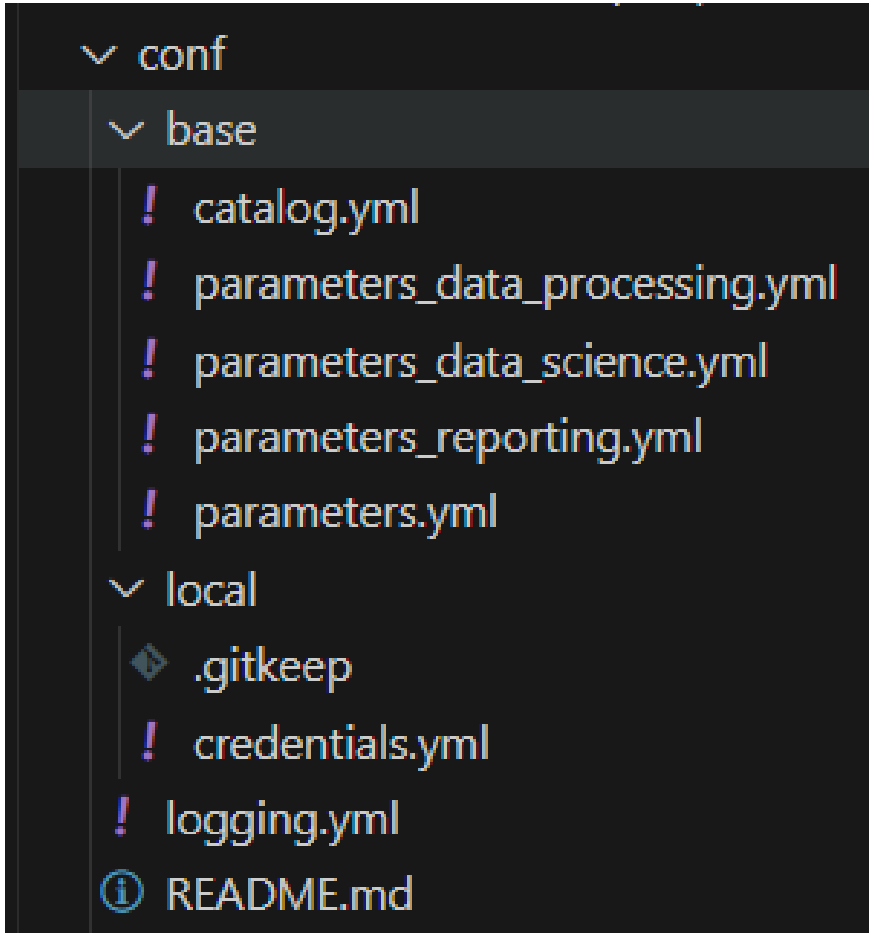
Nosso Primeiro Projeto

```
kedro new \
  --name <nome_do_projeto> \
  --tools "lint,test,log,docs,data" \
  --example y
```


```
[10/15/25 20:12:34] INFO     Using                                                    __init__.py:270
                             \rich_logging.yml' as logging configuration.

Congratulations!
Your project 'kedro-mlflow-tuto' has been created in the directory
C:\Users\gufer\OneDrive\Área de Trabalho\PythonBr2025\pybr2025-kedro-mlflow-new\kedro-mlflow-tuto

You have selected the following project tools: ['Linting', 'Testing', 'Custom Logging', 'Documentation', 'Data Structure']
It has been created with an example pipeline.
[10/15/25 20:12:38] INFO     Kedro is sending anonymous usage data with the sole purpose of       plugin.py:233
                             improving the product. No personal data or IP addresses are stored on
                             our side. If you want to opt out, set the `KEDRO_DISABLE_TELEMETRY`
                             or `DO_NOT_TRACK` environment variables, or create a `.telemetry`
                             file in the current working directory with the contents `consent:
                             false`. Read more at
                             https://docs.kedro.org/en/stable/configuration/telemetry.html
```


Let's Explore
Together

conf/



- Contém todas as configs do projeto separadas por ambientes (pastas)
- Os ambientes padrão são **base** e **local,** seguindo uma hierarquia de precedência (**local sobrescreve base**)
- Essa divisão garante que consigamos rodar o nosso código em ambientes diferentes, com permissões diferentes, sem alterar src.

conf/<level>/catalog.yml

- Descreve todas as fontes de dados usadas no pipeline.
- Explicita **o nome lógico** do dataset, o formato, o caminho e a permissão.
- Suporta opções avançadas como caching, versionamento, etc.
- Separamos o acesso aos dados das operações feitas, tornando as nossas fontes de dados “plugáveis”.

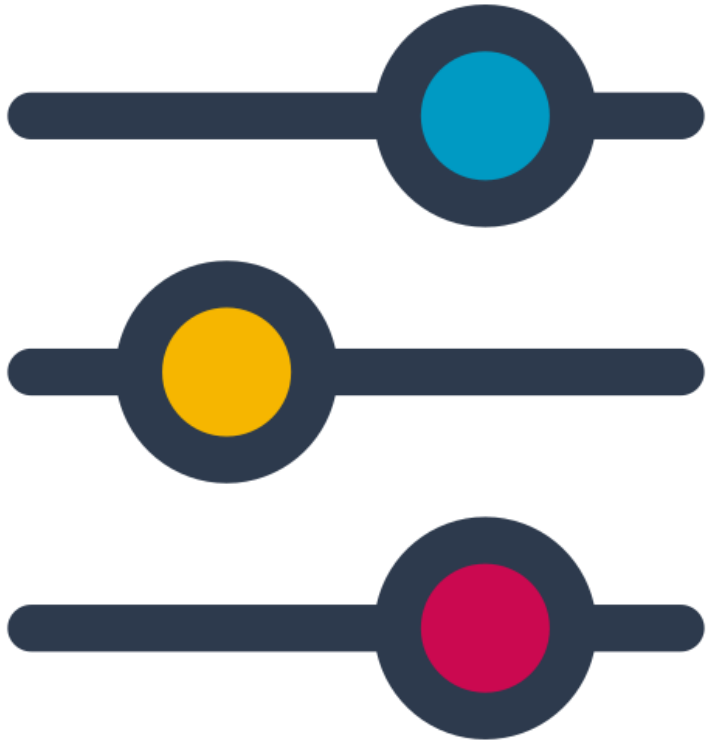
```yaml
################################################
################### CATALOG ####################
################################################

bikes:
  type: pandas.CSVDataset
  filepath: "data/01_raw/bikes.csv"

weather:
  type: spark.SparkDataset
  filepath: s3a://your_bucket/data/01_raw/weather*
  file_format: csv
  credentials: dev_s3
  load_args:
    header: True
    inferSchema: True
  save_args:
    sep: '|'
    header: True
#
scooters:
  type: pandas.SQLTableDataset
  credentials: scooters_credentials
  table_name: scooters
  load_args:
    index_col: ['name']
    columns: ['name', 'gear']
  save_args:
    if_exists: 'replace'
    # if_exists: 'fail'
    # if_exists: 'append'
```

credentials.yml

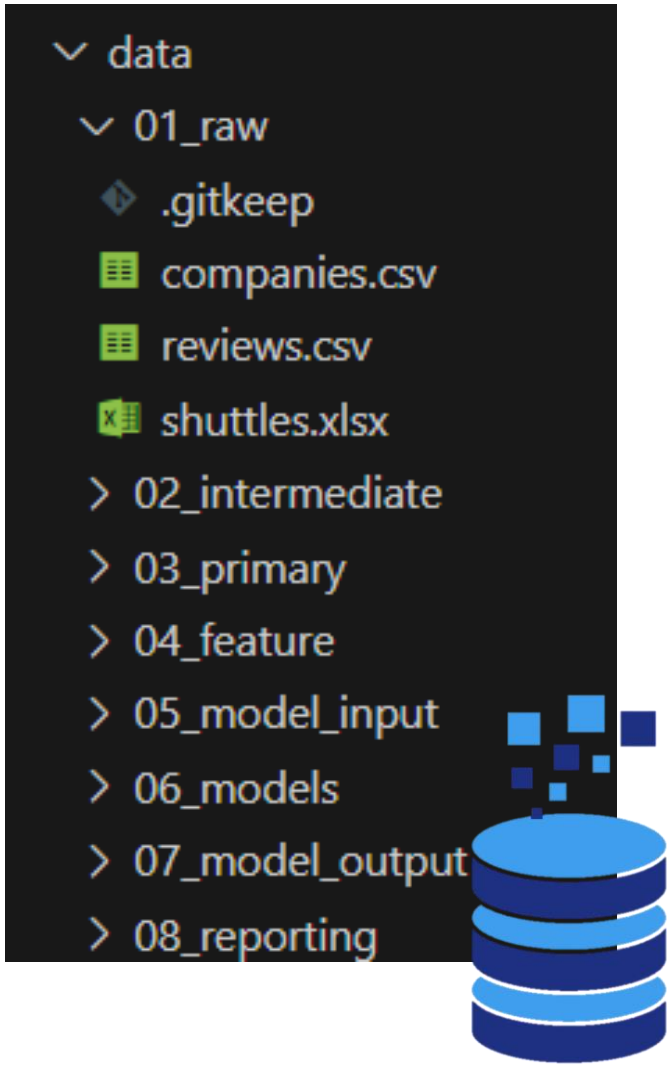
- Contém infos sensíveis, como chaves de APIs, logins e senhas.
- Deixamos as credentials em local por questões de segurança (.gitignore)
- As chaves podem ser referenciadas no catálogo de dados

parameters.yml

- Colocamos aqui parâmetros de configuração usados pelos nodes do próprio pipeline (ex. hiperparâmetros do mdelo, infos de negócios, etc.)
- Pode incluir valores aninhados
- Pode ser dividido em arquivos
- Funcionam como inputs do catálogo e podem ser chamados na pipeline.

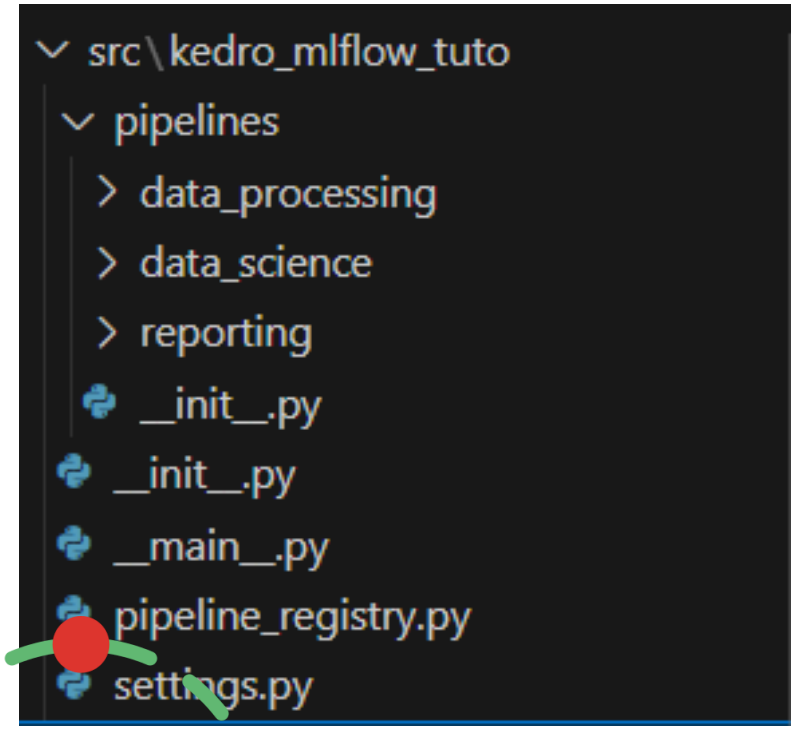


data/



- Organiza os dados consumidos e gerados localmente por camadas de processamento do pipeline
- É o diretório padrão de salvamento via catálogo
- Default de 8 camadas, mas o usuário consegue adicionar/remover, alterando o path de salvamento no catálogo

src/<projeto>/pipelines/



- Onde colocamos as subpastas de cada pipeline do nosso projeto
- Cada pipeline como uma grande etapa do projeto de dados (ex. ingestão, modelagem, avaliação, etc.)
- Facilita pensar o código modularizado
- Tudo integrado no pipeline_registry.py

pipeline_registry.py

- Ponto onde declaramos e mapeamos as pipelines de dados do nosso projeto.
- Permite compor e orquestrar pipelines menores em fluxos mais complexos.
- Em projetos prontos, serve como a **interface principal** entre o framework e o desenvolvedor.
- Permite a rodagem seletiva de pipelines específicos via CLI
  (ex. *kedro run --pipeline name*)

pipeline.py

```python
def create_pipeline(**kwargs) -> Pipeline:
    return Pipeline(
        [
            Node(
                func=preprocess_companies,
                inputs="companies",
                outputs="preprocessed_companies",
                name="preprocess_companies_node",
            ),
            Node(
                func=preprocess_shuttles,
                inputs="shuttles",
                outputs="preprocessed_shuttles",
                name="preprocess_shuttles_node",
            ),
            Node(
                func=create_model_input_table,
                inputs=["preprocessed_shuttles", "preprocessed_companies", "reviews"],
                outputs="model_input_table",
                name="create_model_input_table_node",
            ),
        ]
    )
```

- É onde declaramos a sequência de execução dos nodes.

- Bem como o que entra e sai de cada node pelo nome lógico do catálogo.

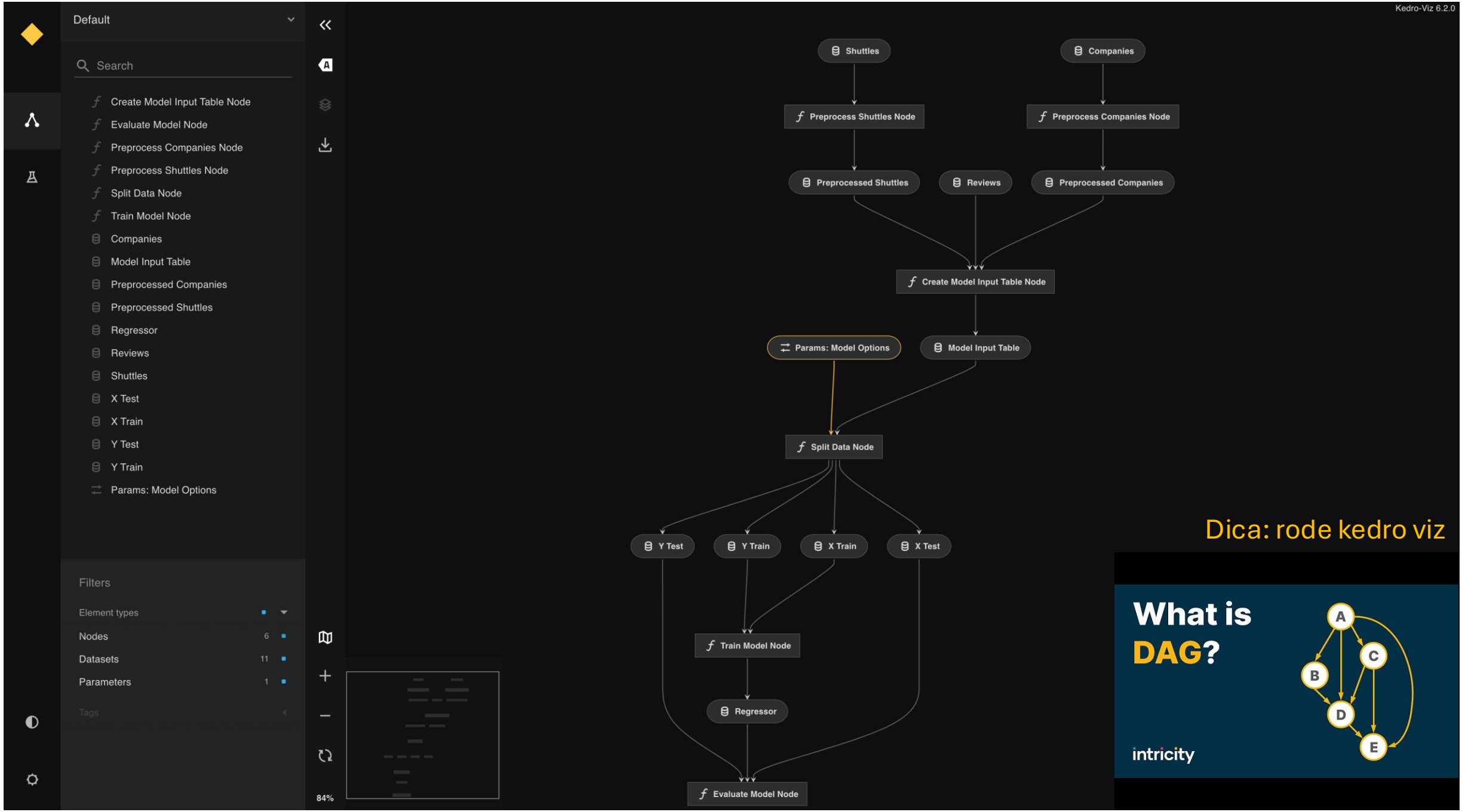
nodes.py

Métodos Internos

```python
import pandas as pd    Import "pandas" could not be resolved from source


def _is_true(x: pd.Series) -> pd.Series:
    return x == "t"


def _parse_percentage(x: pd.Series) -> pd.Series:
    x = x.str.replace("%", "")
    x = x.astype(float) / 100
    return x


def _parse_money(x: pd.Series) -> pd.Series:
    x = x.str.replace("$", "").str.replace(",", "")
    x = x.astype(float)
    return x
```

Node

```python
def preprocess_companies(companies: pd.DataFrame) -> pd.DataFrame:
    """Preprocesses the data for companies.

    Args:
        companies: Raw data.
    Returns:
        Preprocessed data, with `company_rating` converted to a float and
        `iata_approved` converted to boolean.
    """
    companies["iata_approved"] = _is_true(companies["iata_approved"])
    companies["company_rating"] = _parse_percentage(companies["company_rating"])
    return companies
```


Kedro-Viz 6.2.0
Default
Search
Create Model Input Table Node
Evaluate Model Node
Preprocess Companies Node
Preprocess Shuttles Node
Split Data Node
Train Model Node
Companies
Model Input Table
Preprocessed Companies
Preprocessed Shuttles
Regressor
Reviews
Shuttles
X Test
X Train
Y Test
Y Train
Params: Model Options
Filters
Element types
Nodes 6
Datasets 11
Parameters 1
Tags
84%
Shuttles
Companies
Preprocess Shuttles Node
Preprocess Companies Node
Preprocessed Shuttles
Reviews
Preprocessed Companies
Create Model Input Table Node
Params: Model Options
Model Input Table
Split Data Node
Y Test
Y Train
X Train
X Test
Train Model Node
Regressor
Evaluate Model Node
Dica: rode kedro viz
What is DAG?
intricity
A
B
C
D
E

Recap Kedro: Estrutura para Projetos de DS

🎯 **Objetivo**

Facilita a construção de *data pipelines* com foco em organização e boas práticas de engenharia de software.

🧱 **Principais Abstrações**

`Pipeline`, `Node`, `Dataset`, `Catalog`

📁 **Configuração Modular**

Separação clara entre dados, lógica e configuração

Recap Kedro: Estrutura para Projetos de DS

🧪 **Suporte à Prototipação**

Compatível com notebooks e execução via CLI

🔍 **Visualização**

Plugin `kedro-viz` para explorar o pipeline como DAGs

📦 **Empacotamento de Código**

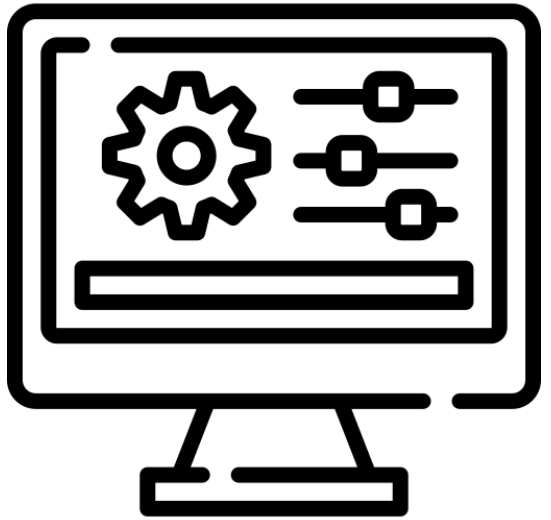
Facilita deploy e reuso com `kedro package`

💡 Ideal para estruturar o ciclo de vida do projeto desde o início do mesmo, com escalabilidade e reprodutibilidade.


mlflow™

Setup mlflow

- Abra um terminal na IDE de sua preferência (recomendamos VS Code)
- Crie um novo ambiente com o Python 3.13 (recomendamos conda + uv)
- Com o ambiente ativado, instale o mlflow



```
conda create -n <env_name> python=3.13 -y
conda activate <env_name>
pip install uv
uv pip install mlflow jupyter
mlflow server --host 127.0.0.1 --port 8080
```

MLFlow: Gerenciamento do ciclo de vida de modelos

Tracking

Registro e acompanhamento de experimentos, parâmetros, métricas e artefatos

Models

Formato padronizado para empacotar modelos, garantindo que o modelo possa ser implementado em diferentes ambientes

Model Registry

Repositório centralizado para gerenciar o ciclo de vida do modelo. Permite versionamento e transição de estágios

Projects

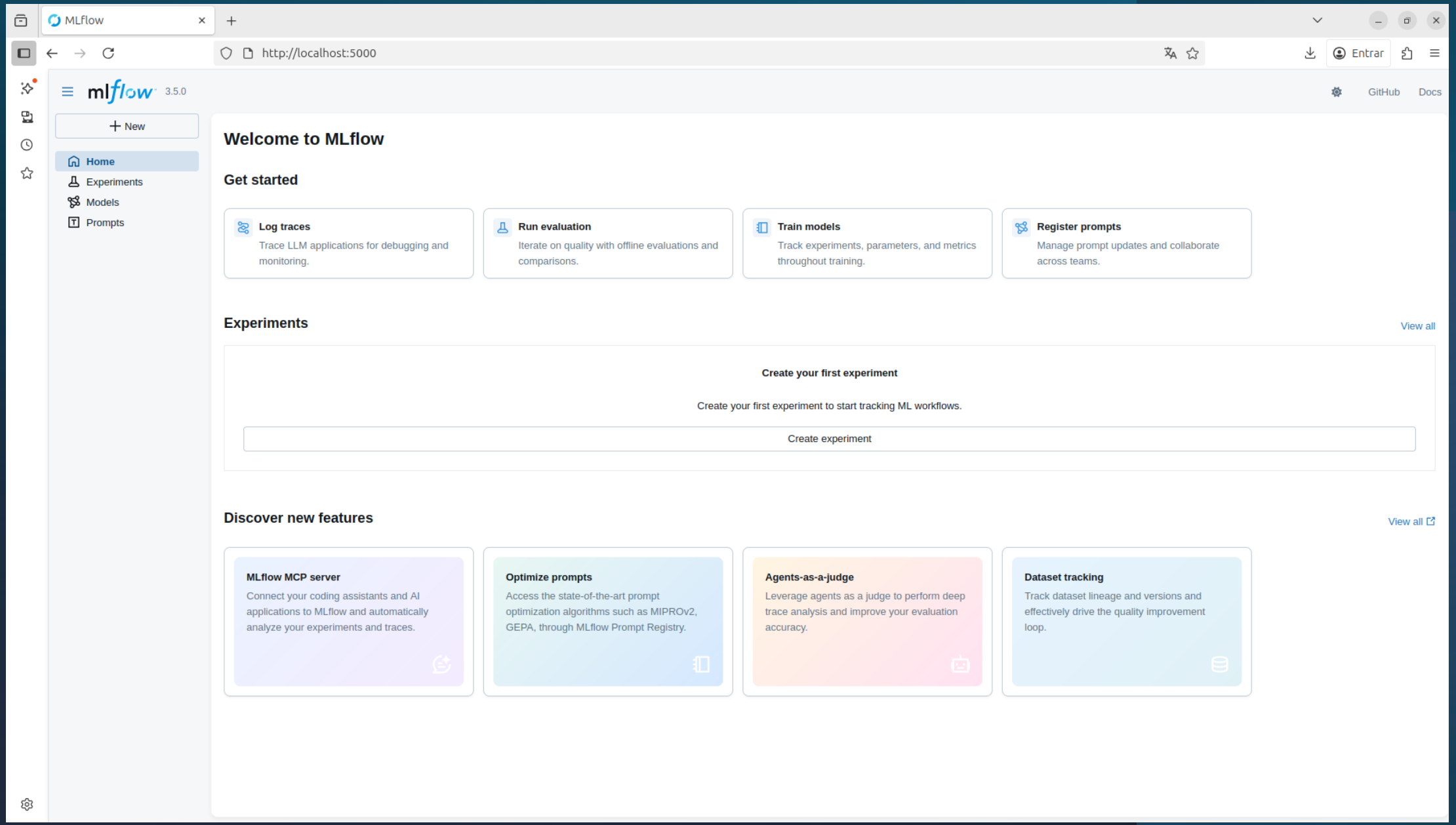
Garante o empacotamento do código em um projeto reproduzível, que define como rodar o código e as configurações de ambiente.

Serviço de Modelos

API local, deploy em Cloud, Databricks, entre outras plataformas.

Interface Gráfica

UI completa para visualizar, comparar e gerenciar todos os experimentos, runs e modelos logados.


MLflow
http://localhost:5000
Entrar
mlflow 3.5.0
GitHub
Docs
New
Home
Experiments
Models
Prompts
Welcome to MLflow
Get started
Log traces
Trace LLM applications for debugging and monitoring.
Run evaluation
Iterate on quality with offline evaluations and comparisons.
Train models
Track experiments, parameters, and metrics throughout training.
Register prompts
Manage prompt updates and collaborate across teams.
Experiments
View all
Create your first experiment
Create your first experiment to start tracking ML workflows.
Create experiment
Discover new features
View all
MLflow MCP server
Connect your coding assistants and AI applications to MLflow and automatically analyze your experiments and traces.
Optimize prompts
Access the state-of-the-art prompt optimization algorithms such as MIPROv2, GEPA, through MLflow Prompt Registry.
Agents-as-a-judge
Leverage agents as a judge to perform deep trace analysis and improve your evaluation accuracy.
Dataset tracking
Track dataset lineage and versions and effectively drive the quality improvement loop.

Recap MLFlow: Gerenciamento de Modelos

🎯 Objetivo

Gerenciar o ciclo de vida de modelos de ML (experimentos, artefatos, versionamento e serving)

🧩 Módulos Principais

- **Tracking**: log_param, log_metric, log_artifact, UI
- **Projects**: Execução do projeto reprodutível via MLproject
- **Models**: Empacotamento e padronização de modelos
- **Model Registry**: Gerenciamento de versões e estágios dos modelos

🔁 Reprodutibilidade

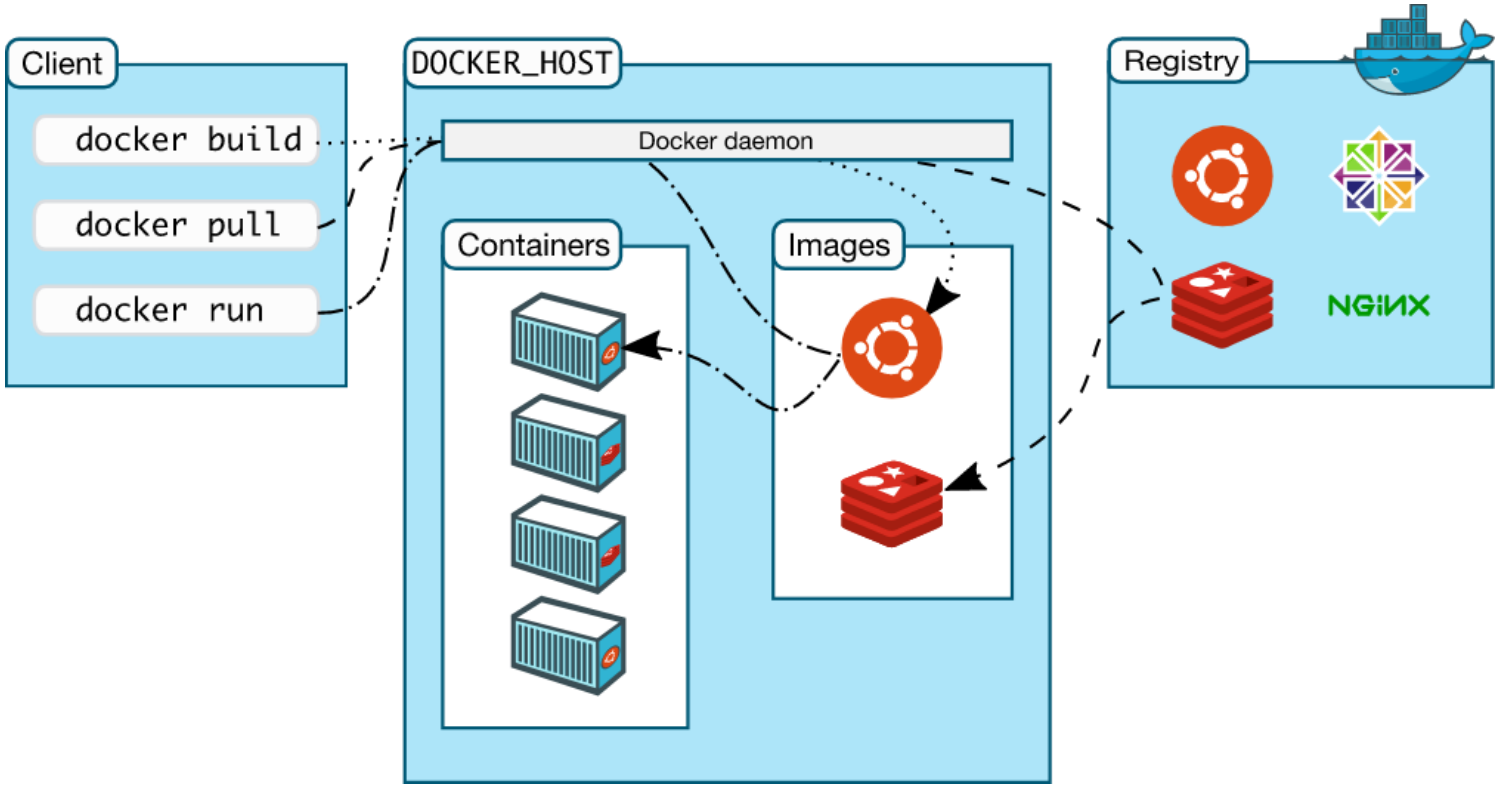
Execução por Git SHA, tracking URI


docker

"Na minha máquina funciona!"

Docker: O que é e por quê usá-lo?

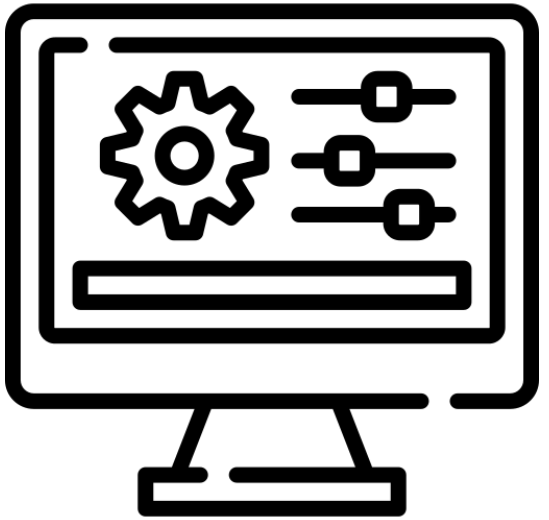
Permite empacotar um projeto (suas dependências, configurações e ambiente) em um pacote padronizado chamado I**magem,** garantindo a **reprodutibilidade** do ambiente. Garante que o ambiente de execução do projeto (Python 3.13, Kedro, MLflow, etc.) seja **idêntico** em qualquer lugar.



Setup docker

- Abra um terminal na IDE de sua preferência (recomendamos VS Code)
- Instale o Docker via terminal (Linux) ou o Docker Desktop + WSL (Windows)
- Build e start nos serviços do docker-compose

```
curl -fsSL https://get.docker.com/ | sh
sudo usermod -aG docker $USER
newgrp docker
docker run hello-world

docker compose build
docker compose up
```




kedro
mlflow™

🧩 Critério	🔷 Kedro	🔶 MLflow
🧱 **Organização de Projeto**	Estrutura modular, clara e opinativa (pipelines/, conf/, data/)	Estrutura via MLproject, mais flexível, menos estruturada
🔄 **Execução via CLI**	Comando kedro run	Comando mlflow run
🧪 **Prototipação Interativa**	Suporte a notebooks e extensões para VS Code	Existente, mas limitado; foco em execução reproduzível
🔍 **Visualização de Pipeline (DAG)**	Sim, com kedro-viz, incluindo preview de datasets	Não possui visualização nativas
🔧 **Integração com CI/CD**	Estrutura limpa e facilmente integrável	Estrutura limpa e facilmente integrável

🧩 Critério	🔷 Kedro	🔶 MLflow
🔁 **Versionamento de Artefatos**	Suporte via AbstractVersionedDataset	Versionamento completo: log_param, log_metric, log_artifact + UI
📈 **Rastreabilidade de Experimentos**	Sem implementação nativa	UI interativa, filtros, gráficos, comparação de execuções
📦 **Empacotamento**	Empacotamento de código com kedro package	Empacotamento de modelos com flavors + metadata
🚀 **Deploy/Serving de Modelos**	Via .pkl, joblib, scripts customizados	Suporte nativo a serving (API REST, batch prediction, integração com cloud)
⚙️ **Foco Principal**	Estruturação e organização de projetos de dados	Gestão do ciclo de vida de modelos de machine learning

Conceitos Fundamentais (ML)

Modelo: o resultado do processo de treinamento. É o cérebro, o motor do nosso sistema de aprendizado de máquina (*log_model*)

Métrica: valores que nos ajudam a avaliar o desempenho do modelo. Analogia com o que medimos no painel do nosso carro - velocidade, rotação, etc. (log_metric)

Artefato: tudo que é produzido ou utilizado ao longo do pipeline de modelagem que não se encaixam nos dois conceitos anteriores. Ex. features que alimentam o modelo, gráficos, etc.

Logging

- Uma das grandes vantages do kedro é centralizar o I/O
- Dito isso, é melhor evitarmos usar mlflow.log_XXX in-code, pois perdemos essa centralização
- ***Recomendado:*** todo o I/O do MLFlow acontecer via catálogo com classe específica

Não bypasse o catálogo!

```
# Initiate the MLflow run context
with mlflow.start_run(run_name=run_name) as run:
    # Log the parameters used for the model fit
    mlflow.log_params(params)

    # Log the error metrics that were calculated during validation
    mlflow.log_metrics(metrics)

    # Log an instance of the trained model for later use
    mlflow.sklearn.log_model(sk_model=rf, input_example=X_va
```

Classe	O que registra	Quando usar	Exemplo de uso
MlflowModelTrackingDataset*	Modelo no *artifact_store* do MLflow	Quando você quer versionar o modelo, rastreá-lo e servi-lo via API do MLflow.	Após treinar um modelo, quer registrál-lo diretamente na UI do MLFlow
MlflowModelLocalFileSystemDataset	Modelo salvo localmente	Quando você deseja manter uma cópia local do modelo já na estrutura do MLFlow	Salvar o modelo como um arquivo .pkl na pasta data/06_models.
MlflowMetricDataset	Uma métrica única (ex: acurácia)	Para logar valores simples como acurácia, F1-score, etc.	Após avaliar o modelo, enviar acurácia (por exemplo, 0.95) para o MLflow.
MlflowMetricHistoryDataset*	Histórico de uma métrica	Quando deseja logar a evolução de uma única métrica ao longo do tempo.	Durante o treinamento de um modelo em que faço várias medidas de F1-Score, manter todas as medidas.
MlflowMetricsHistoryDataset	Histórico de múltiplas métricas	Quando precisa rastrear várias métricas em paralelo, com histórico completo em um único arquivo.	Logar simultaneamente acurácia, precisão e recall em cada época do treinamento.
MlflowArtifactDataset*	Artefatos como imagens, gráficos, arquivos, datasets	Para salvar qualquer arquivo valioso para explicar o modelo.	Após gerar uma matriz de confusão com matplotlib, salvar como imagem .png

```yaml
# catalog.yml

evaluation_metrics:
  type: kedro_mlflow.io.metrics.MlflowMetricsHistoryDataset
```

Logando Métricas

Como configurar:

- O node deve retornar um `dict` com:
  - Nome da métrica
  - Valor
  - Step (passo de execução - para histórico)
- Exemplo:
  "r2_score": {"value": 0.89, "step": 0}

```yaml
regressor:
  type: kedro_mlflow.io.models.MlflowModelTrackingDataset
  flavor: mlflow.sklearn
  artifact_path: regressor
  save_args:
    registered_model_name: spaceflights-regressor
```

Logando Modelos

Como fazer no catalog.yml:

- Substitua `pickle.PickleDataset` por `MlflowModelTrackingDataset`
- Defina o `flavor` (ex: `mlflow.sklearn`) e o `artifact_path`
- Use `save_args` e `registered_model_name` para registrar o modelo no Model Registry
- Zero esforço adicional após o setup, logging a cada `kedro run`

```yaml
clean_board_games:
  type: kedro_mlflow.io.artifacts.MlflowArtifactDataset
  dataset:
    type: pandas.ParquetDataset
    filepath: data/02_intermediate/clean_board_games.parquet
  artifact_path: data_intermediate
```

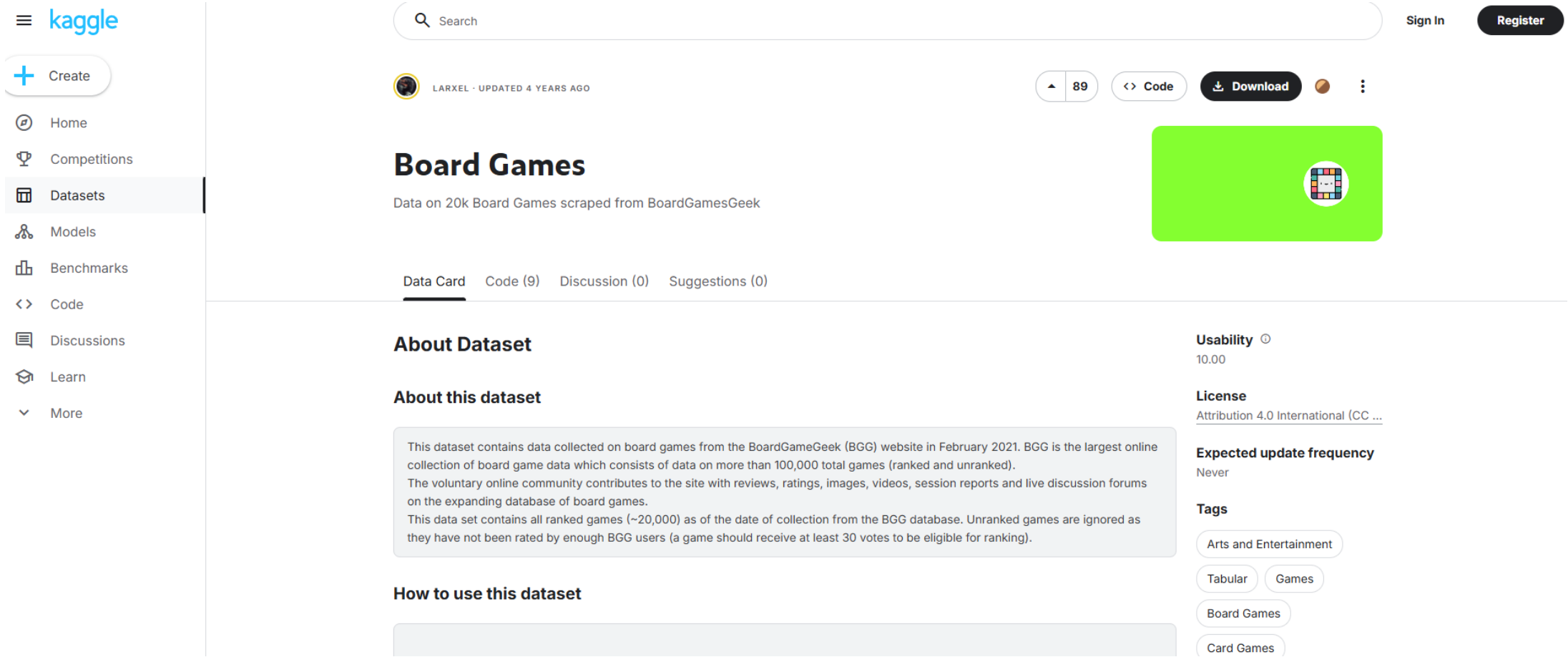
Logando Artefatos

- Use `MlflowArtifactDataset` no lugar de datasets tradicionais
- Artefatos salvos automaticamente no MLflow e organização centralizada na pasta `mlruns`

HANDS ON

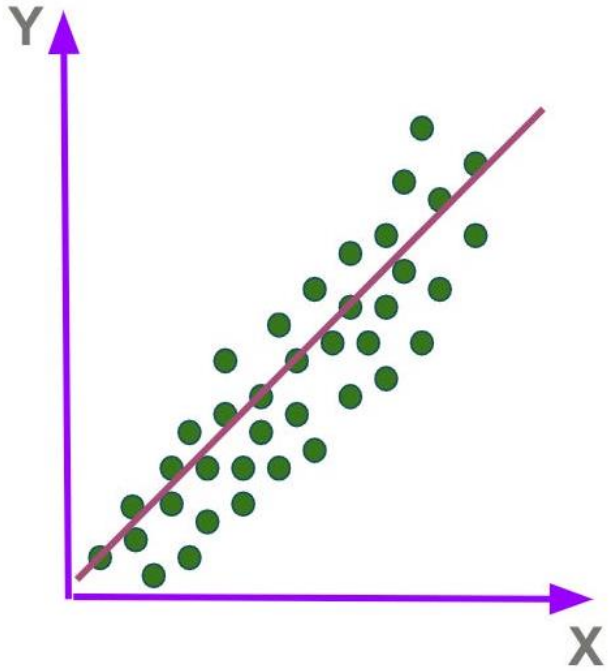
Bora construir um projeto de DS simples com Kedro?

O Dataset

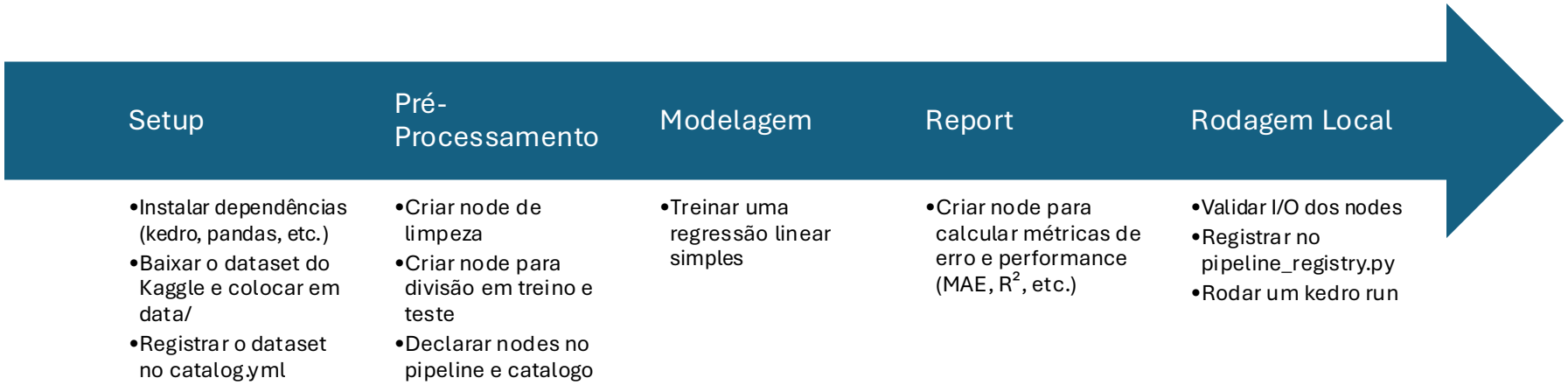

kaggle
Create
Home
Competitions
Datasets
Models
Benchmarks
Code
Discussions
Learn
More
Search
Sign In
Register
LARXEL · UPDATED 4 YEARS AGO
89
Code
Download
Board Games
Data on 20k Board Games scraped from BoardGamesGeek
Data Card
Code (9)
Discussion (0)
Suggestions (0)
About Dataset
About this dataset
This dataset contains data collected on board games from the BoardGameGeek (BGG) website in February 2021. BGG is the largest online collection of board game data which consists of data on more than 100,000 total games (ranked and unranked).
The voluntary online community contributes to the site with reviews, ratings, images, videos, session reports and live discussion forums on the expanding database of board games.
This data set contains all ranked games (~20,000) as of the date of collection from the BGG database. Unranked games are ignored as they have not been rated by enough BGG users (a game should receive at least 30 votes to be eligible for ranking).
How to use this dataset
Usability
10.00
License
Attribution 4.0 International (CC ...
Expected update frequency
Never
Tags
Arts and Entertainment
Tabular
Games
Board Games
Card Games

O Modelo

Faça um modelo de **regressão linear** utilizando o framework **Kedro** para **prever a nota média (average_rating) de um jogo de tabuleiro** a partir de outras variáveis do dataset *Board Games* (Kaggle).

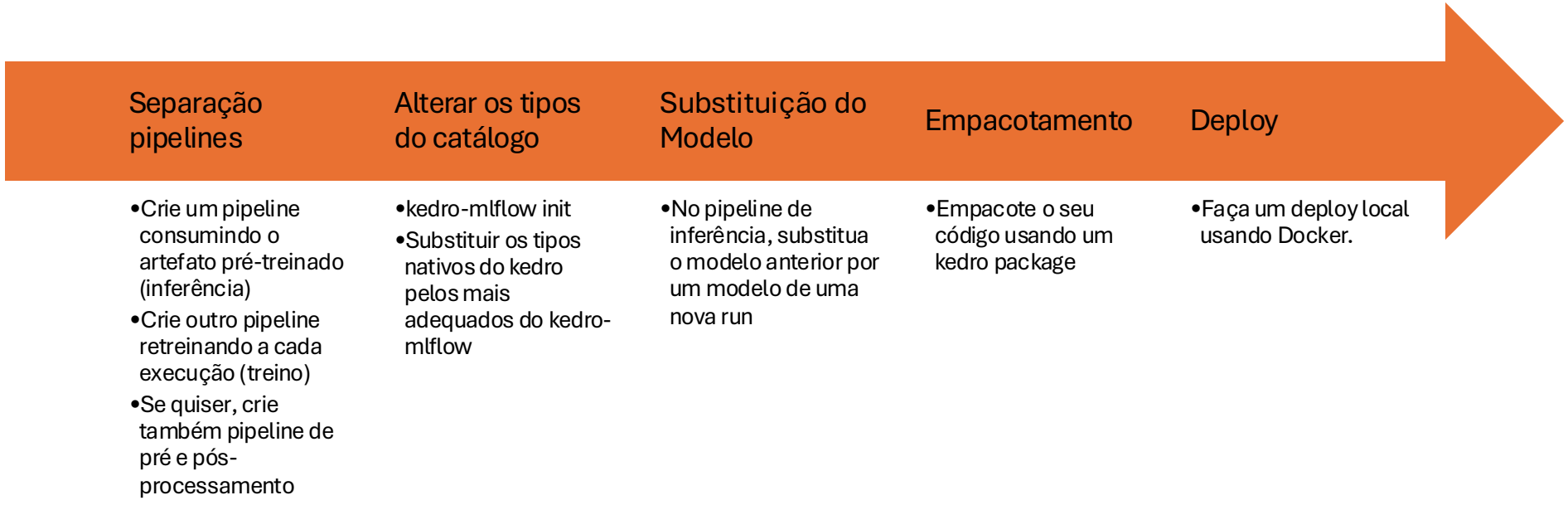


Fluxograma - Exercício 1




HANDS ON
Bora melhorar nosso projeto com MLFlow e Docker?

Fluxograma - Exercício 2




OBRIGADO
THANK YOU
Obrigado
Asante
Mamana
Juspaxar
Chokrane
Grazie
Cam on ban
Kiitos
Dankon
Maake
Ua Tsaug Rau Koj
Mochchakkeram
Terma Kasih
Multumesc
Merci
Raibh Maith Agat
Matondo
Dank Je
Nirringrazzjak
Spasibo
Multumesc
Obrigado
Grazie
Raibh Maith Agat
Merci
Multumesc
Mochchakkeram
Grazie
Vinaka
Obrigado
Thank You
Merci
Arigato
Spasibo
Mochchakkeram
Merci
Matondo
Salamat
Nirringrazzjak
Obrigado
Mochchakkeram
Ua Tsaug Rau Koj
Kiitos
Maake
Matur Nuwun
Chokrane
Raibh Maith Agat
Salamat
Welalin
Multumesc
Spasibo
Cam on ban
Kiitos
Raibh Maith Agat
Merci
Mochchakkeram
Chokrane
Grazie
myloview

PYTHON BRASIL 2025

Gustavo Ferratti
MLOps Engineer @ NTT Data
gmferratti@gmail.com
+55 16 99643 0580
/in/gmferratti

Caroline Zago
Lead AI Engineer @ Creditas
carolineszago@gmail.com
+55 49 9830-5294
/in/caroldbzago