Projeto Final: MLOps na Prática – Previsão de Preço de Imóveis

1. Dataset e Problema

Utilizamos o **Ames Housing** (Kaggle) – um conjunto de 2 930 registros com 82 variáveis – para **estimar o preço de venda (`SalePrice`)** de cada imóvel (tarefa de regressão).

2. Metodologia & Ferramentas

Etapa	Ferramentas
Experiment tracking e registry	**MLflow**
Serving	**FastAPI** (endpoint `/predict`)
Monitoramento de drift	**Evidently AI**
Retreinamento automático	Python + scikit-learn + MLflow
Orquestração	**Docker** & **docker-compose** (dois containers: *api* & *monitor*)
Linguagem / outras libs	Python 3.10, pandas, scikit-learn

Workflow resumido

1. **Exploração → Pré-processamento** (imputação & one-hot)
2. **Treinamento** de 2 modelos (Random Forest & ElasticNet) rastreados no MLflow
3. Registro do **melhor modelo** no *Model Registry*
4. **API** containerizada fornece previsões
5. Container de **monitoramento** roda *cron* ➜ Evidently gera relatório JSON ➜ script decide se **retreina**
6. Novo modelo é logado no MLflow e salvo em `models/model.pkl`

3. Resultados & Métricas

Modelo	RMSE	MAE	R^2
Random Forest	**19 245**	**12 780**	**0.89**
ElasticNet	26 487	18 123	0.81

> **Modelo em produção:** `RandomForestRegressor` (100 árvores, *random_state*=42)

4. Estrutura do Repositório

```
.
├── data/               # raw, processed (train/test)
├── src/
│   ├── training/       # pipeline de ETL + treino
```

```
│   ├── api/            # FastAPI app
│   └── monitoring/     # generate_drift_report.py & auto_retrain.py
├── models/             # model.pkl (backup local)
├── reports/            # data_drift_report.json
├── Dockerfile          # API
├── Dockerfile.monitor  # Monitor + cron
├── docker-compose.yml  # orquestração
└── docs/
    └── mlops-pipeline-diagram.png
```

5. Pipeline Completo

Fluxo detalhado

```
        +-----------+           +---------------+
        |  Dados    |--ETL----->| MLflow Exper. |
        +-----------+           +---------------+
                                      | best
                                      ▼
        +------------------+    +------------+
        |  Registro Modelo |<---|  Treino    |
        +------------------+    +------------+
                 |                    ▲
        docker   | serve              | retrain (cron)
        compose  ▼                    |
        +------------+        +--------------+
        |  FastAPI   |        | Monitor/Drift|
        +------------+        +--------------+
```

6. Monitoramento & Retreinamento

- `generate_drift_report.py` cria **JSON** com métrica `drift_share_detected`
- `auto_retrain.py` abre esse JSON
  - Se `drift_share_detected` > 0.20 ⇒ treina novo Random Forest, loga no MLflow, grava `models/model.pkl`
- **Cron** roda ambos diariamente às **02:00 AM** dentro do container *monitor*

```
0 2 * * * python src/monitoring/generate_drift_report.py &&
python src/monitoring/auto_retrain.py
```

7. Considerações Finais

- Pipeline **modular, reprodutível e versionado** (Git + MLflow + Docker)
- **Adaptativo**: identifica drift e atualiza modelo sem intervenção humana
- Fácil de implantar em nuvem (basta apontar N volumes + registrar MLflow remoto)

GitHub: https://github.com/arduinitavares/house-price-mlops • *Data*: Ames Housing (© Dean De Cock)