Trabalho Final MLOps na Prática

1. Pipeline de MLOps para Previsão de Preços de Imóveis

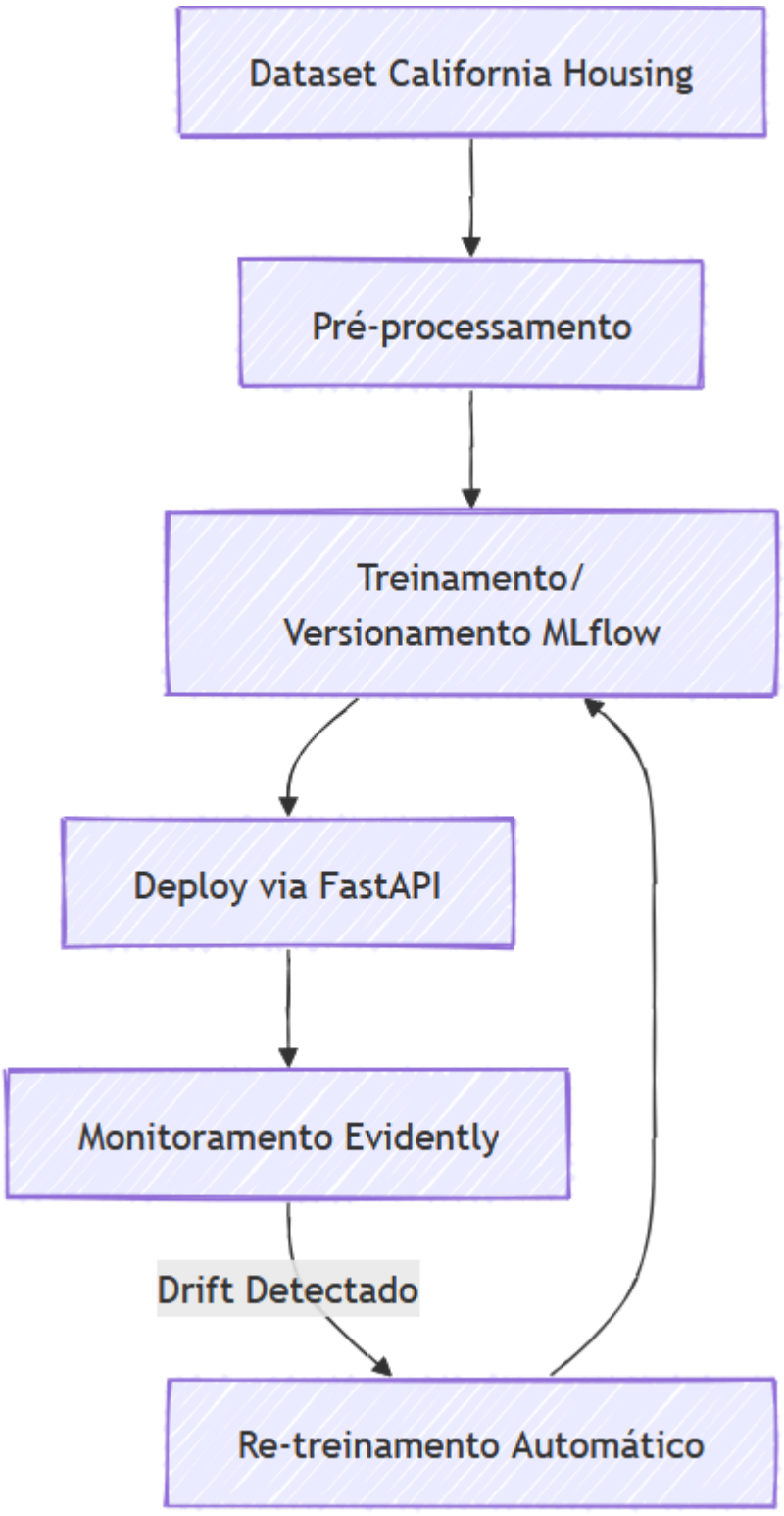
Lucas Henrique Gonçalves Wodtke

2. Introdução

Este projeto implementa um pipeline de Machine Learning seguindo práticas de MLOps, garantindo rastreabilidade, versionamento e monitoramento contínuo. O objetivo é prever preços de imóveis usando um fluxo automatizado que inclui:

- Pré-processamento
- Experimentação com MLflow
- Deploy via API (FastAPI)
- Monitoramento de drift com Evidently AI
- Re-treinamento condicional

3. Pipeline



Este pipeline automatizado cobre todo o ciclo de vida do modelo, desde os dados brutos até a manutenção em produção:

1. Dataset California Housing: Base de dados com características demográficas e geográficas de distritos da Califórnia.
2. Pré-processamento: Limpeza, normalização e divisão dos dados (treino/teste), gerando artefatos reprodutíveis.

3. Treinamento: Desenvolvimento e comparação de modelos (Regressão Linear, Random Forest, Gradient Boosting).
4. Versionamento (MLflow): Rastreamento de experimentos e armazenamento do melhor modelo.
5. Deploy (FastAPI): Disponibilização do modelo como API para integração com aplicações.
6. Monitoramento (Evidently): Detecção contínua de drift nos dados de entrada.
7. Re-treinamento Automático: Atualização do modelo quando necessário, fechando o ciclo MLOps.

4. Implementação por Etapa

4.1. Escolha do Dataset

O conjunto de dados escolhido foi o California Housing Prices, o dataset clássico para problemas de regressão. Os dados originais foram coletados a partir do censo dos Estados Unidos (U.S. Census Bureau), especificamente referentes à Califórnia. Possui as seguintes características:

- **Características**:
    - 20,640 instâncias
    - 9 features:
        - MedInc: Renda média por domicílio
        - HouseAge: Idade média das casas
        - AveRooms: Número médio de cômodos
        - AveBedrms: Número médio de quartos
        - Population: População do setor censitário
        - AveOccup: Número médio de ocupantes por domicílio
        - Latitude: Coordenada geográfica
        - Longitude: Coordenada geográfica
    - Variável alvo: MedHouseVal

- **Justificativas da Escolha:**
  A escolha do conjunto de dados California Housing foi realizada levando em conta os seguintes aspectos. Em termos de praticidade, o dataset já vem bem estruturado e bem documentado, o que reduz significativamente o tempo necessário para a análise e preparação inicial. Além disso, tem em sua origem uma fonte oficial (o censo dos EUA), assegurando que os dados sejam realistas e representem bem a realidade observada. O dataset oferece volume suficiente para treinar modelos eficazes sem exigir grandes recursos computacionais, o que ajuda para executar no ambiente com infraestrutura limitada. Considerando a volatilidade do mercado imobiliário, os modelos construídos sobre esse conjunto exigem atualizações constantes, o que está alinhado com os desafios práticos enfrentados em MLOps.Além disso, o conjunto permite demonstrar todas as etapas do ciclo de MLOps.

- **Problema Abordado**
  Este trabalho tem como objetivo implementar um pipeline de MLOps completo para automatizar o desenvolvimento, deploy e monitoramento de um modelo de predição de preços de imóveis na Califórnia. O desafio principal é criar um sistema que não apenas gera previsões, mas que também se mantenha atualizado ao longo do tempo, detectando automaticamente mudanças nos padrões dos dados (data drift) e retreinando o modelo quando necessário. A solução aborda desde o pré-processamento dos dados e treinamento dos modelos até a disponibilização via API e monitoramento contínuo, garantindo que as previsões permaneçam confiáveis em um cenário real de negócios.

4.2. Exploração e Pré-processamento

- **Script:** *load_data.py*
  Carrega o dataset California Housing Prices do Scikit-learn e o salva em formato CSV e gera dataframe para uso nas demais etapas do pipeline.

  Ferramentas:

  - scikit-learn (fetch_california_housing)
  - pandas (conversão para DataFrame e exportação CSV)

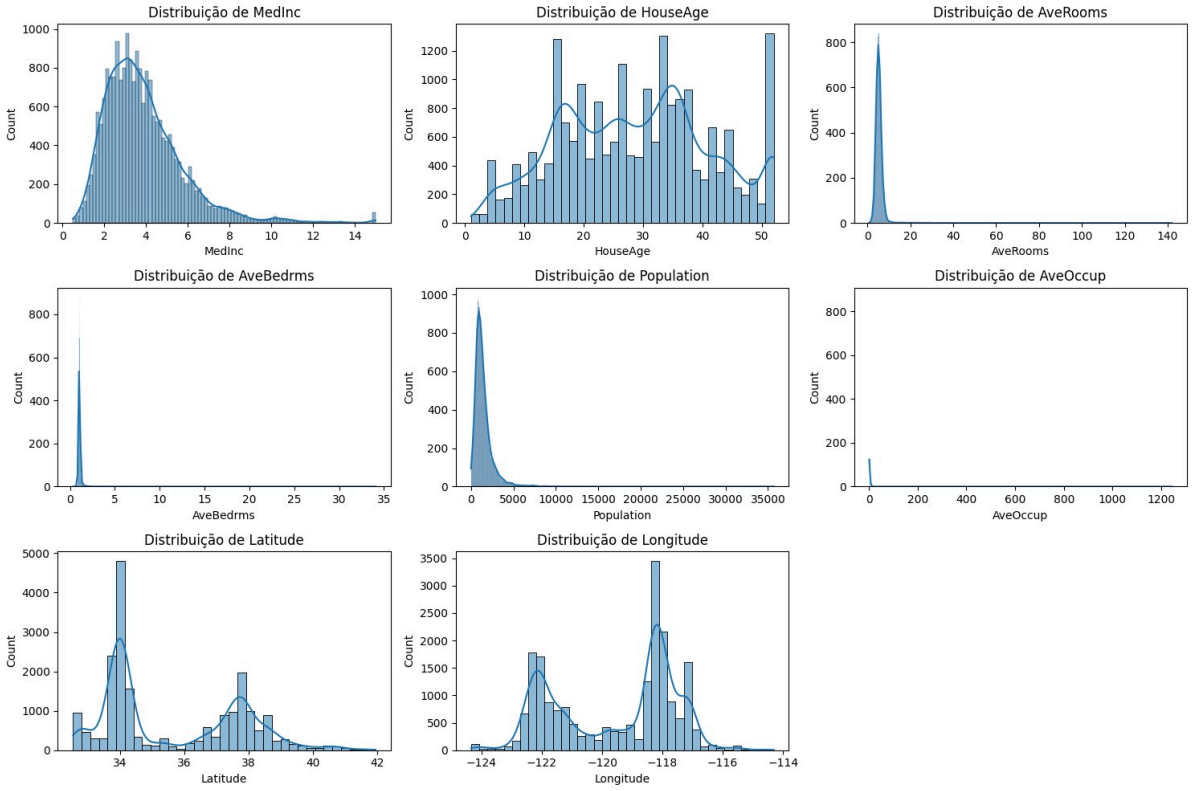
- **Script:** *preprocess.py*

  Transforma os dados brutos em features prontas para treino, com tratamento de outliers, normalização e divisão treino/teste.

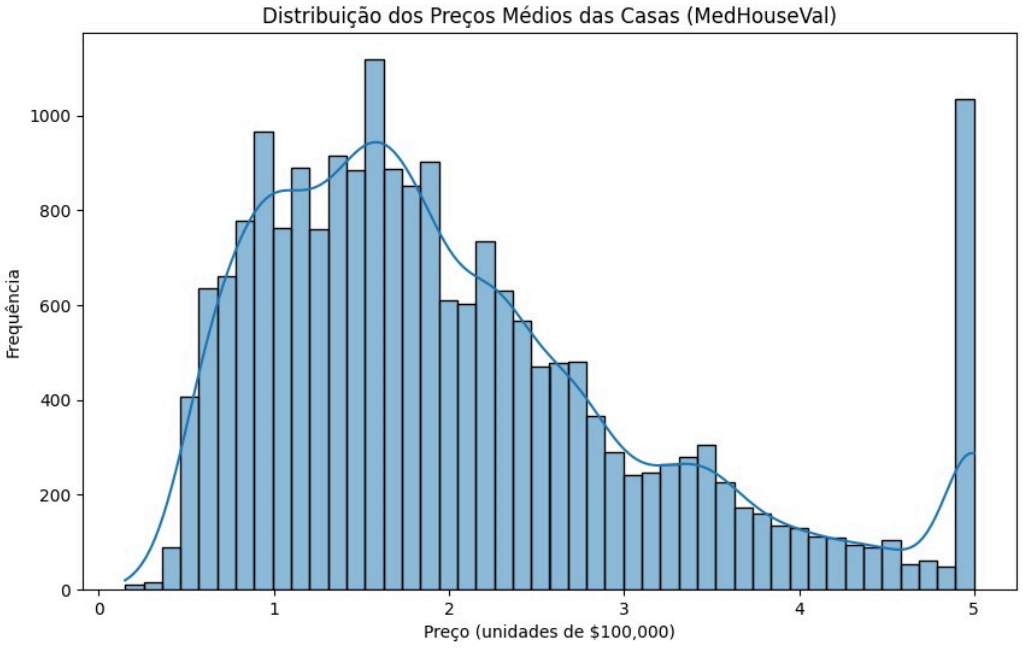
  Ferramentas:

  - pandas (manipulação dos dados)
  - scikit-learn (StandardScaler, train_test_split)
  - matplotlib/seaborn (visualizações do EDA)
  - joblib (salva artefatos: scaler.pkl, datasets processados)

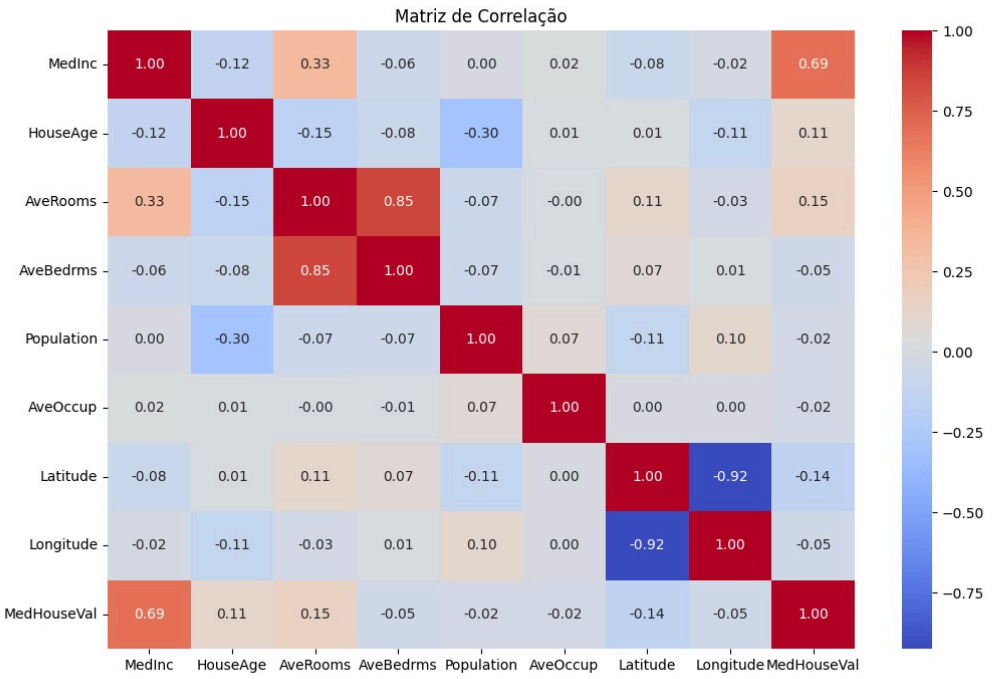
  Distribuição das Features:



Distribuição dos Preços Médios das Casas



Matriz de Correlação:



Resultados da Etapa de Exploração e Pré-processamento:

1. Análise Exploratória (EDA)

Distribuição do Target (MedHouseVal):

- Os preços das casas seguem uma distribuição assimétrica, com concentração entre 1 e 2.5 (em unidades de US$ 100.000).
- Alguns valores extremos (acima de 5) indicam possíveis outliers ou regiões de alto luxo.

Correlação entre Features:

- MedInc (Renda Média) tem a maior correlação positiva com o preço, confirmando que áreas mais ricas tendem a ter imóveis mais caros.
- Latitude e Longitude mostram padrões geográficos claros (ex.: proximidade do litoral aumenta preços).

Distribuição das Features:

- AveRooms e AveBedrms têm distribuições distorcidas (valores altos raros).
- Population e AveOccup apresentam outliers extremos (ex.: distritos com +1.000 ocupantes médios por casa).

2. Pré-processamento

Divisão Treino/Teste (80/20)

- 20.640 amostras -> 16.512 (treino) + 4.128 (teste).

Normalização (StandardScaler):

- Todas as features foram escalonadas para média 0 e desvio padrão 1, evitando que variáveis em escalas diferentes dominem o modelo.

Artefatos Gerados:

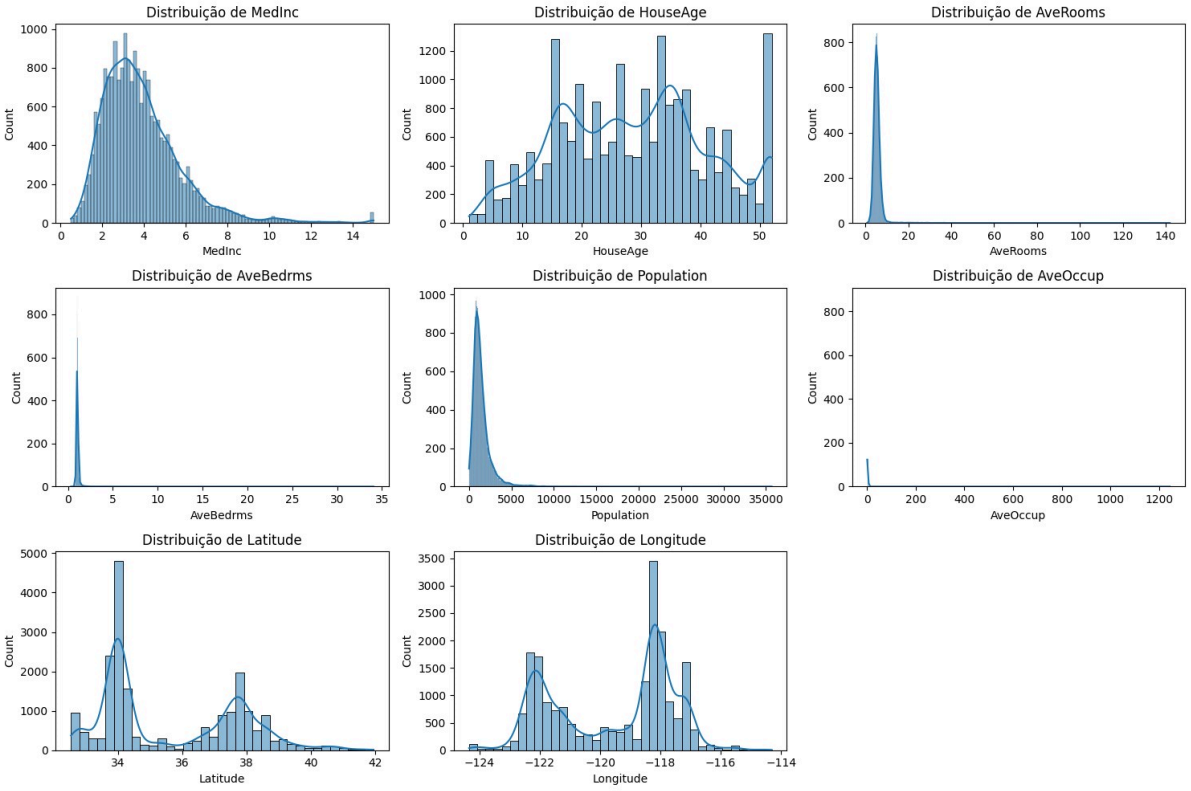
- scaler.pkl: Salva o scaler para aplicar a mesma transformação em produção.
- X_train.pkl, y_train.pkl, X_test.pkl, y_test.pkl: Datasets prontos para treino.
- Gráficos (correlation_matrix.png, features_distribution.png).

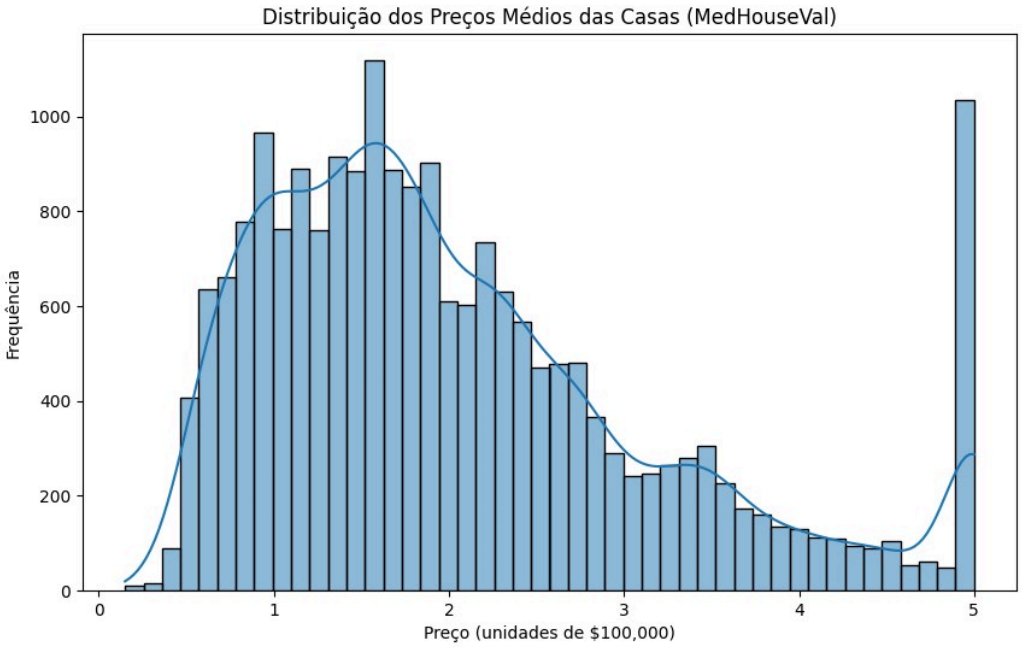
Resultados:

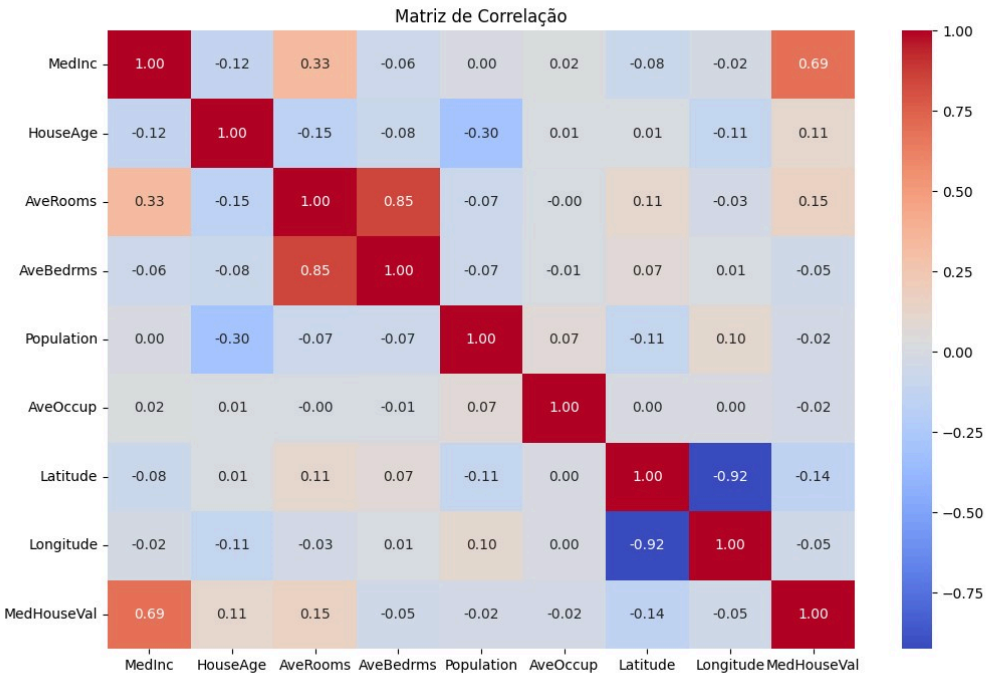
```
======================================
Análise Exploratória dos Dados (EDA)
======================================

1. Primeiras linhas do dataset:
   MedInc  HouseAge  AveRooms  AveBedrms  Population  AveOccup  Latitude  Longitude  MedHouseVal
0  8.3252      41.0  6.984127   1.023810       322.0  2.555556     37.88    -122.23        4.526
1  8.3014      21.0  6.238137   0.971880      2401.0  2.109842     37.86    -122.22        3.585
2  7.2574      52.0  8.288136   1.073446       496.0  2.802260     37.85    -122.24        3.521
3  5.6431      52.0  5.817352   1.073059       558.0  2.547945     37.85    -122.25        3.413
4  3.8462      52.0  6.281853   1.081081       565.0  2.181467     37.85    -122.25        3.422

--------------------------------------

2. Dimensões do dataset: (20640, 9)
Número de registros: 20640
Número de variáveis: 9

--------------------------------------
```

```
----------------------------------------

3. Tipos de dados e valores ausentes:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 9 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   MedInc       20640 non-null  float64
 1   HouseAge     20640 non-null  float64
 2   AveRooms     20640 non-null  float64
 3   AveBedrms    20640 non-null  float64
 4   Population   20640 non-null  float64
 5   AveOccup     20640 non-null  float64
 6   Latitude     20640 non-null  float64
 7   Longitude    20640 non-null  float64
 8   MedHouseVal  20640 non-null  float64
dtypes: float64(9)
memory usage: 1.4 MB
None

Valores ausentes por coluna:
MedInc         0
HouseAge       0
AveRooms       0
AveBedrms      0
Population     0
AveOccup       0
Latitude       0
Longitude      0
MedHouseVal    0
dtype: int64

----------------------------------------
```

```
----------------------------------------

4. Estatísticas descritivas:
              count         mean          std         min         25%         50%         75%         max
MedInc      20640.0     3.870671     1.899822    0.499900    2.563400    3.534800    4.743250   15.000100
HouseAge    20640.0    28.639486    12.585558    1.000000   18.000000   29.000000   37.000000   52.000000
AveRooms    20640.0     5.429000     2.474173    0.846154    4.440716    5.229129    6.052381  141.909091
AveBedrms   20640.0     1.096675     0.473911    0.333333    1.006079    1.048780    1.099526   34.066667
Population  20640.0  1425.476744  1132.462122    3.000000  787.000000 1166.000000 1725.000000 35682.000000
AveOccup    20640.0     3.070655    10.386050    0.692308    2.429741    2.818116    3.282261 1243.333333
Latitude    20640.0    35.631861     2.135952   32.540000   33.930000   34.260000   37.710000   41.950000
Longitude   20640.0  -119.569704     2.003532 -124.350000 -121.800000 -118.490000 -118.010000 -114.310000
MedHouseVal 20640.0     2.068558     1.153956    0.149990    1.196000    1.797000    2.647250    5.000010

----------------------------------------

5. Gráfico de distribuição do target salvo como 'target_distribution.png'
----------------------------------------

6. Matriz de correlação salva como 'correlation_matrix.png'
----------------------------------------

7. Distribuição das features salva como 'features_distribution.png'
----------------------------------------
```


Distribuição de MedInc
MedInc
Distribuição de HouseAge
HouseAge
Distribuição de AveRooms
AveRooms
Distribuição de AveBedrms
AveBedrms
Distribuição de Population
Population
Distribuição de AveOccup
AveOccup
Distribuição de Latitude
Latitude
Distribuição de Longitude
Longitude
Count


Distribuição dos Preços Médios das Casas (MedHouseVal)
Frequência
Preço (unidades de $100,000)




```
========================================
Iniciando Pré-processamento
========================================

Pré-processamento concluído! ✅
Artefatos salvos:
- X_train.pkl
- X_test.pkl
- y_train.pkl
- y_test.pkl
- scaler.pkl
```

4.3. Treinamento e Versionamento

- **Script**: *train.py*

  Treina múltiplos modelos de machine learning, compara seu desempenho.

  Ferramentas:

  - scikit-learn (modelos de regressão)
  - mlflow (rastreamento de experimentos e model registry)
  - joblib (carregamento dos dados pré-processados)

  Experiment Tracking

- Registra todas as execuções de treinamento no banco de dados SQLite.
- Registra métricas, parâmetros e artefatos de cada modelo no MLflow.
- Compara algoritmos de regressão.

Reprodutibilidade

- Usa os mesmos dados pré-processados (X_train.pkl, y_train.pkl) para todos os modelos.
- Fixa random_state nos modelos baseados em árvores para garantir resultados consistentes.

Model Registry

- Versiona modelos como um catálogo organizado (ex.: staging, production).

Fluxo:

- Modelo treinado -> Registrado como "None" (versão inicial).
- Validação manual/automática -> Promovido para "Staging".
- Testes em produção -> Promovido para "Production" (via *promote.py*).

Funcionamento Detalhado:

Modelos Implementados:

- Regressão Linear;
- Random Forest;
- Gradient Boosting.

Métricas-Chave:

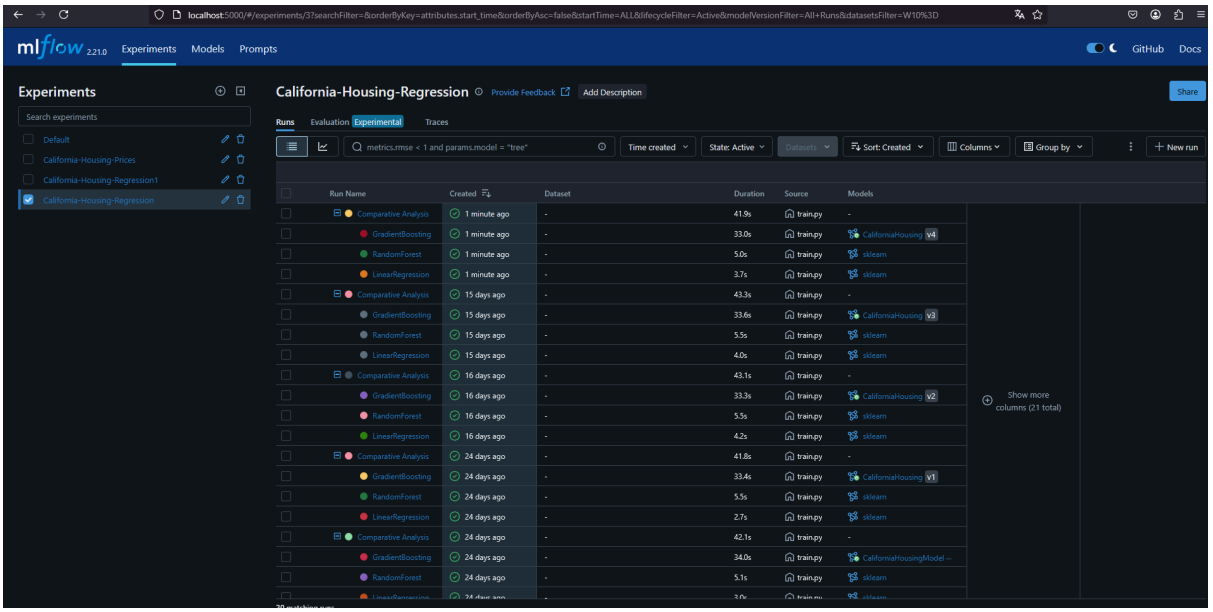
- R^2 (Coeficiente de Determinação): Mede a proporção da variância explicada.
- RMSE (Erro Quadrático Médio): Indica o erro médio em dólares.

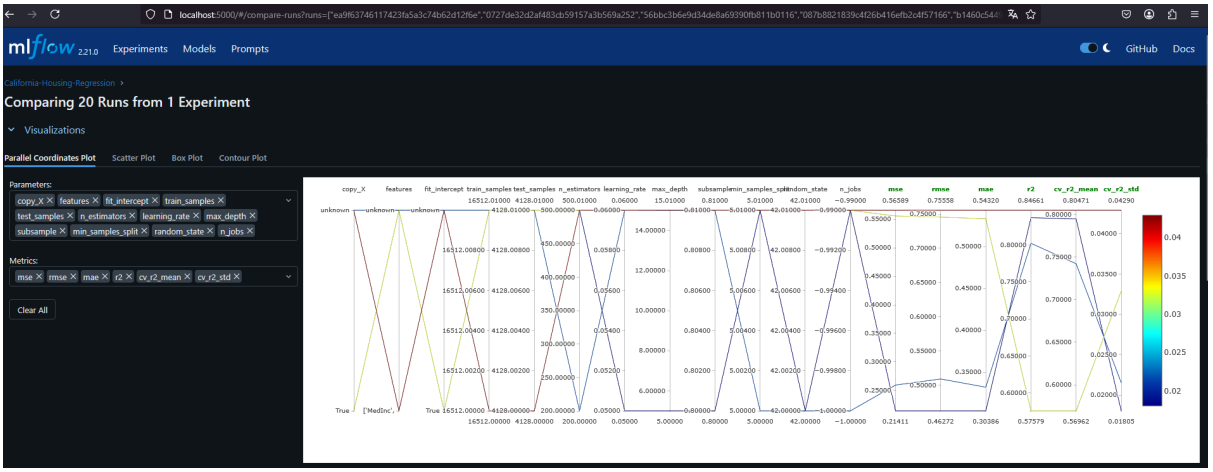
Saídas Importantes:

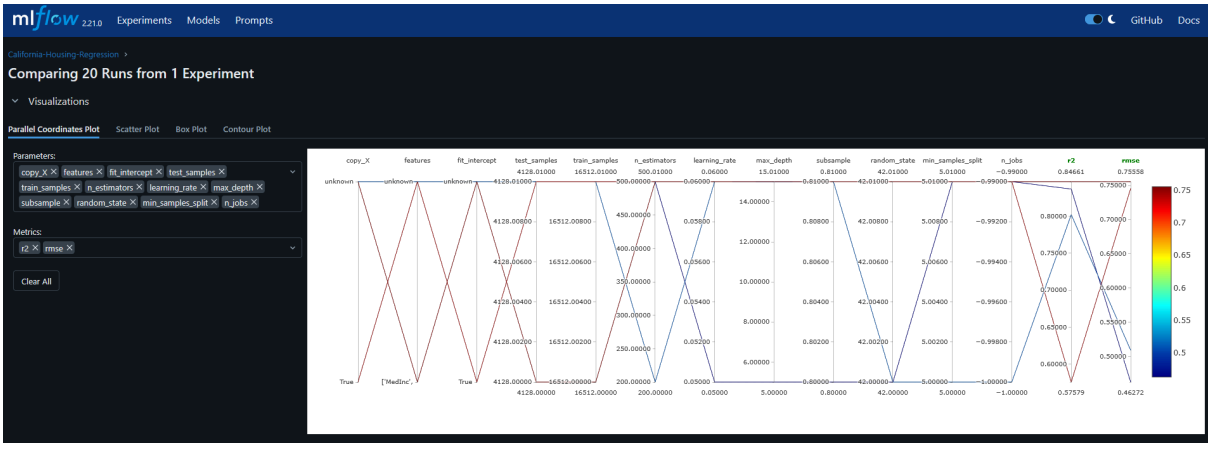
- Modelo registrado no MLflow (models:/CaliforniaHousingModel/).
- Logs de desempenho para cada execução (acessíveis via UI do MLflow).

Resultados:

```
(venv) PS C:\Users\lucas\Desktop\MLOps_California> python train.py
2025/04/19 14:58:34 WARNING mlflow.models.model: Model logged without a signature and input example. Please set `input_example` parameter when logging the model to auto infer the model signature.
View run LinearRegression at: http://localhost:5000/#/experiments/3/runs/087b8821839c4f26b416efb2c4f57166
View experiment at: http://localhost:5000/#/experiments/3
2025/04/19 14:58:39 WARNING mlflow.models.model: Model logged without a signature and input example. Please set `input_example` parameter when logging the model to auto infer the model signature.
View run RandomForest at: http://localhost:5000/#/experiments/3/runs/56bbc3b6e9d34de8a69390fb811b0116
View experiment at: http://localhost:5000/#/experiments/3
2025/04/19 14:59:13 WARNING mlflow.models.model: Model logged without a signature and input example. Please set `input_example` parameter when logging the model to auto infer the model signature.
Registered model 'CaliforniaHousing' already exists. Creating a new version of this model...
2025/04/19 14:59:13 INFO mlflow.store.model_registry.abstract_store: Waiting up to 300 seconds for model version to finish creation. Model name: CaliforniaHousing, version 4
Created version '4' of model 'CaliforniaHousing'.
View run GradientBoosting at: http://localhost:5000/#/experiments/3/runs/0727de32d2af483cb59157a3b569a252
View experiment at: http://localhost:5000/#/experiments/3
View run Comparative Analysis at: http://localhost:5000/#/experiments/3/runs/ea9f63746117423fa5a3c74b62d12f6e
View experiment at: http://localhost:5000/#/experiments/3
Treinamento concluído! Acesse o MLflow UI: http://localhost:5000
(venv) PS C:\Users\lucas\Desktop\MLOps_California>
```







- **Script** *promote.py*
  Gerencia o ciclo de vida dos modelos no MLflow Model Registry, promovendo a melhor versão para produção com base em métricas pré-definidas.

  Ferramentas:

  - mlflow (Client para interagir com o Model Registry)
  - sqlite (backend do MLflow para armazenar metadados)

  Model Registry

  - Define critérios claros para promoção ($R^2 > 0.75$).
  - Compara versões de modelos registrados para selecionar o "Champion".

  Governança de Modelos

  - Garante que apenas modelos validados cheguem à produção.
  - Mantém histórico completo de todas as versões.

  Funcionamento Detalhado:

  Critérios de Promoção:

  - Staging: $R^2 > 0.7$ (threshold mínimo para considerar o modelo).
  - Production: $R^2 > 0.75$ (e selecionar o melhor modelo disponível).

  Fluxo de Decisão:

  - Busca todas as versões registradas do modelo CaliforniaHousingModel.
  - Compara métricas de cada versão (prioriza R^2, depois RMSE).
  - Promove automaticamente a versão com melhor desempenho para Production.

  Recursos-Chave:

  - Fallback Seguro: Se nenhum modelo atingir o threshold, mantém a versão atual em produção.

  Saídas Importantes:

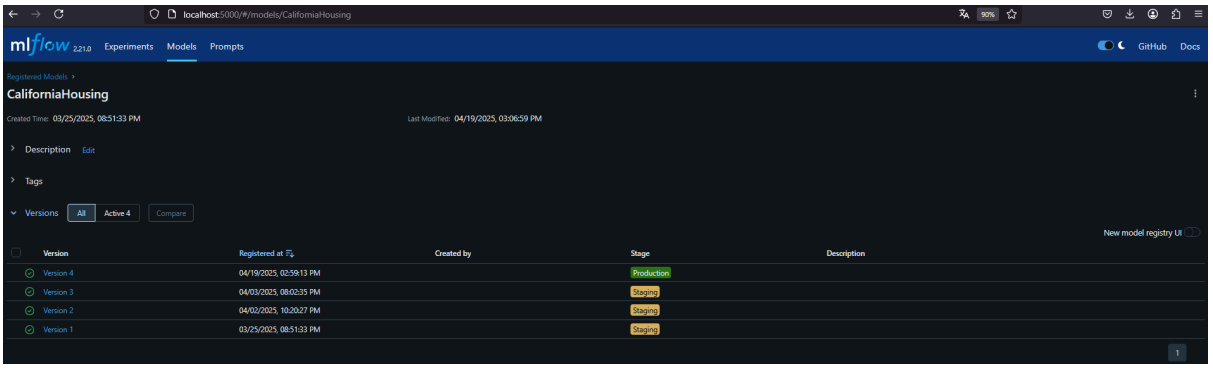
  - Modelo em Produção Atualizado: Acessível via models:/CaliforniaHousingModel/Production.

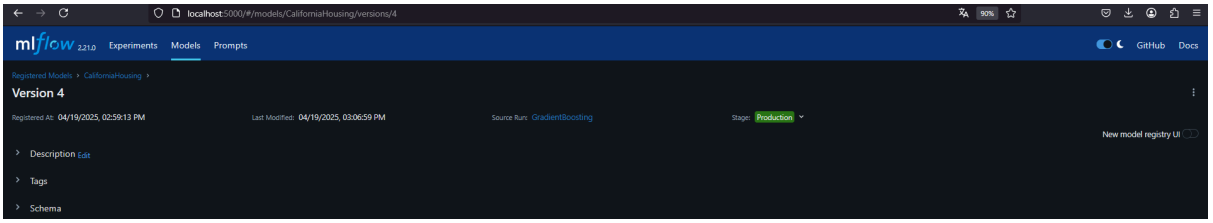
- Lista final com status de todas as versões .

Integração com o Pipeline:

- Entrada: Modelos treinados pelo train.py (registrados no MLflow).
- Saída: Modelo em produção consumido pelo *app.py* e monitorado pelo *monitor.py*.Saídas Importantes:

Resultados:





4.4. Deploy como API

- **Script**: *app.py*

  Disponibiliza o modelo em produção como uma API REST, permitindo previsões em tempo real através de requisições HTTP.

  Ferramentas:

  - FastAPI: Framework para construção da API REST.
  - MLflow: Carregamento do modelo em produção do Model Registry.
  - Joblib: Carregamento do scaler pré-treinado (scaler.pkl).

Funcionamento Detalhado:

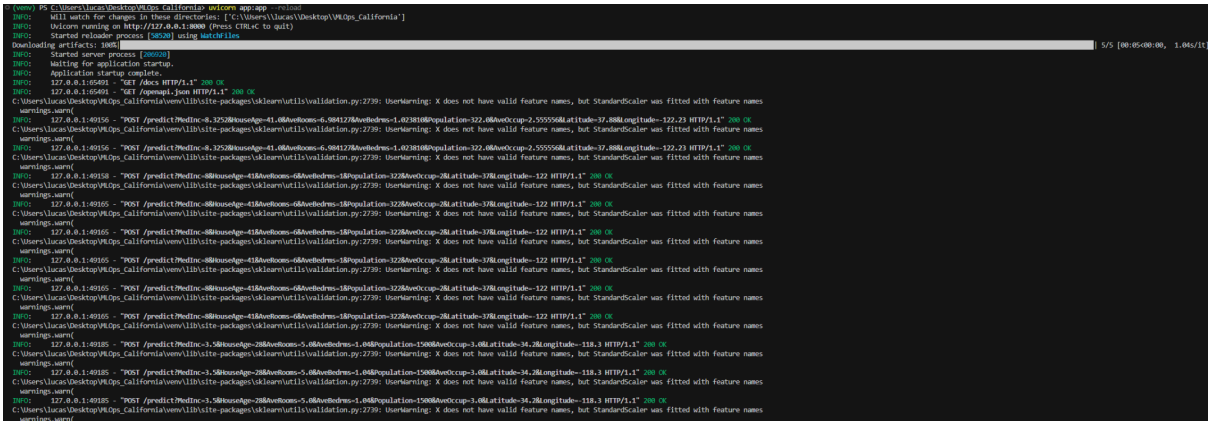
Inicialização:

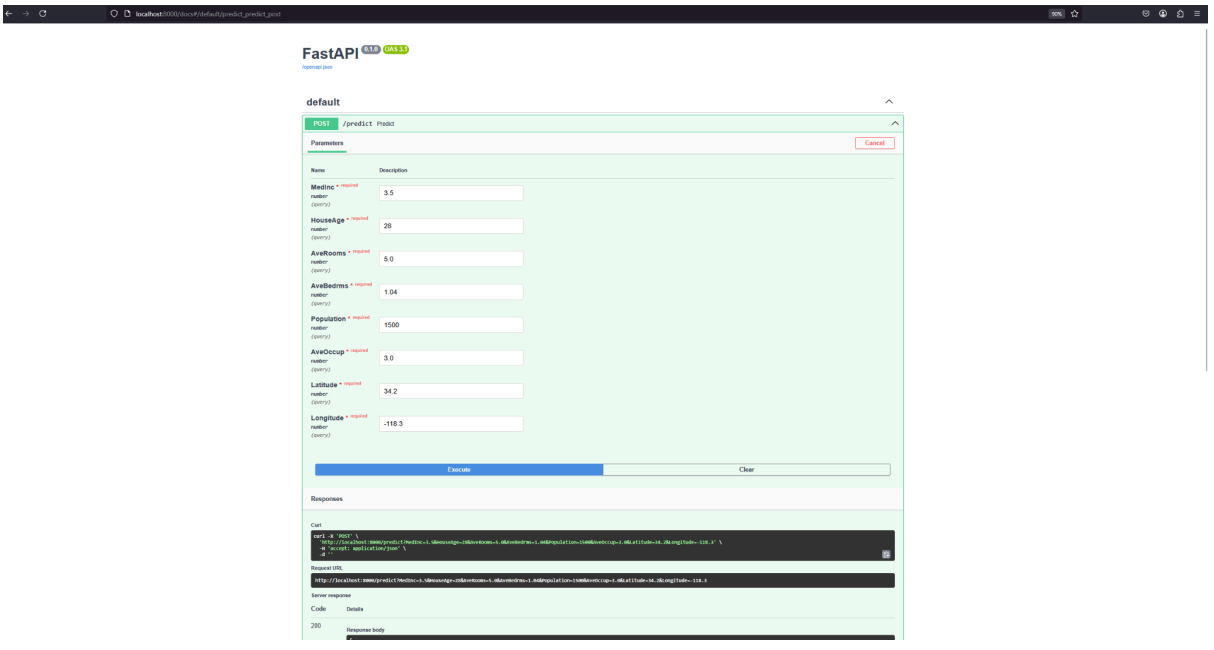
- Carrega o scaler e o modelo do MLflow.

Requisição de Predição:

- Recebe 8 features numéricas via POST.
- Pré-processa os dados com o scaler salvo.
- Retorna o preço previsto em formato JSON.

Resultados:





Curl

```
curl -X 'POST' \
  'http://localhost:8000/predict?MedInc=3.5&HouseAge=28&AveRooms=5.0&AveBedrms=1.04&Population=1500&AveOccup=3.0&Latitude=34.2&Longitude=-118.3' \
  -H 'accept: application/json' \
  -d ''
```

Request URL

```
http://localhost:8000/predict?MedInc=3.5&HouseAge=28&AveRooms=5.0&AveBedrms=1.04&Population=1500&AveOccup=3.0&Latitude=34.2&Longitude=-118.3
```

Server response

Code	Details
200	Response body

```
{
  "previsao": 2.1049099999999994
}
```

Download

Response headers

```
content-length: 31
content-type: application/json
date: Sat,19 Apr 2025 18:16:09 GMT
server: uvicorn
```

Responses

Code	Description	Links
200	Successful Response	*No links*

Media type: application/json

Controls Accept header.

Example Value | Schema

```
"string"
```

Code	Description	Links
422	Validation Error	*No links*

Media type: application/json

Example Value | Schema

```
{
  "detail": [
    {
      "loc": [
        "string",
        0
      ],
      "msg": "string",
      "type": "string"
    }
  ]
}
```

FastAPI 0.1.0 OAS 3.1

/openapi.json

default

POST `/predict` Predict

Parameters — Cancel

Name	Description
MedInc * required number (query)	3.5
HouseAge * required number (query)	15.0
AveRooms * required number (query)	3.5
AveBedrms * required number (query)	1.2
Population * required number (query)	1200.0
AveOccup * required number (query)	2.8
Latitude * required number (query)	34.05
Longitude * required number (query)	-118.25

Execute | Clear

Responses

Curl

```
curl -X 'POST' \
  'http://localhost:8000/predict?MedInc=3.5&HouseAge=15.0&AveRooms=3.5&AveBedrms=1.2&Population=1200.0&AveOccup=2.8&Latitude=34.05&Longitude=-118.25' \
  -H 'accept: application/json' \
  -d ''
```

Request URL

```
http://localhost:8000/predict?MedInc=3.5&HouseAge=15.0&AveRooms=3.5&AveBedrms=1.2&Population=1200.0&AveOccup=2.8&Latitude=34.05&Longitude=-118.25
```

Server response

Code	Details
200	Response body

```
{
  "previsao": 1.9830399999999992
}
```

Download

Response headers

```
content-length: 31
content-type: application/json
date: Sat,19 Apr 2025 18:25:24 GMT
server: uvicorn
```

Responses

Code	Description	Links

```
WARNING:waitress.queue:Task queue depth is 2
WARNING:waitress.queue:Task queue depth is 2
WARNING:waitress.queue:Task queue depth is 2
WARNING:waitress.queue:Task queue depth is 2
WARNING:waitress.queue:Task queue depth is 2
WARNING:waitress.queue:Task queue depth is 2
WARNING:waitress.queue:Task queue depth is 1
WARNING:waitress.queue:Task queue depth is 2
WARNING:waitress.queue:Task queue depth is 3
WARNING:waitress.queue:Task queue depth is 3
WARNING:waitress.queue:Task queue depth is 4
WARNING:waitress.queue:Task queue depth is 5
WARNING:waitress.queue:Task queue depth is 5
WARNING:waitress.queue:Task queue depth is 5
WARNING:waitress.queue:Task queue depth is 5
WARNING:waitress.queue:Task queue depth is 5
WARNING:waitress.queue:Task queue depth is 5
WARNING:waitress.queue:Task queue depth is 4
WARNING:waitress.queue:Task queue depth is 5
WARNING:waitress.queue:Task queue depth is 5
WARNING:waitress.queue:Task queue depth is 5
WARNING:waitress.queue:Task queue depth is 5
WARNING:waitress.queue:Task queue depth is 5
WARNING:waitress.queue:Task queue depth is 1
WARNING:waitress.queue:Task queue depth is 2
WARNING:waitress.queue:Task queue depth is 2
WARNING:waitress.queue:Task queue depth is 2
WARNING:waitress.queue:Task queue depth is 2
WARNING:waitress.queue:Task queue depth is 2
WARNING:waitress.queue:Task queue depth is 1
WARNING:waitress.queue:Task queue depth is 1
WARNING:waitress.queue:Task queue depth is 2
WARNING:waitress.queue:Task queue depth is 2
WARNING:waitress.queue:Task queue depth is 2
WARNING:waitress.queue:Task queue depth is 2
WARNING:waitress.queue:Task queue depth is 2
WARNING:waitress.queue:Task queue depth is 2
WARNING:waitress.queue:Task queue depth is 1
Downloading artifacts: 100%|| 1/1 [00:00<00:00, 18.01it/s]
WARNING:waitress.queue:Task queue depth is 1
Downloading artifacts: 100%|| 1/1 [00:00<00:00, 249.93it/s]
Downloading artifacts: 100%|| 1/1 [00:00<00:00, 100.01it/s]
Downloading artifacts: 100%|| 1/1 [00:00<00:00, 166.63it/s]
Downloading artifacts: 100%|| 1/1 [00:00<00:00, 82.67it/s]
Downloading artifacts: 100%|| 1/1 [00:00<00:00, 1000.31it/s]
Downloading artifacts: 100%|| 1/1 [00:00<00:00, 500.22it/s]
Downloading artifacts: 100%|| 1/1 [00:00<00:00, 999.12it/s]
Downloading artifacts: 100%|| 1/1 [00:00<00:00, 999.12it/s]
Downloading artifacts: 100%|| 1/1 [00:00<00:00, 24.62it/s]
```

4.5. Monitoramento e Re-treinamento

- **Script** *monitor_aut.py*
  Gerencia o ciclo de vida dos modelos no MLflow Model Registry, promovendo a melhor versão para produção com base em métricas pré-definidas.

  Ferramentas:

  - Evidently AI (detecção de drift e relatórios de desempenho)
  - Requests (integração com a API de predição)Evidently AI (detecção de drift e relatórios de desempenho)Evidently AI (detecção de drift e relatórios de desempenho)
  - Joblib (carregamento de dados de referência)
  - Scikit-learn (métricas de regressão)

    Data Drift:

    - Monitora mudanças na distribuição das features (ex.: renda média, idade das casas).
    - Compara dados atuais vs. dados de treino (referência).

    Governança de Modelos

    - Gatilho automático para re-treinamento quando drift excede um threshold (RMSE > 0.5).
    - Mantém modelos alinhados com a realidade dos dados em produção.Scikit-learn (métricas de regressão)

    Feedback Loop:

  - Conecta monitoramento -> re-treinamento -> atualização do modelo em produção.

  Funcionamento Detalhado:

    Coleta de Dados:

    - Data Drift Report.
    - Regression Report.

    Decisão de Re-treinamento:

    - Se RMSE > threshold ou drift score > 0.2.

    Métricas-Chave:

    -

Métrica	Descrição	Threshold
Data Drift Score	Porcentagem de features com distribuição alterada	> 20%
RMSE em Produção	Erro médio nas previsões atuais	> 0.5
R^2 em Produção	Aderência do modelo aos dados recentes	< 0.6

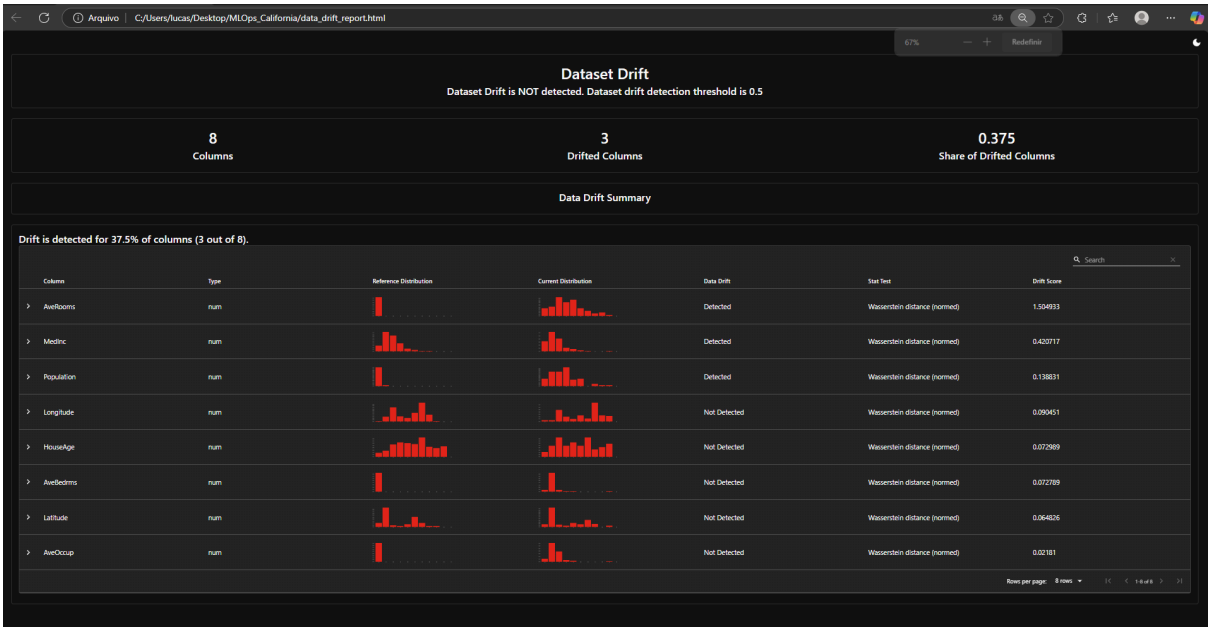
Recursos-Chave:

- Simulação de Drift: current_data['MedInc'] *= 1.2 # Altera renda média para testar sensibilidade
- Fallback Seguro: Mantém o modelo em produção se o novo modelo treinado não atingir o threshold.

Saídas Importantes:

- Relatório HTML: data_drift_report.html (gráficos interativos de drift e desempenho).

Resultados:
(Por conta de problema na execução das requisições, estava levando muito tempo para finalizar e gerar o relatório foi utilizado o *monitor.py*, versão simplificada para demonstrar o uso do evidently.



4.6. Conteinerização e Documentação

- **Scripts:** README.md, requirements.txt

Conteinerização (Não Implementada)

Contexto: O projeto foi executado localmente sem uso de Docker.

Justificativa:

- **Escopo do Projeto**: Foco na validação do pipeline MLOps em ambiente controlado (local).
- **Complexidade Reduzida**: Conseguir mais tempo para viabilizar as demais etapas do pipeline.

5. Conclusão

Concluindo sobre os aspectos do andamento do trabalho:

1. Pipeline Completo (buscando realizar de maneira simplificada e objetiva):
    - Implementei as etapas do fluxo MLOps.
    - Ferramentas como MLflow e FastAPI simplificaram tarefas complexas (ex.: versionamento de modelos).
2. Priorização do Essencial para viabilização do pipeline na prática:
        - API simples para previsões.

        - Monitoramento de drift com Evidently.
        - Re-treinamento manual (sem automação total).
3. Aprendizado Prático:
    - Entendi como os modelos entram em produção e por que o monitoramento é crítico.
    - Vi que MLOps não exige ferramentas perfeitas, mas sim um fluxo funcional.

Onde Melhorar no Futuro:

Assunto	Ação	Impacto
Conteinerização	Usar Docker para isolar ambiente (API, MLflow).	Facilita deploy em qualquer máquina/nuvem.
Mais Métricas	Adicionar mais métricas nos experimentos e em tempo real na API.	Melhora a visibilidade do desempenho.
Novos Modelos	Testar mais modelos e algoritmos de machine learning	Aumenta a precisão e a robustez.
Experimentos	Realizar mais experimentos e testar mais parâmetros.	Define critérios mais realistas para definição do melhor modelo e re-treino.
Monitoramento	Verificar comportamento do drift sem simular	Auxilia a verificar comportamento do pipeline de maneira mais realista e verificar necessidade de melhorias

Melhoria Importante: Evitar Drift Simulado

- **Problema**: No código atual, monitor.py simula drift alterando manualmente features como MedInc e AveRooms.
- **Risco**: Isso não reflete a realidade, pois drift em produção ocorre naturalmente (ex.: mudanças econômicas).
- **Solução Futura**:
    - Coletar dados reais em produção (ex.: logs de predições).
    - Usar um serviço de armazenamento para comparar dados atuais vs. treino.

Por Que o Escopo Foi Reduzido?

- **Limitações Técnicas**: Falta de experiência com programação e automação.
- **Tempo Curto**: Optei por um MVP funcional em vez de um projeto complexo e inacabado.
- **Foco na Prática**: Um fluxo básico, mas executável.

6. Referências Técnicas

Dataset:

California Housing Prices:

- **Fonte Oficial**: Scikit-learn Datasets
- **Descrição**: Dataset clássico para regressão, contendo características demográficas e geográficas de regiões da Califórnia.

Ferramentas e Bibliotecas:

MLflow:

- **Documentação**: mlflow.org/docs/latest/index.html
- **Uso no Projeto**: Rastreamento de experimentos, versionamento de modelos e Model Registry.

Evidently AI:

Documentação: docs.evidentlyai.com

Uso no Projeto: Monitoramento de *data drift* e métricas de desempenho em produção.

FastAPI:

- **Documentação**: fastapi.tiangolo.com
- **Uso no Projeto**: Criação da API para servir previsões do modelo.

Scikit-learn:

- **Documentação**: scikit-learn.org/stable/documentation.html
- **Uso no Projeto**: Implementação de modelos (Regressão Linear, Random Forest) e pré-processamento.

Código-Fonte:

Repositório GitHub:

- **Link**: https://github.com/lucaswodtke/Mlops
- **Conteúdo**:
  - Relatório
  - README.md
  - mermaid-pipeline.png
  - requirements.txt
  - Scripts