Documentação Técnica — Decision Recrutamento AI

# Visão Geral

Este projeto foi desenvolvido para o Datathon Decision - Pós Tech FIAP, com o objetivo de construir uma solução de Inteligência Artificial aplicada ao Recrutamento e Seleção de profissionais de TI. A aplicação integra técnicas de Machine Learning supervisionado, similaridade semântica com embeddings, e uma API robusta desenvolvida com FastAPI, permitindo que a empresa Decision otimize o processo de triagem, entrevistas e contratação de talentos.

# Problemas Enfrentados pela Decision

- Falta de padronização nas entrevistas

- Dificuldade em medir aderência técnica dos candidatos

- Longo tempo de triagem de perfis

- Match impreciso entre candidatos e vagas

# Objetivo da Solução

- Automatizar a triagem de currículos com alto grau de acurácia

- Gerar um ranking de compatibilidade técnica entre candidatos e vagas

- Reduzir o tempo do processo seletivo

- Apoiar a tomada de decisão com base em dados e métricas

# Arquitetura da Solução

**Pipeline Inteligente de Recrutamento:**

1. Data Collection: Leitura de grandes volumes de dados históricos (JSONs de candidatos e vagas).

2. Pré-processamento: Padronização de dados e criação de dataset supervisionado (match binário).

3. Engenharia de Features: TF-IDF, codificação categórica e embeddings com SBERT.

4. Modelo Supervisionado: XGBoost com SMOTE, ajustes de threshold e parâmetros.

5. Geração de Relatório: Métricas de avaliação salvas em HTML.

6. API com FastAPI: Endpoints robustos para classificação e ranking.

7. Interface Streamlit: Visualização e testes pelo time de RH.

# Tecnologias Utilizadas

|  |  |
| --- | --- |
| Categoria | Ferramentas |
| Backend/API | FastAPI, Uvicorn |
| Machine Learning | XGBoost, Scikit-learn, imbalanced-learn |
| Similaridade Semântica | SentenceTransformers (SBERT) |
| Deploy/Container | Docker, GitHub Actions |
| Logging & Relatórios | logging, HTML + Jinja2 |
| Frontend | Streamlit |
| Testes | Pytest |

# Estrutura do Projeto

fiap-posmlops-fase5-datathon-decision

├── .github/

│   └── workflows/

│       └── pipeline.yaml

│

├── backend/

│   ├── app/

│   │   ├── etl/

│   │   │   ├── backblaze\_loader.py

│   │   │   ├── data\_collector.py

│   │   │   └── data\_preprocessing.py

│   │   │

│   │   ├── model/

│   │   │   ├── feature\_engineering.py

│   │   │   ├── model\_evaluation.py

│   │   │   └── model\_training.py

│   │   │

│   │   ├── report/

│   │   │   └── metric\_report.py

│   │   │

│   │   ├── semantic/

│   │   │   ├── semantic\_api\_matcher.py

│   │   │   ├── semantic\_dataset\_builder.py

│   │   │   └── semantic\_matcher.py

│   │   │

│   │   ├── utils/

│   │   │   ├── constants.py

│   │   │   ├── model\_utils.py

│   │   │   └── logging\_config.py

│   │   │

│   │   └── api.py

│   │

│   ├── data/

│   │   ├── dataset\_processado.csv

│   │   └── features\_treinamento.pkl

│   │

│   ├── metric\_reports/

│   │   └── model\_metric\_report\_<timestamp>.html

│   │

│   ├── models/

│   │   ├── model.pkl

│   │   └── feature\_pipeline.pkl

│   │

│   ├── tests/

│   │   ├── test\_api.py

│   │   ├── test\_data\_preprocessing.py

│   │   ├── test\_feature\_engineering.py

│   │   ├── test\_model\_training.py

│   │   └── \_\_init\_\_.py

│   │

│   ├── Dockerfile

│   ├── Procfile

│   ├── requirements.txt

│   └── main.py

│

├── README.md

├── .gitignore

└── Documetacao\_API\_Decision\_AI.pdf

Para mais detalhes, consulte o README.md do projeto.