
POS TECH

PÓS TECH - DATATHON

Olá, estudante! 😊

Chegou o momento de colocar em prática todo o aprendizado adquirido ao longo do curso para desenvolver as habilidades aprendidas até aqui solucionando uma dor real de uma empresa. O tema central deste Datathon será a **aplicação de Inteligência Artificial em processos de recrutamento e seleção**, utilizando como base o estudo de caso da empresa Decision, que atua no setor de bodyshop e possui foco em alocar talentos ideais para os clientes de forma eficiente.

Contexto

A Decision é especializada em serviços de bodyshop e recrutamento, com foco em conectar talentos qualificados às necessidades específicas dos clientes. A Decision atua principalmente no setor de TI, em que a agilidade e a precisão no "match" entre candidatos e vagas são diferenciais essenciais. O objetivo da empresa é entregar profissionais que não apenas atendam aos requisitos técnicos, mas também se alinhem à cultura e aos valores das empresas contratantes.

A Decision desempenha um papel fundamental no processo de recrutamento ao combinar tecnologia e expertise humana para identificar os melhores talentos. O foco da empresa é garantir que cada decisão de contratação seja baseada em dados concretos e análises profundas.

Como funciona o processo atual da Decision?

O time de hunters busca candidatos(as) utilizando a base interna da empresa e plataformas como LinkedIn, InfoJobs, Catho e grupos de WhatsApp, entre outros. Um grande desafio da empresa é encontrar o candidato ou candidata ideal para cada vaga em tempo hábil. Hoje a Decision possui algumas dores, como:

- Falta de padronização em entrevistas, o que pode gerar perda de informações valiosas.
- Dificuldade em identificar o real engajamento dos candidatos e candidatas.

Embora as entrevistas demandem tempo, elas são cruciais para entender o alinhamento do(a) candidato(a) com a vaga e o nível de comprometimento. Muitas

vezes, hunters pulam essa etapa para agilizar o processo, o que pode comprometer a qualidade da seleção.

Fatores que tornam a entrevista parte essencial do processo de match

- **Análise Técnica:** verificamos se o candidato ou candidata possui as habilidades e conhecimentos exigidos pela vaga (ex.: linguagens de programação como Java, Python etc.).
- **Fit Cultural:** avaliamos se o candidato ou candidata se encaixa nos valores e cultura da empresa contratante.
- **Engajamento e Motivação:** identificamos se o candidato ou candidata está realmente interessado na vaga e motivado para assumir o desafio.

Você, como engenheiro(a) de machine learning, tem o desafio de propor uma inteligência artificial para solucionar as principais dores da Decision, criando um algoritmo e disponibilizando-o de forma produtiva. O objetivo é desenvolver soluções criativas e viáveis usando IA para melhorar o processo de recrutamento. Você pode pensar em alguns cases, como por exemplo:

- Melhoria de CV com IA.
- Vaga otimizada com IA.
- Bot de entrevista (escrita/falada) com IA.
- Entrevista no app por vídeo com IA.
- Análise pós-entrevista com IA.
- Agendamento de entrevista com IA.
- Provas de vaga integrada no aplicativo com IA.

Essas são algumas opções propostas pela Decision, mas você pode surpreender na solução e utilizar a criatividade para resolver esse problema 😉.

Lembrando que o foco principal dessa solução é o **desenvolvimento do deployment do modelo**; então, sua entrega deve atender os seguintes requisitos:

- **Treinamento do modelo preditivo:** crie uma pipeline completa para treinamento do modelo, considerando feature engineering, pré-processamento, treinamento e validação. Salve o modelo utilizando pickle

ou joblib para posterior utilização na API. Deixe claro qual é a métrica utilizada na avaliação do modelo, descrevendo por que esse modelo é confiável para ser colocado em produção.

- **Modularização do código:** organize o projeto em arquivos .py separados, mantendo o código limpo e de fácil manutenção. Separe funções de pré-processamento, engenharia de atributos, treinamento, avaliação e utilitários em módulos distintos para facilitar reuso e testes.
- **Crie uma API para deployment do modelo:** crie uma API utilizando Flask ou FastAPI e implemente um endpoint /predict para receber dados e retornar previsões do modelo. Teste a API localmente utilizando Postman ou cURL para validar seu funcionamento.
- **Realize o empacotamento do modelo com Docker:** crie um Dockerfile para empacotar a API e todas as dependências necessárias. Isso garante que o modelo possa ser executado em qualquer ambiente de maneira isolada e replicável.
- **Deploy do modelo:** realize o deploy do modelo localmente ou na nuvem. Caso utilize um serviço de nuvem, você pode optar por AWS, Google Cloud Run, Heroku ou a plataforma de sua preferência.
- **Teste da API:** teste a API para validar sua funcionalidade.
- **Testes unitários:** implemente os testes unitários para verificar o funcionamento correto de cada componente da pipeline, garantindo que seu código tenha maior qualidade (80% de cobertura mínima de testes unitários).
- **Monitoramento Contínuo**: configure logs para monitoramento e disponibilize um painel para o acompanhamento de drift no modelo.
- **Documentação:** sua documentação deve conter os seguintes requisitos:

1) Visão Geral do Projeto

Objetivo: descrever de forma clara o problema de negócio que o modelo resolve (ex.: match de vagas, otimização de entrevista etc.).

Solução Proposta: construção de uma pipeline completa de machine learning, desde o pré-processamento até o deploy do modelo em produção via API.

Stack Tecnológica:

- Linguagem: Python 3.X.
- Frameworks de ML: scikit-learn, pandas, numpy.
- API: Flask ou FastAPI.
- Serialização: pickle ou joblib.
- Testes: pytest.
- Empacotamento: Docker.
- Deploy: Local/Cloud (Heroku, AWS, GCP etc.).
- Monitoramento: logging + dashboard de drift (se implementado).

2) Estrutura do Projeto (Diretórios e Arquivos)

A seguir deixamos um exemplo:

```bash
project-root/
│
├── app/                        # Código da API
│   ├── main.py                 # Arquivo principal da API
│   ├── routes.py               # Rotas e endpoints
│   └── model/                  # Modelos serializados (.pkl/.joblib)
│
├── src/                        # Código da pipeline de ML
│   ├── preprocessing.py        # Funções de limpeza e preparação de dados
│   ├── feature_engineering.py  # Engenharia de features
│   ├── train.py                # Treinamento do modelo
│   ├── evaluate.py             # Avaliação e métricas
│   └── utils.py                # Funções auxiliares
│
├── tests/                      # Testes unitários
│   ├── test_preprocessing.py
│   └── test_model.py
│
├── Dockerfile                  # Dockerfile para empacotamento
├── requirements.txt            # Dependências do projeto
├── README.md                   # Documentação principal
└── notebooks/                  # Jupyter Notebooks (exploração, EDA, etc.)
```

3) Instruções de Deploy (como subir o ambiente)

Deixe claro as instruções para subir o ambiente, ex.: docker build, docker run, kubectl apply.

- Pré-requisitos (Python version, bibliotecas etc.).
- Instalação de dependências (requirements.txt ou environment.yml).
- Comandos para treinar, validar e testar o modelo.

4) Exemplos de Chamadas à API

Coloque exemplos de chamadas à API (ex.: via curl, Postman ou scripts Python), com inputs esperados e outputs gerados.

5) Etapas do Pipeline de Machine Learning

Explique brevemente na sua documentação quais são as etapas de pré-processamento da sua pipeline.

Exemplo:

- Pré-processamento dos Dados.
- Engenharia de Features.
- Treinamento e Validação.
- Seleção de Modelo.
- Pós-processamento (se aplicável).

🎲 **Base de dados: Base de dados Decision**

Como os arquivos se relacionam?

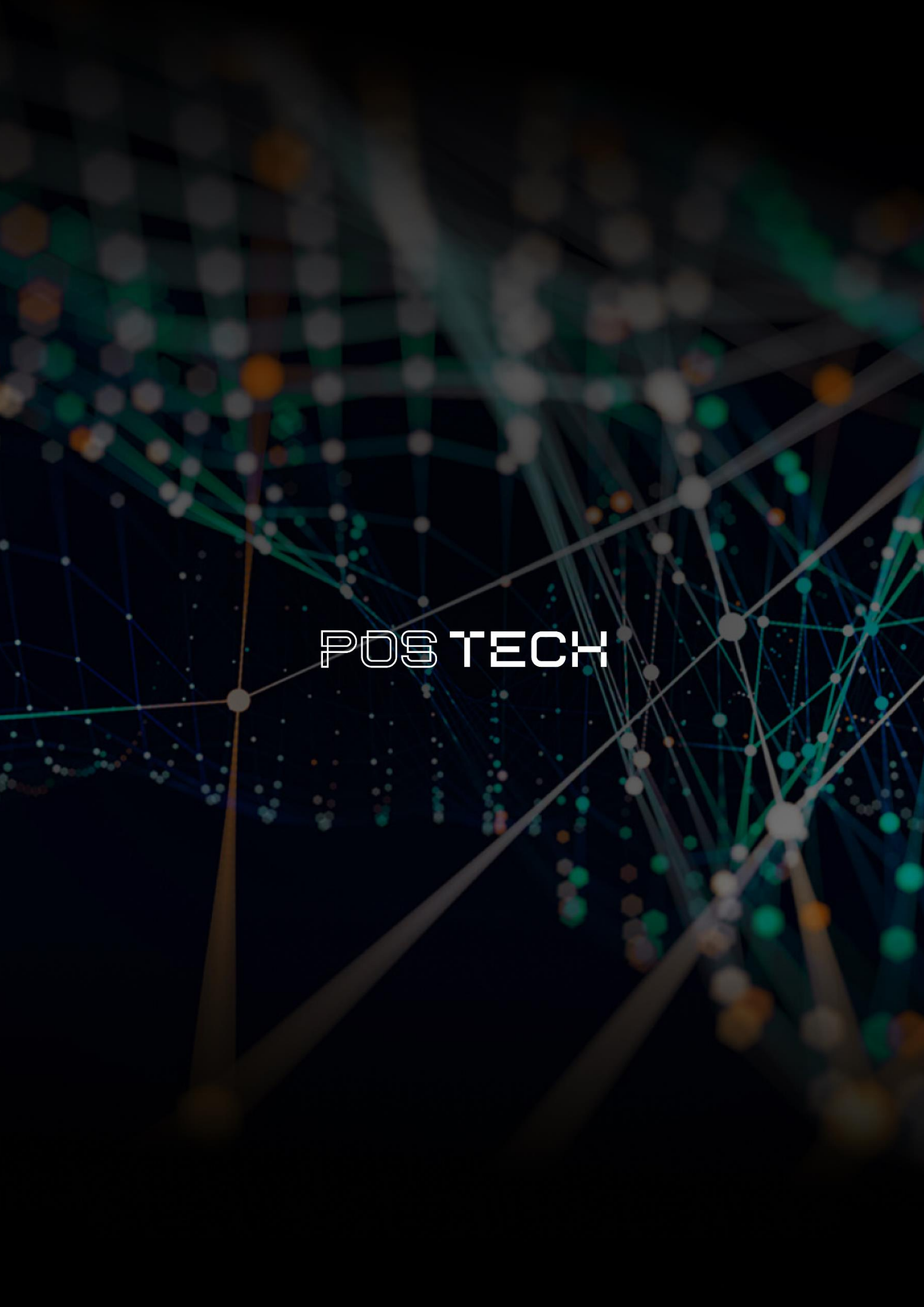
- Cada Vaga em vagas.json tem um conjunto de candidatos inscritos em prospects.json.
- Cada candidato listado em prospects.json tem informações detalhadas que estão armazenadas em applicants.json.
- Exemplo:
  - Para a vaga XPTO em vagas.json, temos:
    - Primeiro a pessoa se cadastra no sistema por meio do applicants.json.
    - Depois os candidatos que passaram (ou foram selecionados/avaliados) para alguma vaga ficam em prospects.json, onde se relaciona com vagas.json.
- PS: você pode ver mais detalhes aqui.

Entrega

- Código-fonte organizado e documentado em um repositório GitHub.
- Documentação do projeto.
- Link para a API.

- Vídeo de até 5 minutos no formato gerencial com a solução proposta do seu projeto.

Boa sorte! 🍀


POS TECH