TECH CHALLENGE

Tech Challenge é o projeto da fase que englobará os conhecimentos obtidos em todas as disciplinas da fase. Esta é uma atividade que, a princípio, deve ser desenvolvida em grupo. É importante atentar-se ao prazo de entrega já que trata-se de uma atividade obrigatória, uma vez que sua pontuação se refere a 90% da nota final.

O problema

Com base nas duas fases anteriores, você será responsável por criar um modelo de ML que siga algumas regras:

- Construa uma API que colete dados (se possível, em tempo real) e armazene isso em um banco de dados convencional, uma estrutura de DW ou até mesmo um Data Lake (se quiser, utilize a mesma ideia da fase 2, inclusive a fonte de dados).
- Construa um modelo de ML à sua escolha que utilize essa base de dados para treinar o mesmo.
- Seu modelo deve seguir com seu **código no github e a devida documentação.**
- Você deve ter uma apresentação visual do storytelling do seu modelo (contando todas as etapas realizadas até a entrega final por meio de um **vídeo explicativo**). O vídeo pode ser entregue através de um link do YouTube junto com o link do seu repositório do github, por meio de um arquivo txt via upload na plataforma online.
- Seu modelo deve ser produtivo (alimentar uma aplicação simples ou um dashboard).

Análise Detalhada: Roteiro vs. Exigências do Professor

1. API de Coleta e Armazenamento de Dados

- **O que o professor pediu:** *"Construa uma API que colete dados (se possível, em tempo real) e armazene isso em um banco de dados convencional, uma estrutura de DW ou até mesmo um Data Lake..."*
- **Minha Sugestão no Roteiro:** Usar **FastAPI** ou **Flask** para a API e **SQLite** ou **PostgreSQL** para o armazenamento.
- **Justificativa:**
  - **API com FastAPI/Flask:** Essas ferramentas são ideais porque são frameworks Python modernos, leves e focados em criar APIs de forma rápida e eficiente. Você consegue criar o *endpoint* para receber os dados com poucas linhas de código, o que atende perfeitamente à exigência de "construir uma API".
  - **Banco de Dados Convencional com SQLite/PostgreSQL:** A sugestão de usar SQLite ou PostgreSQL cumpre diretamente o requisito de "armazenar isso em um banco de dados convencional". SQLite é excelente para começar, pois é um banco de dados autocontido em um único arquivo, sem a necessidade de instalar e gerenciar um servidor. PostgreSQL é a evolução natural para um ambiente mais robusto, mostrando que você também entende de tecnologias de produção.

2. Construção do Modelo de Machine Learning

- **O que o professor pediu:** *"Construa um modelo de ML à sua escolha que utilize essa base de dados para treinar o mesmo."*
- **Minha Sugestão no Roteiro:** Realizar o pré-processamento dos dados e treinar um modelo de classificação (como **Regressão Logística** ou **Random Forest**).
- **Justificativa:**
  - **Modelo de Classificação:** O problema de prever a `satisfaction` (satisfeito vs. insatisfeito) é um problema clássico de **classificação**. Portanto, a sugestão de usar modelos como Regressão Logística (para uma base de comparação) e Random Forest (para maior poder preditivo) é tecnicamente correta e demonstra seu conhecimento em diferentes algoritmos.
  - **Utilizar a Base de Dados:** O roteiro enfatiza o uso dos dados coletados pela API e armazenados no banco para o treinamento. Isso garante que você está cumprindo a regra de que o modelo deve ser treinado com a base de dados que você mesmo montou, fechando o ciclo do projeto.

3. Código no GitHub e Documentação

- **O que o professor pediu:** *"Seu modelo deve seguir com seu código no github e a devida documentação."*
- **Minha Sugestão no Roteiro:** Organizar o projeto em uma estrutura de pastas clara e criar um `README.md` detalhado.
- **Justificativa:**
  - **GitHub:** A plataforma é o padrão do mercado para versionamento de código, então a sugestão é óbvia e correta.
  - **Documentação (`README.md`):** Um projeto não é apenas código. A "devida documentação" exigida pelo professor é perfeitamente atendida por um `README.md` bem escrito. Ele serve como o manual do seu projeto, explicando o objetivo, como instalar as dependências (`requirements.txt`) e como executar a aplicação. Isso mostra profissionalismo e organização, que são habilidades essenciais avaliadas.

4. Storytelling e Apresentação em Vídeo

- **O que o professor pediu:** *"Você deve ter uma apresentação visual do storytelling do seu modelo (contando todas as etapas realizadas...)."*
- **Minha Sugestão no Roteiro:** Estruturar o vídeo em uma narrativa lógica: Problema, Dados, Solução, Demonstração e Resultados.
- **Justificativa:**
  - **Storytelling:** A sugestão transforma um relatório técnico em uma "história". Em vez de apenas listar o que foi feito, você conta o *porquê* (o problema de negócio), o *como* (a solução técnica) e o *resultado* (o que foi alcançado). Essa abordagem narrativa é exatamente o que significa "storytelling" no contexto de dados e cumpre a exigência do professor de forma muito mais eficaz do que uma simples apresentação de slides.

5. Modelo em Produção

- **O que o professor pediu:** *"Seu modelo deve ser produtivo (alimentar uma aplicação simples ou um dashboard)."*
- **Minha Sugestão no Roteiro:** Usar **Streamlit** ou **Dash** para criar um dashboard interativo.
- **Justificativa:**
  - **Modelo Produtivo:** Um modelo "na gaveta" (em um notebook Jupyter, por exemplo) não é produtivo. A exigência é que ele seja usado em uma aplicação real.
  - **Dashboard com Streamlit:** Streamlit é a ferramenta perfeita para isso, pois permite criar aplicações web interativas usando apenas Python, de forma muito rápida. Um dashboard onde o usuário pode inserir dados e receber uma previsão em tempo real é a personificação de um "modelo produtivo alimentando uma aplicação simples", atendendo a este requisito de maneira visual, prática e impressionante.