Relatório de Desenvolvimento do Projeto de Análise de Ocorrências Aeronáuticas

Integrantes:

- Vitor Manoel Santos Moura
- Victor Santos Chagas
- Cauan Santos Silva

Visão Geral do Projeto

Este projeto tem como objetivo principal analisar dados de ocorrências aeronáuticas da aviação civil brasileira para prever o nível de dano de uma aeronave em um incidente. Para isso, foi desenvolvida uma solução completa que engloba desde a coleta e tratamento dos dados até a criação de um modelo de *Machine Learning* e sua disponibilização através de uma API REST e uma interface web interativa.

A aplicação permite que um usuário insira informações sobre uma ocorrência e receba uma predição sobre a gravidade do dano à aeronave, auxiliando na avaliação de riscos e na tomada de decisões.

Análise de Dados e Treinamento do Modelo

A base do projeto foi a análise detalhada dos dados de ocorrências, seguida pelo treinamento de um modelo preditivo. Todo esse processo foi documentado no notebook `dataplane-api/model/notebook/projeto_ia_retificado.py`.

1. Coleta e Preparação dos Dados

Os dados foram obtidos de múltiplos arquivos CSV, incluindo informações sobre ocorrências, tipos de ocorrência, aeronaves, fatores contribuintes e recomendações.

O processo de preparação dos dados envolveu as seguintes etapas:

- **Limpeza e Padronização:** Remoção de colunas duplicadas ou irrelevantes, tratamento de valores ausentes (Nulos, `***`, `NULL`) e padronização de strings.
- **Transformação de Dados:** Conversão de campos de data e hora para formatos adequados (`datetime`) para possibilitar análises temporais.
- **Merge de Datasets:** Unificação dos diferentes datasets em um único DataFrame, utilizando a coluna `codigo_ocorrencia` como chave primária. Linhas com dados essenciais ausentes (como `fator_area`) foram removidas para garantir a qualidade do modelo.

2. Treinamento e Avaliação do Modelo

O objetivo era prever a variável `aeronave_nivel_dano`. Para isso, foi escolhido um modelo de classificação.

- **Seleção de Features:** Foram selecionadas as colunas mais relevantes para a predição, como `aeronave_tipo_operacao`, `fator_area`, `aeronave_tipo_veiculo`, `ocorrencia_uf`, `aeronave_ano_fabricacao` e `aeronave_fatalidades_total`.
- **Codificação de Variáveis Categóricas:** As variáveis categóricas (não numéricas) foram transformadas em números utilizando `LabelEncoder` da biblioteca `scikit-learn`, para que pudessem ser utilizadas pelo modelo.
- **Escolha do Modelo:** Foram testados os algoritmos `DecisionTreeClassifier` e `RandomForestClassifier`. O `RandomForestClassifier` apresentou um desempenho superior, com uma acurácia média de aproximadamente 89% nos testes e na validação cruzada, demonstrando um bom equilíbrio entre viés e variância.
- **Treinamento:** O modelo final (`RandomForestClassifier` com `max_depth=15`) foi treinado com 80% dos dados e testado com os 20% restantes.
- **Salvando o Modelo:** O modelo treinado, juntamente com os `LabelEncoders` (para as features e para o target), foram salvos em arquivos `.joblib` para serem posteriormente carregados e utilizados pela API.

Arquitetura do Sistema

A solução foi desenvolvida com uma arquitetura de microsserviços, composta por um back-end (API) e um front-end (aplicação web), ambos containerizados com Docker.

- **Back-end:** Uma API REST desenvolvida em Python com o framework **FastAPI**. É responsável por servir o modelo de Machine Learning, processar as requisições de predição e fornecer dados para a aplicação web.
- **Front-end:** Uma Single Page Application (SPA) desenvolvida com **Next.js** (React) e **TypeScript**. Oferece a interface do usuário para interagir com o modelo, visualizar dados e obter as predições.
- **Banco de Dados: MongoDB** é utilizado para armazenar os dados das ocorrências, que são consumidos pela API para popular os dashboards e mapas da aplicação.

Back-end (dataplane-api)

A API é o cérebro da aplicação, onde a lógica de negócio e o modelo de IA residem.

Endpoints da API

A API expõe vários endpoints, sendo os mais importantes para a funcionalidade de IA:

- `POST /api/v1/predict`: Recebe os dados de uma ocorrência em formato JSON, os processa e retorna a predição do nível de dano e a confiança do modelo.
- `GET /api/v1/predict/form-options`: Fornece as opções de preenchimento para os campos do formulário do front-end (ex: lista de UFs, tipos de

operação, etc.), buscando-as diretamente das classes dos `LabelEncoders` salvos.
- `GET /api/v1/ocurrence/coordinates`: Retorna as coordenadas geográficas das ocorrências para exibição no mapa.
- `GET /api/v1/health`: Endpoint para verificação de status da API.

Lógica de Predição

O serviço `AIService` (`dataplane-api/app/services/ai_service.py`) implementa a lógica de predição:

1. **Carregamento Singleton:** O modelo e os encoders são carregados uma única vez na inicialização da aplicação, utilizando um padrão Singleton, para evitar o alto custo de carregá-los a cada requisição.
2. **Pré-processamento:** Quando uma requisição chega ao endpoint `/predict`, os dados recebidos são transformados em um DataFrame do Pandas. As features categóricas são codificadas usando os `LabelEncoders` previamente carregados.
3. **Predição e Confiança:** O DataFrame processado é passado para o modelo `RandomForestClassifier`, que retorna a predição do nível de dano. Além da predição, a probabilidade (confiança) da classe prevista também é calculada.
4. **Resposta:** A API retorna a predição decodificada (o texto original do nível de dano, ex: "LEVE") e o score de confiança.

A autenticação na API é feita via Bearer Token, garantindo que apenas aplicações autorizadas possam consumir os endpoints.

Front-end (dataplane-webapp)

O front-end oferece uma experiência de usuário rica e interativa.

Página de Predição

A página de predição (`dataplane-webapp/src/components/PredictionPage.tsx`) é o principal ponto de interação do usuário com o modelo de IA.

- **Formulário Dinâmico:** O formulário de predição é construído com campos `Select` que são populados dinamicamente com as opções obtidas do endpoint `/api/v1/predict/form-options`.
- **Gerenciamento de Estado:** A biblioteca `react-hook-form` é utilizada para gerenciar o estado do formulário, validações e submissão dos dados.
- **Comunicação com a API:** Um cliente de API (`dataplane-webapp/src/lib/api.ts`), construído sobre o `axios`, gerencia a comunicação com o back-end. Ele é responsável por adicionar o token de autenticação e fazer as chamadas para os endpoints de predição e obtenção de opções.
- **Exibição do Resultado:** Após a submissão, o resultado da predição (nível de dano e confiança) é exibido de forma clara e intuitiva, com cores que indicam a severidade do dano e o nível de confiança do modelo.

Outras Funcionalidades

Além da predição, o front-end conta com:

- **Dashboard de Análise:** Gráficos e visualizações que apresentam estatísticas sobre as ocorrências.
- **Mapa de Ocorrências:** Um mapa interativo que mostra a distribuição geográfica dos incidentes.

Como Executar o Projeto

Para executar o projeto localmente, é necessário ter o Docker e o Docker Compose instalados.

1. **Configurar Variáveis de Ambiente:**
    - Crie um arquivo `.env` na raiz do diretório `dataplane-api`, baseando-se no `env.example`. Defina um `API_TOKEN`.
    - Crie um arquivo `.env` na raiz do diretório `dataplane-webapp`, baseando-se no `env.example`. Use o mesmo `API_TOKEN` na variável `NEXT_PUBLIC_API_TOKEN`.
2. **Subir os Contêineres:** Execute o seguinte comando na raiz de cada subdiretório (`dataplane-api` e `dataplane-webapp`):

    ```
    docker-compose up --build
    ```

3. **Acessar a Aplicação:**
    - A API estará disponível em `http://localhost:8000`.
    - A aplicação web estará disponível em `http://localhost:3000`.