Documentação da Arquitetura de Software

1. Introdução

Este documento detalha a arquitetura de software proposta para a aplicação de Machine Learning, que utiliza Streamlit para a interface do usuário e PyCaret para o backend de modelagem. A aplicação visa fornecer uma plataforma interativa para upload de dados, análise exploratória, seleção de variáveis, treinamento e comparação de modelos de Machine Learning (classificação, regressão e clusterização), análise de modelos e previsão com novos dados.

2. Visão Geral da Arquitetura

A arquitetura da aplicação é baseada em um modelo cliente-servidor simplificado, onde o Streamlit atua como a camada de apresentação (frontend) e o PyCaret, executado no mesmo ambiente, serve como a camada de lógica de negócios (backend). A persistência de dados será inicialmente tratada através de arquivos (CSV/Excel) e, futuramente, com a integração de um banco de dados SQL dockerizado.

Componentes Principais:

- **Streamlit (Frontend):** Responsável pela interface gráfica do usuário, permitindo a interação com a aplicação através de upload de arquivos, seleção de opções e visualização de resultados.
- **PyCaret (Backend):** Biblioteca de Machine Learning low-code que abstrai o processo de modelagem, incluindo pré-processamento de dados, treinamento, avaliação e comparação de modelos.
- **Pandas:** Utilizado para manipulação e análise de dados em memória.
- **Docker:** Ferramenta de conteinerização para empacotar a aplicação e suas dependências, garantindo portabilidade e reprodutibilidade.
- **Banco de Dados SQL (Opcional/Futuro):** Para persistência de dados e modelos, permitindo escalabilidade e gerenciamento eficiente de grandes volumes de informação.

3. Diagrama de Arquitetura

```
graph TD
  A[Usuário] -->|Interage com| B(Streamlit App)
  B -->|Upload de Dados| C{Arquivos CSV/Excel}
  B -->|Chama Funções PyCaret| D(PyCaret Library)
  D -->|Processa Dados| E[Pandas DataFrames]
  D -->|Treina/Avalia Modelos| F[Modelos de ML]
  B -->|Exibe Resultados| A
  subgraph Docker Container
    B
    D
    E
    F
  end
  C -- Opcional --> G[Banco de Dados SQL]
  G -- Opcional --> D
```

4. Detalhamento dos Componentes

4.1. Streamlit App

O Streamlit é a espinha dorsal da interface do usuário. Ele permite a criação rápida de aplicações web interativas com código Python puro. As principais funcionalidades implementadas no Streamlit incluem:

- **Upload de Dados:** Permite que o usuário carregue arquivos CSV ou Excel. Os dados são lidos e convertidos em DataFrames do Pandas.
- **Análise Exploratória de Dados (EDA):** Apresenta estatísticas descritivas, tipos de dados e informações sobre valores ausentes, fornecendo insights iniciais sobre o dataset.
- **Seleção de Variáveis:** O usuário pode selecionar a variável alvo para problemas de classificação e regressão.
- **Seleção do Tipo de Problema:** Permite alternar entre classificação, regressão e clusterização, adaptando as opções e o fluxo de trabalho do PyCaret.
- **Execução do PyCaret:** Inicia o processo de configuração do ambiente PyCaret, comparação de modelos e treinamento.
- **Análise do Modelo:** Exibe os resultados da comparação de modelos e permite a visualização de gráficos específicos para cada tipo de problema (e.g., curva ROC para classificação, resíduos para regressão, elbow method para clusterização).
- **Previsão com Novos Dados:** Permite o upload de novos datasets para realizar previsões com o modelo treinado.

4.2. PyCaret Library

O PyCaret é uma biblioteca Python low-code que simplifica o ciclo de vida do Machine Learning. Ele é utilizado como o motor de backend para as operações de ML. As funções chave do PyCaret utilizadas são:

- `setup()`: Inicializa o ambiente do PyCaret, preparando os dados para modelagem (tratamento de valores ausentes, codificação de variáveis categóricas, etc.).
- `compare_models()`: Treina e avalia múltiplos modelos de Machine Learning e retorna o melhor modelo com base em métricas padrão.
- `create_model()`: Cria um modelo específico (usado para clusterização, onde não há uma variável alvo).
- `assign_model()`: Atribui clusters aos dados no caso de problemas de clusterização.
- `pull()`: Recupera os resultados da última operação do PyCaret (e.g., tabela de comparação de modelos).
- `save_model()` e `load_model()`: Permitem a persistência e o carregamento de modelos treinados, o que é crucial para a funcionalidade de previsão.
- `plot_model()`: Gera visualizações para análise de modelos, como curvas ROC, matrizes de confusão, gráficos de resíduos, etc.

4.3. Pandas

O Pandas é fundamental para a manipulação de dados. Ele é usado para:

- Ler os arquivos CSV e Excel carregados pelo usuário.
- Realizar operações de pré-processamento de dados antes de passá-los para o PyCaret.
- Exibir prévias e estatísticas descritivas dos dados na interface do Streamlit.

4.4. Docker

O Docker será utilizado para conteinerizar a aplicação, garantindo que ela possa ser executada de forma consistente em diferentes ambientes. Isso inclui:

- **Dockerfile:** Para construir a imagem da aplicação, contendo o sistema operacional base, Python, as bibliotecas Streamlit, PyCaret, Pandas e todas as suas dependências.
- **Docker Compose (Opcional/Futuro):** Para orquestrar múltiplos contêineres, como o da aplicação Streamlit e um contêiner de banco de dados SQL.

4.5. Banco de Dados SQL (Opcional/Futuro)

Embora a versão inicial da aplicação utilize upload de arquivos, a arquitetura prevê a integração com um banco de dados SQL. Isso permitiria:

- Armazenar datasets de forma persistente.
- Gerenciar múltiplos datasets e modelos.
- Suportar bases de dados maiores e mais complexas.
- Facilitar a integração com outras ferramentas e sistemas.

5. Fluxo de Dados e Interações

1. **Início da Aplicação:** O usuário acessa a aplicação Streamlit através de um navegador web.
2. **Upload de Dados:** O usuário faz o upload de um arquivo CSV ou Excel. O Streamlit lê o arquivo usando Pandas.
3. **EDA:** O Streamlit exibe estatísticas descritivas e informações sobre os dados.
4. **Seleção de Parâmetros:** O usuário seleciona a variável alvo e o tipo de problema (classificação, regressão, clusterização).
5. **Execução do PyCaret:** Ao clicar no botão

"Executar PyCaret", o Streamlit invoca as funções apropriadas do PyCaret (`setup` , `compare_models` , `create_model`).
6. **Treinamento e Avaliação de Modelos:** O PyCaret realiza o pré-processamento, treinamento e avaliação dos modelos. Os resultados (tabela de comparação de modelos, métricas) são retornados ao Streamlit.
7. **Análise do Modelo:** O Streamlit exibe os resultados e permite a geração de gráficos de análise de modelo usando `plot_model` do PyCaret.
8. **Persistência do Modelo:** O melhor modelo treinado é salvo em disco usando `save_model` do PyCaret.
9. **Previsão:** Se o usuário fizer upload de novos dados e clicar em "Fazer Previsão", o Streamlit carrega o modelo salvo (`load_model`) e utiliza-o para gerar previsões nos novos dados.

6. Considerações de Segurança e Escalabilidade

Segurança:

- **Validação de Entrada:** Implementar validação robusta para os arquivos de entrada e seleções do usuário para prevenir vulnerabilidades.

- **Isolamento:** A conteinerização com Docker oferece um nível de isolamento, mas é crucial garantir que o contêiner seja executado com os privilégios mínimos necessários.
- **Autenticação/Autorização (Futuro):** Para um ambiente de produção, seria necessário adicionar mecanismos de autenticação e autorização para controlar o acesso à aplicação e aos dados.

Escalabilidade:

- **Docker:** Facilita a escalabilidade horizontal, permitindo a execução de múltiplas instâncias da aplicação em diferentes servidores.
- **PyCaret:** Embora o PyCaret seja eficiente para prototipagem e desenvolvimento rápido, para grandes volumes de dados ou modelos complexos, pode ser necessário otimizar o desempenho ou considerar frameworks de ML distribuídos.
- **Banco de Dados:** A migração para um banco de dados SQL robusto é essencial para a escalabilidade da persistência de dados.
- **Cloud Deployment:** A arquitetura é compatível com implantações em nuvem (AWS, GCP, Azure), onde serviços gerenciados podem ser utilizados para escalabilidade e alta disponibilidade.

7. Conclusão

A arquitetura proposta, baseada em Streamlit e PyCaret, oferece uma solução ágil e eficiente para o desenvolvimento de uma aplicação de Machine Learning interativa. A utilização de Docker garante a portabilidade e reprodutibilidade do ambiente. As considerações futuras sobre banco de dados e segurança/escalabilidade preparam a aplicação para um crescimento e uso em cenários mais exigentes.