## **Sistema Inteligente de Análise e Modelagem de Dados com PyCaret**

Luciano Franzoi filho RA: 22021768-2

### **Visão Geral**

Este sistema é uma aplicação interativa desenvolvida em Python com Streamlit, voltada para análise exploratória de dados (EDA), download de datasets, e modelagem preditiva (classificação, regressão e clusterização). O objetivo é democratizar o acesso à ciência de dados, permitindo que usuários de diferentes níveis possam explorar, visualizar e modelar dados tabulares de forma intuitiva e automatizada.

Interface gráfica do usuário, Aplicativo

O conteúdo gerado por IA pode estar incorreto.

Tela inicial do sistema

### **Estrutura do Projeto**

streamlit-data-science-app/

│

├── main.py

├── requirements.txt

│

├── dados/ # Armazena datasets baixados ou carregados

│ └── (datasets.csv, etc)

│

├── results/ # Relatórios EDA (Sweetviz, YData)

│ ├── sweetviz\_report.html

│ ├── ydata\_profile.html

│ └── (outros\_relatorios.html)

│

└── src/

│

├── controllers/ # Lógica de controle (download, EDA, modelagem)

│ ├── data\_controller.py

│ ├── eda\_controller.py

│ ├── model\_controller.py

│ └── \_\_init\_\_.py

│

├── models/ # Estruturas de dados e resultados

│ ├── dataset\_model.py

│ ├── result\_model.py

│ └── \_\_init\_\_.py

│

├── utils/ # Utilitários (logger, download Kaggle, SQL)

│ ├── kaggle\_downloader.py

│ ├── logger.py

│ └── \_\_init\_\_.py

│

└── views/ # Interfaces Streamlit (sidebar, EDA, ML, download)

├── sidebar.py

├── eda\_view.py

├── machine\_learning.py

├── download\_view.py

└── \_\_init\_\_.py

### **Organização do Código**

O projeto segue o padrão de arquitetura MVC (Model-View-Controller), promovendo separação de responsabilidades, modularidade e facilidade de manutenção. Abaixo, detalhamos o papel de cada diretório e exemplos de arquivos típicos:

**models/**

Esta pasta contém as estruturas de dados e objetos de resultado utilizados em todo o sistema. Os arquivos aqui definem classes e funções responsáveis por representar datasets, resultados de modelagem, configurações de experimentos e outros objetos essenciais para o fluxo de dados da aplicação.

* **dataset\_model.py:**  
  Define a estrutura de um dataset carregado, incluindo atributos como nome, caminho, dataframe em memória, tipos de variáveis e metadados relevantes. Pode incluir métodos para validação e transformação dos dados.
* **result\_model.py:**  
  Estrutura os resultados de experimentos de machine learning, armazenando métricas, modelos treinados, parâmetros utilizados, gráficos gerados e caminhos para relatórios exportados.

Sugestão de detalhamento:  
Classes aqui facilitam a troca de informações entre controladores e visualizações, centralizando a lógica de dados e resultados.

**views/**

A pasta views é responsável por toda a interface com o usuário, implementada com Streamlit. Cada arquivo corresponde a uma tela, componente ou funcionalidade específica do sistema, tornando a experiência do usuário intuitiva e organizada.

* **sidebar.py:**  
  Implementa a barra lateral de navegação, permitindo ao usuário alternar entre funcionalidades como upload de dados, EDA e modelagem.
* **eda\_view.py:**  
  Exibe os resultados da análise exploratória, gráficos interativos, sumários estatísticos e links para download de relatórios EDA.
* **machine\_learning.py:**  
  Interface para seleção de variáveis, configuração de tarefas de ML, visualização de métricas, gráficos de avaliação e exportação de modelos.
* **download\_view.py:**  
  Tela dedicada ao download de datasets do Kaggle ou de outras fontes, com feedback visual sobre o progresso e sucesso da operação.

**controllers/**

Os controllers concentram a lógica de negócio do sistema. Eles fazem a ponte entre os dados (models) e a interface (views), processando requisições do usuário, executando operações de EDA, modelagem, download de dados e orquestrando o fluxo da aplicação.

* **data\_controller.py:**  
  Gerencia o carregamento, validação e armazenamento de datasets, além de controlar o download de arquivos do Kaggle para a pasta de dados.
* **eda\_controller.py:**  
  Executa a análise exploratória automatizada, gerando relatórios com YData Profiling, Sweetviz e D-Tale, além de processar estatísticas e gráficos para exibição.
* **model\_controller.py:**  
  Responsável pelo pipeline de machine learning: seleção de variáveis, configuração de tarefas, treinamento, avaliação, tuning de hiperparâmetros e exportação de modelos.

**utils/**

A pasta utils reúne funções utilitárias e módulos auxiliares que suportam as demais camadas do sistema. São scripts reutilizáveis, focados em tarefas específicas que não pertencem diretamente à lógica de negócio principal.

* **kaggle\_downloader.py:**  
  Implementa a integração com a API do Kaggle, permitindo baixar datasets públicos de forma automatizada e segura.
* **logger.py:**  
  Gerencia o registro de logs do sistema, facilitando o monitoramento de operações, rastreamento de erros e auditoria de processos.

**Fundamentos Teóricos**

O desenvolvimento deste sistema está fundamentado em três pilares centrais, que orientam tanto a arquitetura do código quanto a experiência do usuário:

**1. Análise Exploratória Automatizada (EDA)**

A análise exploratória de dados (EDA) é uma etapa fundamental para compreender a estrutura, padrões e possíveis anomalias dos dados antes da modelagem. Inspirado em Hyndman & Athanasopoulos (2018), o sistema automatiza o EDA utilizando bibliotecas como **YData Profiling**, **Sweetviz** e **D-Tale**.  
Essas ferramentas permitem:

* Geração automática de sumários estatísticos;
* Identificação de outliers, valores ausentes e colunas com baixa variabilidade;
* Visualização de distribuições, correlações e tendências temporais;
* Relatórios interativos salvos na pasta results/ para consulta e compartilhamento.

No código, essa etapa é orquestrada pelos controladores de EDA e exibida nas views específicas, tornando o processo acessível mesmo para usuários sem experiência em programação.

**2. Modelagem Automatizada com Scikit-learn**

A modelagem preditiva do sistema é baseada na biblioteca **scikit-learn**, referência em machine learning em Python (Pedregosa et al., 2011).  
O sistema implementa pipelines que:

* Automatizam o pré-processamento dos dados (tratamento de nulos, encoding, normalização);
* Permitem a seleção do tipo de tarefa (classificação, regressão, clusterização);
* Comparam múltiplos algoritmos e apresentam as melhores métricas;
* Oferecem tuning de hiperparâmetros e geração de gráficos de avaliação;
* Exportam modelos treinados e resultados para uso futuro.

A lógica de modelagem está centralizada nos controllers, enquanto a seleção de variáveis, configuração e visualização dos resultados são feitas nas views, garantindo flexibilidade e transparência.

**3. Interface Interativa com Upload Simples**

A interface do sistema foi projetada com foco em usabilidade, seguindo recomendações de Kluyver et al. (2019) para aplicações interativas em ciência de dados.  
Utilizando **Streamlit**, o sistema oferece:

* Upload de arquivos por drag-and-drop ou download direto do Kaggle;
* Navegação intuitiva por meio de uma sidebar;
* Feedback visual imediato sobre operações realizadas (carregamento, análise, modelagem);
* Visualizações interativas e exportação facilitada de relatórios e modelos.

Essa abordagem reduz a barreira de entrada para usuários iniciantes e agiliza o fluxo de trabalho para profissionais, promovendo uma experiência fluida e produtiva.

### Interface gráfica do usuário, Texto, Aplicativo O conteúdo gerado por IA pode estar incorreto.**Funcionalidades** **1. Upload e Gerenciamento de Dados Download direto do Kaggle:** O usuário pode baixar datasets públicos do Kaggle diretamente pela interface, usando o nome do dataset. O arquivo é salvo na pasta ***dados/.* Carregamento local:** O sistema busca automaticamente arquivos .csv na pasta ***dados/*** para análise e modelagem. **Controle de dados:** Implementado em ***data\_controller.py***, com funções para baixar, salvar e carregar datasets.

Tela Download Dataset

#### **2. Análise Exploratória de Dados (EDA)**

#### **Resumo automático:** Ao selecionar um dataset, o sistema exibe informações como número de linhas/colunas, tipos de variáveis, valores ausentes e únicos.

#### **Relatórios interativos:** O usuário pode gerar relatórios EDA com:

#### **YData Profiling:**Relatório HTML detalhado.

#### **Sweetviz:**Relatório visual comparativo.

#### **D-Tale:**Visualização interativa dos dados.

#### **Geração e armazenamento:** Relatórios são salvos na pasta ***results/*** e podem ser visualizados na interface.

#### **Implementação:** Controlada por ***eda\_controller.py*** e interface em ***eda\_view.py.***

Interface gráfica do usuário, Texto

O conteúdo gerado por IA pode estar incorreto.

Tela EDA

#### **3. Seleção de Variáveis e Modelagem**

#### **Seleção de features e alvo:** O usuário escolhe as variáveis preditoras e a variável alvo (target) para regressão ou classificação.

#### **Tipos de tarefa:**

#### **Classificação:**Ex: predição de categorias.

#### **Regressão:**Ex: predição de valores contínuos.

#### **Clusterização:**Agrupamento não supervisionado.

#### **Modelos disponíveis:**

#### **Classificação:**Logistic Regression, Random Forest, etc.

#### **Regressão:**Linear Regression, Random Forest, etc.

#### **Clusterização:**KMeans.

#### **Treinamento e avaliação:** O sistema treina o modelo, exibe métricas (acurácia, RMSE, R², etc.), matriz de confusão, gráficos de dispersão e permite prever novos valores.

#### **Implementação:** Controlada por ***model\_controller.py*** e interface em ***machine\_learning.py.***

Tela de celular com aplicativo aberto

O conteúdo gerado por IA pode estar incorreto.

Tela Machine Learning

#### **4. Visualização e Exportação de Resultados**

#### **Visualizações:**

#### Gráficos interativos (matriz de confusão, dispersão, distribuição de clusters).

#### Relatórios HTML (EDA).

#### Tabelas de resultados.

#### **Exportação:**

#### Resultados e modelos podem ser salvos para uso futuro.

#### Relatórios EDA disponíveis em HTML.

#### Possibilidade de salvar datasets processados.

#### **Implementação:** Utiliza funções dos módulos ***models/*** e ***utils/*** para salvar resultados e logs.

Interface gráfica do usuário

O conteúdo gerado por IA pode estar incorreto.

Tela resultados Machine Learning

Interface gráfica do usuário, Texto

O conteúdo gerado por IA pode estar incorreto.

Tela resultados EDA Ydata

### **Roadmap de Desenvolvimento**

|  |  |  |
| --- | --- | --- |
| Etapa | Funcionalidade Principal | Entregáveis/Detalhes |
| Versão 1 | Download e EDA Local | - Download de datasets (Kaggle ou upload) - Análise exploratória automatizada - Relatórios EDA (YData, Sweetviz, D-Tale) - Visualização básica dos dados |
| Versão 2 | Seleção de Variáveis e Execução de Modelos | - Interface para seleção de features e alvo  - Implementação de pipelines de ML (classificação, regressão, clusterização)  - Avaliação automática de modelos  - Visualização de métricas iniciais |
| Versão 3 | Dashboard Interativo e Exportação de Resultados | - Dashboard com métricas detalhadas  - Gráficos interativos (matriz de confusão, dispersão, etc.)  - Exportação de relatórios e modelos treinados  - Download de resultados processados |
| Versão 4 | Integração Avançada e Historização de Experimentos | - Integração com API REST para automação e consumo externo  - Armazenamento e versionamento de experimentos  - Logs detalhados e rastreabilidade  - Suporte a múltiplos usuários (opcional) |
| Versão 5 | Personalização e Expansão de Funcionalidades | - Customização de pipelines e hiperparâmetros  - Templates de experimentos  - Integração com notebooks e exportação para outros formatos |

### 

### **Conclusão**

O sistema oferece uma solução completa para análise e modelagem de dados tabulares, integrando download, EDA e machine learning em uma interface amigável. Sua arquitetura modular facilita a manutenção e expansão, tornando-o ideal para ensino, prototipagem e uso profissional.

Além disso, a separação clara entre camadas (controllers, models, views e utils) garante que novas funcionalidades possam ser adicionadas de forma organizada e sem comprometer a estabilidade do sistema. O uso de bibliotecas modernas como Streamlit, scikit-learn, YData Profiling e Sweetviz proporciona ao usuário acesso a recursos avançados de análise e visualização, mesmo sem conhecimento aprofundado em programação.

A interface intuitiva permite que usuários iniciantes realizem tarefas complexas de ciência de dados com poucos cliques, enquanto profissionais podem aproveitar a flexibilidade para customizar experimentos e exportar resultados detalhados. O suporte ao download direto de datasets, geração automática de relatórios e exportação de modelos treinados amplia as possibilidades de aplicação em ambientes acadêmicos, corporativos e de pesquisa.

Por fim, o projeto serve como base sólida para futuras integrações, como APIs externas, bancos de dados ou módulos de automação, promovendo a escalabilidade e a adaptação a diferentes cenários de uso. Dessa forma, o sistema contribui para democratizar o acesso à ciência de dados, tornando o processo de análise e modelagem mais acessível, eficiente e transparente.

**Link para o repositório do GitHub**

<https://github.com/Luciano-Franzoi-Filho/streamlit>

### **Referências**

BISHOP, Christopher M. Pattern Recognition and Machine Learning. New York: Springer, 2006.

HYNDMAN, Rob J.; ATHANASOPOULOS, George. Forecasting: principles and practice. 2. ed. Melbourne: OTexts, 2018. Disponível em: [https://otexts.com/fpp2/](vscode-file://vscode-app/c:/Users/lucia/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html). Acesso em: 16 jun. 2025.

KLUYVER, Thomas et al. Jupyter Notebooks as a publishing format for reproducible workflows. In: LOIZIDES, Fernando; SCHMIDT, Birgit (ed.). Positioning and Power in Academic Publishing: Players, Agents and Agendas. Amsterdam: IOS Press, 2016. p. 87-90. Disponível em: [https://eprints.soton.ac.uk/403913/](vscode-file://vscode-app/c:/Users/lucia/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html). Acesso em: 12 jun. 2025.

NORMAN, Donald A. The Design of Everyday Things. Revised and Expanded Edition. New York: Basic Books, 2020.

DOSHI-VELEZ, Finale; KIM, Been. Towards a rigorous science of interpretable machine learning. arXiv preprint arXiv:1702.08608, 2017. Disponível em: [https://arxiv.org/abs/1702.08608](vscode-file://vscode-app/c:/Users/lucia/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html). Acesso em: 15 jun. 2025.

PEDREGOSA, Fabian et al. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, v. 12, p. 2825-2830, 2011. Disponível em: [https://jmlr.org/papers/v12/pedregosa11a.html](vscode-file://vscode-app/c:/Users/lucia/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html). Acesso em: 1 jun. 2025.