Pontifícia Universidade Católica
do Rio de Janeiro

Antonio Pedro Santos Alves

Naming the Pain in Machine-Learning Enabled Systems Engineering

Projeto Final de Programação

Projeto apresentado ao Programa de Pós-graduação em Informática do Departamento de Informática da PUC-Rio como requisito parcial para obtenção do grau de Mestre em Informática.

Orientador: Prof. Marcos Kalinowski

Rio de Janeiro
Dezembro 2022

Sumário

1
Introdução

Os avanços e benefícios da área de Machine Learning na indústria e sociedade são inquestionáveis. Além da velocidade para geração e tomada de decisões, estes sistemas inteligentes oferecem resultados e conclusões que nós, humanos, poderíamos demorar demasiadamente para descobrir.

Entretanto, com avanços tão significativos e rápidos em ML, uma área muito importante na Computação foi sendo pouco relacionada com este tópico tão proeminente, a Engenharia de Software. (SCULLEY et al., 2015) trouxe ao mundo a preocupação de engenheiros de software do Google sobre a manutenção e evolução de sistemas de ML. Apesar do sucesso dos seus produtos, os pesquisadores identificaram uma série de causas que afetavam desde o desenvolvimento até o monitoramento desses sistemas, aspectos que são objetos de estudo da Engenharia de Software e não estavam sendo seguidos de maneira adequada, ou, simplesmente, eram ignorados.

2
Especificação

Com o objetivo de fazer novas descobertas sobre o desenvolvimento de projetos de ML na atualidade e avaliar o comportamento observado nas equipes e projetos do Google na indústria como um todo, o professor Marcos Kalinowski, eu e uma série de outros pesquisadores, nacionais e internacionais, disponibilizamos um *survey* de avaliação sobre as dores encontradas no desenvolvimento de sistemas de ML entre membros da indústria e outros pesquisadores. O *survey* foi elaborado com auxílio da ferramenta UniPark [1], que facilita a criação das perguntas mas não oferece tanto suporte para a visualização dos resultados.

2.1
Escopo

Dado a dificuldade de análise dos dados deste *survey*, este projeto define um arcabouço de soluções e visualizações pré-estabelecidas para melhor entendimento sobre os dados coletados deste *survey*, garantindo ainda que o usuário tenha a capacidade de explorar e desenvolver novas formas de visualização dos dados.

O sistema deve ser capaz de permitir que o usuário faça uso tanto dos recursos desenvolvidos e que trazem visões padronizadas para diferentes tipos de questões, bem como permitir que o mesmo seja capaz de desenvolver novas visualizações e *insights* a partir de outros recursos e bibliotecas disponíveis na Internet.

2.2
Especificação de Requisitos

Esta seção apresenta tanto os requisitos funcionais quanto os não-funcionais do projeto desenvolvido.

2.2.1
Requisitos Funcionais

[RF1] O sistema deve ser capaz de gerar pelo menos uma visualização gráfica para cada questão.

[RF2] O sistema deve permitir que o usuário formate o nome das colunas de outras maneiras.

[RF3] O sistema deve permitir que o usuário altere quais colunas devem ser removidas do arquivo com respostas do *survey*.

[RF4] O sistema deve ser capaz de aplicar a técnica de *bootstrapping* para toda questão fechada disponível (i.e, questões que não são de texto livre).

[1] https://www.unipark.com/

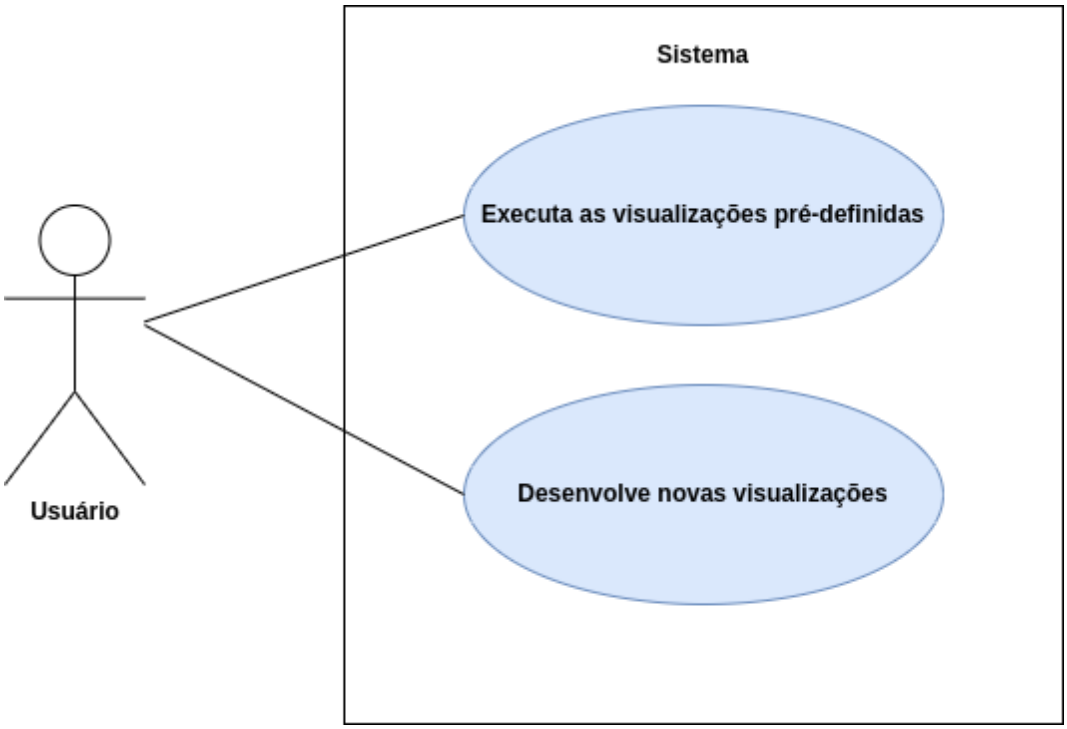

Figure 2.1: Diagrama de Caso de Uso

[RF5] O sistema deve fornecer a confiança estatística, com valor mínimo, média e máximo, para cada questão que passou pelo processo de *bootstrapping*.

2.2.2 Requisitos Não-Funcionais

[RNF1] O sistema deve executar em qualquer uma das três distribuições: Windows, Linux e MacOS.

[RNF2] O sistema não deve aceitar arquivos que não estejam na extensão CSV.

[RNF3] O sistema deve ser possível de ser utilizado após a leitura do Guia do Usuário.

[RNF4] O sistema deve executar dentro de um ambiente virtual criado a partir do arquivo *Pipfile* estabelecido.

2.2.3 Casos de Uso

O diagrama de caso de uso é mostrado na Figura 2.1

2.2.4 Descrição dos Casos de Uso

Esta subseção descreve os casos de uso encontrados na Figura 2.1.

O primeiro caso de uso descreve o relacionamento do usuário com as visualizações pré-estabelecidas, como segue:

UC1 - Executa as visualizações pré-definidas

Ator: Usuário

Visão Geral: O usuário abre o terminal dentro da pasta do projeto e digitar o comando, *pipenv shell*. Após o comando ser digitado, o usuário deve esperar a conclusão do mesmo para então digitar o comando: *jupyter notebook*. Uma aba no navegador irá se abrir e o usuário terá acesso às análises já desenvolvidas.

Trigger: Não existe nenhum.

Pré-Condições: O usuário deve assegurar que sua máquina possui o *pipenv* e *Python 3.10* instalados

Pós-Condições: O usuário terá acesso a todas as análises já estabelecidas.

Sequência Esperada de Ações:

Ações do Ator	**Respostas do Sistema**
1. Ator digita os comandos no terminal	
	2. O sistema instala as dependências e abre uma aba no navegador padrão do usuário
3. Ator seleciona uma das análises de questão	
4. Ator executa todas as células	
	5. Sistema calcula e exibe os resultados das análises pré-estabelecidas

Fluxos Alternativos:

- 2a: O sistema não consegue instalar todas as dependências e exibe uma mensagem de erro.
- 4a: O ator não executa todas as células, mas somente algumas em específicos.

O segundo, e último, caso de uso descreve o relacionamento do usuário com as novas visualizações, como segue:

UC2 - Desenvolve novas visualizações

Ator: Usuário

Visão Geral: O usuário abre o terminal dentro da pasta do projeto e digitar o comando, *pipenv shell*. Após o comando ser digitado, o usuário deve esperar a conclusão do mesmo para então digitar o comando: *jupyter notebook*. Uma aba no navegador irá se abrir e o usuário terá acesso às análises já desenvolvidas.

Trigger: Não existe nenhum.

Pré-Condições:

- O usuário deve assegurar que sua máquina possui o *pipenv* e *Python 3.10* instalados
- O usuário deve assegurar que toda biblioteca nova e fora das especificadas no *Pipfile* devem ser instaladas dentro do ambiente virtual com o comando *pipenv install <nome-biblioteca>*

Pós-Condições: O usuário será capaz de desenvolver e visualizar novos gráficos sobre as análises e variáveis já desenvolvidas.

Sequência Esperada de Ações:

Ações do Ator	**Respostas do Sistema**
1. Ator digita os comandos no terminal	
	2. O sistema instala as dependências e abre uma aba no navegador padrão do usuário
3. Ator seleciona uma das análises de questão	
4. Ator executa todas as células	
	5. Sistema calcula e exibe os resultados das análises pré-estabelecidas
6. Ator cria uma nova célula com o intuito de visualizar o dado com uma outra biblioteca	
7. Ator instala biblioteca no ambiente virtual	
	8. Sistema anexa a biblioteca nova no ambiente
9. Ator escreve e modifica as variáveis já definidas para adequá-las à nova biblioteca instalada e executa a célula	
	10. Sistema calcula e exibe os novos resultados da análise realizada.

Fluxos Alternativos:

- 2a: O sistema não consegue instalar todas as dependências e exibe uma mensagem de erro.
- 7a: O sistema não consegue instalar a nova biblioteca e exibe uma mensagem de erro.

3
Projeto do Programa

Para atingir os objetivos do sistema, bem como os requisitos especificados, o sistema foi desenvolvido em ***Python*** em conjunto com os ***notebooks*** da biblioteca ***Jupyter***. A liberdade e flexibilidade que os *notebooks* oferecem permitiu que fosse desenvolvido um *notebook* base com funções e classes úteis para o gerenciamento dos dados do *survey* para cada uma das questões. Contudo, essa estrutura não é restrita apenas às visualizações e estratégias já definidas, dado que o usuário pode aproveitar as variáveis e métodos definidos para criar novos gráficos e gerar novos *insights* com esforço reduzido. A Figura 3.1 apresenta uma visão macro da arquitetura da solução construída.

3.1
Diagrama de Classes

Os recursos desenvolvidos em Python foram baseados em três classes principais, que reúnem atributos e métodos específicos para gerenciamento dos dados obtidos. As classes são definidas como segue e de acordo com a Figura 3.2.

- **DataframeUtils**, responsável pelo armazenamento e gerenciamento das respostas do *survey* em um DataFrame, com diversos métodos facilitados para operações complexas.
- **BootstrappingUtils**, responsável pela aplicação da técnica de *bootstrapping* sobre as questões do *survey*.
- **PlotUtils**, responsável pela gestão e criação de visualizações com parâmetros simplificados.

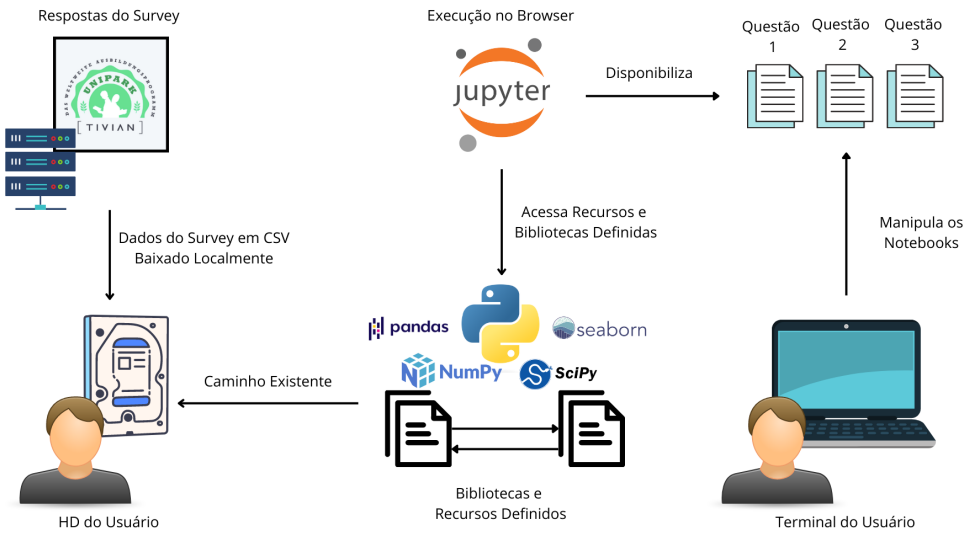

Figure 3.1: Arquitetura da Solução

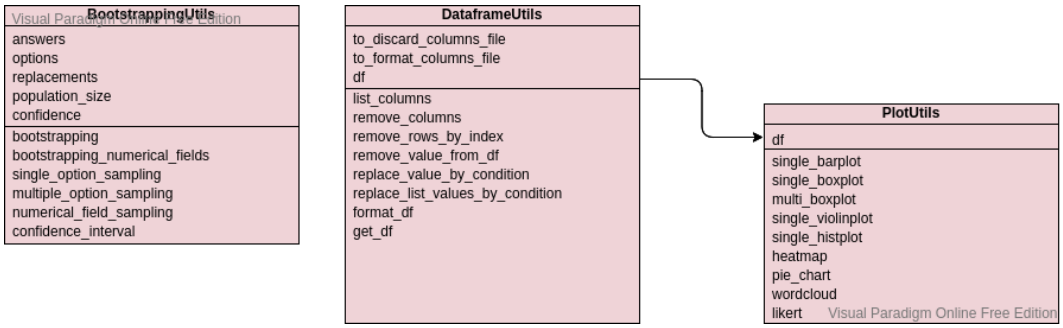

Figure 3.2: Diagrama de Classe

3.2 Tecnologias Utilizadas

Por se tratar de um sistema desenvolvido em Python, que possui diversas bibliotecas disponíveis, é importante listar as que são específicas do projeto. Neste caso:

- ***pandas***
- ***notebook***
- ***matplotlib***
- ***scipy***
- ***numpy***
- ***seaborn***
- ***pexpect***
- ***ipynb***
- ***plot-likert***
- ***wordcloud***
- ***coverage***

3.3 Modelagem de Dados

Uma vez que as análises e imagens geradas ficam armazenadas diretamente nos *notebooks* e não em um banco de dados específico para cada questão, não houve necessidade de realizar uma modelagem de dados.

4
Código Fonte

O código deste projeto está disponível no GitHub [1], ferramenta usada tanto para acesso dos usuários, quanto para versionamento do mesmo. Destaca-se que o código foi escrito em Python e seguindo os padrões de estilo e design de código estabelecidos na PEP8[2], referência para a linguagem. Ademais, apesar de ser uma linguagem dinamicamente tipada, é possível especificar nos cabeçalhos de funções e retornos das mesmas, tipos esperados. Isso aumenta a confiabilidade das funções e melhora a legibilidade do código durante o desenvolvimento e chamada dessas funções. Em conjunto com os comentários no início de cada função, o resultado pro usuário quando vemos os detalhes de uma função seguem o da Figura 4.1.

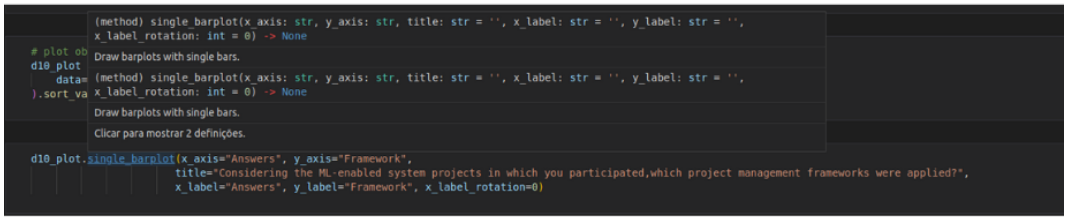

Figure 4.1: Anotações e Tipagem

Outro aspecto importante do código é que ele foi desenvolvido dentro de um ambiente virtual em Python, ou seja, basta ativar o ambiente que as bibliotecas utilizadas e as corretas versões das mesmas são utilizadas, assegurando o funcionamento correto do código e escapando de possíveis conflitos de versão.

[1] https://github.com/antoniopedro98/pfp-antonio-pedro
[2] https://peps.python.org/pep-0008/

5
Testes

Testes Unitários automatizados foram realizados de modo a assegurar a confiabilidade do sistema. Por se tratar se um sistema em Python, foi utilizada a biblioteca ***unittest***, que realiza os testes unitários propriamente ditos, em conjunto com a biblioteca ***coverage***, responsável por realizar um teste de cobertura sobre o código testado.

5.1
Critérios de Teste

Os critérios de teste utilizados consideraram todos os módulos disponíveis e passíveis de terem o seu retorno, de alguma forma, avaliados. Nesse caso, apenas o módulo ***PlotUtils*** e a função *listcolumns* do módulo ***BasicUtils*** não foram incluídas nos testes unitários pois não eram claros sobre como poderiam ser testados, já que se tratavam de criação de gráficos e *prints* na tela, respectivamente. Em outras palavras, somente os módulos e métodos que resultavam em mudança de estado de variáveis foram testados.

Os testes estão disponíveis na pasta *tests* do repositório do GitHub e cada um possui descrições e motivações para terem sido realizados.

5.2
Resultado dos Testes

Os resultados dos testes unitários são obtidos de maneira automática ao executar o comando na pasta base do projeto, *python -m unittest -v*. Contudo, foi implementado uma funcionalidade que realiza uma cobertura dos testes realizados, permitindo que se avalie de modo dinâmico os trechos de código não executados e executados. Para isso, basta executar os seguintes comandos na pasta base do projeto,

1. *python -m coverage run -m unittest*
2. *python -m coverage html*

Após estes comandos, uma pasta chamada ***htmlconv*** será gerada e dentro dela um arquivo chamado *index.html*, que contém uma página Web simples contendo o relatório de execução de todos os testes realizados. No repositório deste projeto, esta pasta já está criada e contém o último relatório de testes realizados sobre o sistema, como mostra a Figura 5.1

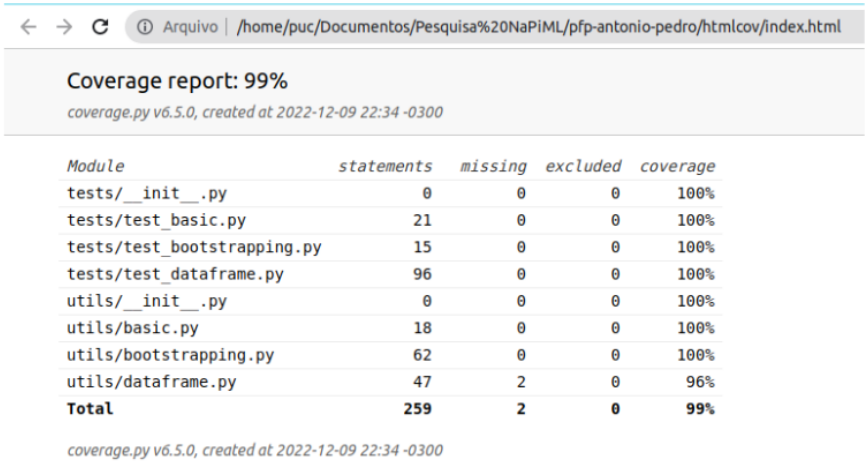

Figure 5.1: Cobertura de Testes

6
Documentação para o Usuário

Nesta seção, alguns pontos sobre a utilidade e utilização do sistema serão respondidas.

6.1
Público-Alvo

Os usuários do projeto são principalmente o grupo de pesquisadores envolvidos no *survey* elaborado. Por se tratar de uma equipe extensa e multidisciplinar, o projeto foi desenvolvido com recursos mais básicos, para os que estão pouco familiarizados com a linguagem Python, e robustos, para os que possuem maior expertise com programação.

Contudo, o projeto também pode ser útil para outros pesquisadores que adotam a ferramenta UniPark para elaboração de *surveys*, uma vez que os arquivos de respostas dos mesmos são padronizados. A adequação do projeto para estes *surveys* se resumiria a troca do nome das colunas e algumas poucas modificações em notebooks específicos.

6.2
Contexto de Atividade

O projeto está no contexto de avaliação de resultados do *survey*. Ele não apoia a construção ou edição do mesmo, somente análise das respostas já recuperadas e armazenadas como CSV. Neste sentido, o programa ajuda no entendimento dos dados com gráficos diversos e acesso ao conteúdo do arquivo CSV de modo facilitado e rápido.

6.3
Instalação e Execução

A instalação e execução do projeto depende exclusivamente do *clone* do projeto do GitHub e acopmanhamento das instruções presentes no *README* do repositório. De modo resumido, para instalar corretamente o projeto é necessário ter a versão do ***Python 3.10***, bem como o utilitário ***pipenv***.

Em caso de insucesso na instalação, o GitHub conta com a aba *Issues*, na qual eu serei notificado caso algum usuário tenha algum problema nesta etapa.

6.4
Tarefas Apoiadas

As tarefas apoiadas residem, sobretudo, na análise dos dados do *survey* elaborado, permitindo análises sofisticadas e completas com poucos comandos, desde um gráfico de barra ao comportamento estatístico da questão sobre uma população (obtido com *bootstrapping*).

6.5 Contato

Em caso de melhorias e sugestões, na descrição do projeto no GitHub, está o meu e-mail para contato.

7
Bibliografia

SCULLEY, D. et al. Hidden technical debt in machine learning systems. In: CORTES, C. et al. (Ed.). **Advances in Neural Information Processing Systems**. Curran Associates, Inc., 2015. v. 28. Disponível em: <https://proceedings.neurips.cc/paper/2015/file/86df7dcfd896fcaf2674f757a2463eba-Paper.pdf>.
Cited in page 3.