Universidade Federal de Ouro Preto
Escola de Minas
CECAU - Colegiado do Curso de
Engenharia de Controle e Automação

Victória Mara da Silva Araújo

XAI *Web*: Aplicação *Web* para Explicar Previsões Realizadas por Modelos de Aprendizado de Máquina

Monografia de Graduação

Ouro Preto, 2023

Victória Mara da Silva Araújo

XAI *Web*: Aplicação *Web* para Explicar Previsões Realizadas por Modelos de Aprendizado de Máquina

Trabalho apresentado ao Colegiado do Curso de Engenharia de Controle e Automação da Universidade Federal de Ouro Preto como parte dos requisitos para a obtenção do Grau de Engenheira(o) de Controle e Automação.

Universidade Federal de Ouro Preto

Orientador: Prof. Rodrigo Cesar Pedrosa Silva

Coorientador: Prof. Adrielle De Carvalho Santana

Ouro Preto

2023

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE OURO PRETO
REITORIA
ESCOLA DE MINAS
DEPARTAMENTO DE ENGENHARIA CONTROLE E AUTOMACAO

FOLHA DE APROVAÇÃO

Victória Mara da Silva Araújo

XAI web: Aplicação web para explicar previsões realizadas por modelos de aprendizado de máquina

Monografia apresentada ao Curso de Engenharia de Controle e Automação da Universidade Federal de Ouro Preto como requisito parcial para obtenção do título de bacharel em Engenharia de Controle e Automação

Aprovada em 15 de agosto de 2023

Membros da banca

Prof. Dr. Rodrigo Cesar Pedrosa Silva - Orientador (DECOM - Universidade Federal de Ouro Preto)
Profa. Dra. Adrielle de Carvalho Santana - Coorientadora (DECAT - Universidade Federal de Ouro Preto)
M. Sc. Henrique Oliveira Duarte - Convidado (Universidade Federal de Ouro Preto)
Prof. Dr. Agnaldo José da Rocha Reis - Convidado (DECAT - Universidade Federal de Ouro Preto)

Rodrigo Cesar Pedrosa Silva, orientador do trabalho, aprovou a versão final e autorizou seu depósito na Biblioteca Digital de Trabalhos de Conclusão de Curso da UFOP em 21/08/2023

Documento assinado eletronicamente por **Rodrigo Cesar Pedrosa Silva**, **PROFESSOR DE MAGISTERIO SUPERIOR**, em 21/08/2023, às 15:17, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0 , informando o código verificador **0573394** e o código CRC **3F9DCE13**.

Resumo

A explicabilidade e a interpretabilidade de modelos de inteligência artificial tem crescido em importância quanto mais a utilização desses modelos e suas aplicações ganham destaque na sociedade. Contudo, uma vez desenvolvido um modelo não interpretável, chamado de caixa-preta, essa interpretabilidade só poderá ser alcançada aplicando-se métodos de inteligência artificial explicável. Tais métodos, para serem utilizados, têm como requisito básico a programação, o que dificulta às diversas análises que podem ser feitas em cima de um modelo. Os modelos podem ser desenvolvidos para diferentes áreas de conhecimento e os seus resultados podem ser posteriormente avaliados por diferentes profissionais capacitados, afim de validar o modo como o modelo tratou as informações. Biólogos, economistas, publicitários e diversos outros profissionais ou estudantes podem eventualmente precisar de uma maior transparência ao avaliar e aplicar os resultados de modelos, mesmo que não possuam conhecimento em programação. Neste trabalho foi desenvolvida uma aplicação *web*: o XAI *web*, responsável por aplicar os métodos de interpretabilidade *permutation*, SHAP e LIME em modelos de aprendizado de máquina supervisionado com o objetivo de tornar esses métodos mais acessíveis para usuários não programadores. Ao final do trabalho, as informações disponibilizadas pela aplicação, através de visualizações gráficas e metadados, podem auxiliar os profissionais na detecção de vieses, permitindo auditoria aos modelos fornecendo melhor entendimento dos modelos de inteligência artificial e assim aumentando a confiança e responsabilidade em suas utilizações.

Palavras-chaves: inteligência artificial explicável, aprendizado de máquina, aprendizado supervisionado, *permutation*, LIME, SHAP.

Abstract

The explainability and interpretability of artificial intelligence models have been growing in importance as the use of these models and their applications become more prominent in society. However, once a non-interpretable model, termed a ”black-box”, is developed, this interpretability can only be achieved by applying explainable artificial intelligence methods. To use such methods, basic programming is a requirement, which complicates various analyses that can be conducted on a model. This is because models can be developed for different fields of knowledge, and their results can later be evaluated by various qualified professionals to validate the way the model handled the information. Biologists, economists, advertisers, and many other professionals or students might occasionally need greater transparency when evaluating and applying the results of models, even if they have no knowledge of programming. In this study, a web application was developed: the XAI web, responsible for applying the interpretability methods permutation, SHAP, and LIME on supervised machine learning models with the aim of making these methods more accessible to non-programmers. At the end of the study, the information provided by the application, through graphical visualizations and metadata, can assist professionals in bias detection, allowing for model auditing, providing a better understanding of artificial intelligence models, and thereby increasing trust and responsibility in their uses.

Key-words: explainable artificial intelligence. machine learning, supervised learning, permutation, LIME, SHAP.

Sumário

Lista de ilustrações

Lista de tabelas

1 Introdução

Segundo Zhou (2021) algoritmos de aprendizado de máquina, do inglês *Machine Learning*(ML), têm como objetivo aumentar o desempenho de sistemas utilizando dados para criar modelos, ou seja, são algoritmos que aprendem com base em conjuntos de dados fornecidos e conseguem, assim, realizar previsões, regressões e classificações. Esse funcionamento faz com que essa ferramenta seja útil em diferentes áreas, como, por exemplo: recomendação de produtos (LUDERMIR, 2021), veículos autônomos (FENG et al., 2021) e reconhecimento de imagens (MIRANDA, 2011).

Alguns dos algoritmos de aprendizado de máquina funcionam com aprendizagem supervisionada que, conforme Monard e Baranauskas (2003) explicam, é um processo de aprendizagem no qual, dentro do conjunto de amostras de treinamento, são fornecidos dados com rótulos de classes conhecidos previamente. O sistema tem então o desafio de aprender a mapear os novos dados para as saídas desejadas com base nos exemplos. Para isso, o algoritmo encontra os padrões dos dados e os representa em um modelo matemático (WOLFGANG, 2017).

Nos últimos anos, a utilização dessas ferramentas cresceu consideravelmente, visto tanto o aumento do poder computacional quanto as diversas aplicações possíveis (NETO, 2021). Nesse sentido, quando uma previsão é realizada a partir de um algoritmo de ML, muitas vezes, é interessante entender melhor esse resultado. Entretanto, nem sempre os modelos criados são de fácil compreensão, à medida que os dados e o poder computacional aumentam, os modelos ficam mais complexos e mais difíceis de interpretar (NETO, 2021). É nesse contexto que técnicas de inteligência artificial explicável, do inglês *Explainable Artificial Intelligence* - XAI estão ficando cada vez mais relevantes.

De acordo com Langer et al. (2021), pode-se entender XAI como uma área de pesquisa multidisciplinar que tem como propósito desenvolver ferramentas capazes de tornar modelos de Inteligência Artificial (IA) mais compreensíveis para os seres humanos. Assim, esses métodos podem aumentar a confiabilidade nos resultados, facilitar a detecção de viéses, possibilitar maior entendimento dos modelos e das predições e se adequarem melhor em aplicações de maior risco, como segurança, finanças e saúde (NETO, 2021).

Para Molnar (2022), essa interpretabilidade pode ser alcançada entendendo como as decisões estão sendo feitas pelo algoritmo, isto é, observando suas características, os padrões aprendidos, os pesos e a estrutura do modelo. Esse conceito é a base do funcionamento das ferramentas XAI. Exemplificando, existem métodos tais como: (1) *permutation feature importance* que procura demonstrar a importância de cada atributo para o modelo (KANEKO, 2022), (2) SHAP que calcula qual a contribuição positiva ou negativa dos atri-

butos para o modelo (ULLAH et al., 2023), e (3) LIME, um método local, que avalia como o algoritmo classifica exemplos simulados (RIBEIRO; SINGH; GUESTRIN, 2016). Além disso, esses algoritmos facilitam a compreensão de resultados através de visualizações gráficas.

Atualmente, essas metodologias são aplicadas, em geral, por meio de desenvolvimento em Python, pois existem diversas bibliotecas implementadas nessa linguagem para essa finalidade. Por exemplo, bibliotecas como *explainerdashboard* (EXPLAINERDASHBOARD, 2020), *Shapash* (DOCUMENTATION, Shapash's, 2020) e *Dalex* (SETH, 2021) que reúnem vários métodos de interpretabilidade gerando diversos gráficos interativos para melhorar o entendimento dos modelos. Contudo, para a utilização de todas essas bibliotecas é necessário realizar programação em Python e, nos ambientes corporativos, a análise das respostas encontradas e da aplicação desenvolvida poderá ser feita por outras pessoas além dos desenvolvedores que implementaram o modelo. Além disso, por se tratar de uma área muito importante no mercado, na educação e no desenvolvimento tecnológico, é importante que o acesso a essas explicações esteja cada vez mais facilitado.

Analisando-se todos os pontos apresentados e considerando-se a relevância dos métodos *permutation*, SHAP e LIME que são todos utilizados pelas bibliotecas *explainerdashboard* (EXPLAINERDASHBOARD, 2020), *Shapash* (DOCUMENTATION, Shapash's, 2020) e *Dalex* (SETH, 2021), encontrou-se a oportunidade de criar recursos para agregar na utilização desses métodos, sob a forma de um sistema que não precisa de programação para ser utilizado e que permite ao usuário interpretar seu modelo pelos três métodos de maneira mais simples e rápida, justificando-se assim a importância de se realizar este trabalho.

1.1 Objetivos

1.1.1 Objetivo Geral

Tornar algumas técnicas de XAI, para modelos de ML de aprendizado supervisionado, mais acessíveis aos usuários, desenvolvendo uma aplicação *web* que demonstre graficamente a influência dos parâmetros nas previsões realizadas por esses modelos: o XAI *web*.

1.1.2 Objetivos Específicos

Como objetivos específicos, destacam-se:

- Pesquisar técnicas de XAI para modelos de aprendizado de máquina;
- Desenvolver o *front-end* que permita ao usuário enviar o código e os dados e visualizar os resultados obtidos;

- Desenvolver o *back-end* que aplique os métodos: Permutation, SHAP e LIME e retorne os resultados para o usuário visualizar graficamente;
- Analisar a aplicação desenvolvida e seu funcionamento.

1.2 Estrutura do trabalho

O presente trabalho foi dividido em cinco capítulos. O Capítulo 1 introduz o assunto abordado em todo o projeto e discute a motivação de sua realização. No Capítulo 2 é apresentado o referencial teórico utilizado como base para o desenvolvimento da aplicação descrevendo os conceitos envolvidos, os métodos de interpretabilidade e explicabilidade utilizados e o estudo de soluções existentes com funcionamento similar a aplicação proposta. No Capítulo 3 foi apresentado o desenvolvimento da aplicação junto com as ferramentas utilizadas para a criação do sistema. No Capítulo 4 e 5, foram apresentados, respectivamente, os resultados alcançados e a conclusão, além das sugestões de trabalhos futuros.

2 Fundamentação Teórica

2.1 Aprendizado de máquina - ML

O aprendizado de máquina é uma parte da inteligência artificial, na qual, os algoritmos recebem um objetivo e uma grande quantidade de dados e precisam aprender a alcançar a saída desejada (SOCIETY, 2019). O aprendizado de máquina pode ser dividido em; (1) aprendizado não supervisionado, no qual o algoritmo tenta realizar agrupamentos dos dados encontrando características em comum; (2) supervisionado, no qual o algoritmo recebe alguns atributos rotulados e tem como função rotular os atributos que não tem rótulos conhecidos; (3) semi-supervisionado, no qual existem tanto dados rotulados como não rotulados, sendo essa técnica a mistura das técnicas (1) e (2); e (4) por reforço, no qual o sistema recebe recompensas ou punições do ambiente que está inserido à cada ação realizada e deve aprender a maximizar as recompensas agindo corretamente (MONARD; BARANAUSKAS, 2003).

No aprendizado supervisionado têm-se um processo de aprendizagem conhecido como aprendizado indutivo, no qual, utilizando-se do conjunto de treinamento como exemplo o algoritmo cria generalizações e especializações e consegue rotular novos dados, sendo por conta disso conhecido como classificador. Quando essa classificação é feita para classes discretas, o problema é identificado como classificação e quando a classe é contínua o problema é identificado como regressão (SANCHES, 2003).

2.2 Inteligência Artificial Explicável - XAI

Conforme é ilustrado por Das e Rad (2020) a explicabilidade é uma forma de compreender as decisões tomadas por algoritmos de inteligência artificial que busca garantir maior confiança no modelo. Alguns modelos de aprendizagem de máquina, por exemplo, são tão complexos que são denominados de modelos caixa-preta, e isso acontece porque existe uma relação direta do aumento da acurácia e a diminuição da interpretabilidade do modelo (GILPIN et al., 2018). Nesse sentido, pode-se entender os métodos XAI como ferramentas que buscam tornar essa caixa-preta numa caixa transparente, na qual é possível analisar o porquê do algoritmo ter chegado em certo resultado.

Essas ferramentas fornecem diferentes tipos de explicação, como descrições do funcionamento do sistema, visões gerais de como as representações são criadas e sistemas paralelos que geram uma saída e uma explicação usando diferentes modelos (SOCIETY, 2019). Considerando que a IA utiliza entradas para definir saídas, a IA explicável procura

demonstrar a importância das entradas para definição das saídas obtidas (ANKARSTAD, 2020).

Os ganhos alcançados com a explicabilidade são inúmeros pois ela permite aos desenvolvedores avaliarem se o algoritmo está funcionando como o esperado, assim como serve de ferramenta para evidenciar características ou padrões que podem fornecer percepções para melhoria e otimização dos resultados. Além disso, é necessária para que certas aplicações atendam regulamentações, para identificação de preconceitos e para um uso responsável (IBM, 2022).

Por fim, os algoritmos de explicabilidade recebem algumas classificações conforme seu funcionamento e suas utilidades que são descritas a seguir.

2.2.1 Global x Local

Essa categoria divide os métodos naqueles que são aplicados; (1) localmente, realizando uma análise de um parâmetro de entrada individualmente; (2) globalmente, analisando o modelo a partir de todos os seus parâmetros; e (3) métodos, que podem ser tanto global quanto local dependendo da sua implementação (DAS; RAD, 2020).

2.2.2 *Transparent* x *Post-Hoc*

O método de interpretação pode ser classificado como *transparent* quando ele está inserido no próprio modelo, ou seja, quando o modelo é simples o suficiente para que seja possível entender o resultado analisando o próprio funcionamento do algoritmo. Um exemplo, são os modelos de árvores de decisão. Enquanto métodos *post-hoc* são aplicados depois do treinamento do modelo para tornar os resultados mais claros (MOLNAR, 2022).

2.2.3 Modelo-Específico x Modelo-Agnóstico

A classificação em modelo-específico, do inglês *model-specific*, ou a classificação em modelo-agnóstico, do inglês *model-agnostic* está relacionada com os modelos para os quais o método pode ser aplicado. Modelos específicos podem ser usados apenas para o tipo de modelo para o qual foi desenvolvido, enquanto, modelos agnósticos podem ser usados em qualquer modelo (NETO, 2021).

2.3 Métodos XAI

2.3.1 *Permutation feature importance*

Inicialmente proposto por Breiman (2001), o *permutation feature importance* é um método de aplicação global, *post-hoc* e modelo-agnóstico. A ideia por trás do método é realizar

permutações nos atributos e calcular o erro, dessa forma, cria-se um comparativo dos valores de erro encontrados. Quanto maior o valor do erro para aquela alteração mais relevante é aquele atributo, e, por outro lado, quando o erro não varia muito trata-se de uma característica menos significativa para o modelo em questão.

Matematicamente a estimativa do erro para o modelo original é calculada pela equação:

$$e_{orig} = L(y, f(X)) \tag{2.1}$$

Em que, *f*(X) é o modelo treinado; X é a matriz dos atributos; y é o valor de resultado esperado; e *L(y,f(X))* é a medida do erro.

Para calcular o *permutation* primeiro deve ser gerada para cada atributo j pertencente ao conjunto $\{1, .., p\}$ a matriz X_{perm} permutando os valores dos dados de X individualmente e de forma aleatória. O valor do erro para os dados permutados, será dado por:

$$e_{perm} = L(Y, f(X_{perm})) \tag{2.2}$$

Por fim, o cálculo do *permutation* será dado pela diferença:

$$FI_j = e_{perm} - e_{orig} \tag{2.3}$$

O FI é a sigla do inglês *Feature Importance* e representa a medida da importância relativa de cada atributo do modelo de aprendizagem, isso porque, representa a diferença no erro antes e depois de ter seu valor permutado. Depois de realizados todos os cálculos o método irá ordenar os valores encontrados do maior FI para o menor (MOLNAR, 2022). Dessa forma, será retornada a lista ordenada das características mais influentes no modelo e seus respectivos valores de influência.

O cálculo do *permutation importance* poderá ser feito com o conjunto de treinamento ou de validação ou teste. Para o conjunto de teste, por exemplo, o resultado demonstra atributos importantes para as generalizações do modelo. Outra análise que pode ser feita é que atributos importantes no conjunto de treinamento mas não importantes no conjunto de teste podem ser causa de *overfitting* no modelo porque isso indica que o sistema está super ajustado ao conjunto de treinamento e não está realizando boas generalizações de novos dados (SCIKIT-LEARN, 2023).

2.3.2 SHAP

O *Shapley Additive Explanations* - SHAP é um método *post-hoc* e modelo-agnóstico proposto por Lundberg e Lee (2017) que possui aplicação tanto global quanto local.

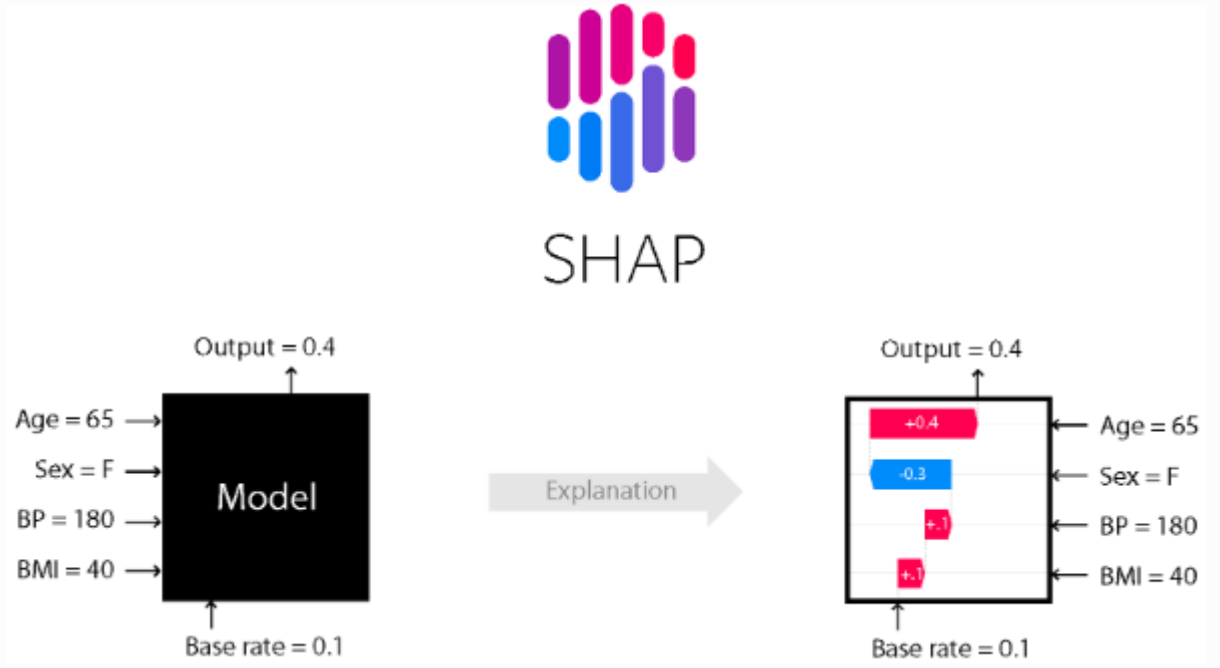


Figura 1 – Funcionamento do SHAP
Fonte: SHAP (2018)

Na figura 1, pode-se observar o objetivo geral das técnicas XAI aplicada ao SHAP: um modelo não interpretável caixa-preta sendo transformado em um modelo interpretável transparente. No primeiro momento, conhecem-se as entradas e a saída de um certo modelo e com a aplicação do SHAP novas informações dos valores de contribuição tanto positiva como negativa de cada parâmetro de entrada para o resultado ficam disponíveis, tornando possível uma análise do modelo e de seu resultado, de forma mais completa e aprofundada.

O SHAP funciona calculando as contribuições de cada atributo no resultado, baseando-se, na teoria dos valores Shapley de Lloyd Shapley. Isto é, é criado um paralelo entre os jogadores e suas contribuições para a equipe com os atributos e suas importâncias para o modelo (MENEGOL, 2020).

Os valores Shapley representam a contribuição de cada jogador individualmente no resultado do jogo. Para seu cálculo, é necessário descobrir o resultado com todas as configurações de jogadores possíveis, podendo incluir ou não o jogador avaliado e então realizar a soma ponderada da diferença entre as pontuações obtidas quando o jogador participou e quando não participou do jogo. Ao final, é aplicado um peso ao valor obtido (AMOROSO, 2023). A fórmula dos valores Shapley é definida pela equação 2.4 (SHAPLEY, 1951).

$$\varphi_i(v) = \frac{1}{n} \sum_{S \subseteq N \{i\}} \left(\frac{n-1}{|S|} \right)^{-1} [v(S \cup \{i\}) - v(S)] \tag{2.4}$$

Na qual, φ_i representa a contribuição do jogador i; v é a função que calcula a

contribuição da equipe; N é o conjunto de todos os jogadores; n é o total de jogadores; e S é o subconjunto de N.

Por fim, para avaliar a importância de um atributo para o resultado final de um modelo então, considera-se as variáveis como os jogadores e o valor de contribuição para a equipe seu impacto.

2.3.3 LIME

O *Local Interpretable Model-Agnostic Explanation* - LIME é um método modelo-agnóstico, *post-hoc* de aplicação local proposto por Ribeiro, Singh e Guestrin (2016). A ideia do método é explicar previsões individuais por meio de modelos locais substitutos do modelo caixa preta, ou seja, cria-se um novo conjunto de dados com amostras perturbadas e treina-se um novo modelo interpretável que é ponderado pela proximidade dos exemplos amostradas comparado à instância avaliada (SETH, 2021).

O funcionamento do LIME é explicado passo a passo em Molnar (2022):

1. É selecionada a instância para a qual será feita a explicação.

2. O conjunto de dados é perturbado e são feitas novas previsões com os dados alterados.

3. O novo conjunto de amostras é avaliado e descobre-se sua relevância de acordo com a proximidade da instância de interesse.

4. Um modelo ponderado e interpretável é treinado com o conjunto de dados com as variações.

5. A previsão pode ser explicada interpretando o modelo local.

As perturbações criadas pelo método no passo 2 vão variar de acordo com os dados, para imagens, por exemplo, poderá remover ou adicionar pixels assim como para arquivos de texto poderá adicionar ou remover palavras. O LIME também permite variar a escolha do modelo que vai ser treinado no passo 4 para permitir a escolha de um que melhor se adapta aos requisitos dos dados, podendo ser regressão linear, regressão logística ou árvores de decisão. O desempenho do método está muito relacionado com a quantidade de instâncias geradas próximo aquela de interesse, contudo, essa quantidade não é muito simples de ser definida o que torna o método um pouco mais suscetível a falhas. Ainda assim, é um método muito importante pois fornece explicações bastante amigáveis para os humanos e é um dos poucos que funciona para dados tabulares, textos e imagens.

2.4 Ferramentas XAI

Atualmente, desenvolvedores que procuram aplicar métodos de interpretabilidade podem usar bibliotecas Python como a *explainerdashboard* (EXPLAINERDASHBOARD, 2020), a *Shapash* (DOCUMENTATION, Shapash's, 2020) e a *Dalex* (DRWHY.AI, 2021). Todas elas têm uma funcionalidade similar ao sistema proposto neste trabalho reunindo métodos de interpretabilidade em uma só ferramenta. Por conta disso, nos próximos tópicos essas ferramentas serão discutidas e comparadas ao XAI *web*.

2.4.1 *explainerdashboard*

explainerdashboard é uma biblioteca implementada em Python que une diversos métodos de explicabilidade e interpretabilidade. Trata-se de uma ferramenta personalizável e para sua utilização é necessário a utilização de alguns comandos na linguagem Python, um ambiente de desenvolvimento Python e a configuração do ambiente com a instalação das bibliotecas necessárias.

Na figura 2 pode-se visualizar como é apresentada essa ferramenta para o usuário. Suas funcionalidades são todas divididas em abas, nas quais, diferentes gráficos serão disponibilizados. Os métodos SHAP e *permutation importance* estão incorporados na ferramenta, assim como vários outros (EXPLAINERDASHBOARD, 2020).

Para sua utilização, são necessárias as mesmas informações do XAI *web*, o modelo e os dados de testes divididos nos atributos X e nas saídas Y. Para uma maior compreensão do seu funcionamento e da necessidade de utilizar programação Python foi realizado um caso de teste descrito a seguir. Para isso, utilizou-se o conjunto de dados bastante utilizado na literatura, o *Breast Cancer dataset*. Inicialmente, foi preciso fazer a instalação da biblioteca utilizada:

```
pip install explainerdashboard
```

Logo após, as importações:

```
from explainerdashboard import explainerdashboard
from explainerdashboard import ClassifierExplainer
import pandas as pd
```

explainerdashboard e *ClassifierExplainer* são os dois métodos importados da biblioteca *explainerdashboard* que são necessários para a utilização inicial da ferramenta. Já o *pandas* é a biblioteca utilizada para fazer a leitura dos arquivos: o modelo , o conjunto de atributos "Xteste" e as saídas alvo "yteste". Após a leitura dos arquivos é feita a aplicação do modelo, conforme demonstrado nos comandos a seguir.

```
Xteste = pd.read_csv('X.csv')
```

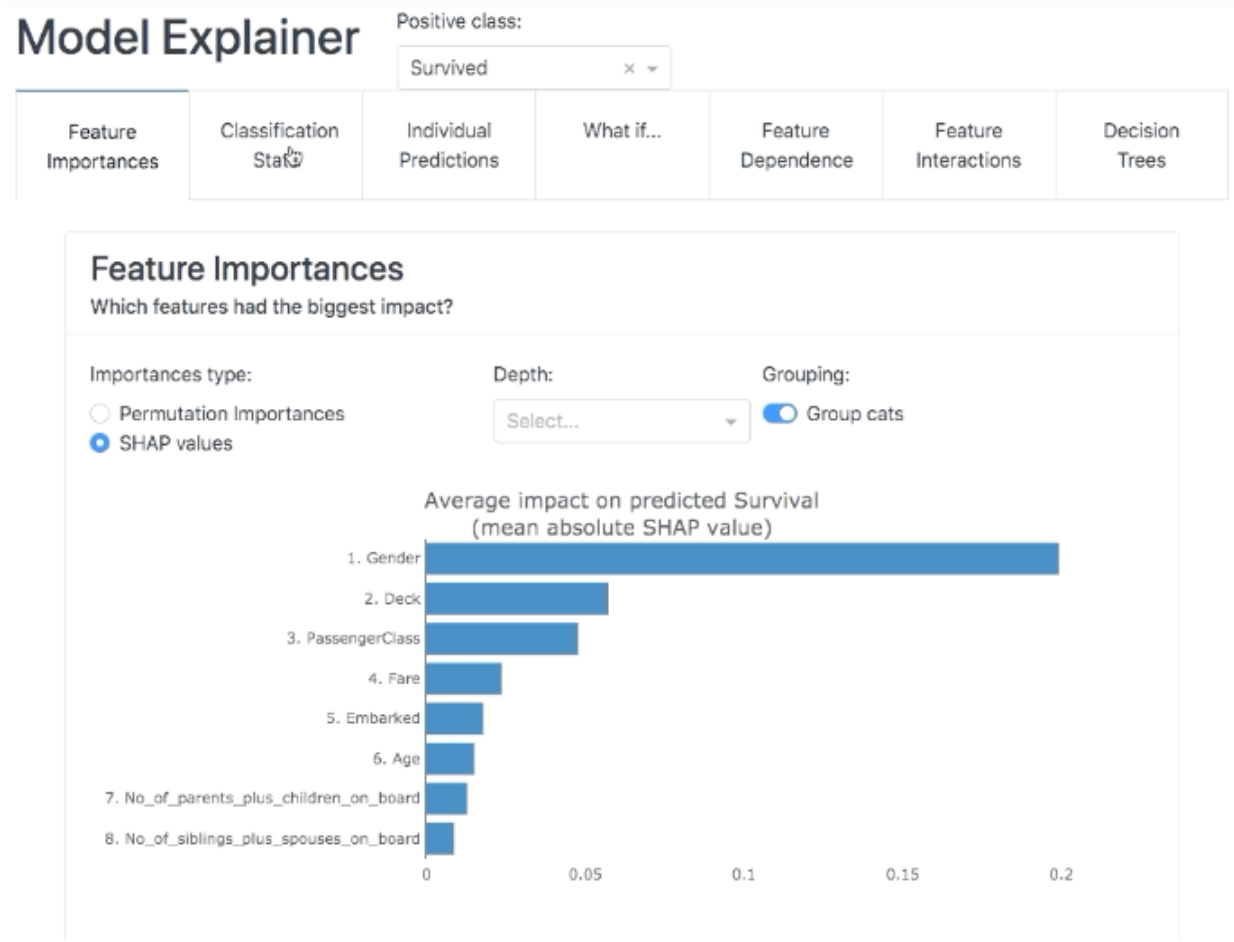


Figura 2 – Exemplo da biblioteca *explainerdashboard*
Fonte: explainerdashboard (2020)

```
Yteste = pd.read_csv('y.csv')
modelo = pd.read_pickle('teste.pck')
explicacao = ClassifierExplainer(modelo, Xteste, yteste)
```

No *ClassifierExplainer* é construído um objeto com as explicações do modelo e os dados de teste o "explicacao". Em seguida, esse objeto é passado como parâmetro do *explainerdashboard* para obtenção da *dashboard*:

```
explainerdashboard(explicacao).run()
```

Após executados todos esses passos, obtêm-se os gráficos conforme mostrado nas figuras 3 e 4.

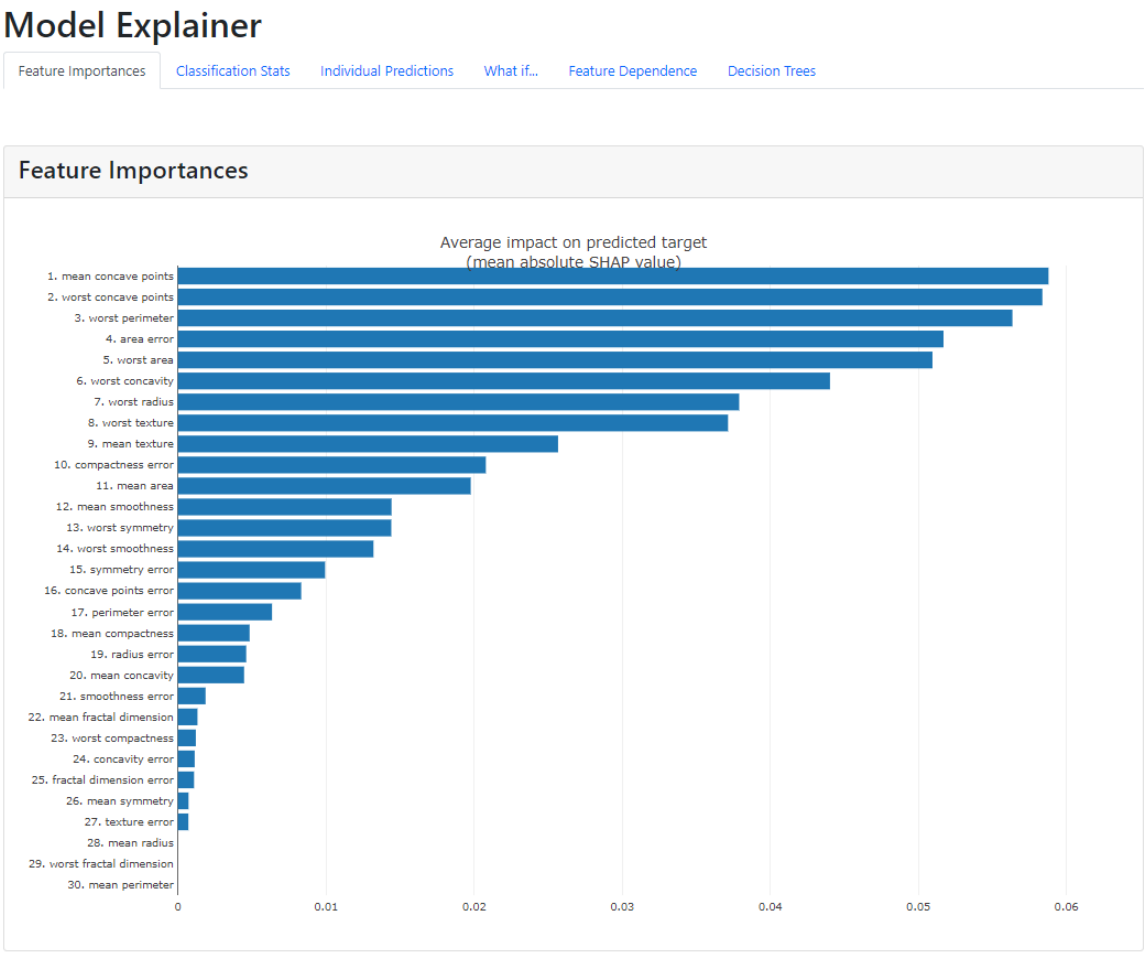


Figura 3 – Exemplo da *explainerdashboard* gerado com o *Breast Cancer dataset*
Fonte: Elaboração própria

Além do desenvolvimento básico demonstrado, é possível utilizar diversos outros comandos para personalizar o estilo do painel, títulos, alterar a ordem das seções ou plotar gráficos específicos com a biblioteca. Assim como a utilização mais básica demonstrada no exemplo, todos os comandos de personalização precisam ser feitos com desenvolvimento Python, demonstrando a dependência que a ferramenta tem da utilização de programação.

2.4.2 *Shapash*

A *Shapash* é outra biblioteca implementada em Python que permite aos usuários visualizarem diversos métodos de interpretabilidade de aprendizado de máquina em formato de uma *dashboard* interativa. Ela inclui os métodos Shap, *feature importance* e LIME e para sua utilização também é necessário desenvolvimento Python (DOCUMENTATION, Shapash's, 2020).

A figura 5 demonstra como a ferramenta é apresentada.

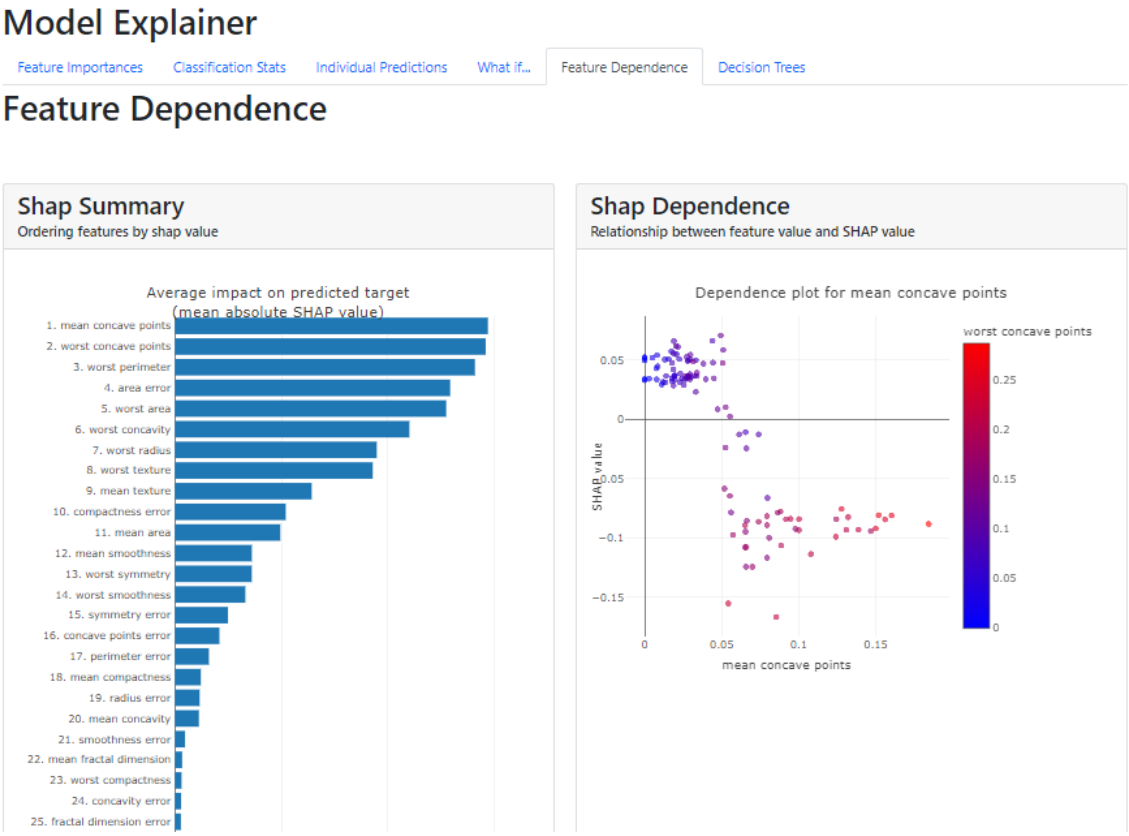


Figura 4 – Exemplo de outra aba do *explainerdashboard* gerado com o *Breast Cancer dataset*

Fonte: Elaboração própria

Figura 5 – Exemplo da biblioteca *Shapash*
Fonte: Shapash's Documentation (2020)

Para exemplificar seu uso e a necessidade de desenvolvimento Python, foi feito um

exemplo também com o *Breast Cancer dataset*. Inicialmente, a biblioteca foi instalada:

```
pip install Shapash
```

Logo após, foram feitas as importações e a leitura dos arquivos:

```
from Shapash import SmartExplainer
import pandas as pd
Xteste = pd.read_csv('X.csv')
Yteste = pd.read_csv('y.csv')
modelo = pd.read_pickle('teste.pck')
```

Em seguida, com o comando *SmartExplainer* é gerado uma instância do objeto chamada xpl que tem o modelo como parâmetro. A instância é, então, compilada com os dados de teste conforme mostra os comandos a seguir.

```
xpl = SmartExplainer(model=modelo)
xpl.compile(
    x=Xteste
)
```

Por fim, os gráficos são gerados pelo comando:

```
app = xpl.run_app(title_story='Breast Cancer example')
```

Após executados todos esses passos, obtêm-se os gráficos conforme mostrado na figura 6.

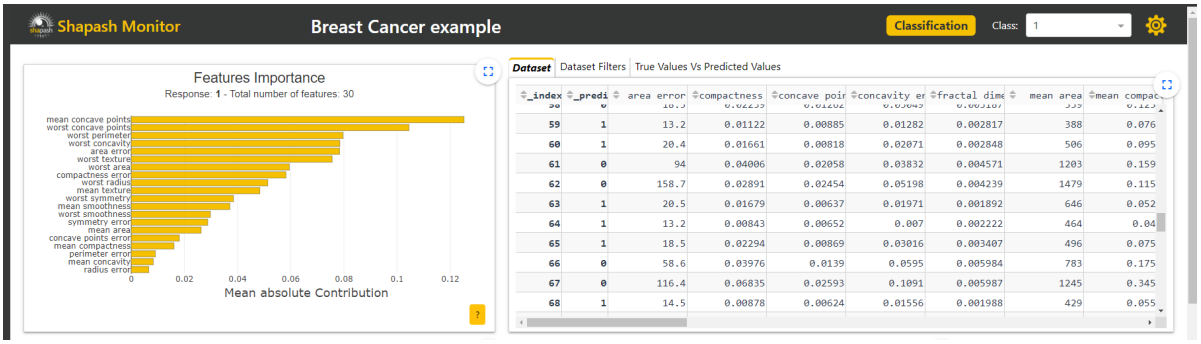


Figura 6 – Exemplo da *Shapash* gerado com o *Breast Cancer dataset*
Fonte: Elaboração própria

Assim como a *explainerdashboard*, a utilização da *Shapash* demonstrou-se dependente da realização de alguns passos com programação em Python.

2.4.3 *Dalex*

O nome *Dalex* é sigla para *Data Driven Analysis of Learning Experiments* que no português significa Análise Orientada a Dados de Experimentos de Aprendizado. A *Dalex*

também é uma biblioteca Python utilizada para conseguir interpretar modelos de aprendizado de máquinas (SETH, 2021).

Na figura 7, são apresentados alguns gráficos obtidos com a utilização dessa biblioteca. Para visualizar os gráficos em conjunto como uma *dashboard*, é preciso utilizar a ferramenta Arena que faz essa integração do *Dalex* em uma interface organizada.

Figura 7 – Exemplo do arena: *Dalex*
Fonte: DrWhy.AI (2021)

Exemplificando a utilização do *Dalex* junto ao Arena com o *Breast Cancer dataset*, inicialmente são feitas as instalações:

```
pip install Dalex
pip install flask
pip install flask_cors
```

Em seguida, são feitas as importações:

```
import Dalex as dx
from Dalex import Explainer
import flask
import flask_cors
import pandas as pd
```

Os arquivos do modelo e dos dados são lidos:

```
Xteste = pd.read_csv('X.csv')
Yteste = pd.read_csv('y.csv')
```

```
modelo = pd.read_pickle('teste.pck')
```

O objeto de explicação é criado a partir do modelo e dos dados:

```
expl = dx.Explainer(modelo, Xteste, Yteste)
```

No caso do *Dalex* cada gráfico desejado deverá ser criado por código, conforme demonstrado a seguir:

```
# Criação do Variable Importance:
expl.model_parts().plot(max_vars=30)
# Criação do breakdown:
expl.predict_parts(X.iloc[79, :],
type='break_down_interactions').plot(max_vars=20)
# Criação do SHAP:
expl.predict_parts(X.iloc[79, :],type=``shap'')
.plot(min_max=[0,1],max_vars=15)
```

Após esses passos, é preciso passar os gráficos para a *dashboard* Arena. Inicialmente é criada uma instância da arena:

```
arena=dx.Arena()
```

Em seguida, os valores do objeto de explicação "expl" criado pelo *Dalex* e os dados de teste são passados para o arena:

```
arena.push_model(expl)
arena.push_observations(Xteste)
```

Por fim, é preciso colocar a Arena para rodar em uma porta local.

```
arena.run_server(port=8080)
```

O resultado obtido pode ser visualizado na figura 8.

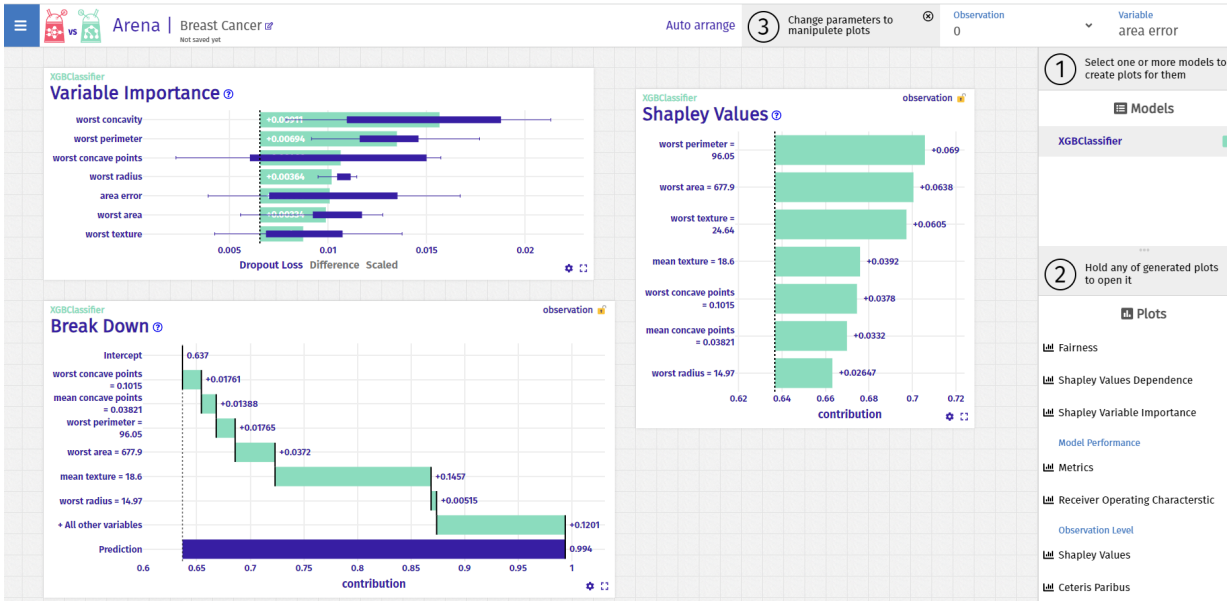


Figura 8 – Exemplo do *Dalex* com o Arena gerado com o *Breast Cancer dataset*
Fonte: DrWhy.AI (2021)

Nesse exemplo, é possível ver que a biblioteca *Dalex* necessita de ainda mais programação em Python que as demais ferramentas discutidas anteriormente, necessitando uma quantidade maior de ferramentas auxiliares que demandam instalações e importações a mais. Além disso, cada gráfico que se deseja traçar com a utilização dessa biblioteca é feito individualmente por algum comando específico. Dessa forma, percebe-se que a biblioteca apesar de permitir uma análise do modelo com diversos métodos de interpretabilidade, tem grande dependência da programação Python.

3 Desenvolvimento da aplicação

3.1 Planejamento

Com base no estudo dos sistemas e métodos de explicabilidade e interpretabilidade existentes, planejou-se o desenvolvimento de um sistema de fácil utilização, com *design* simples e que reunisse alguns métodos relevantes e amplamente utilizados. Dessa forma, escolheu-se os métodos *Permutation*, LIME e SHAP e idealizou-se um site conforme o esboço da figura 9 abaixo demonstra, um menu lateral à esquerda e os gráficos à direita.

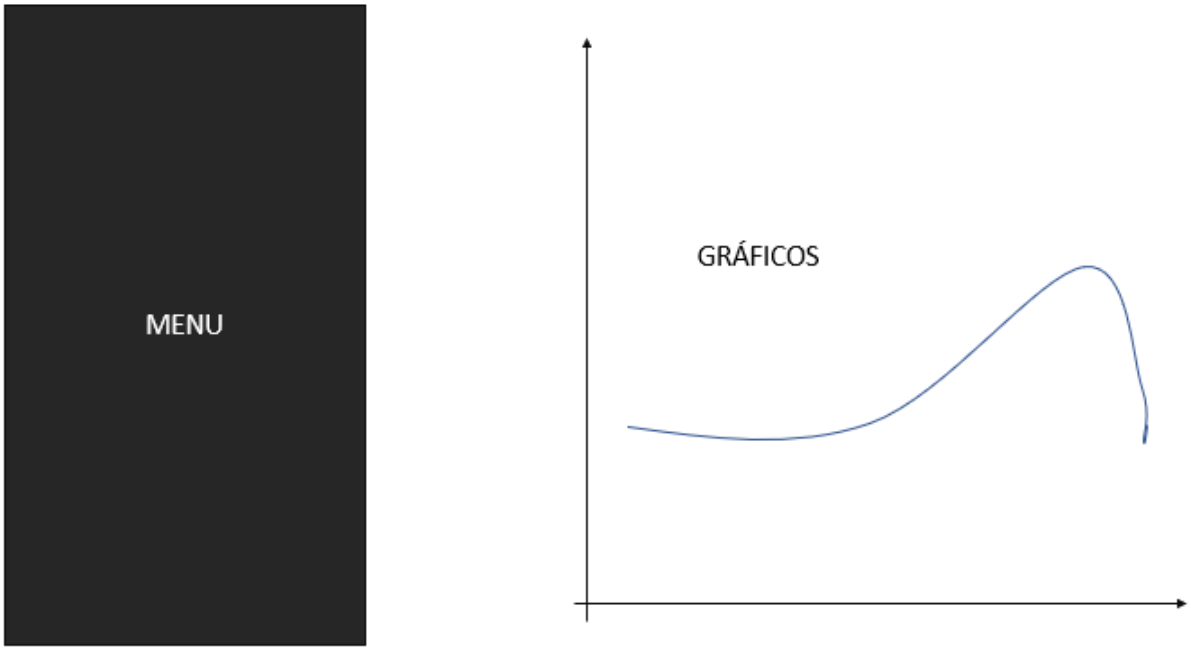


Figura 9 – Planejamento: esboço da aplicação
Fonte: Elaboração própria

3.2 Ferramentas de desenvolvimento

No desenvolvimento de *software* existem diversas ferramentas utilizadas; a linguagem de programação, necessária para a implementação da lógica do sistema; o *framework* que cria uma estrutura base para a aplicação permitindo um desenvolvimento mais rápido e eficiente e diversas bibliotecas, que são ferramentas já implementadas que podem ser instaladas e utilizadas no seu código de forma personalizada para sua aplicação. De acordo com as demandas, requisitos do sistema e pesquisas, alguns recursos foram selecionados para serem utilizados no XAI *web*, os principais são listados a seguir:

- **Angular**: framework de desenvolvimento de *front-end* de aplicações *web* (DOCUMENTATION, A., 2023).
- **Angular material**: biblioteca de componentes customizáveis do angular (DOCUMENTATION, A. M., s.d.).
- **TypeScript**: linguagem de programação (DOCUMENTATION, T., 2023).
- **Plotly**: biblioteca de gráficos para *JavaScript* (DOCUMENTATION, P. J., 2023).
- **SweetAlert2**: biblioteca de *pop-up* de alertas para *JavaScript* (DOCUMENTATION, SweetAlert2, 2023).
- **Flask**: framework de desenvolvimento de *back-end* de aplicações *web* (PROJECTS, 2023).
- **Python**: linguagem de programação (DOCUMENTATION, P., 2023).
- **Scikit-learn**: biblioteca de desenvolvimento Python que reúne ferramentas de aprendizado de máquina (SCIKIT-LEARN DEVELOPMENT TEAM, 2023).
- **Pandas**: biblioteca de desenvolvimento Python para manipulação de dados (PANDAS DEVELOPMENT TEAM, 2023).
- **Eli5**: biblioteca de desenvolvimento Python que reúne ferramentas de explicabilidade de modelos de ML (ELI5 DEVELOPMENT TEAM, 2023).
- **LIME**: biblioteca de desenvolvimento Python para explicabilidade de modelos de ML (MARCO TULIO RIBEIRO, 2023).
- **SHAP**: biblioteca de desenvolvimento Python para explicabilidade de modelos de ML (SHAP, 2018).
- **Anaconda**: ambiente de desenvolvimento e gerenciamento de pacotes Python (DISTRIBUTION, 2023).

Além disso, todos os códigos foram desenvolvidos no editor de código-fonte Visual Studio Code da Microsoft (DOCUMENTATION, V. S. C., 2021).

3.3 Metodologia para desenvolvimento da aplicação

A aplicação foi desenvolvida em duas etapas paralelas uma para o *front-end* e uma para o *back-end*. Para a etapa do *front-end* foi escolhido o desenvolvimento com o *framework* Angular e a linguagem Typescript, procurando criar um *design* simples e intuitivo. Conforme demonstrado na figura 10, o *front-end* possui (1) cabeçalho com o nome do projeto,

(2) um menu lateral, no qual o usuário pode fazer o envio dos arquivos necessários para o método e (3) a área onde são disponibilizados os gráficos resultantes dos métodos aplicados. Na segunda etapa, por sua vez, desenvoleu-se o *back-end* utilizando o *framework* Flask, uma vez que ele utiliza a linguagem de programação Python, que é a mesma na qual estão implementadas as bibliotecas de explicabilidade de aprendizado de máquina utilizadas nesse projeto.

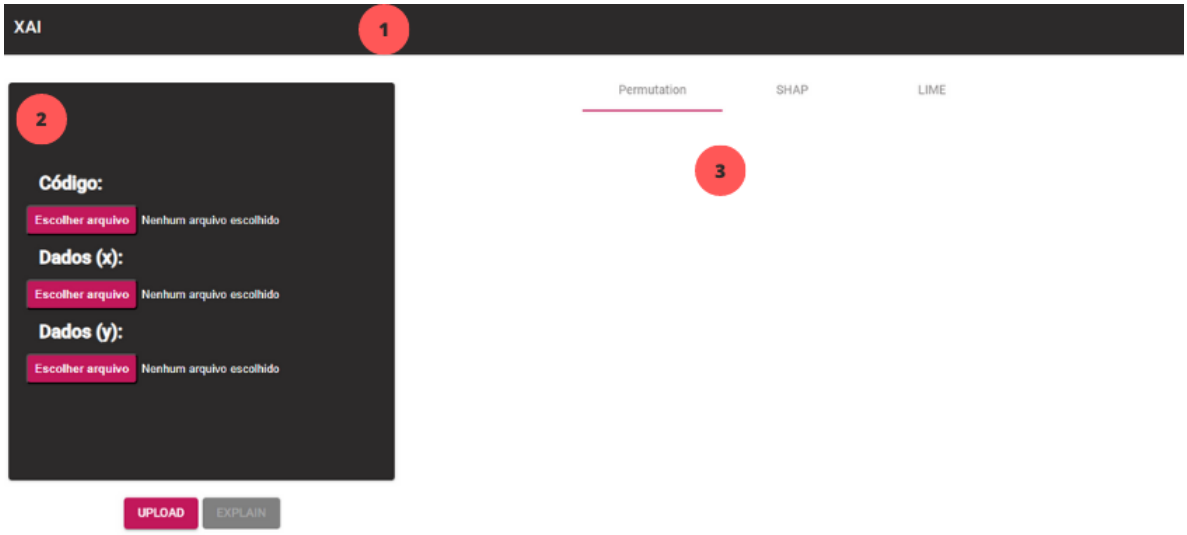


Figura 10 – A aplicação *web*
Fonte: Elaboração própria.

3.3.1 *Front-end*

O *front-end* da aplicação possui uma estrutura de componentes. O principal componente do sistema desenvolvido é a tela principal, na figura 11 vê-se sua organização. Dentro do código, essa tela foi nomeada como *home* e é dividida em um arquivo principal; home.component.ts, no qual está implementada a lógica de funcionamento da tela; home.component.html, responsável pela estrutura html da tela; home.component.css, que é o arquivo *css* onde fica estruturado todos os estilos utilizados no componente desde cores, fontes até posicionamentos; e por fim, o arquivo home.component.spec.ts, que é um arquivo de testes para o próprio componente.

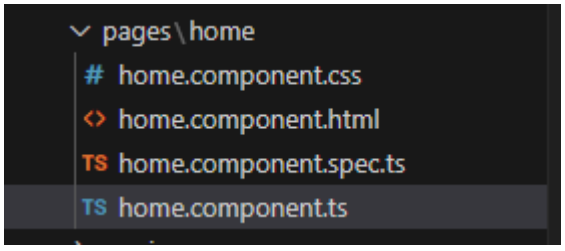


Figura 11 – Componente *home* do sistema
Fonte: Elaboração própria.

Uma outra parte muito importante implementada no *front-end* é a parte dos serviços porque é dentro dela que ficam inseridas as chamadas para as rotas criadas no *back-end*. Na figura 12, vê-se o exemplo do método responsável por enviar os atributos para a rota "dataX" e na figura 13 o exemplo da função de chamada da rota "/permutation" que receberá os resultados obtidos pelo método *permutation* implementado no *back-end*.

```
postDataX(dataX: FormData): Observable<any> {
    const apiUrl = `${this.API_URL}/dataX`;
    return this.http.post(apiUrl, dataX, {
        reportProgress: true,
        observe: 'events'
    })
}
```

Figura 12 – Método de envio dos atributos para o *back-end*
Fonte: Elaboração própria.

```
getPermutationResult() {
const apiUrl = `${this.API_URL}/permutation`;
return this.http.get(apiUrl);
}
```

Figura 13 – Método de chamada do *permutation*
Fonte: Elaboração própria.

3.3.2 *Back-end*

Para o desenvolvimento do *back-end*, criou-se um ambiente virtual com o auxilio do Anaconda. Dessa forma, todos os pacotes, extensões e bibliotecas necessários para o funcionamento do sistema ficaram reunidos e organizados nas versões corretas. No *back-end* foram criados os métodos de recebimento e leitura dos arquivos e de aplicação dos métodos XAI.

Exemplificando-se com o *permutation*, têm-se os passos:

- Passo 1: Leitura dos arquivos, o código para isso é mostrado na figura 14. A rota "/dataX" definida no início desse trecho de código define a rota que deve ser chamada pelo *front-end* para ser feito o envio dos atributos. Logo em seguida é definida a função upload_file_csv_x() que vai fazer a verificação do arquivo recebido e retornar para o *front-end* se o arquivo estava dentro do esperado ou se houve algum erro. Se caso estiver dentro do esperado, o arquivo será salvo em uma pasta auxiliar chamada UPLOAD_FOLDER para poder ser usado posteriormente. Esse código também é repetido para o recebimento das saídas esperadas e para o código do modelo, cada um com seus devidos ajustes.

```python
@app.route('/dataX', methods=['POST'])
def upload_file_csv_x():
    # check if the post request has the file part
    if 'file' not in request.files:
        resp = jsonify({'message' : 'No file part in the request'})
        resp.status_code = 400
        return resp
    file = request.files['file']
    if file.filename == '':
        resp = jsonify({'message' : 'No file selected for uploading'})
        resp.status_code = 400
        return resp
    if file and allowed_file(file.filename):
        file.save(os.path.join(app.config['UPLOAD_FOLDER'], 'data-x.csv'))
        resp = jsonify({'message' : 'File successfully uploaded'})
        resp.status_code = 201
        return resp
    else:
        resp = jsonify({'message' : 'Allowed file types are csv and pck'})
        resp.status_code = 400
        return resp
```

Figura 14 – Código de *upload* dos atributos X
Fonte: Elaboração própria.

- Passo 2: Cada método XAI implementado também é uma rota dentro do código do *back-end*. Para o permutation, conforme mostrado na figura 15, têm-se a rota: "/permutation" e a função predict(). Nela inicialmente serão lidos os parâmetros salvos na pasta UPLOAD_FOLDER com o auxílio da biblioteca *pandas* no código definida como pd. Com os arquivos lidos, na linha:

```
perm =
PermutationImportance(loaded_model, random_state=1).fit(
df_feature, df_target)
```

o sistema faz o cálculo da importância das características com o *Permutation Importance* e na linha seguinte:

```
result_permutation =
eli5.explain_weights_df(
perm, feature_names=df_feature.columns.tolist())
```

são gerados os resultados da importância das características calculadas. O nome das colunas obtidas na tabela de resultados é alterado para uma melhor usabilidade e o resultado é enviado para o *front-end*.

```python
@app.route('/permutation', methods=['GET'])
def predict():

    df_feature = pd.read_csv(os.path.join(app.config['UPLOAD_FOLDER'], 'data-x.csv'))
    df_target = pd.read_csv(os.path.join(app.config['UPLOAD_FOLDER'], 'data-y.csv'))
    loaded_model = pd.read_pickle(os.path.join(app.config['UPLOAD_FOLDER'], 'file.pck'))

    perm = PermutationImportance(loaded_model, random_state=1).fit(df_feature, df_target)

    result_permutation = eli5.explain_weights_df(perm, feature_names=df_feature.columns.tolist())
    result = result_permutation.rename(columns={"feature": "Feature", "weight": "Weight"}).to_dict()
    result['Message'] = 'Model predicted'


    resp = jsonify(result)
    resp.status_code = 201
    return resp
```

Figura 15 – Código do *permutation*
Fonte: Elaboração própria.

Os códigos dos métodos SHAP e LIME também seguem o mesmo padrão do permutation; inicia-se com a definição da rota e têm-se uma função responsável por aplicar os métodos. Essa função inicialmente lê os arquivos do modelo e dos dados salvos e em seguida o modelo é explicado pelo método. A principal diferença é que cada um deles têm suas bibliotecas, métodos e forma de utilização específicas. O LIME, por exemplo, utiliza-se da biblioteca LIME e necessita da definição de qual parâmetro e qual método vão ser utilizados, uma vez que, trata-se de um método local que avalia os atributos separadamente e permite utilizar diferentes modelos para gerar essa explicação. Isso pode ser observado na figura 16, com a definição da variável *limeMode* como o modelo utilizado e *columnValue* como índice do *df_feature.values* que é o índice do atributo para o qual será feita a explicação local.

No caso da aplicação desenvolvida o *limeMode* é definido como *"regression"* por padrão ou seja, será feito um modelo de regressão para atender os dados lineares dos modelos de ML de aprendizado supervisionado e o *columnValue* vai ser correspondente ao índice do atributo selecionado pelo usuário na tela da aplicação.

```python
explainer = lime.lime_tabular.LimeTabularExplainer(df_feature.values, verbose=True, mode=limeMode)
exp = explainer.explain_instance(df_feature.values[columnValue], loaded_model.predict)
```

Figura 16 – Código do LIME
Fonte: Elaboração própria.

4 A aplicação: XAI *web*

4.1 Diagrama de componentes

O diagrama de componentes UML apresentado na figura 17 apresenta a arquitetura do sistema desenvolvido neste trabalho. Este diagrama destaca os dois componentes principais, (1) a aplicação *Flask* que corresponde ao *back-end* ou servidor do sistema e (2) a interface do usuário que corresponde ao *front-end*. Dentro do componente (1) encontram-se os três componentes referentes a cada um dos três métodos de interpretabilidade disponíveis na aplicação, sendo responsáveis por a partir dos dados recebidos aplicar o método correspondente, tratar a resposta e enviá-la ao requisitante. No componente (2), por sua vez, estão englobados o componente de formulário para a realização de requisições e o componente gráficos responsável por disponibilizar a visualização dos resultados na interface do usuário quando for recebida uma resposta da aplicação. As setas indicam o fluxo e direção dos dados e a interação entre os componentes.

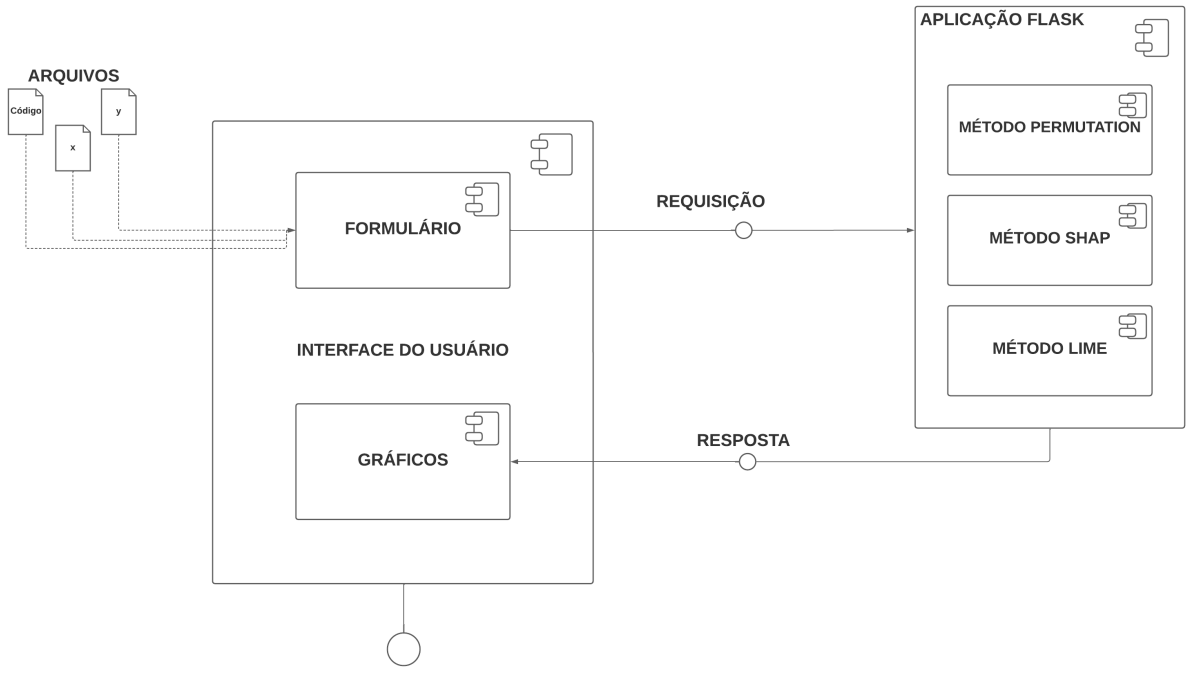
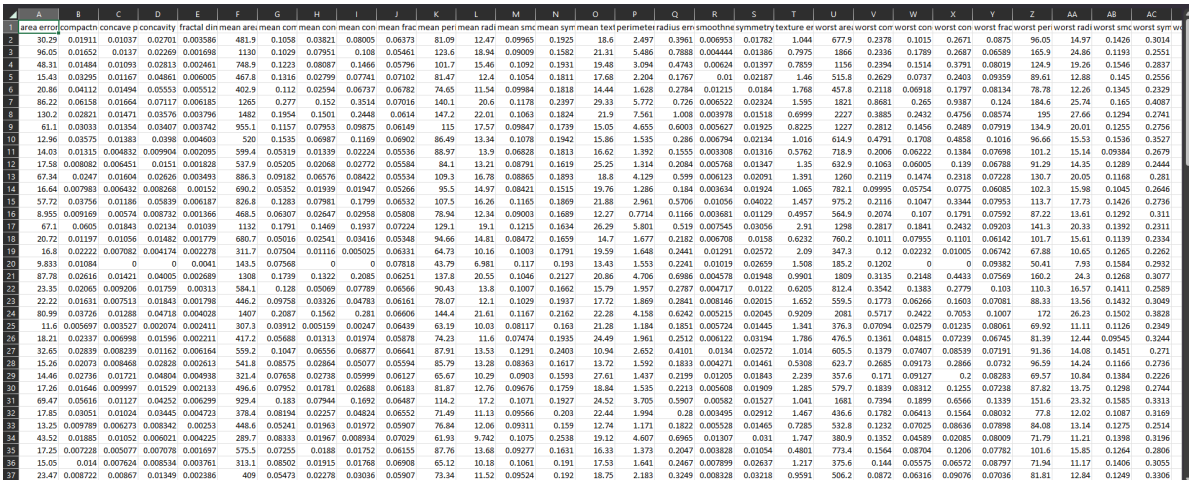


Figura 17 – Diagrama de componentes do sistema
Fonte: Elaboração própria.

4.2 Funcionamento

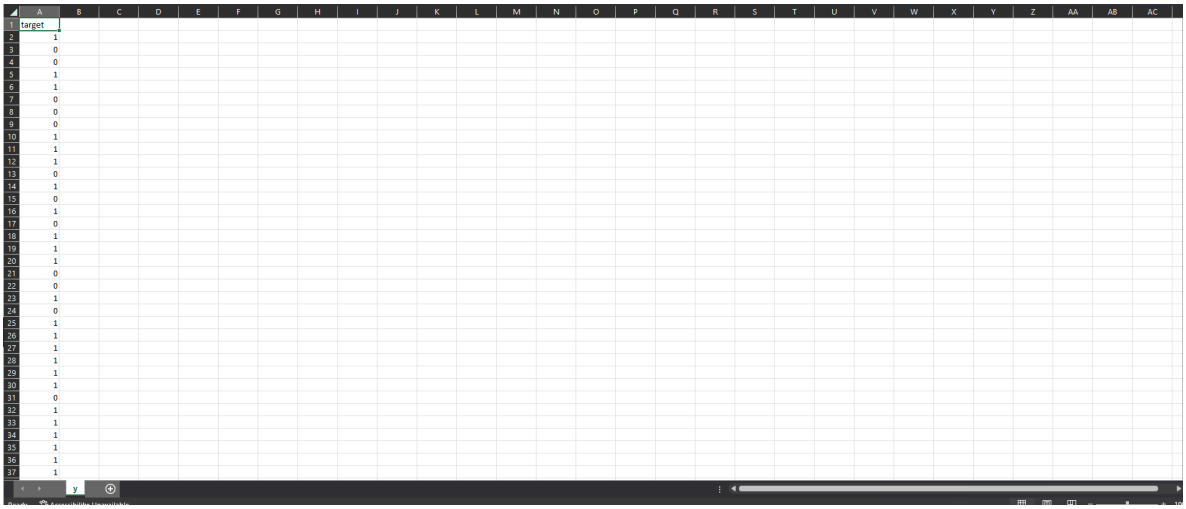
Todos os métodos implementados no projeto são *post-hoc*, ou seja, são criadas explicações para modelos já treinados. Para utilizar a aplicação é necessário fazer o envio de 3 arquivos

no sistema: o modelo treinado no formato pickle (pickle é uma ferramenta de serialização Python para ler e salvar códigos e objetos da linguagem), o vetor de atributos X de teste (figura 18) e o vetor com as saídas esperadas Y de teste (figura 19).



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC
1	area error	compactn	concave p	concavity	fractal dim	mean area	mean com	mean con	mean con	mean frac	mean peri	mean radi	mean smo	mean symr	mean text	perimeter	radius err	smoothne	symmetry	texture er	worst area	worst com	worst con	worst con	worst frac	worst peri	worst radi	worst smo	worst symr
2	30.29	0.01911	0.01037	0.02701	0.003586	481.9	0.1058	0.03821	0.08005	0.06373	81.09	12.47	0.09965	0.1925	18.6	2.497	0.3961	0.006953	0.01782	1.044	677.9	0.2378	0.1015	0.2671	0.0875	96.05	14.97	0.1426	0.3014
3	96.05	0.01652	0.0137	0.02269	0.001698	1130	0.1029	0.07951	0.108	0.05461	123.6	18.94	0.09009	0.1582	21.31	5.486	0.7888	0.004444	0.01386	0.7975	1866	0.2336	0.1789	0.2687	0.06589	165.9	24.86	0.1193	0.2551
4	48.31	0.01484	0.01093	0.02813	0.002461	748.9	0.1223	0.08087	0.1466	0.05796	101.7	15.46	0.1092	0.1931	19.48	3.094	0.4743	0.00624	0.01397	0.7859	1156	0.2394	0.1514	0.3791	0.08019	124.9	19.26	0.1546	0.2837
5	15.43	0.03295	0.01167	0.04861	0.006005	467.8	0.1316	0.02799	0.07741	0.07102	81.47	12.4	0.1054	0.1811	17.68	2.204	0.1767	0.01	0.02187	1.46	515.8	0.2629	0.0737	0.2403	0.09359	89.61	12.88	0.145	0.2556
6	20.86	0.04112	0.01494	0.05553	0.005512	402.9	0.112	0.02594	0.06737	0.06782	74.65	11.54	0.09984	0.1818	14.44	1.628	0.2784	0.01215	0.0184	1.768	457.8	0.2118	0.06918	0.1797	0.08134	78.78	12.26	0.1345	0.2329
7	86.22	0.06158	0.01664	0.07117	0.006185	1265	0.277	0.152	0.3514	0.07016	140.1	20.6	0.1178	0.2397	29.33	5.772	0.726	0.006522	0.02324	1.595	1821	0.8681	0.265	0.9387	0.124	184.6	25.74	0.165	0.4087
8	130.2	0.02821	0.01471	0.03576	0.003796	1482	0.1954	0.1501	0.2448	0.0614	147.2	22.01	0.1063	0.1824	21.9	7.561	1.008	0.003978	0.01518	0.6999	2227	0.3885	0.2432	0.4756	0.08574	195	27.66	0.1294	0.2741
9	61.1	0.03033	0.01354	0.03407	0.003742	955.1	0.1157	0.07953	0.09875	0.06149	115	17.57	0.09847	0.1739	15.05	4.655	0.6003	0.005627	0.01925	0.8225	1227	0.2812	0.1456	0.2489	0.07919	134.9	20.01	0.1255	0.2756
10	12.96	0.03575	0.01383	0.0398	0.004603	520	0.1535	0.06987	0.1169	0.06902	86.49	13.34	0.1078	0.1942	15.86	1.535	0.286	0.006794	0.02134	1.016	614.9	0.4791	0.1708	0.4858	0.1016	96.66	15.53	0.1536	0.3527
11	14.03	0.01315	0.004832	0.009904	0.002095	599.4	0.05319	0.01339	0.02224	0.05536	88.97	13.9	0.06828	0.1813	16.62	1.392	0.1555	0.003308	0.01316	0.5762	718.9	0.2006	0.06222	0.1384	0.07698	101.2	15.14	0.09384	0.2679
12	17.58	0.008082	0.006451	0.0151	0.001828	537.9	0.05205	0.02068	0.02772	0.05584	84.1	13.21	0.08791	0.1619	25.25	1.314	0.2084	0.005768	0.01347	1.35	632.9	0.1063	0.06005	0.139	0.06788	91.29	14.35	0.1289	0.2444
13	67.34	0.0247	0.01604	0.02626	0.003493	886.3	0.09182	0.06576	0.08422	0.05534	109.3	16.78	0.08865	0.1893	18.8	4.129	0.599	0.006123	0.02091	1.391	1260	0.2119	0.1474	0.2318	0.07228	130.7	20.05	0.1168	0.281
14	16.64	0.007983	0.006432	0.008268	0.00152	690.2	0.05352	0.01939	0.01947	0.05266	95.5	14.97	0.08421	0.1515	19.76	1.286	0.184	0.003634	0.01924	1.065	782.1	0.09995	0.05754	0.0775	0.06085	102.3	15.98	0.1045	0.2646
15	57.72	0.03756	0.01186	0.05839	0.006187	826.8	0.1283	0.07981	0.1799	0.06532	107.5	16.26	0.1165	0.1869	21.88	2.961	0.5706	0.01056	0.04022	1.457	975.2	0.2116	0.1047	0.3344	0.07953	113.7	17.73	0.1426	0.2736
16	8.955	0.009169	0.00574	0.008732	0.001366	468.5	0.06307	0.02647	0.02958	0.05808	78.94	12.34	0.09003	0.1689	12.27	0.7714	0.1166	0.003681	0.01129	0.4957	564.9	0.2074	0.107	0.1791	0.07592	87.22	13.61	0.1292	0.311
17	67.1	0.0605	0.01843	0.02134	0.01039	1132	0.1791	0.1469	0.1937	0.07224	129.1	19.1	0.1215	0.1634	26.29	5.801	0.519	0.007545	0.03056	2.91	1298	0.2817	0.1841	0.2432	0.09203	141.3	20.33	0.1392	0.2311
18	20.72	0.01197	0.01056	0.01482	0.001779	680.7	0.05016	0.02541	0.03416	0.05348	94.66	14.81	0.08472	0.1659	14.7	1.677	0.2182	0.006708	0.0158	0.6232	760.2	0.1011	0.07955	0.1101	0.06142	101.7	15.61	0.1139	0.2334
19	16.8	0.02222	0.007082	0.004174	0.002278	311.7	0.07504	0.01116	0.005025	0.06311	64.73	10.16	0.1003	0.1791	19.59	1.648	0.2441	0.01291	0.02572	2.09	347.3	0.12	0.02232	0.01005	0.06742	67.88	10.65	0.1265	0.2262
20	9.833	0.01084	0	0	0.0041	143.5	0.07568	0	0	0.07818	43.79	6.981	0.117	0.193	13.43	1.553	0.2241	0.01019	0.02659	1.508	185.2	0.1202	0	0	0.09382	50.41	7.93	0.1584	0.2932
21	87.78	0.02616	0.01421	0.04005	0.002689	1308	0.1739	0.1322	0.2085	0.06251	137.8	20.55	0.1046	0.2127	20.86	4.706	0.6986	0.004578	0.01948	0.9901	1809	0.3135	0.2148	0.4433	0.07569	160.2	24.3	0.1268	0.3077
22	23.35	0.02065	0.009206	0.01759	0.00313	584.1	0.128	0.05069	0.07789	0.06566	90.43	13.8	0.1007	0.1662	15.79	1.957	0.2787	0.004717	0.0122	0.6205	812.4	0.3542	0.1383	0.2779	0.103	110.3	16.57	0.1411	0.2589
23	22.22	0.01631	0.007513	0.01843	0.001798	446.2	0.09758	0.03326	0.04783	0.06161	78.07	12.1	0.1029	0.1937	17.72	1.869	0.2841	0.008146	0.02015	1.652	559.5	0.1773	0.06266	0.1603	0.07081	88.33	13.56	0.1432	0.3049
24	80.99	0.03726	0.01288	0.04718	0.004028	1407	0.2087	0.1562	0.281	0.06606	144.4	21.61	0.1167	0.2162	22.28	4.158	0.6242	0.005215	0.02045	0.9209	2081	0.5717	0.2422	0.7053	0.1007	172	26.23	0.1502	0.3828
25	11.6	0.005697	0.003527	0.002074	0.002411	307.3	0.03912	0.005159	0.00247	0.06439	63.19	10.03	0.08117	0.163	21.28	1.184	0.1851	0.005724	0.01445	1.341	376.3	0.07094	0.02579	0.01235	0.08061	69.92	11.11	0.1126	0.2349
26	18.21	0.02337	0.006998	0.01596	0.002211	417.2	0.05688	0.01313	0.01974	0.05878	74.23	11.6	0.07474	0.1935	24.49	1.961	0.2512	0.006122	0.03194	1.786	476.5	0.1361	0.04815	0.07239	0.06745	81.39	12.44	0.09545	0.3244
27	32.65	0.02839	0.008239	0.01162	0.006164	559.2	0.1047	0.06556	0.06877	0.06641	87.91	13.53	0.1291	0.2403	10.94	2.652	0.4101	0.0134	0.02572	1.014	605.5	0.1379	0.07407	0.08539	0.07191	91.36	14.08	0.1451	0.271
28	15.26	0.02073	0.008468	0.02828	0.002613	541.8	0.08575	0.02864	0.05077	0.05594	85.79	13.28	0.08363	0.1617	13.72	1.592	0.1833	0.004271	0.01461	0.5308	623.7	0.2685	0.09173	0.2866	0.0732	96.59	14.24	0.1166	0.2736
29	14.46	0.02736	0.01721	0.04804	0.004938	321.4	0.07658	0.02738	0.05999	0.06127	65.67	10.29	0.0903	0.1593	27.61	1.437	0.2199	0.01205	0.01843	2.239	357.6	0.171	0.09127	0.2	0.08283	69.57	10.84	0.1384	0.2226
30	17.26	0.01646	0.009997	0.01529	0.002133	496.6	0.07952	0.01781	0.02688	0.06183	81.87	12.76	0.09676	0.1759	18.84	1.535	0.2213	0.005608	0.01909	1.285	579.7	0.1839	0.08312	0.1255	0.07238	87.82	13.75	0.1298	0.2744
31	69.47	0.05616	0.01127	0.04252	0.006299	929.4	0.183	0.07944	0.1692	0.06487	114.2	17.2	0.1071	0.1927	24.52	3.705	0.5907	0.00582	0.01527	1.041	1681	0.7394	0.1899	0.6566	0.1339	151.6	23.32	0.1585	0.3313
32	17.85	0.03051	0.01024	0.03445	0.004723	378.4	0.08194	0.02257	0.04824	0.06552	71.49	11.13	0.09566	0.203	22.44	1.994	0.28	0.003495	0.02912	1.467	436.6	0.1782	0.06413	0.1564	0.08032	77.8	12.02	0.1087	0.3169
33	13.25	0.009789	0.006273	0.008342	0.00253	448.6	0.05241	0.01963	0.01972	0.05907	76.84	12.06	0.09311	0.159	12.74	1.171	0.1822	0.005528	0.01465	0.7285	532.8	0.1232	0.07025	0.08636	0.07898	84.08	13.14	0.1275	0.2514
34	43.52	0.01885	0.01052	0.006021	0.004225	289.7	0.08333	0.01967	0.008934	0.07029	61.93	9.742	0.1075	0.2538	19.12	4.607	0.6965	0.01307	0.031	1.747	380.9	0.1352	0.04589	0.02085	0.08009	71.79	11.21	0.1398	0.3196
35	17.25	0.007228	0.005077	0.007078	0.001697	575.5	0.07255	0.0188	0.01752	0.06155	87.76	13.68	0.09277	0.1631	16.33	1.373	0.2047	0.003828	0.01054	0.4801	773.4	0.1564	0.08704	0.1206	0.07782	101.6	15.85	0.1264	0.2806
36	15.05	0.014	0.007624	0.008534	0.003761	313.1	0.08502	0.01915	0.01768	0.06908	65.12	10.18	0.1061	0.191	17.53	1.641	0.2467	0.007899	0.02637	1.217	375.6	0.144	0.05575	0.06572	0.08797	71.94	11.17	0.1406	0.3055
37	23.47	0.008722	0.00867	0.01349	0.002386	409	0.05473	0.02278	0.03036	0.05907	73.34	11.52	0.09524	0.192	18.75	2.183	0.3249	0.008328	0.03218	0.9591	506.2	0.0872	0.06316	0.09076	0.07036	81.81	12.84	0.1249	0.3306

Figura 18 – Atributos x enviados para aplicação
Fonte: Elaboração própria.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC
1	target																												
2	1																												
3	0																												
4	0																												
5	1																												
6	1																												
7	0																												
8	0																												
9	0																												
10	1																												
11	1																												
12	1																												
13	0																												
14	1																												
15	0																												
16	1																												
17	0																												
18	1																												
19	1																												
20	0																												
21	0																												
22	1																												
23	0																												
24	1																												
25	1																												
26	1																												
27	1																												
28	1																												
29	1																												
30	0																												
31	1																												
32	1																												
33	1																												
34	1																												
35	1																												
36	1																												
37	1																												

Figura 19 – Saídas esperadas y enviados para aplicação
Fonte: Elaboração própria.

Após selecionados os 3 arquivos, o usuário deve submetê-los selecionando o botão *upload*. Após realizado o envio o sistema irá liberar o botão *explain* que quando acionado, retornará os gráficos resultantes dos métodos na aba lateral direita. O usuário poderá então, navegar pelas abas para visualizar os gráficos obtidos pelos diferentes métodos, conforme é mostrado na imagem 20. É importante ressaltar ainda que, em caso de erro, o sistema retornará uma modal informando o problema conforme os exemplos das figuras 21 e 22.

Na figura 20 têm-se o resultado do funcionamento esperado da aplicação quando todos os requisitos de funcionamento são atendidos: os arquivos foram enviados e validados

pelo *back-end* conforme os ícones verdes de verificado no menu lateral demonstram e as explicações já foram geradas. Na figura, o usuário encontra-se acessando a aba do resultado do SHAP e por isso está visualizando o gráfico correspondente a este método. Navegando pelas demais abas, o usuário tem acesso aos outros métodos disponíveis: o *permutation* e o LIME.

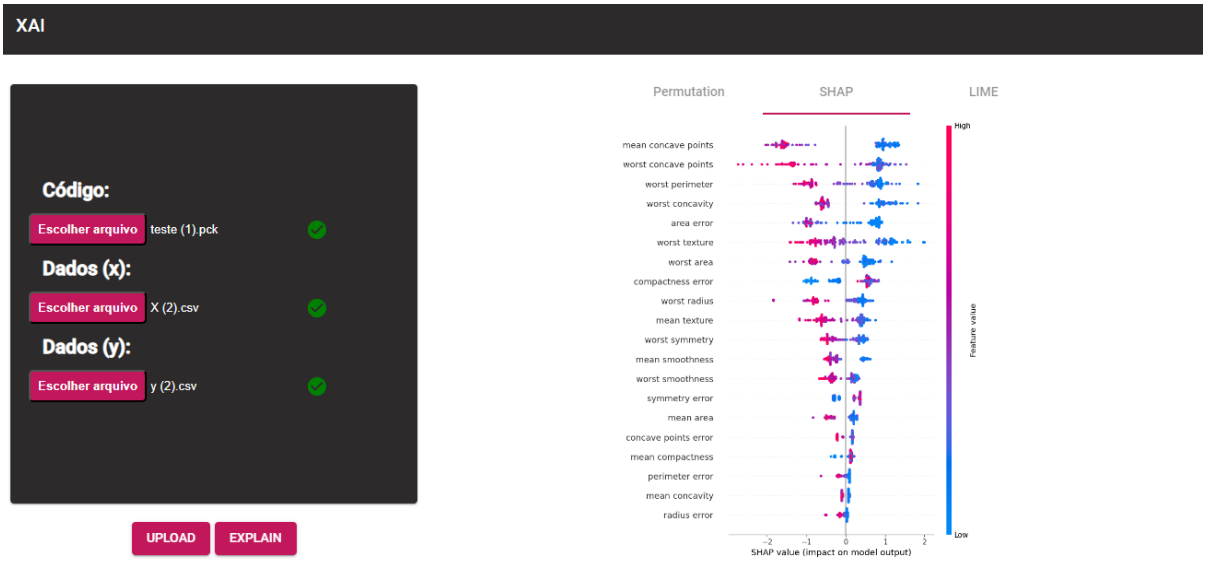


Figura 20 – A aplicação *web*: exemplo de funcionamento
Fonte: Elaboração própria.

Quando o usuário tenta fazer o envio clicando em *upload* mas sem selecionar um ou mais arquivos, o sistema abrirá um modal de tratamento do erro com a mensagem em inglês *"No file part in the request"*, que avisa ao usuário que os arquivos não foram enviados (figura 21).

Outro erro que pode ocorrer ao fazer o envio dos arquivos, é a seleção de arquivos com extensões diferentes das esperadas pelo sistema, uma vez que, o *back-end* está configurado para receber arquivos somente nas extensões .csv para os dados e .pck para o modelo. Caso o usuário tente fazer envio de arquivo em outra extensão o sistema retornará o modal com o erro "Allowed file types are csv and pck", traduzido para o português como "Tipos de arquivos permitidos são csv e pck", conforme a figura 22 demonstra. Além disso, quando um ou mais arquivos apresentam algum erro, um ícone de x em vermelho aparecerá logo a sua frente para identificar que o arquivo não pode ser lido e o botão *explain* só será liberado quando todos os três arquivos necessários forem lidos com sucesso.

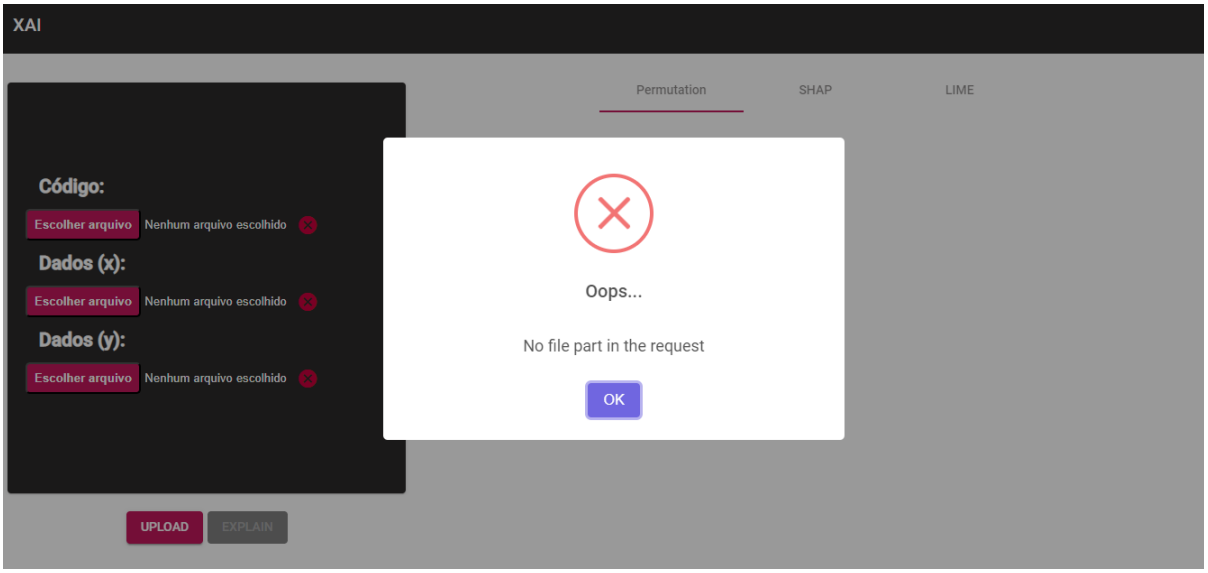


Figura 21 – A aplicação *web*: erro quando o usuário tenta fazer envio sem selecionar os arquivos

Fonte: Elaboração própria.

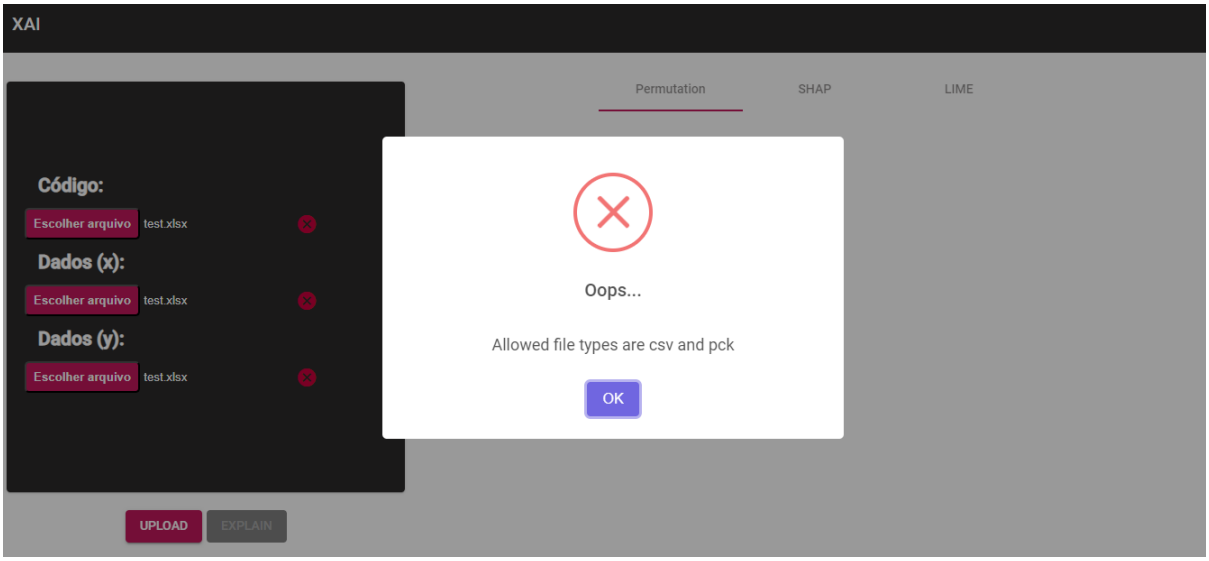


Figura 22 – A aplicação *web*: erro quando o usuário tenta fazer envio de arquivos com extensões não permitidas

Fonte: Elaboração própria.

4.3 Caso de teste

Para testar o funcionamento da aplicação XAI *web* desenvolvida nesse trabalho, também foi utilizado o conjunto de dados *Breast Cancer dataset* (DOCUMENTATION, 2021). Esse conjunto de dados reúne informações sobre diagnósticos de câncer de mama. Nele diversas características são avaliadas a partir de imagens de uma amostra de aspiração de uma agulha fina da mama e o tumor recebe uma classificação em: benigno ou não cancerígeno representado pelo valor 0, ou maligno ou cancerígeno representado pelo valor 1. Os valores

classificatórios 0 ou 1 representam no conjunto de dados as saídas esperadas y, e as características são os valores da variável que podem ser por exemplo área, textura, pontos côncavos, simetria e suavidade (SCIKIT-LEARN DEVELOPMENT TEAM, 2023).

Após realizado o envio dos arquivos do código, os dados X e as saídas y, obteve-se os resultados apresentados nas figuras 23, 24, 26. É importante ressaltar aqui, que os métodos do sistema vão procurar demonstrar como os atributos estão influenciando os resultados do modelo de aprendizagem que tenta classificar novos dados como cancerígeno ou não cancerígeno baseado no padrão que ele aprendeu com os dados desse banco de dados. Dessa forma, todos os gráficos vão trazer informações sobre as características dessa base de dados (os valores das variáveis X).

Na primeira aba do sistema (figura 23) é apresentado o resultado do método *permutation feature importance* da biblioteca *scikit-learn*. Essa aba já é selecionada inicialmente como padrão, então esse é o primeiro gráfico obtido. Nele, têm-se as FIs, ou seja, as importâncias das características para a permutação realizada pelo método. Trata-se de um gráfico de barras simples que ordena da esquerda para a direita a importância dos atributos de forma crescente. O valor da importância é uma medida relativa obtida da comparação entre as características e pode variar de 0 a 1. No exemplo feito, pode-se observar que a característica de maior importância relativa para a permutação realizada é a *worst radius* (maior raio encontrado nas células presentes) e diversas outras não tiveram impacto no resultado gerando um valor de importância igual a 0.

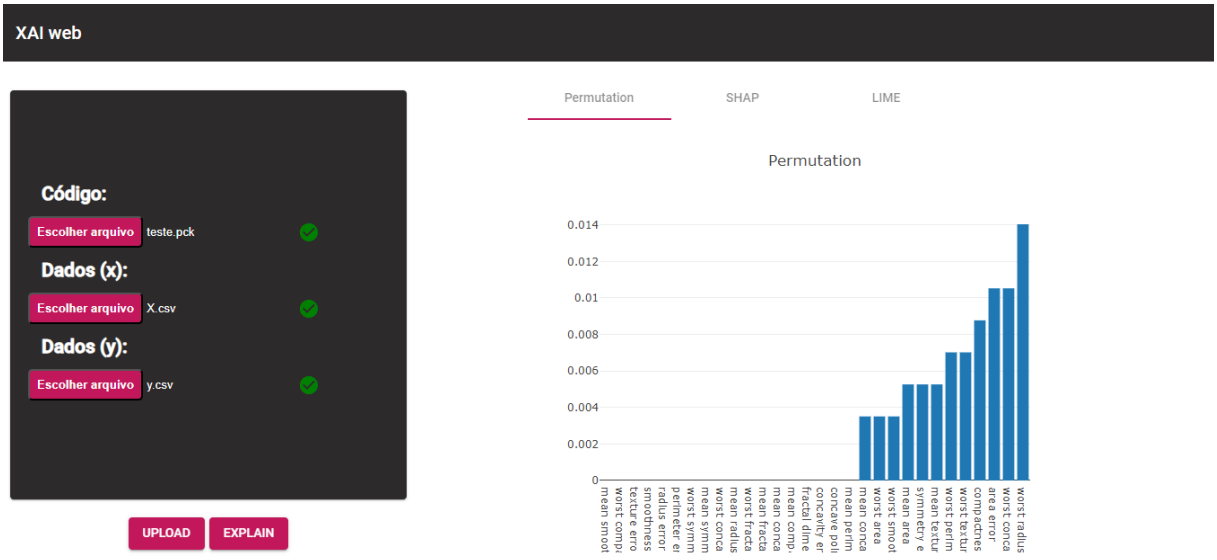


Figura 23 – A aplicação *web*: exemplo de funcionamento *permutation*
Fonte: Elaboração própria.

O gráfico apresentado na figura 24 é o resultado de uma explicação global obtida com o método SHAP. Esse gráfico é o *summary plot*, nele, podem ser analisadas as importâncias das características junto de seus efeitos em um modelo de aprendizagem de

máquina. No eixo horizontal x, têm-se a representação dos valores Shapley, ou seja, os valores de contribuição individual de cada atributo na predição. No eixo vertical y, têm-se as características que são ordenadas de cima para baixo quanto maior for sua importância para o modelo. Já as cores, indicam o valor da variável, quanto menor o valor mais forte o tom de azul e quanto maior o valor mais forte, o tom de vermelho. Além disso, quando existem pontos sobrepostos, os pontos são deslocados na direção vertical para uma melhor visualização e compreensão da distribuição dos valores Shapley para a característica (MOLNAR, 2022).

No exemplo realizado com o conjunto de dados de câncer de mama utilizando-se do SHAP têm-se o *mean concave points* (média dos pontos côncavos) com maior contribuição. Além dessa informação, pelas cores do gráfico percebe-se uma variação dessa contribuição: quando a variável assume valores positivos maiores (tons de rosa até vermelho) sua contribuição para o modelo é negativa (valores a esquerda do eixo y) e quando assume valores pequenos (tons de azul) sua contribuição é positiva (valores a direita do eixo y). No gráfico têm-se também a *radius error* (erro padrão dos valores de raio das células) com menor contribuição e uma variação bem pequena do resultado de acordo com a alteração de seus valores (percebe-se que quase todos os seus pontos estão próximos do eixo y, ou seja, estão próximas do valor 0 de contribuição).

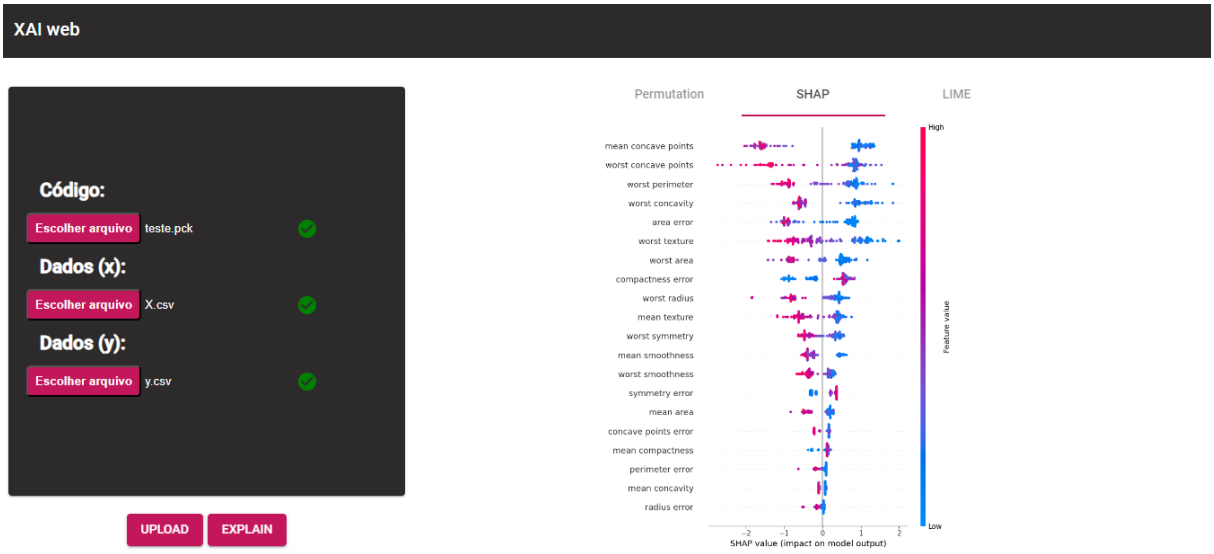


Figura 24 – A aplicação *web*: exemplo de funcionamento do SHAP
Fonte: Elaboração própria.

Para o LIME, inicialmente, o sistema apenas retorna a lista de atributos (figura 25), isso porque, por ser um método local, ele gera explicações para os atributos individualmente e no XAI *web*, o usuário pode escolher o atributo para o qual ele quer gerar a explicação. Na figura 26 escolheu-se o atributo *area error* (variação dos valores de área das células) como exemplo para gerar o gráfico de explicação local. Esse gráfico é o LIME Tabular que é uma versão do LIME para dados tabulares, nele o eixo y representa as faixas

de valores que o atributo analisado está assumindo e o eixo x o valor de influência para essa faixa, quando a influência é positiva a barra é representada na cor verde e quando a influência é negativa é representada na cor vermelha (GARREAU; LUXBURG, 2020).

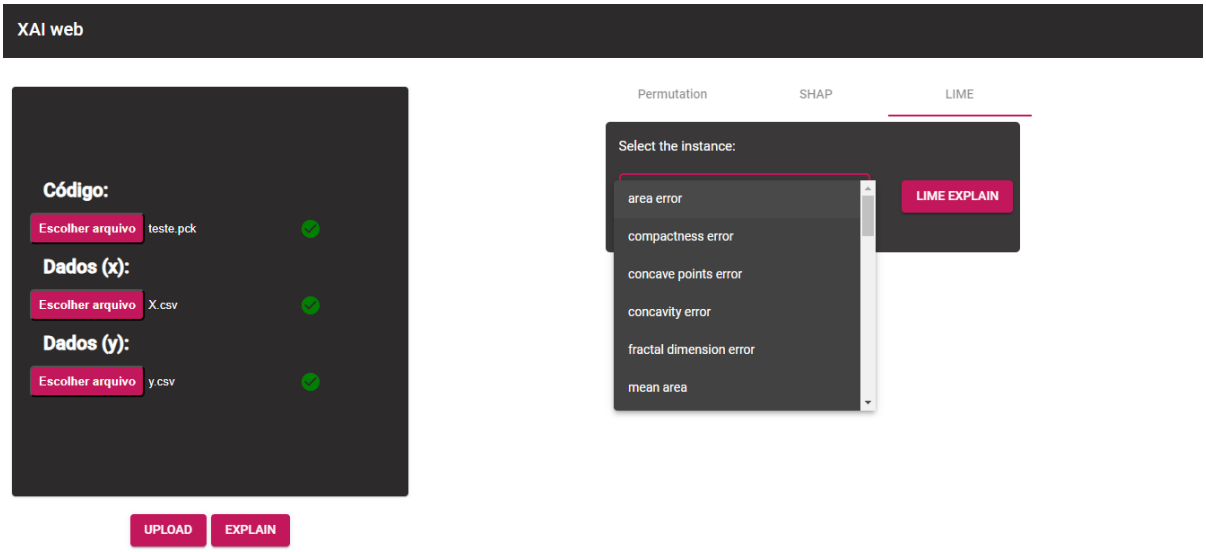


Figura 25 – A aplicação *web*: exemplo de funcionamento do LIME 1
Fonte: Elaboração própria.

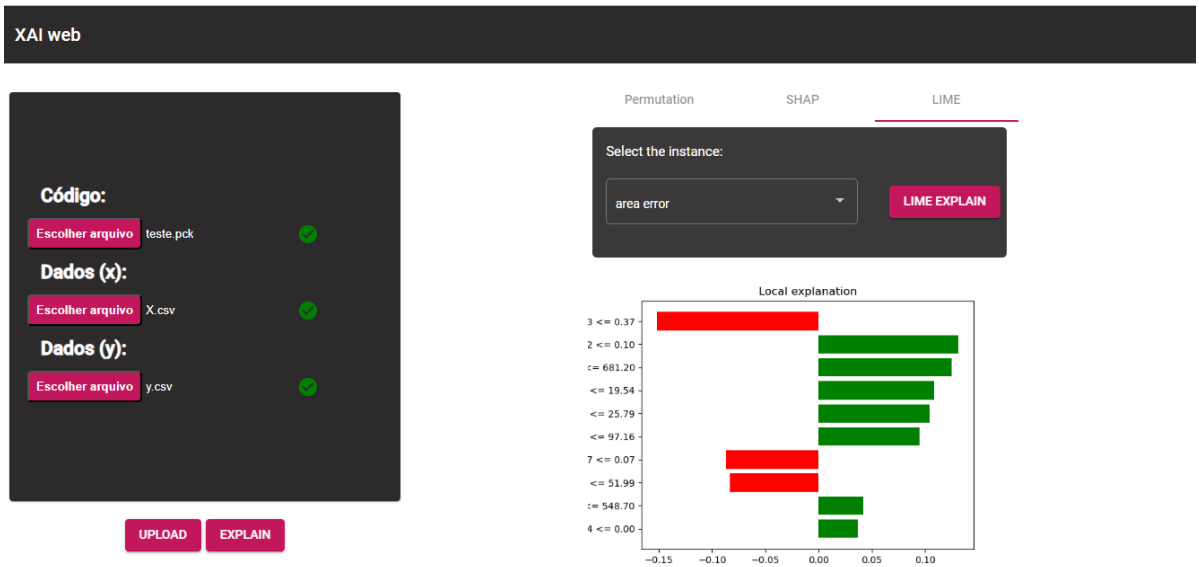


Figura 26 – A aplicação *web*: exemplo de funcionamento do LIME 2
Fonte: Elaboração própria.

4.4 Análise

Ao final do trabalho, o sistema XAI *web* proposto foi desenvolvido com o *front-end* criado com o *framework* Angular e o *back-end* com o *framework* Flask. Ambos totalmente integrados e com bom funcionamento. A aplicação seguiu o esboço inicial e manteve-se

uma organização simples com as funcionalidades bem intuitivas e organizadas na tela do usuário.

Na seção 4.2, foi apresentado um exemplo que demonstra todo o funcionamento da aplicação. O usuário foi capaz de acessar a aplicação, fazer o envio dos arquivos do seu modelo de aprendizagem de máquina supervisionado e visualizar os gráficos de interpretabilidade dos métodos *permutation*, SHAP e LIME sem precisar de desenvolver nenhum código em Python ou outra linguagem de programação. Comparado as bibliotecas *explainerdashboard*, *Shapash* e *Dalex* que precisam de código Python para serem utilizadas, o XAI *web* apresentou o diferencial de permitir a utilização de métodos de interpretabilidade apenas enviando os arquivos necessários.

A aplicação também incluiu um pouco de interatividade, permitindo que no método LIME fosse selecionado o atributo para o qual seria realizada a explicação local. Mas, é importante ressaltar que o escopo do sistema precisou ser limitado devido a restrições de tempo, a complexidade de modelos de IA e o número grande de métodos de interpretabilidade e explicabilidade existentes. Ainda assim, o sistema demonstrou-se uma ferramenta promissora para contribuir com informações mais claras sobre as características e suas influências nos resultados das predições dos modelos de forma bem simplificada.

4.4.1 Comparativo

Na Tabela 1, é possível visualizar um comparativo do funcionamento das ferramentas discutidas no capítulo 2 e o XAI web.

Vantagens	XAI *web*	*explainerdashboard* (EXPLAINERDASHBOARD, 2020)	*Shapash* (DOCUMENTATION, Shapash's, 2020)	*Dalex* (DRWHY.AI, 2021)
Explicação Global	✓	✓	✓	✓
Explicação Local	✓	✓	✓	✓
Não é necessário desenvolvimento Python	✓	X	X	X

Tabela 1 – Tabela comparativa

Pode-se observar que todas elas apresentam explicações globais e locais mas a principal diferença é que o sistema XAI *web* é o único que não necessita de desenvolvimento em Python ou outra linguagem de programação para sua utilização. Por isso, seu funcionamento torna mais simples a utilização da explicabilidade por não desenvolvedores que tiverem essa necessidade. Dentro de empresas, projetos e Universidades, os usuários que estiverem criando e analisando projetos com modelos de aprendizagem podem acessar a aplicação fazer o envio dos arquivos necessários e obterem de forma muito rápida e descomplicada gráficos de interpretabilidade com alguns dos métodos mais usados para esse propósito atualmente.

5 Conclusão

Considerando a relevância do uso de aprendizado de máquina atualmente, o desenvolvimento do sistema XAI *web* representa um passo importante para área de inteligência artificial explicável (XAI). O principal intuito do presente trabalho foi tornar algumas técnicas de XAI mais acessíveis para usuários sem conhecimento técnico de programação criando um sistema que permite o uso de métodos de interpretabilidade sem ter a programação como requisito.

Para isso, os métodos *permutation feature importance*, SHAP e LIME foram estudados e escolhidos por serem amplamente citados e com muita relevância na área. Conforme os resultados demonstraram o sistema permitiu aos usuários visualizar os gráficos gerados por esses métodos de interpretabilidade e explicabilidade para seus modelos apenas fazendo envio do modelo, dos atributos X e das saídas esperadas y. Tudo isso com um *design* intuitivo e boa acessibilidade.

Contudo, o desenvolvimento do presente trabalho precisou limitar a quantidade de métodos, a interatividade e os tipos de inteligência artificial para quais o sistema cria explicações. Por conta disso, sugere-se a continuidade do sistema em trabalhos e pesquisas futuras com a inclusão de novos métodos, estudo aprofundado dos resultados para diferentes aplicações e aumento de sua aplicabilidade para outros modelos de IA.

Por fim, avaliando-se os resultados alcançados verifica-se que o objetivo do trabalho foi alcançado e a criação do XAI *web* tornou mais simples e rápido o acesso a alguns métodos de interpretabilidade para modelos de aprendizagem supervisionada. Dessa forma, o sistema desenvolvido, suas possíveis aplicações e sua continuidade representam um importante avanço na construção de um uso da inteligência artificial mais responsável, com maior transparência e com aplicações cada vez mais assertivas.

Referências

AMOROSO, Fabrício Steinle. Inteligência artificial explicável com LIME e SHAP aplicada à rede neural convolucional. Universidade Estadual Paulista (Unesp), 2023. Citado 1 vez na página 14.

ANKARSTAD, Nicklas. *What is Explainable AI (XAI)?* [S.l.: s.n.], 2020. urlhttps: //towards datascience.com/what-is-explainable-ai-xai-afc56938d513. Citado 1 vez na página 12.

BREIMAN, Leo. Random forests. *Machine learning*, Springer, v. 45, p. 5–32, 2001. Citado 1 vez na página 12.

DAS, Arun; RAD, Paul. Opportunities and challenges in explainable artificial intelligence (xai): A survey. *arXiv preprint arXiv:2006.11371*, 2020. Citado 2 vezes nas páginas 11, 12.

DISTRIBUTION, Anaconda Software. *Anaconda Software Distribution*. [S.l.: s.n.], 2023. https://www.anaconda.com/. Citado 1 vez na página 25.

DOCUMENTATION, Angular. *Angular Documentation*. [S.l.: s.n.], 2023. https://angular.io/docs. Citado 1 vez na página 25.

DOCUMENTATION, Angular Material. *Angular Material Documentation*. [S.l.: s.n.]. https://material.angular.io/. Accessed on: 2023. Citado 1 vez na página 25.

DOCUMENTATION, Plotly JavaScript. *Plotly JavaScript Documentation*. [S.l.: s.n.], 2023. https://plotly.com/javascript/. Citado 1 vez na página 25.

DOCUMENTATION, Python. *Python Documentation*. [S.l.: s.n.], 2023. https://www.python.org/doc/. Citado 1 vez na página 25.

DOCUMENTATION, scikit-learn. *sklearn.datasets.load_breast_cancer*. [S.l.: s.n.], 2021. Disponível em: https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_breast_cancer.html. Citado 1 vez na página 33.

DOCUMENTATION, Shapash's. *Welcome to Shapash's documentation !* [S.l.: s.n.], 2020. urlhttps://shapash.readthedocs.io/en/latest/. Citado 4 vezes nas páginas 9, 16, 18, 19, 37.

DOCUMENTATION, SweetAlert2. *SweetAlert2 Documentation*. [S.l.: s.n.], 2023. https://sweetalert2.github.io/. Citado 1 vez na página 25.

DOCUMENTATION, TypeScript. *TypeScript Documentation*. [S.l.: s.n.], 2023. https://www.typescriptlang.org/docs/. Citado 1 vez na página 25.

DOCUMENTATION, Visual Studio Code. *Visual Studio Code Documentation*. [S.l.: s.n.], 2021. Disponível em: https://code.visualstudio.com/docs. Citado 1 vez na página 25.

DRWHY.AI. *Arena - Documentation*. 2021. Disponível em: https://arena.drwhy.ai/docs/. Citado 1 vez nas páginas 16, 21, 23, 37.

ELI5 DEVELOPMENT TEAM. *ELI5 Documentation*. [S.l.: s.n.], 2023. Website. Disponível em: https://eli5.readthedocs.io/en/latest/overview.html. Citado 1 vez na página 25.

EXPLAINERDASHBOARD, Documentation. *explainerdashboard*. [S.l.: s.n.], 2020. https://explainerdashboard.readthedocs.io/en/latest/. Citado 4 vezes nas páginas 9, 16, 17, 37.

FENG, Shuo et al. Intelligent driving intelligence test for autonomous vehicles with naturalistic and adversarial environment. *Nature communications*, Nature Publishing Group UK London, v. 12, n. 1, p. 748, 2021. Citado 1 vez na página 8.

GARREAU, Damien; LUXBURG, Ulrike von. Looking deeper into tabular LIME. *arXiv preprint arXiv:2008.11092*, 2020. Citado 1 vez na página 36.

GILPIN, Leilani H et al. Explaining explanations: An overview of interpretability of machine learning. In: IEEE. 2018 IEEE 5th International Conference on data science and advanced analytics (DSAA). [S.l.: s.n.], 2018. P. 80–89. Citado 1 vez na página 11.

IBM. *Explainable AI (XAI)*. [S.l.: s.n.], 2022. urlhttps://www.ibm.com/watson/explainable-ai. Citado 1 vez na página 12.

KANEKO, Hiromasa. Cross-validated permutation feature importance considering correlation between features. *Analytical Science Advances*, Wiley Online Library, v. 3, n. 9-10, p. 278–287, 2022. Citado 1 vez na página 8.

LANGER, Markus et al. What do we want from Explainable Artificial Intelligence (XAI)?–A stakeholder perspective on XAI and a conceptual model guiding interdisciplinary XAI research. *Artificial Intelligence*, Elsevier, v. 296, p. 103473, 2021. Citado 1 vez na página 8.

LUDERMIR, Teresa Bernarda. Inteligência Artificial e Aprendizado de Máquina: estado atual e tendências. *Estudos Avançados*, SciELO Brasil, v. 35, p. 85–94, 2021. Citado 1 vez na página 8.

LUNDBERG, Scott M; LEE, Su-In. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, v. 30, 2017. Citado 1 vez na página 13.

MARCO TULIO RIBEIRO. *LIME GitHub Repository*. [S.l.: s.n.], 2023. GitHub repository. Disponível em: https://github.com/marcotcr/lime. Citado 1 vez na página 25.

MENEGOL, Marcelo S. *SHAP: O que é e por que usar*. [S.l.: s.n.], 2020. urlhttps://medium.com/big-data-blog/. Citado 1 vez na página 14.

MIRANDA, Bruno de Souza. Algoritmos clássicos de aprendizado de maquina aplicados ao problema do reconhecimento de imagens. Universidade Federal do Pampa, 2011. Citado 1 vez na página 8.

MOLNAR, Christoph. *Interpretable Machine learning*. [S.l.]: BOOKDOWN, 2022. Citado 5 vezes nas páginas 8, 12, 13, 15, 35.

MONARD, Maria Carolina; BARANAUSKAS, José Augusto. Conceitos sobre aprendizado de máquina. *Sistemas inteligentes-Fundamentos e aplicações*, Manole, v. 1, n. 1, p. 32, 2003. Citado 2 vezes nas páginas 8, 11.

NETO, Milton Gama. *Explainable AI: extraindo explicações e aumentando a confiança dos modelos de ML*. [S.l.: s.n.], 2021. urlhttps://medium.com/data-hackers. Citado 4 vezes nas páginas 8, 12.

PANDAS DEVELOPMENT TEAM. *Pandas Documentation*. [S.l.: s.n.], 2023. Website. Disponível em: https://pandas.pydata.org/docs/. Citado 1 vez na página 25.

PROJECTS, Pallets. *Flask Documentation*. [S.l.: s.n.], 2023. https://flask.palletsprojects.com/en/2.3.x/. Citado 1 vez na página 25.

RIBEIRO, Marco Tulio; SINGH, Sameer; GUESTRIN, Carlos. "Why should i trust you?" Explaining the predictions of any classifier. In: PROCEEDINGS of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. [S.l.: s.n.], 2016. P. 1135–1144. Citado 2 vezes nas páginas 9, 15.

SANCHES, Marcelo Kaminski. *Aprendizado de máquina semi-supervisionado: proposta de um algoritmo para rotular exemplos a partir de poucos exemplos rotulados*. 2003. Tese (Doutorado) – Universidade de São Paulo. Citado 1 vez na página 11.

SCIKIT-LEARN. *Permutation feature importance*. [S.l.: s.n.], 2023. urlhttps://scikit-learn.org/stable/modules/permutation_importance.html. Citado 1 vez na página 13.

SCIKIT-LEARN DEVELOPMENT TEAM. *scikit-learn Documentation*. [S.l.: s.n.], 2023. Website. Disponível em: https://scikit-learn.org/stable/. Citado 2 vezes nas páginas 25, 34.

SETH, Yashu. *Explainable AI (XAI): A Guide to 7 Packages in Python to Explain Your Models*. 2021. Disponível em: https:%20//towardsdatascience.com/explainable-ai-xai-a-guide-to-7-packages-in-python-to-explain-your-models-932967f0634b. Citado 4 vezes nas páginas 9, 15, 21.

SHAP, Documentation. *Welcome to the SHAP documentation*. [S.l.: s.n.], 2018. urlhttps://shap.readthedocs.io/en/latest/. Citado 1 vez nas páginas 14, 25.

SHAPLEY, Lloyd S. Notes on the n-Person Game – II: The Value of an n-Person Game. *Contributions to the Theory of Games*, v. II, p. 307–317, 1951. Citado 1 vez na página 14.

SOCIETY, The Royal. *Explainable AI: the basics*. [S.l.]: The Royal Society, 2019. Citado 2 vezes na página 11.

ULLAH, Irfan et al. Modeling of machine learning with SHAP approach for electric vehicle charging station choice behavior prediction. *Travel Behaviour and Society*, Elsevier, v. 31, p. 78–92, 2023. Citado 1 vez na página 9.

WOLFGANG, Ertel. *Introduction to artificial intelligence*. [S.l.]: Springer, 2017. Citado 1 vez na página 8.

ZHOU, Zhi-Hua. *Machine learning*. [S.l.]: Springer Nature, 2021. Citado 1 vez na página 8.