Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Projeto Orientado em Computação

Visualizando Falhas Críticas em Modelos de Aprendizado

Aluno: Igor Lacerda Faria da Silva
Orientador: Renato Vimieiro

Sumário

Resumo

Os recentes avanços no desenvolvimento de algoritmos de Aprendizado de Máquina, bem como o aumento da adoção destes modelos, geraram uma grande demanda por explicabilidade. Este trabalho apresenta uma ferramenta desenvolvida para auxiliar na explicabilidade de modelos de previsão (mais especificamente, de classificação), inspirada em (PRUDÊNCIO; SILVA FILHO, 2022). Diferentemente da maioria das publicações na área, são exploradas as regiões de falha dos modelos (ao invés de se estudar as *features* que levaram a uma dada previsão). O funcionamento básico da aplicação consiste em encontrar regiões nos dados onde as previsões são estatisticamente excepcionais, ou seja, regiões cujo desempenho é muito abaixo do esperado, e exibi-las por meio de uma visualização interativa. Em comparação com o artigo original, é usado um algoritmo diferente (*Beam Search*) para a descoberta das regiões.

Palavras Chave: explicabilidade, descoberta de subgrupos, visualização de dados

Abstract

Recent advancements in Machine Learning, coupled with its widespread adoption have created a demand for responsible, accountable and explainable AI. This work presents a tool to aid in explainability of prediction models (more specifically, classification models), inspired by (PRUDÊNCIO; SILVA FILHO, 2022). Although publications about explaining why a given prediction was made, based on the dataset's features are far more common, this work focuses on describing failure regions. The application works by discovering regions where predictions are statistically interesting, that is, regions where performance is particularly worse than usual and then showing these regions in an interactive visualization. The classic Beam Search algorithm is used for subgroup discovery, instead of the one proposed in the original article.

Keywords: *explainability, subgroup discovery, data visualization*

Visualizando Falhas Críticas em Modelos de Aprendizado

Igor L. F. da Silva
igorlfs@ufmg.br
Orientador: Renato Vimieiro

1. Introdução

Com o aumento do poder computacional e o desenvolvimento de algoritmos cada vez melhores, o uso de modelos de aprendizado tem aumentado consideravelmente nos últimos anos. Isso tornou as aplicações desses modelos mais diversas, incluindo áreas como biologia, saúde e direito. Além disso, esses algoritmos têm sido usados em tarefas cada vez mais elaboradas e, muitas vezes, os modelos são complexos e de funcionamento opaco, o que pode dificultar seu uso. Apesar da ampla adoção dos modelos, as previsões podem ser vistas com ceticismo, especialmente quando não há explicação de como o modelo as obteve.

Tendo em vista esse cenário, é imprescindível o desenvolvimento de ferramentas que facilitem o uso destes algoritmos. Desse modo, o objetivo deste trabalho é auxiliar na compreensão destes modelos complexos. Para atingir isso, foi desenvolvida uma adaptação do clássico *pipeline* de Aprendizado de Máquina: a obtenção de dados, adaptação dos dados a um ou mais modelos e validação dos resultados. Em particular, a ferramenta desenvolvida auxilia na validação, permitindo a avaliação das regiões críticas (ou seja, regiões em que o desempenho é muito diferente da média). Em suma, o objetivo é tornar o uso de modelos mais acessível e confiável.

Na literatura, existem diferentes abordagens para lidar com a alta complexidade dos modelos. A mais simples é optar por outro tipo de modelo, uma vez que existem algoritmos que são mais explicáveis por natureza. No entanto, essa abordagem é ineficaz do ponto de vista de performance preditiva, pois podem existir algoritmos complexos que modelam melhor um dado problema. Nesses casos, se torna necessário escolher entre explicabilidade e qualidade das previsões. Portanto, é desejável trabalhar com soluções agnósticas, que consigam tratar todos os tipos de modelo. A análise proposta permite estudar os modelos dessa maneira pois leva em conta somente as entradas e saídas dos mesmos.

A área de compreensão de modelos complexos pode ser subdividida em duas subáreas: o estudo das explicações de predições e o estudo regiões de falha. Na primeira, busca-se entender quais características são mais ou menos relevantes para o desempenho dos modelos. Já a segunda busca entender em quais circunstâncias o desempenho do modelo foi fraco. Este trabalho implementa uma aplicação para estudo de regiões de falha, baseada em (PRUDÊNCIO; SILVA FILHO, 2022). Em comparação com o trabalho original, a aplicação desenvolvida usa um algoritmo diferente para a descoberta de subgrupos: o *Beam Search* (mais detalhes são discutidos na Seção 2). A implementação usada faz parte da biblioteca `pysubgroup`.

Além de identificar as regiões com desempenho aquém do esperado, que são mais mais valiosas para os usuários, a aplicação também identifica regiões de desempenho elevado. As regiões de baixa performance revelam em quais situações a saída do modelo pode não ser confiável. Com isso, a comparação entre modelos permite explorar se a dificuldade da região é intrínseca ao problema. Por outro lado, o conhecimento dos pontos fracos de um modelo permite que outro modelo (ou outra abordagem) seja escolhido(a) para a região em questão.

Um resumo do funcionamento da ferramenta é apresentado na Figura 1. O *dataset* faz parte da entrada, assim como um arquivo para descrever os erros do modelo[1], enquanto o modelo em si é um elemento exterior. O programa então realiza a descoberta de subgrupos, faz uma filtragem intermediária e, por fim, faz a construção da visualização, permitindo assim a comparação de subgrupos.

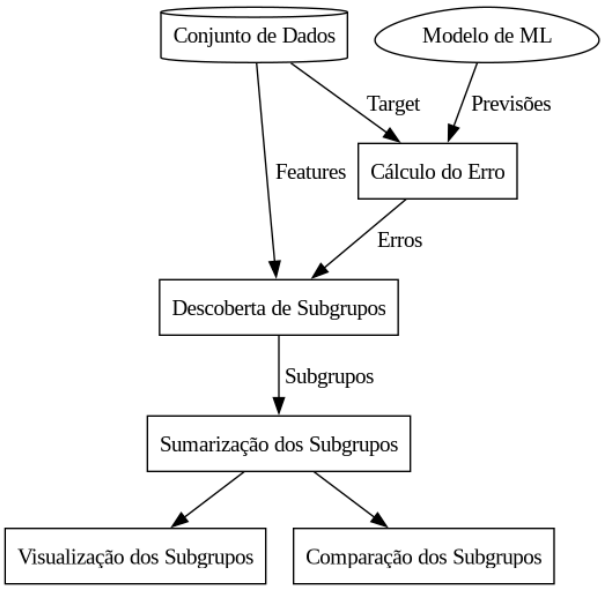

Figura 1: *Pipeline* da ferramenta

O trabalho foi desenvolvido em Python 3.12, com o *framework* `dash`. A estilização foi feita com o *framework* `tailwind`. O repositório está disponível `aqui`. O funcionamento é simples (a entrada será detalhada na Seção 3): ao abrir a visualização, o usuário seleciona um limiar para a similaridade das regiões; seleciona as regiões que deseja analisar e clica em um botão para desenhar os retângulos nas regiões selecionadas. A ferramenta desenvolvida pode ser usada com modelos de classificação em que as *features* são numéricas.

Além deste capítulo de introdução, este relatório está dividido da seguinte maneira: a Seção 2 discute trabalhos relacionados e conceitos teóricos usados durante o desenvolvimento da aplicação; a Seção 3 descreve as atividades conduzidas e a Seção 4 apresenta conclusões e sugestões de trabalhos futuros.

2. Referencial

Há duas possibilidades no que diz respeito à explicabilidade de modelos: a intrínseca e a *post-hoc* (PIMENTEL; AZEVEDO; TORGO, 2023). Na intrínseca, a própria natureza do modelo provê as interpretações. Já a *post-hoc* consiste em usar ferramentas para analisar um algoritmo com base nas suas entradas e saídas, assim atingindo independência de um tipo particular de modelo. Como comentado na introdução, o trabalho segue esta abordagem.

A maioria das técnicas publicadas na literatura estudam os efeitos de uma ou mais *features* nas previsões. Por exemplo, os gráficos de *Individual Conditional Expectation* (ICE) ou os *Partial Dependence Plots* (PDP). Os PDP mostram a relação entre uma variável contínua e a previsão do modelo. O ICE é semelhante, mas avalia o efeito para cada exemplo de entrada (PIMEN-

[1]Essa etapa deve ser realizada a priori.

TEL; AZEVEDO; TORGO, 2023). Essas análises de efeitos nas previsões são complementares à análise de regiões críticas.

Em termos de técnicas de análise de erro (de forma geral), também há grande diversidade de técnicas, mas elas deixam a desejar em alguns aspectos. As mais simples são estatísticas como a taxa de erro, o erro médio absoluto e o erro médio quadrático. Além disso, existem métodos visuais, como as curvas *Receiver Operating Characteristic* (ROC) para modelos de classificação e *Regression Receiver Operating Characteristic* (RROC) para modelos de regressão (PIMENTEL; AZEVEDO; TORGO, 2023). Um aspecto negativo dessas abordagens é que, frequentemente, elas produzem uma visão generalista dos dados, quando pode ser mais interessante para o usuário uma perspectiva mais local.

Para lidar com essa questão, a solução desenvolvida é uma implementação de uma técnica inspirada em (PRUDÊNCIO; SILVA FILHO, 2022). Inicialmente, é realizada uma descoberta de subgrupos para encontrar as regiões em que o desempenho do modelo é anormal, podendo tanto ser abaixo do esperado (*Local Hard Region* - LHR) ou acima do esperado (*Local Easy Region* - LER). O artigo original propõe seu próprio algoritmo para esta fase, mas neste trabalho foi usado o algoritmo de *Beam Search* da biblioteca `pysubgroup`.

Para a função de qualidade, foi usada uma modificação de uma das medidas de qualidade padrão do `pysubgroup`: o `StandardQFNumeric`. Ela considera o número de instâncias no subgrupo (n), elevado a uma constante α, multiplicado pela diferença entre o erro médio do subgrupo e ($\overline{s}$) o erro médio global[2] ($\overline{s}$). A modificação consiste em considerar o valor absoluto da diferença (pois tanto valores acima quanto abaixo do esperado são considerados interessantes). Além disso, foi usado o parâmetro de $\alpha = 0.5$, obtendo-se a seguinte fórmula:

$$Q = \sqrt{n} \mid \overline{s} - \overline{x} \mid \tag{1}$$

É importante considerar o tamanho do subgrupo na medida de qualidade, para que se possa filtrar subgrupos que, apesar de excepcionais, cobrem poucas instâncias. A literatura apresenta diversas outras métricas, como o WRAcc e o *lift* (MARVIN MEENG; ARNO KNOBBE, 2020).

De forma geral, a descoberta de subgrupos pode ser definida como: dada uma população de indivíduos e uma propriedade de interesse destes indivíduos, a descoberta de subgrupos é a tarefa de encontrar subgrupos da população que são "estatisticamente interessantes" com base na propriedade em questão (WROBEL, 2001).

O algoritmo clássico (e usado neste trabalho) para a descoberta de subgrupos é o *Beam Search*. O *Beam Search* é uma heurística gulosa, similar a uma *best-first search*, mas ao invés de explorar todo o espaço de busca, somente uma fração promissora é considerada, o que reduz drasticamente a demanda computacional. Para isso, uma busca por níveis é realizada, como em uma *breadth-first search*, mas somente um conjunto (cujo tamanho é definido por um parâmetro *beam width*) seleto de soluções parciais é mantido como soluções candidatas (LEEUWEN; ARNO KNOBBE, 2012).

A saída do *Beam Search* é uma lista de conjunções de seletores: regiões que descrevem o espaço de *features*. Apesar do uso amplo, o algoritmo possui algumas limitações. A principal delas

[2]Uma outra variação poderia ser considerar o erro médio excluindo o subgrupo em questão.

é o fato de o resultado final ser consideravelmente redundante. Em bases de dados de *benchmark*, o *Beam Search* é incapaz de discernir essa redundância. Por exemplo, em um *benchmark* conduzido por (LEEUWEN; ARNO KNOBBE, 2012), o *Beam Search* descreveu 100 grupos diferentes quando, em essência, apenas 2 eram suficientes para explicar a região.

Após a descoberta de subgrupos, é realizada uma filtragem preliminar dos subgrupos (na visualização é possível ter um controle mais fino). O objetivo desta etapa é remover redundâncias. Ou seja, regras que já são cobertas por outras regras na lista de regiões gerada anteriormente. Isso é importante, pois é possível que os principais grupos sejam muito semelhantes, de modo que usá-los na visualização apenas introduza complexidade desnecessária. Por fim, após o processamento, é exibida a visualização propriamente dita. Seu funcionamento é descrito detalhadamente na Seção 3.

Em suma, essa abordagem busca regiões de baixo (ou alto) desempenho e as seleciona de maneira a evitar redundância, para que se possa entender quais condições cobrem os desvios apresentados no modelo. Na seção seguinte, a implementação em si é explorada mais detalhadamente.

3. Contribuição

O programa desenvolvido pode ser dividido em duas partes: o cálculo dos subgrupos e a visualização. O cálculo dos subgrupos, por sua vez, é dividido nas duas etapas apresentadas na seção anterior: a descoberta dos subgrupos e a filtragem de subgrupos redundantes.

Na primeira etapa foi usado o *Beam Search*, mas para o POC II, está planejado o desenvolvimento de um algoritmo de busca próprio. A saída desta primeira etapa é um conjunto de subgrupos, em que cada grupo é descrito como a conjunção de dois seletores. Para dados numéricos, os seletores geralmente são compostos por intervalos, mas também podem conter um único elemento.

O programa aceita até 5 argumentos. São eles: o *path* de um CSV contendo o *dataset*, o *path* de um CSV contendo os erros do modelo, uma *string* para selecionar a classe de interesse, o tamanho do *beam width* (e consequentemente a quantidade máxima de subgrupos) e a coluna que representa o y no *dataset* (que, por padrão é *"target"*). Mais detalhes sobre como deve ser o formato dos arquivos estão no README do projeto, no GitHub, disponível aqui.

A visualização possui três elementos principais: um dendrograma, uma tabela e um *scatter plot*. A tabela é o elemento mais simples da visualização. Seu intuito é oferecer uma visão geral dos subgrupos encontrados, assim como estatísticas básicas. Ela possui 4 colunas: a primeira apresenta os seletores que descrevem o subgrupo; a segunda é o tamanho do subgrupo; a terceira o erro médio do grupo e a quarta, a qualidade, com base na Equação 1.

SUBGROUP	SIZE	AVG ERROR	QUALITY
petal width: [1.50:1.90[AND sepal length: [5.0:5.60[	1	0.020	0.020
petal length<1.50 AND sepal width<2.70	1	0.020	0.020
sepal length: [5.0:5.60[AND sepal width: [3.0:3.10[	2	0.010	0.014
sepal length<5.0 AND sepal width<2.70	3	0.007	0.011
petal length: [3.90:4.70[AND sepal length: [5.0:5.60[	5	0.004	0.008
petal length: [3.90:4.70[AND sepal width: [3.0:3.10[	7	0.003	0.007
petal width: [1.50:1.90[AND sepal width: [3.0:3.10[	9	0.002	0.005
petal length: [3.90:4.70[AND petal width: [1.50:1.90[	9	0.002	0.005
petal width: [0.20:1.20[AND sepal width<2.70	10	0.002	0.005
petal length<1.50 AND petal width: [0.20:1.20[	21	0.001	0.005
sepal length: [5.60:6.10[AND sepal width>=3.40	4	0.003	0.004
petal length: [1.50:3.90[AND sepal length: [5.60:6.10[	4	0.003	0.004
petal length<1.50 AND sepal length<5.0	13	0.002	0.004
petal width: [0.20:1.20[AND sepal length<5.0	17	0.001	0.003
petal width: [0.20:1.20[AND sepal length: [5.60:6.10[	7	0.001	0.003

Figura 2: Tabela com subgrupos (e estatísticas) para o *dataset* iris, carregado via `sklearn`[3].

O dendrograma (Figura 3), por outro lado, é o elemento mais complexo da visualização. A ideia do dendrograma é ilustrar como a cobertura de instâncias está distribuída, e usar disso para ter um controle mais fino da visualização. Para a construção do dendrograma, é fundamental que a função de distância esteja bem definida. A distância entre os subgrupos é calculada com base na diferença entre as instâncias cobertas, usando o índice de Jaccard. De fato, para calcular o índice de Jaccard, tanto a diferença entre as instâncias cobertas, como também a diferença entre os próprios seletores de cada subgrupo são considerados (é escolhido o máximo entre essa opções). Os seletores, nesse contexto, são os intervalos que descrevem a região.

[3]Este exemplo será usado nas outras figuras.

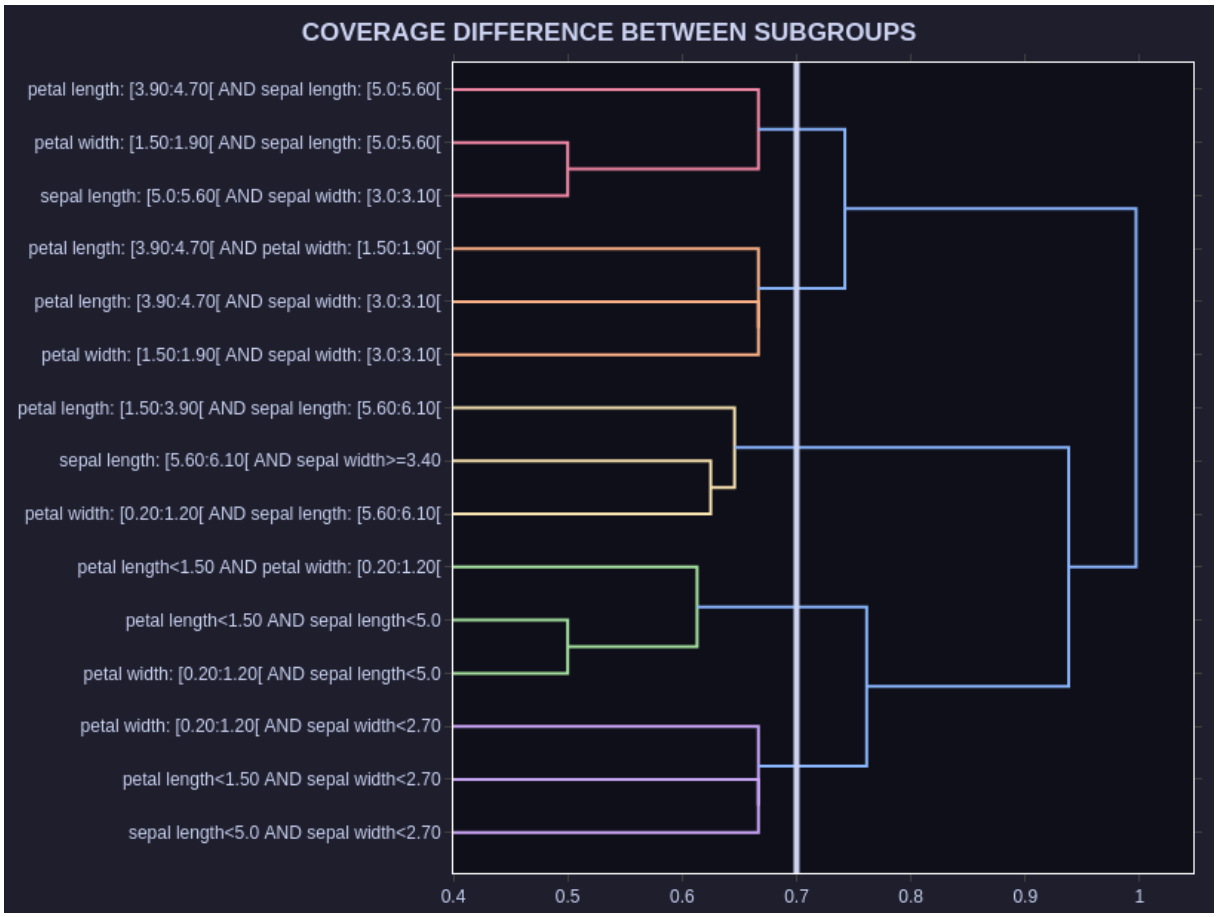

Figura 3: Dendrograma listando subgrupos para o *dataset* iris, para a classe `setosa`.
A linha branca representa o limiar.

Além disso, o dendrograma é interativo, aliado ao componente de seleção de limiar. A ideia do limiar é proporcionar um filtro complementar, além da remoção de subgrupos redundantes apresentada na seção anterior. O limiar é representado pela linha branca e cada interseção com o dendrograma seleciona um subgrupo (dado que há uma escolha entre pelo menos dois caminhos). O subgrupo é selecionado com base em um algoritmo que elege aquele de maior qualidade no caminho, via Equação 1.

A princípio, o limiar não tem nenhum valor, o que permite explorar todos os subgrupos para a classe escolhida. O componente de seleção de limiar aparece na Figura 4. Logo abaixo dele está a caixa de seleção para devidamente escolher os subgrupos. Na Figura 5, um subgrupo está selecionado e o limiar está configurado. Quando o limiar possui algum valor, é possível reiniciá-lo ao clicar no botão “CLEAR”, assim permitindo maior explorabilidade da aplicação.

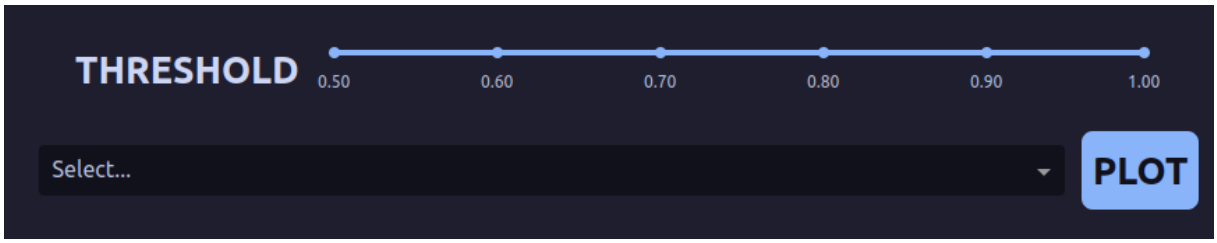

Figura 4: Componentes de seleção de limiar e subgrupos.
O Limiar contém apenas o intervalo válido para a separação dos grupos.

Figura 5: Componentes de seleção de limiar e subgrupos, com os dados selecionados.

Resumindo, o usuário segue o fluxo: observar a tabela com os subgrupos, selecionar um ou mais subgrupos usando o componente de seleção e clicar em "PLOT". Caso o usuário se sinta intimidado com a quantidade de opções, ele pode clicar no seletor do limiar e ver que a linha é desenhada no dendrograma. Ao perceber isso, o usuário clica novamente no seletor e percebe que as opções foram filtradas. Ao explorar as diferentes possibilidades, é possível que o usuário se encontre em um estado indesejável devido ao filtro, bastando clicar em "CLEAR" para reverter ao estado original.

Por fim, temos o *scatter plot*. Inicialmente ele apresenta apenas um *plot* das duas primeiras dimensões do *dataset*, sem nenhum subgrupo desenhado. Conforme se seleciona os subgrupos, as regiões são *plotadas*. A caixa de seleção possui um filtro para que subgrupos de colunas distintas não sejam selecionadas, para impedir que dois gráficos incompatíveis sejam gerados. LERs são destacadas de verde, enquanto LHR são destacadas de vermelho. No canto inferior esquerdo, é indicado erro médio do subgrupo, enquanto que no canto superior direito, é indicado o erro médio global. Além disso, no centro da região, os seletores do grupo aparecem, o que facilita a distinção quando muitos seletores estão ativos.

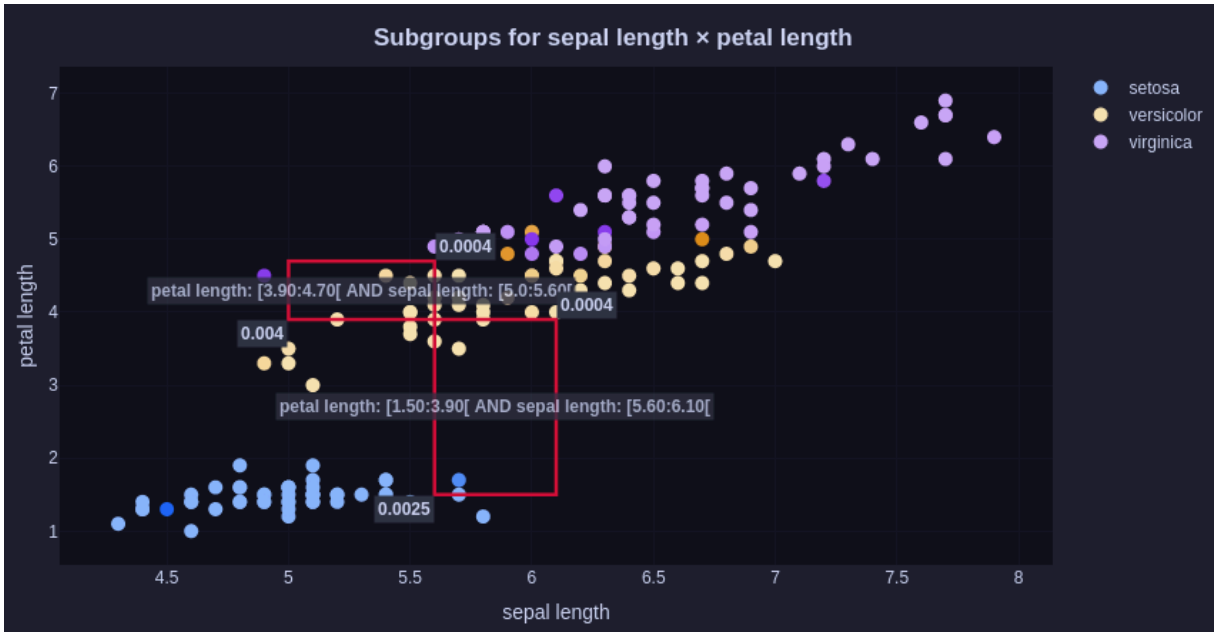

Figura 6: *Scatter plot* para o *dataset* de exemplo, `iris`. São ilustradas duas LHRs (de vermelho).

Outro aspecto da visualização é sua responsividade. Em particular, com o uso do *framework* `tailwind`, foram realizadas adaptações para telas pequenas: a página assume um *layout* de coluna, com a tabela no topo, o dendrograma em seguida, o limiar e o restante da aplicação. Essa ordem favorece com que o dendrograma esteja próximo do seletor de limiar, facilitando o

uso. Outros ajustes finos nos tamanhos dos componentes também foram realizados. Para telas maiores, o dendrograma divide a linha com a tabela.

4. Fechamento

O objetivo desta etapa de desenvolvimento foi atingido. Foi desenvolvido um Produto Mínimo Viável para ser usado em análises de validação de modelos de aprendizado, no contexto de regiões de falha. Para as próximas etapas do projeto, estão planejadas para o POC II:

1. O desenvolvimento de um algoritmo de busca próprio
2. Maior interatividade, com possibilidade de atualizar os arquivos de entrada dinamicamente.
3. Comparação entre múltiplos modelos, com uso de um diagrama de Venn para explicar as interseções entre as falhas.

Bibliografia

LEEUWEN; ARNO KNOBBE. **Diverse subgroup set discovery**. Springer, , 31 out. 2012.

MARVIN MEENG; ARNO KNOBBE. **For real: a thorough look at numeric attributes in subgroup discovery**. Springer, , 21 set. 2020.

PIMENTEL, J. ; AZEVEDO, P. J. ; TORGO, L. Subgroup mining for performance analysis of regression models. **Expert Systems**, v. 40, p. e13118, 2023.

PRUDÊNCIO, R. B. C. ; SILVA FILHO, T. M. Explaining Learning Performance with Local Performance Regions and Maximally Relevant Meta-Rules. **Intelligent Systems**, v. 13653, p. 550–564, 2022.

WROBEL, S. **Inductive logic programming for knowledge discovery in databases**. Springer, , 2001.