DOI: 10.5748/20CONTECSI/PSE/AIT/7244

eLocator: e207244

FERRAMENTA GRÁFICA PARA A AVALIAÇÃO DE MODELOS E CONJUNTOS DE DADOS DE APRENDIZADO DE MÁQUINA PELA TEORIA DE RESPOSTA AO ITEM

Thiago Paracampo De Castro – https://orcid.org/0000-0002-7432-2104
Universidade Federal Do Pará (Ufpa)

Lucas Cardoso – https://orcid.org/0000-0003-3838-3214
Universidade Federal Do Pará (Ufpa)

Ronnie Alves – https://orcid.org/0000-0003-4139-0562
Instituto Tecnológico Vale (Itv)

Regiane Kawasaki – https://orcid.org/0000-0003-3958-064X
Universidade Federal Do Pará (Ufpa)

GRAPHICAL TOOL FOR EVALUATING MACHINE LEARNING MODELS AND DATASETS BY ITEM RESPONSE THEORY

ABSTRACT

The use of Artificial Intelligence systems that involve Machine Learning are increasingly common and this is due to the versatility of the application of these systems in different spheres, whether with academic interest or market interest. However, it is still not a simple task to define the best model for a given problem and whether the generated model can be explained to the end user who uses it. Given this, recent research has applied psychometric concepts to make model assessments more robust. Among recent research is the development of the decodIRT tool that automatically implements the use of Item Response Theory applied to Machine Learning. Given its increasing use, this work aims to develop a user-centered graphical interface version of the decodIRT tool aimed at helping end users understand the application of Item Response Theory to evaluate datasets and algorithms of Machine Learning and explanation of the generated models.

Keywords: Artificial Intelligence, Machine Learning, Item Response Theory, Classification and Graphical Interface.

FERRAMENTA GRÁFICA PARA A AVALIAÇÃO DE MODELOS E CONJUNTOS DE DADOS DE APRENDIZADO DE MÁQUINA PELA TEORIA DE RESPOSTA AO ITEM

RESUMO

O uso de sistemas de Inteligência Artificial que envolvem Aprendizado de Máquina são cada vez mais comuns e isso se deve a versatilidade da aplicação desses sistemas em diferentes esferas, seja com interesse acadêmico ou por interesse de mercado. Entretanto, ainda não é tarefa simples definir qual o melhor modelo para um determinado problema e se o modelo gerado pode ser explicado para o usuário final que utilizá-lo. Diante disso, pesquisas recentes aplicaram conceitos da psicometria para poder dar mais robustez as avaliações dos modelos. Dentre as pesquisas recentes está o desenvolvimento da ferramenta decodIRT que implementa de forma automatizada o uso da Teoria de Resposta ao Item aplicada ao Aprendizado de Máquina. Dado o seu uso crescente, este trabalho tem como objetivo o desenvolvimento de uma versão com interface gráfica centrada no usuário da ferramenta decodIRT voltada para auxiliar o entendimento do usuário final sobre a aplicação da Teoria de Resposta ao Item para avaliação de conjuntos de dados e algoritmos de Aprendizado de Máquina e explicação dos modelos gerados.

Palavras-chave: Inteligência Artificial, Aprendizado de Máquina, Teoria de Resposta ao Item, Classificação e Interface Gráfica.

1. Introdução

A popularização de técnicas de Aprendizado de Máquina nos últimos anos se dá devido a sua utilidade crescente em diversas áreas do conhecimento. Isso, aliado à grande quantidade de dados gerados diariamente, contribui para sistemas mais robustos e com resultados favoráveis. No entanto, a qualidade de criação de um modelo de Aprendizado de Máquina (AM) está intrinsecamente atrelada à qualidade dos seus dados de treinamento, desse modo, a qualidade dos modelos está diretamente associada a complexidade dos dados. Dessa forma, entende-se que a avaliação de modelos deve também considerar também o dado utilizado.

Além disso, destaca-se que existem diversos tipos de algoritmos que implementam técnicas diferentes de aprendizado. Logo, tendem a ter desempenhos distintos a depender do tipo e da complexidade dos dados utilizados. Entretanto, as métricas clássicas de Aprendizado de Máquina comumente não consideram a complexidade da instância ao realizar a medição de habilidade de um modelo. Entender como se dá a relação dado e classificador, pode ser útil para entender qual algoritmo escolher para um determinado problema. Trabalhos recentes exploram esse problema utilizando de conceitos de avaliação da psicometria por meio da Teoria de Resposta ao Item (TRI) [Martínez-Plumed et al. 2016, Martínez-Plumed et al. 2019], dada a sua capacidade de avaliar respondentes considerando a complexidade do item [de Andrade et al. 2000].

Para entender se um modelo é mais habilidoso que outro para determinado tipo de dado implica entender se o modelo realmente aprendeu e se realmente é possível confiar nas suas predições. A área do conhecimento especializada em realizar essa análise é chamada de *Explanable Artificial Inteligence* (XAI) [Gunning e Aha 2019]. O XAI é uma abordagem *user-centric* que desenvolve técnicas capazes de explicar a predição de um modelo com o objetivo de tornar o resultado da predição mais confiável para o usuário final [Gunning e Aha 2019].

A ferramenta decodIRT [Cardoso et al. 2020] foi desenvolvida para apoiar análises de classificadores e *datasets* de AM pela TRI de forma automatizada. Desde então o decodIRT têm sido utilizado em mais trabalhos recentes, por exemplo no estudo [Cardoso et al. 2022] o decodIRT é utilizado para realizar análises de XAI a partir dos resultados da TRI. Assim como o trabalho de [Araujo Santos et al. 2023] que também utiliza a ferramenta decodIRT para gerar os estimadores da TRI para avaliação de desempenho de modelos de AM.

A ferramenta, além de permitir o uso de dados coletados pelo próprio usuário, utiliza o OpenML, um ecossistema online que visa facilitar a pesquisa na área de Aprendizado de Máquina através da disponibilização de dados e desafios diversos para uso dos pesquisadores. O foco deste trabalho é utilizar as metodologias propostas nos artigos supracitados para remodelar o decodIRT em uma ferramenta com interface gráfica *user-centric* de forma a facilitar as análises de resultados obtidos pela TRI para algoritmos e *datasets* de AM.

O restante deste trabalho está dividido da seguinte forma: A seção 2 apresenta os conceitos teóricos necessários sobre a TRI, XAI e a ferramenta decodIRT; A seção 3 apresenta a metodologia utilizada no desenvolvimento deste trabalho; A seção 4 expõe os resultados obtidos e a seção 5 conclui a pesquisa e traz os trabalhos futuros.

2. Referencial Teórico

2.1 Teoria de Resposta ao Item

Segundo [de Andrade et al. 2000], para se avaliar o desempenho de indivíduos em um teste, tradicionalmente utiliza-se a quantidade total de acertos. Apesar de ser comum, essa abordagem possui limitações para avaliar a real habilidade de um indivíduo. Em contrapartida, a TRI permite avaliar as características latentes de um indivíduo que não podem ser observadas diretamente e visa apresentar a relação entre a probabilidade de um indivíduo responder corretamente a um item e sua habilidade na área de conhecimento avaliada. Uma das principais características da TRI é ter como elementos centrais os itens e não o teste como um todo, isto é, o desempenho de um indivíduo é avaliado a partir da sua capacidade de acertar determinados itens de um teste e não quantos itens ele acerta.

A TRI é um conjunto de modelos matemáticos que buscam representar a probabilidade de um indivíduo acertar um item em função dos parâmetros do item e da habilidade do respondente, onde quanto maior for a habilidade do indivíduo, maior a chance de acerto. Dentre os tipos de item existentes, os dicotômicos são os mais utilizados, onde apenas considera-se se o item foi respondido corretamente ou não [de Andrade et al. 2000]. A TRI permite avaliar simultaneamente tanto os itens quanto os respondentes.

Para avaliar os itens, dispõe-se dos parâmetros que descrevem o item: Discriminação (a_i), que determina o quão bem um item *i* consegue discriminar os respondentes mais habilidosos dos menos habilidosos; Dificuldade (b_i), que determina o quão difícil é o item *i*; Adivinhação (c_i), que representa a probabilidade de um item *i* ser respondido corretamente em um acerto casual.

Para estimar os parâmetros de item, utiliza-se o conjunto de resposta de todos os indivíduos para todos os itens a serem avaliados. A avaliação dos respondentes é feita a partir da habilidade estimada (θ) e a probabilidade de resposta correta calculada em função da habilidade de um indivíduo *j* e dos parâmetros do item *i*. O modelo logístico que utiliza os três parâmetros (3PL) é definido pela equação 1:

$$P(U_{ij} = 1|\theta_j) = c_i + (1 - c_i)\frac{1}{1 + e^{-a_i(\theta_j - b_i)}}$$

A TRI então pode ser entendida como sendo uma “lupa” que permite observar o desempenho do indivíduo de forma específica sobre cada item e estimar um provável nível de habilidade na área que está sendo avaliada. A TRI também dispõe do cálculo do True-Score [Lord e Wingersky 1984] que visa gerar uma pontuação final por meio da somatória das probabilidades de acerto calculadas para cada item do teste.

Assim como as métricas clássicas de Aprendizado de Máquina, por exemplo a acurácia, a forma clássica de avaliação de indivíduos soma 1 para cada acerto e 0 para erro. Esse formato não considera a complexidade do item ao dar o mesmo peso para itens fáceis e difíceis. A partir da analogia que os modelos são os indivíduos, as instâncias são os itens e o *dataset* é o teste, é possível aplicar conceitos robustos da psicometria para avaliar modelos e *datasets* de AM. Dessa forma, é como se cada instância de teste fosse uma questão de uma prova de múltipla escolha que o modelo está fazendo, onde as alternativas são as possíveis classes do *dataset* (Ver Figura 1).

Feature 1, *Feature* 2, *Feature* 3, *Feature* 5, *Feature* 6, *Feature* 7, *Feature* 8, *Feature* 9, *Feature* 10...

(A) Classe A;
(B) Classe B;
(C) Classe C;
(D) Classe D;
(E) Classe E.

Figura 1. Exemplo de como cada instância é vista como um item.

2.2 Inteligência Artificial Explicada

A expansão e uso cada vez maior de sistemas de AM cria avanços que permitem que tais sistemas inteligentes sejam capazes de perceber, aprender e tomar decisões por conta própria [Gunning e Aha 2019]. Para que esse avanço continue crescendo, toda a comunidade está diante da barreira da explicabilidade do modelo devido ao crescente uso de modelos do tipo *black-box*. Tais modelos, comumente possuem uma alta acurácia para diversos problemas, logo são preferíveis sobre os demais. Entretanto, apenas são capazes de responder perguntas de sim ou não, sem explicar como a resposta foi obtida [Gohel et al. 2021]. E segundo [Arrieta et al. 2020], quanto mais habilidoso for um modelo, mais difícil será interpretar e explicar a sua decisão. Para contornar esse problema um novo campo de estudo chamado de Inteligência Artificial Explicada (XAI) vem crescendo recentemente.

O XAI é um campo de estudo centrado no usuário que visa desenvolver técnicas de explicação capazes de permitir que o usuário consiga entender o motivo da decisão tomada pelo modelo e tornar a IA mais transparente [Arrieta et al. 2020]. O fato de cada vez, mais sistemas de AM estarem tomando decisões importantes ligadas a vida das pessoas torna ainda maior a importância de estudos em XAI, onde já existem regulamentações no mundo que defendem que o indivíduo tem o direito a explicação do porquê da decisão de um modelo, como o Regulamento Geral sobre a Proteção de Dados implementado em 2018 pela União Europeia.

2.3 Ferramenta decodIRT

Para estimar os parâmetros de item, a habilidade dos modelos e calcular a probabilidade de acerto, foi utilizado a ferramenta decodIRT proposta por [Cardoso et al. 2020]. O decodIRT é uma ferramenta escrita em Python 3.7 que funciona em linha de comando e tem como objetivo principal automatizar a análise de conjuntos de dados existentes na plataforma OpenML bem como na proficiência de diferentes classificadores. Para isso, ele depende da probabilidade de acerto derivada do modelo logístico da TRI, bem como dos parâmetros de item e da habilidade dos respondentes.

A ferramenta consiste em um total de três *scripts* principais projetados para serem usados em sequência. O primeiro *script* é responsável por baixar os *datasets* do OpenML, gerar os modelos de ML e colocá-los para classificar os *datasets*. Por definição, será feito um split estratificado de 70% para treino e 30% para teste. Em seguida, é gerado uma matriz de resposta, que contém o resultado da classificação de todos os classificadores para cada instância de teste. No total são: 120 modelos de Redes Neurais, aumentando gradativamente a profundidade das redes; 12 modelos de diferentes famílias de algoritmos; 7 classificadores artificiais propostos por [Martínez-Plumed et al. 2016] para servirem de indicadores de

limite, sendo: um classificador ótimo (acerta todas as classificações), um péssimo (erra todas), um majoritário (classifica todas as instâncias com a classe majoritária), um minoritário (classifica com a classe minoritária) e três classificadores randômicos (classificam aleatoriamente). Totalizando 139 classificadores que terão as suas respostas coletadas.

A matriz de resposta é a entrada para o segundo *script*, que por sua vez é responsável pelo cálculo dos parâmetros do item. De forma que, para cada instância que foi respondida pelos classificadores será estimado os parâmetros de dificuldade, discriminação e adivinhação. O último *script* usará os dados gerados pelos anteriores para avaliar os *datasets* usando os parâmetros do item, estimar a habilidade dos classificadores e calcular a probabilidade de acerto de cada modelo.

3. Materiais e Métodos

3.1. Arquitetura e Planejamento do Software

Dados os requisitos da ferramenta, sendo esses a simplicidade de uso e visualização de dados, uma arquitetura e seu eventual planejamento foram concebidos, a fim de facilitar seu processo de construção. Devido a sua simplicidade e escalabilidade, uma infraestrutura de visualização *Web* foi proposta para a interface, complementada por uma API (*Application Programming Interface*) que realiza a comunicação entre o decodIRT e a interação com o usuário.

Portanto, tem-se uma Arquitetura de Duas Camadas, no qual uma API recebe e envia mensagens a um servidor *Web*, dedicado à exibição da interface. No entanto, a organização dessa API também é um desafio. É necessário que ela cumpra o papel de mediadora entre o decodIRT e a interface. Desse modo, definiu-se que essa seria escrita em Python, mesma linguagem na qual decodIRT fora escrito, assim como seria um servidor *Web* capaz de conversar com o provedor da interface. Nesse sentido, a arquitetura proposta na Figura 2 foi fixada.

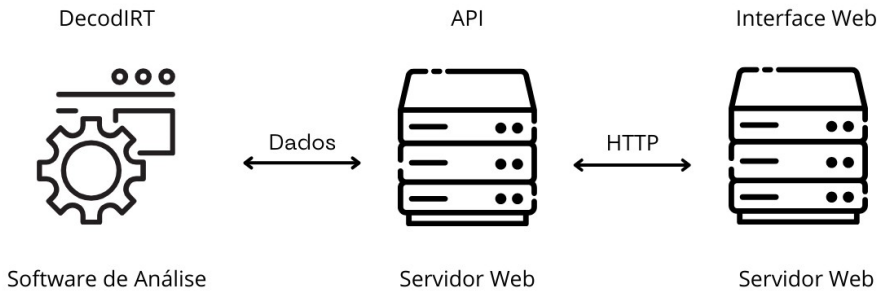

Figura 2. Representação visual da arquitetura de duas camadas escolhida.

Baseado nessas escolhas, o desenvolvimento do *software* se tornou mais fácil. As ferramentas escolhidas não só são simples e escaláveis, mas cumprem os requisitos de um pacote de instalação e uso descomplicado. Apesar disso, é necessário que a API e a interface também sejam devidamente planejadas, a fim de evitar futuros retrabalhos.

O próximo passo é ter uma noção de que tipos de serviço a ferramenta proverá ao usuário. Para retratar isso, utilizou-se um diagrama de casos de uso com o intuito de entender melhor sua organização e funcionalidades. Ver a Figura 3 para o diagrama completo desenvolvido.

Figura 3. Diagrama de casos de uso completo.

3.2. Concepção da API Web

Uma API é um conceito abstrato usado em ciência da computação que denota uma série de regras que media a interação entre dois *softwares*, as quais podem ser definidas por outro programa. No caso deste trabalho, era necessário que uma interface *Web* interagisse com um programa cliente já existente, por isso a fim de simplificar o processo e evitar retrabalhos no decodIRT usou-se outro servidor *Web* que realiza esse papel.

Como dito anteriormente, esse servidor *Web* deveria ser escrito em Python, portanto, escolheu-se o microframework Flask. Que é simples e enxuta, algo necessário para a criação de um ambiente minimalista livre de complexidade. Como consequência, o desenvolvimento começou, tendo como referência o diagrama exposto na Figura 3.

3.2.1. Arquitetura da API

A API é dividida em três módulos: *GetDataset*, *Execute* e *Analyse*. Os módulos da API foram desenvolvidos para se adequarem a divisão de três módulos definidas no decodIRT, no qual cada um tem objetivo único. A fim de simplificar e modularizar o software, o mesmo acontece aqui. É importante notar que, na maioria das vezes, em um caso de execução normal, esses são executados na ordem apresentada. A seguir são apresentados os módulos e suas respectivas funcionalidades:

- **GetDataset**: O primeiro módulo é responsável por conversar com a API do OpenML, a fim de obter os *datasets* necessários para análise. Além disso, é responsável por armazenar, em um arquivo, quais *datasets* estão disponíveis para análise;
- **Execute**: O segundo módulo realiza a execução do decodIRT. Os principais *scripts* do programa executados aqui são o primeiro, para baixar os dados do OpenML e gerar os modelos de AM, e o segundo, que calcula os parâmetros de item da TRI;
- **Analyse**: O terceiro módulo é o de análise dos dados gerados pelos anteriores. Ele gera representações gráficas de interesse para estudo de determinado *dataset*, apresentando gráficos de True-Scores e dos parâmetros estudados na Teoria de Resposta ao Item.

3.2.2. Implementação da API

Embora a API esteja bem organizada em uma camada superficial, é necessário que o código esteja adequado para padrões modernos de engenharia. Desse modo, concluiu-se que o padrão de projeto MVC (*Model-View-Controller*) seria utilizado para organizar a implementação. Este modelo divide a aplicação em três partes: o modelo, a visualização e o controlador.

O modelo é aquele que realiza a atualização e recebimento de dados, geralmente de um banco de dados. No caso da interface, um banco completo foi considerado desnecessário, pois o único armazenamento do programa é composto de metadados de *datasets*. Portanto, usou-se um arquivo de texto formatado em uma linguagem de marcação chamada TOML (*Tom's Obvious Minimal Language*) para essa função. Um fator de interesse é o fato de esse tipo de arquivo ser facilmente convertido para um dicionário da linguagem Python.

A visualização é considerada a interface da aplicação. Neste caso, esse componente foi separado para utilização de outra tecnologia especializada em *frontend*. No entanto, os módulos da API entendem esta como uma parte do MVC. Ela não toma nenhuma decisão de controle e só interage com o usuário, portanto, essa separação não afetou a estrutura como um todo.

O controlador é o coração da aplicação. É nele que a lógica das regras de negócio são armazenadas, portanto, este faz o intermédio entre os outros dois módulos. Neste trabalho, o programa decodIRT é um pacote que é chamado pelo controlador quando oportuno. A utilização desse modelo garante mais robustez à ferramenta e facilita futuras manutenções.

3.2.3. Concepção da Interface

A interface foi definida como uma parte separada, a fim de priorizar o uso de uma tecnologia específica para *frontend*. Nesse contexto, utilizou-se a *framework* VueJs, composta por um ecossistema completo e de uso intuitivo, por isso escolhida. Muitas das complexidades relacionadas ao desenvolvimento *Web* foram abstraídas com o uso dessa tecnologia. Com isso, entende-se que uma das restrições impostas era de que a interface deveria se comportar como uma SPA (Single Page Application), isso tornaria o aspecto do programa mais moderno e rápido.

Uma SPA carrega somente a parte afetada por uma interação com o usuário, o que facilita a presença de um painel lateral em várias páginas conectadas, por exemplo. Para tanto, empregou-se o módulo *VueRouter*, o qual simplifica esse processo. Ele funciona a partir da criação de um *layout*, um padrão que se repetirá em uma sequência de páginas relacionadas. Dentro desse *layout*, um objeto é chamado para representar o componente que deve ser carregado, no caso, a parte principal, que se repete de forma contínua. A partir disso, apenas a página com novo conteúdo é carregada.

3.2.4. Exibição de Dados

O VueJs, assim como outras bibliotecas *frontend*, é capaz de definir uma estrutura de pequenos componentes reutilizáveis, os quais, juntos, formam uma página interativa. Nesse contexto, toda a interface foi concebida a fim de facilitar a reutilização dessas pequenas peças. Apesar de ser um paradigma diferente, a arquitetura da interface se baseia completamente na arquitetura da API, definida na seção 3.2.1, com três módulos principais que realizam uma tarefa exclusiva.

O diferencial da interface é a análise gráfica exposta ao usuário. Para tanto, o ecossistema de visualização Vega, capaz de realizar a construção de vários tipos de gráficos de forma declarativa, foi escolhida. Por ser um ecossistema grande, para sua utilização foram necessários dois componentes: *Vega-Altair* e *Vega-Embed*.

O primeiro é o item associado à geração da visualização a partir de uma entrada de dados, geralmente um *DataFrame* do Pandas. O ecossistema Vega utiliza uma linguagem própria denotada a partir de um documento JSON, o qual determina a aparência do gráfico. Por sua natureza, é interessante que esta peça esteja contida na API, mais próxima da ferramenta que gera os dados analisados. Ela está incluída no módulo *Analyse*, citado anteriormente.

O segundo é um instrumento que converte o documento JSON gerado pelo anterior para uma exibição verdadeira. Devido a essa característica, ele está presente como um componente da interface, a qual se comunica com a API, recebendo a definição do gráfico através de uma requisição HTTP.

4. Resultados e Discussão

O resultado deste trabalho trata-se de um MVP funcional que implementa as funções principais do programa, ou seja, a visualização da análise de conjuntos de dados e de modelos de AM.

4.1. Introdução da Ferramenta

Ao abrir a interface, a primeira tela é composta de uma visão geral do programa. No entanto, existe uma ordem certa para que a análise seja feita. É preciso, primeiro, obter um conjunto

de dados da internet – através do OpenML – ou fornecido localmente pelo usuário. Segundo, é necessário calcular os parâmetros da TRI para determinado conjunto de dados, utilizando os resultados obtidos no primeiro passo. A partir disso, é possível fazer a análise desses resultados. A seguir serão explicados os caminhos para esses três passos, os quais estão devidamente explícitos no seu componente.

Figura 4. Tela inicial da interface.

4.2. Obter um Conjunto de Dados

O primeiro passo para se utilizar a ferramenta é obter um conjunto de dados e treinar os modelos de AM com eles. Para tanto, o usuário deve, na tela da Figura 4, clicar no botão "Obter um conjunto de dados". Com isso, a tela da Figura 5 será renderizada.

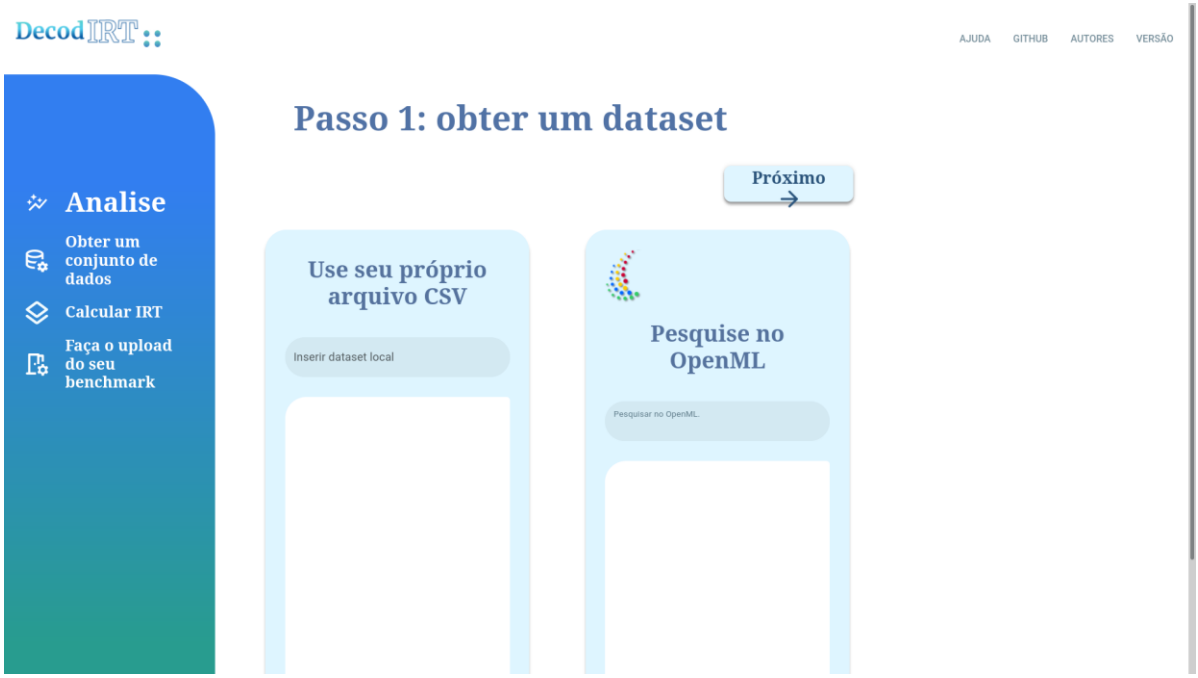

Figura 5. Tela para obter os dados.

Nesta tela, observa-se alguns pontos interessantes: o menu à esquerda, conhecida como sidebar, conduz o usuário a qualquer outra parte do programa, independente de onde estiver – evitando retorno desnecessário a páginas anteriores; um cabeçalho indicando onde ele está no programa; dois componentes com descrições diferentes no centro da página.

A parte principal desse passo é a escolha de um conjunto de dados, portanto, o usuário, através dos componentes do centro, pode escolher utilizar dados locais da máquina – utilizando o componente à esquerda – ou obtê-los no ambiente OpenML – a partir do componente à direita.

No componente à esquerda, o usuário deve clicar no botão “Inserir dataset local” e escolher um arquivo da sua máquina. No componente à direita, ele deve clicar no botão “Pesquisar no OpenML”. O botão se expandirá e mostrará o semelhante à Figura 6.

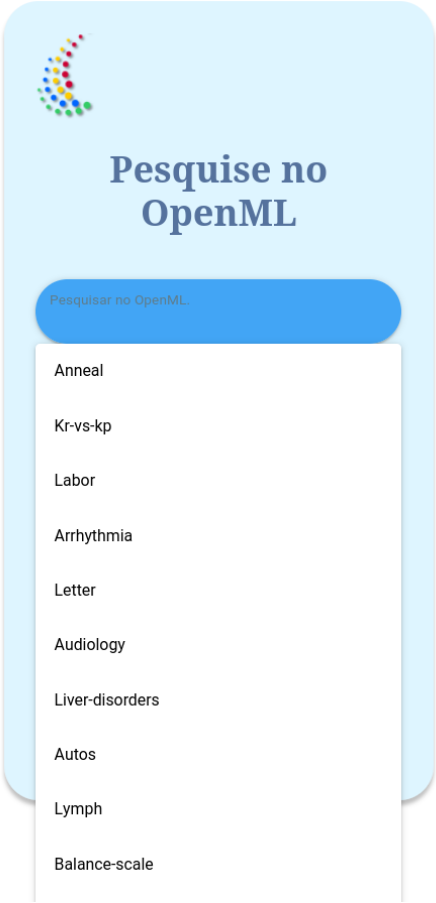

Figura 6. Menu para escolher um conjunto de dados do OpenML.

Esses são alguns conjuntos de dados disponíveis no OpenML. Um nome pode ser digitado no campo do botão, a fim de procurar por um em específico. Depois da escolha, clica-se no botão “+” para adicioná-los ao programa. A partir disso, os escolhidos aparecerão logo abaixo, como ilustrado na Figura 7.

Após a decisão, pressiona-se o botão “Próximo” na parte superior direita da tela (vide Figura 5). Uma tela nova aparecerá intitulada “Passo 2: revise sua escolha”, ela está disposta na Figura 8.

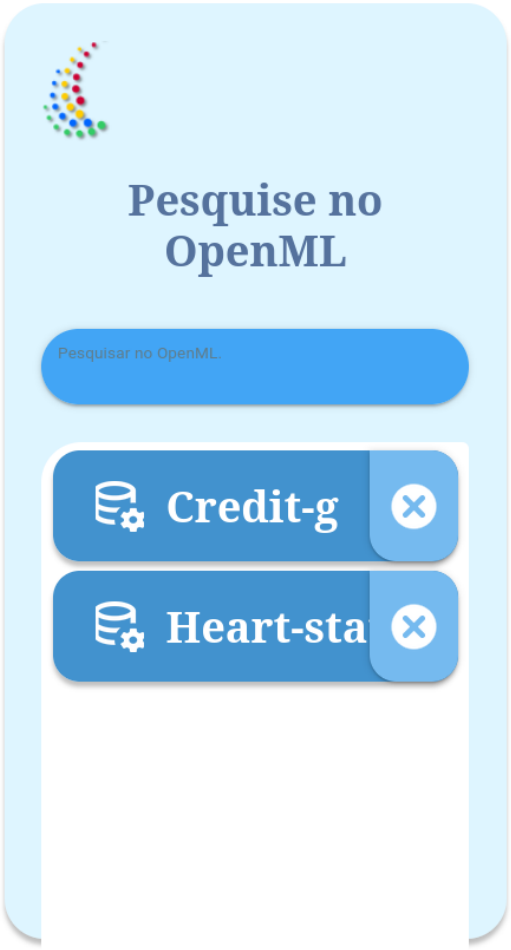

Figura 7. Conjunto de dados escolhidos.

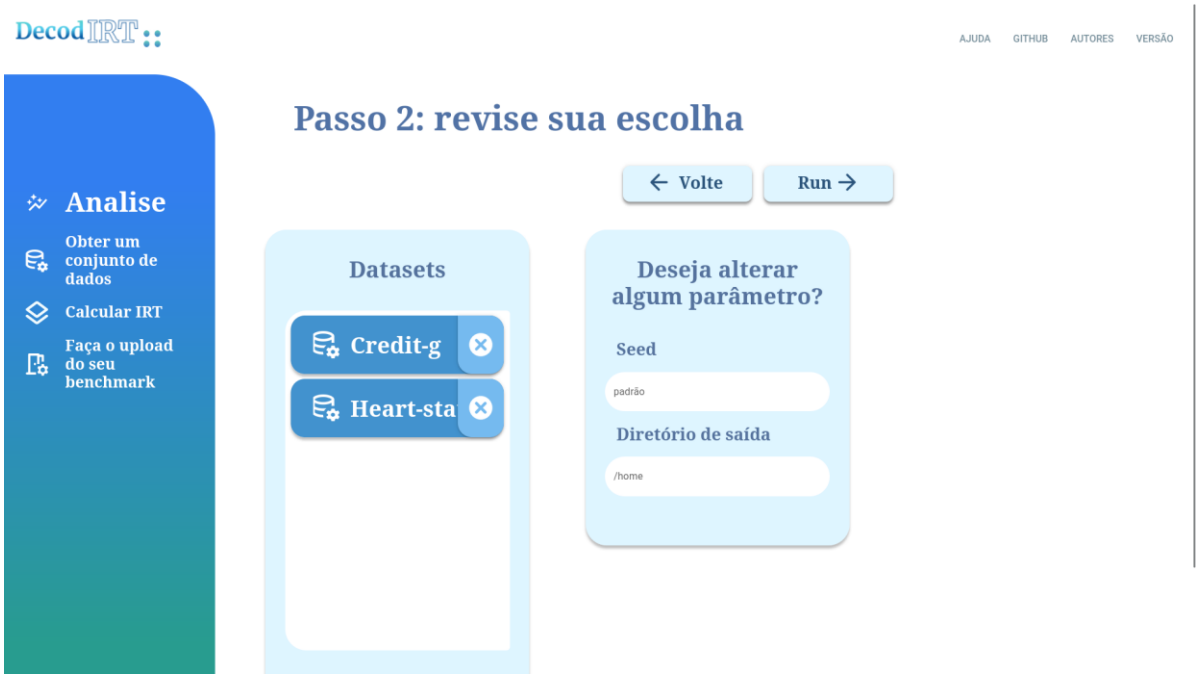

Figura 8. Tela com a revisão da escolha dos dados.

Neste passo, escolhe-se os dados nos quais os algoritmos serão treinados, além de permitir alterar parâmetros extras caso seja necessário. Após essa escolha, clica-se no botão “Run” e o decodIRT executará seu primeiro módulo. Uma tela de carregamento aparecerá, indicando a execução do programa, vide Figura 9.

Figura 9. Tela de carregamento que representa a execução do decodIRT.

4.3. Calculando Parâmetros da TRI

Depois da execução, o usuário será movido para o módulo “Calcular IRT”, o qual mostrará os arquivos que foram criados com a execução (Figura 10), além de uma opção de executar o segundo módulo do decodIRT nos dados, a partir do botão “Run”. Essa movimentação gera mais praticidade à ferramenta, já que – normalmente – o usuário executará esse módulo de qualquer maneira, a fim de analisar seus dados.

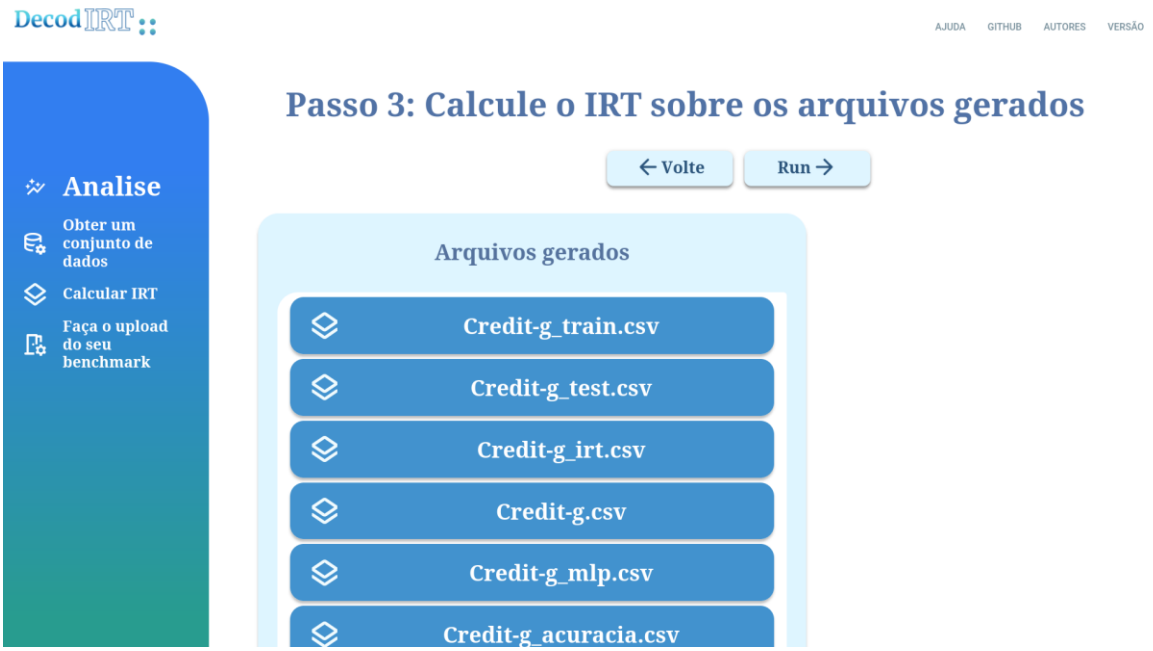

Figura 10. Tela com resumo dos dados gerados pelo decodIRT.

Neste caso, é importante notar que esta tela é o mesmo destino do botão “Calcular IRT” no painel à esquerda. Ao pressionar “Run”, uma tela semelhante à Figura 9 aparecerá, indicando a execução do segundo módulo do decodIRT, o qual gera os parâmetros gerais da TRI para cada algoritmo e instância do dado. Ao término da execução, a tela de análise já pode ser acessada.

4.4. Análise dos Resultados

Na tela de análise, disposta na Figura 11, encontram-se as informações e os gráficos utilizados pelos trabalhos de referência para análise dos modelos e *datasets*. Nessa tela, encontram-se três gráficos à esquerda da página. Esses são os Curva de Características de Classificadores [Martínez-Plumed et al. 2016], no qual a probabilidade de acerto dos algoritmos de classificação está em função de parâmetros da TRI dos itens, neste caso dificuldade, discriminação e adivinhação. Esses gráficos permitem analisar o comportamento de modelos de diferentes algoritmos de aprendizado sobre as instâncias dos *datasets* a medida que a dificuldade aumenta, por exemplo. Esse mesmo plot já foi utilizado no trabalho de [Cardoso et al. 2022] para explicar a partir de exemplos quais os conjuntos de instâncias que o modelo apresenta ser mais confiável para classificar corretamente. E quais são os tipos de instâncias menos confiáveis. No exemplo da Figura 11 utiliza-se o *dataset Heart-Statlog* de doenças cardíacas, um possível uso da ferramenta seria indicar para um médico quando uma IA não estaria apta para diagnosticar um paciente.

No outro lado, à direita, encontram-se duas informações importantes. Acima, a informação das porcentagens de instâncias difíceis, discriminatórias e de fáceis adivinhação estão presentes. Logo abaixo, a pontuação dos classificadores no conceito True-Score [Cardoso et al. 2020] é disposta, a fim de conceder uma visão geral dos algoritmos e seu desempenho. Conforme explicado em seções anteriores, a TRI visa avaliar o desempenho a nível de instância, porém a medida de True-Score pode ser utilizada para calcular uma nota final para os respondentes em questão. No futuro, pretende-se permitir que o usuário escolha qual modelo específico ele gostaria de analisar.

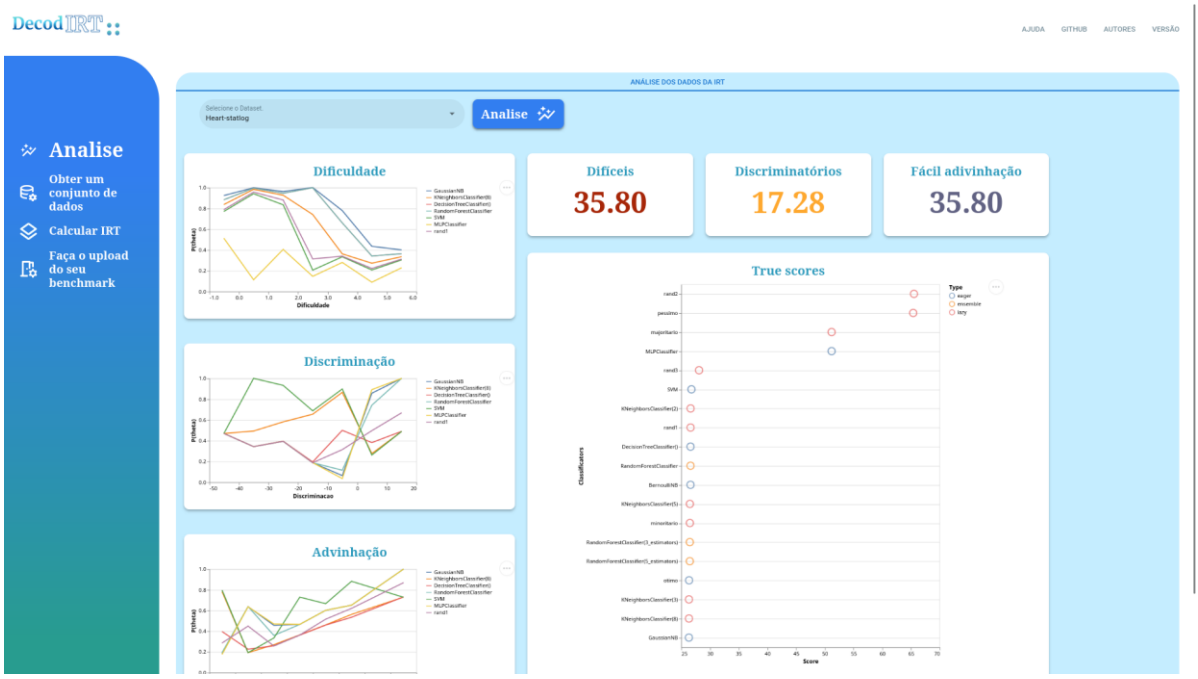

Figura 11. Tela de análise gráfica.

5. Considerações Finais e Trabalhos Futuros

Este trabalho apresenta uma ferramenta visual e interativa para análise de conjuntos de dados e modelos de AM a partir da TRI, baseada na ferramenta decodIRT que já é utilizada por outras pesquisas recentes. A ferramenta visa ajudar o pesquisador que procura entender melhor os seus dados e algoritmos, principalmente se este já pretende utilizar a TRI de antemão. Ademais, devido ao crescimento do Aprendizado de Máquina e da importância da qualidade dos dados para tal fim, o decodIRT e sua interface tem um futuro promissor na área, contanto que devidas manutenções sejam feitas. Além disso, a interface e API desenvolvidas apresentam documentação capaz de auxiliar o usuário no uso da ferramenta.

Referências Bibliográficas

Araujo Santos, V. C., Cardoso, L., & Alves, R. (2023). The quest for the reliability of machine learning models in binary classification on tabular data. Scientific Reports, 13(1), 18464.

Arrieta, A. B., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., ... & Herrera, F. (2020). Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. Information fusion, 58, 82-115.

Cardoso, L. F., de S. Ribeiro, J., Santos, V. C. A., Silva, R. L., Mota, M. P., Prudêncio, R. B., & Alves, R. C. (2022, November). Explanation-by-Example Based on Item Response Theory. In Brazilian Conference on Intelligent Systems (pp. 283-297). Cham: Springer International Publishing.

Cardoso, L. F., Santos, V. C., Francês, R. S. K., Prudêncio, R. B., & Alves, R. C. (2020, October). Decoding machine learning benchmarks. In Brazilian Conference on Intelligent Systems (pp. 412-425). Cham: Springer International Publishing.

de Andrade, D. F., Tavares, H. R., & da Cunha Valle, R. (2000). Teoria da Resposta ao Item: conceitos e aplicações. ABE, Sao Paulo.

Gohel, P., Singh, P., & Mohanty, M. (2021). Explainable AI: current status and future directions. arXiv preprint arXiv:2107.07045.

DW, G. D. A. (2019). DARPA's explainable artificial intelligence program. AI Mag, 40(2), 44.

Lord, F. M., & Wingersky, M. S. (1984). Comparison of IRT true-score and equipercentile observed-score" equatings". Applied psychological measurement, 8(4), 453-461.

Martínez-Plumed, F., Prudêncio, R. B., Martínez-Usó, A., & Hernández-Orallo, J. (2016). Making sense of item response theory in machine learning. In ECAI 2016 (pp. 1140-1148). IOS Press.

Martínez-Plumed, F., Prudêncio, R. B., Martínez-Usó, A., & Hernández-Orallo, J. (2019). Item response theory in AI: Analysing machine learning classifiers at the instance level. Artificial intelligence, 271, 18-42.