Workflow autonômico para aplicações de *Machine Learning* utilizando métricas de *Fairness*

Thales Eduardo Nazatto[1], Cecília Mary Fischer Rubira[2], Leonardo Montecchi[3]

[1] UNICAMP, Campinas, Brasil, tenazatto@gmail.com
[2] UNICAMP, Campinas, Brasil, cmrubira@ic.unicamp.br
[3] NTNU, Trondheim, Noruega, leonardo.montecchi@ntnu.no

14 de fevereiro de 2023

Resumo

O uso de Inteligência Artificial (IA) envolvendo grandes volumes de dados vem crescendo conforme nossa sociedade migra processos manuais de trabalho para soluções digitais e necessita de tomadas de decisão mais rápidas e assertivas, mas, devido a barreiras éticas e legais, métricas usadas inicialmente para definir a eficácia de um algoritmo se mostraram limitadas para medir vieses que refletem a sociedade de maneira que não era esperada pelos desenvolvedores da solução. Para resolver tal problema, novos algoritmos foram desenvolvidos e um novo conjunto de métricas, denominado como métricas de Fairness, é utilizado para determinar um equilíbrio entre grupos que sofrem discriminações. Com a introdução deste novo conjunto de algoritmos e métricas, novos problemas surgem, aumentando a complexidade da análise do Cientista de Dados para obter modelos de forma otimizada. Esta dissertação de Mestrado possui o objetivo de contribuir com o tema de Inteligência Artificial (IA) do ponto de vista da Engenharia de Software, apresentando um *Workflow* para aplicações de *Machine Learning* que pode ser executado de maneira autônoma sem a necessidade de experimentar uma grande quantidade de técnicas e pode ser mais assertivo em encontrar opções mais otimizadas para diferentes contextos. Para isso, é utilizado a arquitetura Pipe-and-Filter para realizar o *Workflow*, utilizando proveniência de dados para gravação de metadados, e a arquitetura MAPE-K para determinar essa autonomia. Foram realizados diversos estudos de caso para determinar se o MAPE-K pode ser viável na resolução destes problemas, e se o *Workflow* pode ser evoluído sem grande complexidade, e foi possível notar que as arquiteturas propostas conseguiram ser robustas e modulares, possibilitando estudos de Engenharia de Software em aplicações com o uso responsável de dados.

Keywords — *Workflow*, *Machine Learning*, Inteligência Artificial, Computação Autônoma, Métricas de *Fairness*

1 Introdução

Técnicas de Inteligência Artificial e Aprendizado de Máquina já são utilizadas há bastante tempo no ramo da Computação. Ramos como robótica e jogos são grandes exemplos, dada a necessidade nos mesmos de automatizar comportamentos que seriam tidos como triviais para um ser humano. Entretanto, nos últimos anos ocorreu um crescimento no uso dessas tecnologias em aplicações tradicionais, devido principalmente à grande quantidade de dados processada diariamente pelas empresas e pela quantidade de processamento disponível a custos baixos. Diferentes perfis podem traçados com esses dados e usar soluções de IA gera tomadas de decisão mais assertivas com o objetivo de melhorar a experiência de usuário e cor-

rigir problemas. Porém, muitas dessas soluções foram projetadas sem pensar em governança de dados como requisito de projeto, e se mostram ineficientes quando ela é tratada em consideração. E nesse ponto muitas aplicações de IA falham: muitas implementações foram implementadas como *black boxes*, onde o determinante para estabelecer a confiança no modelo implementado é sua entrada e sua saída.

Um outro efeito colateral dessa estratégia é a exposição de vieses que, embora sejam vistos como não-intencionais pelos desenvolvedores por ter a possibilidade de ser um *outlier* no modelo treinado, refletem preconceitos escancarados da sociedade atual. Uma entrada de dados enviesada resulta em um algoritmo que realiza discriminações em sua classificação [10], e uma vez que as métricas utilizadas para medir a qualidade de um modelo *black box* são geralmente baseadas em acurácia, precisão e recall, discriminações não são facilmente percebidas por tais métricas. Ao mesmo tempo, um modelo *black box* pode ter uma alta dependência de poucos dados, determinando problemas de acoplamento. Para resolver este problema, é possível que a criação de um novo modelo seja uma melhor opção que realizar a correção em apenas parte dele.

Devido a esses tipos de problemas, o termo *Explainable AI* (XAI) ganha força para envolver o desenvolvimento de uma IA que seja acurada e simultaneamente transparente. Como IA possui diversos tipos de métodos diferentes para enquadrar diversos tipos de dados, o mesmo acaba se aplicando em Explainable AI, podendo enquadrar em diversos tipos de dados [20], ou dados específicos como imagens [14] e tabelas [16]. Como o objetivo em XAI é fazer com que os resultados alcançados pela solução de IA sejam compreendidos por humanos, é possível considerar este fato como requisito no design de uma solução de IA, fazendo com que a mesma seja reusável e testável.

No mesmo tema, é possível estabelecer métricas para determinar o quão o modelo está preparado para dados sensíveis [8], termo que é conhecido como *Fairness*. Com a evolução das pesquisas na comunidade acadêmica, foram descobertos algoritmos para redução dos vieses presentes nos conjuntos de dados, como *Reweighing* [12], *Adversarial Debiasing* [22] e *Reject Option Classification* [13]. Por consequência, ocorre melhora nas métricas em questão, mas pode desfavorecer métricas que já são amplamente utilizadas como garantia de um bom modelo desenvolvido com técnicas de Aprendizado de Máquina.

O objetivo desta dissertação de mestrado é desenvolver uma estrutura de *Workflow* para aplicações de *Machine Learning* que seja completamente autônoma, por três fatores principais:

- Facilitar a criação de modelos justos e confiáveis com a automatização da escolha dos algoritmos, cuja complexidade aumenta com a escolha dos algoritmos a serem utilizados e suas execuções nas etapas corretas do processo, onde eles foram escolhidos para atuar.
- Estabelecer um balanceamento entre métricas para avaliar bons modelos com métricas para avaliar modelos justos.
- Considerar proveniência de dados como requisito no design de uma solução de IA, e como uma alternativa a XAI através da utilização de metadados.

Para isso, foi desenvolvido um sistema, que pode ser dividido em 4 etapas principais: Engenharia de dados, *Workflow* de IA, Autonomia do *Workflow* e Interface Humano-Computador. Para o desenvolvimento do *workflow*, será utilizada a arquitetura *Pipe-and-Filter*. Para a autonomia deste, será criado um componente utilizando a arquitetura MAPE-K [6] para analisar uma base de conhecimento e prover o melhor pipeline seguindo regras pré-determinadas. Para a interface, ela foi criada nos moldes de uma aplicação web. O código deste desenvolvimento foi disponibilizado no GitHub[1] para avaliação e testes em estudos posteriores.

O restante deste artigo se organizará da seguinte forma: o Capítulo 2 descreve os conceitos que irão ser abordados neste projeto; o Capítulo 3 mostra a metodologia e detalhamento do processo de desenvolvimento; o Capítulo 4 discute os resultados obtidos e, finalmente, o Capítulo 5 estabelece as conclusões,

[1]Repositório Git contendo os códigos deste projeto: `https://github.com/tenazatto/MsC`

considerações finais e sugestões de trabalhos futuros e evoluções.

2 Conceitos Principais

2.1 Computação Autônoma

Em 2001, Paul Horn introduziu o conceito de Computação Autônoma como alternativa a solução para a crescente complexidade dos sistemas da época, onde previa-se que os mesmos se tornariam muito grandes e complexos até mesmo para os profissionais mais qualificados configurarem e realizarem manutenção. Tal conceito qualifica sistemas de computação que podem se autogerenciar com relação aos objetivos de alto nível dados pelos administradores e é derivado da biologia, dado a grande variedade e hierarquia de sistemas autônomos presentes na natureza e na sociedade [15].

Para a Computação Autônoma acontecer, é implementado um Elemento Autônomo [7], um componente de software que gerencia partes do sistema baseando-se em um *loop* MAPE-K (*Monitor, Analyze, Plan, Execute, and Knowledge*), ilustrado na Figura 1. O MAPE-K é um conceito que constitui um *loop* de controle, usado para monitorar e controlar um ou mais elementos gerenciados. Um elemento gerenciado (*Managed Element*) pode ser um hardware, como uma impressora, um software, como um banco de dados, outro Elemento Autônomo ou funcões específicas, como balanceamento de carga. Um *loop* de controle MAPE-K é dividido da seguinte forma:

- **Monitoramento (*Monitor*):** Responsável por monitorar os recursos gerenciados e coletar, agregar e filtrar dados. O monitoramento é feito por meio de um sensor (*Sensor*) ou mais sensores.
- **Análise (*Analyze*):** Analisa os dados relatados pela parte do monitor. A análise visa compreender qual é o estado atual do sistema e se há medidas para serem tomadas.
- **Planejamento (*Plan*):** Um plano de ação é preparado no base dos resultados da análise. O plano é uma série de medidas que irão mover o sistema de seu estado atual para um estado desejado.
- **Execução (*Execute*):** O plano é executado e controlado. Um efetor (*Effector*) ou mais executam as ações planejadas no recurso.
- **Conhecimento (*Knowledge*):** A base de conhecimento é central e acessível por todas as partes do *loop*. Separado a partir de dados coletados e analisados, ele contém conhecimento adicional, como modelos de arquitetura, modelos de metas, políticas e planos de mudança.

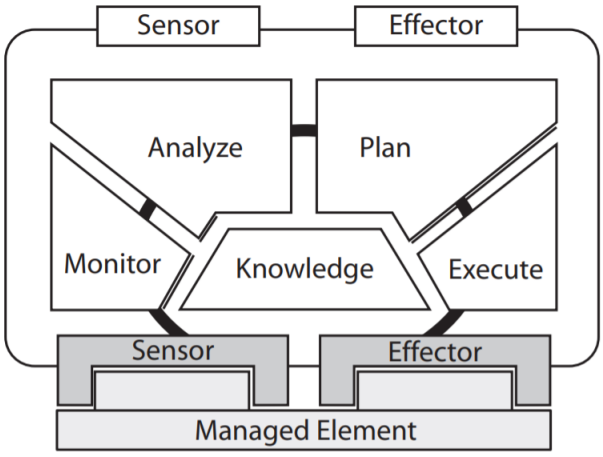

Figura 1: Diagrama de funcionamento da arquitetura MAPE-K [7].

2.2 *Machine Learning*

Aprendizado de Máquina (*Machine Learning*, em inglês) pode ser definido como "a prática de usar algoritmos para coletar dados, aprender com eles, e então fazer uma determinação ou predição sobre alguma coisa no mundo. Em vez de implementar as rotinas de software manualmente, com um gama específica de instruções para completar uma tarefa em particular, a máquina é 'treinada' usando uma quantidade grande de dados e algoritmos que dão e ela a

habilidade de aprender como executar a tarefa" [1], podendo ser dividido em processos de coleta, limpeza e refinamento de dados, treinamento e avaliação. As tarefas de aprendizado podem ser classificadas em três categorias básicas [3] [2]:

- **Aprendizado supervisionado**: O treinamento é realizado por meio de exemplos rotulados, como uma entrada na qual a saída desejada é conhecida. Através de métodos como classificação, regressão e *gradient boosting*, o aprendizado supervisionado utiliza padrões para prever os valores de rótulos em dados não-rotulados adicionais. O aprendizado supervisionado é comumente empregado em aplicações nas quais dados históricos preveem eventos futuros prováveis.

- **Aprendizado não-supervisionado**: É utilizado em dados que não possuem rótulos históricos. A "resposta certa" não é informada ao sistema, o algoritmo deve descobrir o que está sendo mostrado. O objetivo é explorar os dados e encontrar alguma estrutura dentro deles. Técnicas populares incluem mapas auto-organizáveis, mapeamento por proximidade, agrupamento *k-means* e decomposição em valores singulares. Esses algoritmos também são utilizados para segmentar tópicos de texto, recomendar itens e identificar pontos discrepantes nos dados.

- **Aprendizado por reforço**: O algoritmo descobre através de testes do tipo "tentativa e erro" quais ações rendem as maiores recompensas. Este tipo de aprendizado possui três componentes principais: o agente (o aprendiz ou tomador de decisão), o ambiente (tudo com que o agente interage) e ações (o que o agente pode fazer). O objetivo é que o agente escolha ações que maximizem a recompensa esperada em um período de tempo determinado. O agente atingirá o objetivo muito mais rápido se seguir uma boa política, então o foco do aprendizado por reforço é descobrir a melhor política.

2.3 *Fairness* em *Machine Learning*

Com o aumento do uso de *Machine Learning* para tomadas de decisão, o uso de dados sensíveis em um contexto determinado também aumentou, e temas como uma IA ética e conceitos como vieses nos dados e *Fairness* passaram a serem discutidas não apenas na Computação, mas em áreas como Direito. Algoritmos são mais objetivos, rápidos e são capazes de considerar uma grande magnitude de recursos que pessoas não são capazes. Entretanto, até o presente momento eles não são capazes de diferenciar contextos sociais, onde um resultado mais eficiente de acordo com os dados disponíveis podem amplificar as desigualdades sociais e tomar decisões de modo injusto [17]. Estes dados sensíveis, tendo como exemplos cor de pele, raça, sexo, idade e altura, são considerados atributos protegidos, que precisam ser classificados e processados antes da execução de um algoritmo de *Machine Learning*, determinarão como o algoritmo se comportará e, consequentemente, afetará suas métricas [18]. Os grupos de dados provenientes destes atributos protegidos são considerados grupos protegidos, que podem ser divididos em dois grupos: o grupo privilegiado, que possui vantagens no contexto do problema, e o grupo não-privilegiado, que possui desvantagens no contexto do problema e, portanto, sujeito a discriminação.

É possível descrever o conceito de *Fairness* no contexto de aprendizagem supervisionada, onde um modelo f pode prever um conjunto de resultados y a partir de um conjunto de *features* x, evitando discriminação injusta em relação a um atributo protegido a. É permitido, mas não exigido, que a seja um componente de x [8]. Em outras palavras, um modelo de ML considerado justo é aquele onde a correlação de seu resultado é baixa em relação a dados de entrada considerados como sensíveis a discriminações. As métricas de *Fairness* diferem das métricas utilizadas para avaliação do modelo, que possuem o propósito de verificar se um modelo tem previsões confiáveis ou não. Enquanto as métricas mais tradicionais avaliam a performance do modelo e seus dados como um todo e seus resultados gerais, as métricas de *Fairness* avaliam se os resultados gerais também se refletem em grupos específicos, para verificar se não há dis-

paridade ou discriminação nos resultados propostos. Exemplos de métricas utilizadas para isso são a Taxa de Verdadeiros Positivos e a Taxa de Falsos Positivos. Enquanto a **Taxa de Verdadeiros Positivos** (TVP, ou TPR pelo termo em inglês **True Positive Rate**) é a fração de casos positivos reais (TP) de todos os casos negativos previstos incorretamente como estando na classe positiva (FN), a **Taxa de Falsos Positivos** (TFP, ou FPR pelo termo em inglês **False Positive Rate**) é a fração de casos negativos previstos incorretamente como estando na classe positiva (FP) de todos os casos positivos reais (TN):

$$TPR = \frac{TP}{TP + FN} FPR = \frac{FP}{FP + TN} \tag{1}$$

Dada essas métricas iniciais, considerando $Y = 1$ a classe positiva, $Z = 0$ o grupo não-privilegiado e $Z = 1$ o grupo privilegiado, algumas das definições de *Fairness* mais usadas são as seguintes:

- **Diferença de paridade estatística (*Statistical parity difference*), ou discriminação [21]:** Esta métrica é baseada na seguinte fórmula:

  $$Pr(Y = 1|Z = 0) - Pr(Y = 1|Z = 1) \tag{2}$$

  Aqui, o viés ou paridade estatística é a diferença entre a probabilidade de que um indivíduo aleatório retirado dos não-privilegiados seja rotulado como 1 e a probabilidade de que um indivíduo aleatório dos privilegiados seja rotulado como 1. Portanto, um valor próximo de 0 é considerado justo.

- **Diferença de oportunidade igual (*Equal opportunity difference*) [9]:** É a diferença entre a taxa positiva verdadeira do grupo não privilegiado e a taxa positiva verdadeira do grupo privilegiado:

  $$TPR_{Z=0} - TPR_{Z=1} \tag{3}$$

  Um valor próximo de 0 é considerado justo. Um classificador binário satisfaz a igualdade de oportunidades quando a taxa positiva verdadeira de ambos os grupos são iguais [11]

- **Diferença de probabilidade média (*Average odds difference*) [9]:** Essa métrica usa a taxa de falsos positivos e a taxa positiva verdadeira para calcular a tendência, calculando a igualdade de probabilidades com a fórmula:

  $$\frac{1}{2}(|FPR_{Z=0} - FPR_{Z=1}| + |TPR_{Z=0} - TPR_{Z=1}|) \tag{4}$$

  Precisa ser próximo a 0 para ser considerado justo.

- **Impacto de disparidade (*Disparate impact*) [9]:** Para esta métrica, é usada a seguinte fórmula:

  $$\frac{Pr(Y = 1|Z = 0)}{Pr(Y = 1|Z = 1)}$$

  Usa as mesmas probabilidades da diferença de paridade estatística, mas aqui são calculadas como proporção. Desta forma, um valor próximo de 1 é considerado justo.

- **Índice de Theil (*Theil index*) [19]:** Esta medida também é conhecida como índice de entropia generalizado, mas com α igual a 1 [19]. É calculado com a seguinte fórmula:

  $$\frac{1}{n} \sum_{i=0}^{n} \frac{b_i}{\mu} \ln \frac{b_i}{\mu}$$

  Onde $b_i = \hat{y}_i - y_i + 1$, y_i é o conjunto de saídas e $\hat{y}_i$ é o conjunto de previsões dadas pelo modelo. Também precisa ser próximo a 0 para ser considerado justo.

3 Solução Proposta

3.1 Arquitetura

Foi desenvolvido um sistema, cujo desenvolvimento pode ser dividido em 4 etapas principais:

- **Engenharia de dados:** Etapa criada com o objetivo de simular processos de transformação e limpeza de dados.

- **Workflow de IA:** Etapa para execução de um Pipeline que simula o desenvolvimento de uma aplicação automatizada de IA, com etapas de preparação dos dados (Pré-processamento), treinamento (Processamento) e avaliação dos resultados (Pós-processamento), salvando tais dados para geração de uma base de conhecimento.

- **Autonomia do Workflow:** Etapa que executa um Elemento Autônomo que controla o Workflow como elemento gerenciado para automatizar todas as suas etapas, com o objetivo de evitar com que perca-se tempo em execuções manuais que podem demorar dependendo do algoritmo e do conjunto de dados utilizado.

- **Interface:** Etapa criada com o objetivo de simular a etapa anterior, porém de modo a proporcionar uma experiência de usuário mais simples e intuitiva, onde sua integração com as outras etapas é mostrada na Figura 2. É dividida em duas partes:

  - **Frontend:** Parte visual, exibida em um navegador.

  - **Backend:** Parte onde o Frontend se comunica para obter os dados e montar o visual corretamente, de forma que corresponda a configurações utilizadas pelo Elemento Autônomo.

Para o desenvolvimento do workflow, foi utilizada a arquitetura Pipe-and-Filter. Para a autonomia deste, foi criado um componente utilizando a arquitetura MAPE-K para analisar uma base de conhecimento e prover o melhor pipeline seguindo regras pré-determinadas. Para a interface, ela foi criada nos moldes de uma aplicação web.

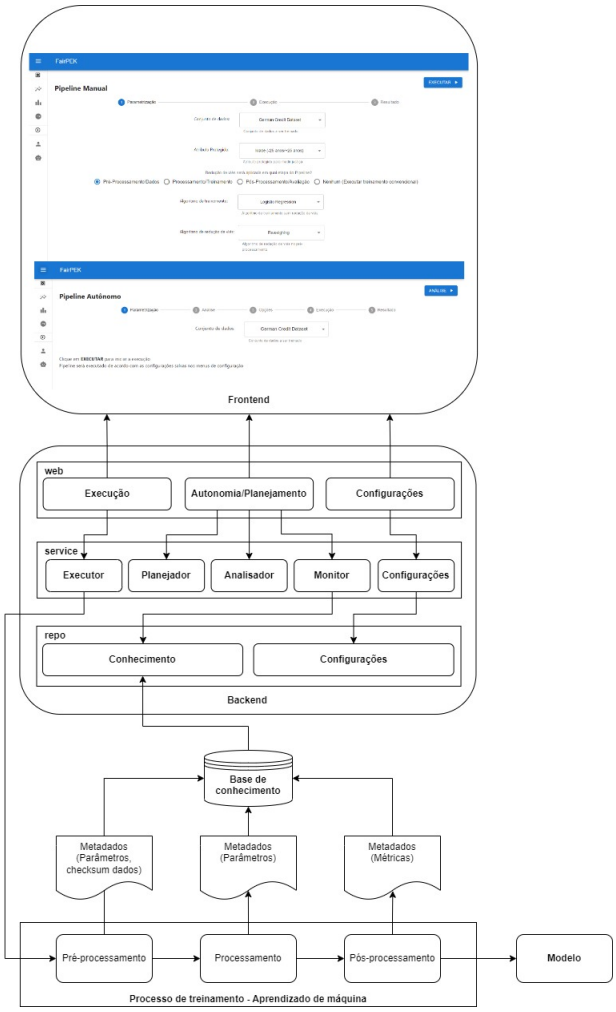

Figura 2: Comunicação entre Interface, autonomia e *Workflow* de IA

3.2 Autonomia

Para determinar a autonomia deste workflow, é realizado um cálculo de pesos baseado em uma seleção livre das métricas. O motivo de existir esse cálculo é mensurar o contexto do problema de acordo com uma análise prévia do Cientista de Dados e do Especialista de Domínio, e consolidar todas as métricas para simplificar as estratégias de planejamento. As métricas são divididas em dois grupos (Métricas de perfor-

mance e Metricas de Fairness), e dentro desse grupo pode-se colocar quantas métricas forem necessárias, desde que seja respeitado o contexto de cada grupo. A cada grupo é atribuído pesos diferentes, e a cada métrica desse grupo também é atribuído pesos diferentes.

Primeiro, normaliza-se as métricas m_{F_i} para m'_{F_i}, referentes às métricas de Fairness, para todas ficarem em um intervalo de 0 a 1, conforme exibido na equação 5. Dessa forma, seus resultados ficam uniformes e é possível aplicar os pesos sem haver distorções no cálculo. No caso das métricas de Performance, todas possuem a mesma escala, por isso as métricas m_{P_i} não são normalizadas. Depois, multiplica-se cada uma por seus pesos correspondentes w_{P_i} e w_{F_i}, e realiza-se uma média ponderada dentro do grupo para atribuir uma pontuação S_P para o grupo das Métricas de Performance e S_F para o grupo das Metricas de Fairness, conforme exibido na equação 6. Para facilitar a visualização das pontuações, multiplica-se as pontuações por um fator $X = 1000$ para o intervalo da pontuação ser de 0 a 1000 e arredonda-se o número. Após tais pontuações serem obtidas, a pontuação geral S é calculada multiplicando-as por seus pesos correspondentes w_P e w_F e realizando a média ponderada, conforme exibido na equação 7.

$$m'_{F_i} = \begin{cases} 1 - |m_{F_i}| & \text{caso } m_{F_i} \text{ envolva diferença e } -1 < m_{F_i} < 1 \\ 0 & \text{caso } m_{F_i} \text{ envolva diferença, e } m_{F_i} >= 1 \text{ ou } m_{F_i} <= -1 \\ 1 - |\frac{1}{m_{F_i}} - 1| & \text{caso } m_{F_i} \text{ envolva razão e } m_{F_i} > 1 \\ 1 - |m_{F_i} - 1| & \text{caso } m_{F_i} \text{ envolva razão e } m_{F_i} <= 1 \\ m_{F_i} & \text{caso contrário} \end{cases} \tag{5}$$

$$\begin{aligned} S_F &= \lfloor X \times \frac{\sum_{i=1}^{n_F} w_{m'_{F_i}} \times m'_{F_i}}{\sum_{i=1}^{n} w_{m'_{F_i}}} \rceil \\ S_P &= \lfloor X \times \frac{\sum_{i=1}^{n_P} w_{m_{P_i}} \times m_{P_i}}{\sum_{i=1}^{n} w_{m_{P_i}}} \rceil \end{aligned} \tag{6}$$

$$S = \frac{w_F \times S_F + w_P \times S_P}{w_F + w_P} \tag{7}$$

Para este cálculo ser realizado, é necessário como pré-requisito execuções anteriores realizadas no workflow, para que o mesmo grave as métricas necessárias e determine quais as melhores combinações através da pontuação.

3.3 Determinação do Planejamento

Para determinação do que vai ser executado no Workflow, algumas estratégias foram implantadas:

- **Filtragem de algoritmos**: Alguns algoritmos podem estar mal implementados ou seus modelos podem estar com métricas que necessitam uma melhor análise do Cientista de Dados para serem consideradas confiáveis. Para isso não acontecer, é possível realizar um filtro de acordo com as combinações de algoritmos consideradas confiáveis antes de selecionar os modelos ideais.

- **Filtragem de resultados da análise**: Pelo mesmo motivo da estratégia anterior, métricas não confiáveis significam uma distorção na pontuação final. Para esse caso, foi criado um limiar de pontuação mínimo e máximo para determinar pontuações que podem ser consideradas confiáveis para avaliação.

Após a realização destas estratégias, é possível escolher as 5 melhores pontuações presentes na interface.

4 Avaliação da Solução

Foram realizados 3 Estudos de Caso para determinar a viabilidade de escolha das arquiteturas, manutenções futuras e funcionamento da autonomia na escolha de melhores opções em diferentes contextos. Em todos eles, o objetivo foi a obtenção de uma classificação de crédito (boa ou ruim), através de uma série de *features* e utilizando diferentes conjuntos de dados. No caso de alterações no sistema, também foram contadas as linhas de código realizadas em cada alteração, para definir se tais alterações são simples de serem feitas.

A execução para determinar as melhores opções foi realizada com 3 pesagens diferentes na pontuação geral:

- 50% para métricas de Performance e 50% para métricas de Fairness, para uma configuração equilibrada.

- 75% para métricas de Performance e 25% para métricas de Fairness, para uma configuração que prioriza a performance em detrimento da justiça.

- 25% para métricas de Performance e 75% para métricas de Fairness, para uma configuração que prioriza a justiça em detrimento da performance.

Todas as execuções foram realizadas utilizando uma base de conhecimento cerca de 600 execuções prévias com diversas opções de conjuntos de dados, atributos protegidos e algoritmos, e são utilizadas as métricas Acurácia, Precisão, *Recall*, *F1-Score* e AUC como métricas de Performance e as métricas *Statistical Parity Difference*, *Equal Opportunity Difference*, *Average Odds Difference*, *Disparate Impact* e *Theil Index* como métricas de *Fairness*, todas com pesagens iguais em seu respectivo agrupamento. Como não foram encontrados estudos onde são utilizados redução de viés em mais de uma parte do processo de Machine Learning (pré-processamento, processamento ou pós-processamento), é realizada a execução de apenas um algoritmo de redução de viés por *workflow*.

4.1 Estudo de Caso 1: Viabilidade e utilidade do MAPE-K

Neste Estudo de Caso, o maior foco foi colocado em testar e verificar como o MAPE-K se comporta em um cenário com diversas execuções prévias do Workflow, utilizando o German Credit Dataset [5] como conjunto de dados, Idade e Nacionalidade como opções de atributos protegidos e um intervalo de pontuação entre 500 e 950. Os resultados estão presentes abaixo nas Tabelas 1, 2 e 3:

Tabela 1: Melhores opções escolhidas pelo modelo MAPE-K
Todos os métodos - 50% Performance/50% Fairness

Workflow				Pontuação		
Atributo protegido	Pré-processamento	Treinamento	Pós-processamento	Performance	Fairness	**Geral**
Idade	Nenhum	Regressão Logística	Equalized Odds	968	860	**914**
Nacionalidade	Nenhum	Random Forest	Calibrated Equalized Odds	902	922	**912**
Nacionalidade	Nenhum	Gradient Boosting	Calibrated Equalized Odds	870	925	**898**
Idade	Nenhum	Gradient Boosting	Equalized Odds	927	862	**894**
Idade	Reweighing	Gradient Boosting	Nenhum	804	931	**868**

Tabela 2: Melhores opções escolhidas pelo modelo MAPE-K
Todos os métodos - 75% Performance/25% Fairness

Workflow				Pontuação		
Atributo protegido	Pré-processamento	Treinamento	Pós-processamento	Performance	Fairness	**Geral**
Idade	Nenhum	Regressão Logística	Equalized Odds	968	860	**941**
Idade	Nenhum	Gradient Boosting	Equalized Odds	927	862	**910**
Nacionalidade	Nenhum	Random Forest	Calibrated Equalized Odds	902	922	**907**
Nacionalidade	Nenhum	Gradient Boosting	Calibrated Equalized Odds	870	925	**883**
Idade	Nenhum	Random Forest	Equalized Odds	898	799	**874**

Tabela 3: Melhores opções escolhidas pelo modelo MAPE-K
Todos os métodos - 25% Performance/75% Fairness

Workflow				Pontuação		
Atributo protegido	Pré-processamento	Treinamento	Pós-processamento	Performance	Fairness	**Geral**
Idade	Disparate Impact Remover	Support Vector Machines	Nenhum	747	989	**928**
Nacionalidade	Disparate Impact Remover	Support Vector Machines	Nenhum	747	989	**928**
Idade	Nenhum	Adversarial Debiasing	Nenhum	742	979	**920**
Nacionalidade	Reweighing	Support Vector Machines	Nenhum	755	972	**918**
Nacionalidade	Learning Fair Representations	Support Vector Machines	Nenhum	755	972	**918**

Nessas execuções, surpreende 2 observações. A primeira é o fato da predominância de algoritmos com redução de viés no pós-processamento/resultado especialmente em configurações que priorizavam performance, contrariando o esperado de que os algoritmos com redução de viés aumentavam justiça em detrimento da performance. A segunda é a predominância de algoritmos com redução de viés no pré-processamento/dado em configurações que priorizavam justiça, principalmente pois todas as execuções usavam *Support Vector Machines* como algoritmo de treinamento.

Diante destas 2 predominâncias envolvendo todos os *workflows* executados, novos experimentos com restrições adicionais foram realizados para obter observações mais detalhadas a respeito dos resultados:

- Uso apenas de *workflows* com o uso de algoritmos com redução de viés no pré - processamento/dado.
- Uso apenas de *workflows* com o uso de algoritmos com redução de viés no processamento/treinamento.
- Uso apenas de *workflows* com o uso de algoritmos com redução de viés no pós - processamento/resultado.
- Uso apenas de *workflows* sem o uso de algoritmos com redução de viés.

Nos *workflows* utilizando apenas algoritmos com redução de viés no dado, percebe-se a predominância dos algoritmos de treinamento *Gradient Boosting* e *Support Vector Machines*, sendo o *Gradient Boosting* predominante em configurações priorizando performance e o *Support Vector Machines* predominante em configurações priorizando justiça, o que começa a explicar a sua predominância também presente no resultado geral. Também é possível perceber mais 2 observações: A primeira observação é que a análise de apenas uma categoria de algoritmos dá mais clareza em ver como o cálculo utilizado nas 3 configurações faz com que o equilíbrio de ambas as métricas se torna mais importante do que a prioridade apenas em performance ou apenas em justiça, uma vez que há exemplos de conjuntos de algoritmos com pontuações ligeiramente maiores em performance que acabaram sendo pior avaliados pois a pontuação em *Fairness* está bem menor, e vice-versa. A segunda observação é que o uso de um atributo protegido diferente (e, por consequência, com tratamento de dados diferente) e de um algoritmo de treinamento parecem impactar tanto quanto ou até mais que o próprio algoritmo com redução de viés no dado.

Nos *workflows* utilizando apenas algoritmos com redução de viés no treinamento, percebe-se uma variedade maior nos algoritmos, até porque não há usos de redução de viés em um pré ou um pós-processamento, com destaque para o *Adversarial Debiasing* que foi bem avaliado pela pontuação alta em *Fairness*. Nestes *workflows*, as 2 observações percebidas nos *workflows* utilizando apenas algoritmos com redução de viés no dado são reforçadas por uma maior variedade de pontuações e pelo algoritmo *Exponentiated Gradient Reduction* com 2 exemplos diferentes na Tabela **??**, onde o uso da Nacionalidade como atributo protegido possui pontuações de performance e *Fairness* piores que a Idade e conclundo que o processamento utilizado no atributo protegido pode afetar todas as métricas.

Nos *workflows* utilizando apenas algoritmos com redução de viés no resultado, surpreende o fato de que os *workflows* obtiveram as melhores pontuações em Performance e as piores pontuações em *Fairness*, podendo indicar uma característica dos algoritmos *Equalized Odds* e *Calibrated Equalized Odds*. Entretanto, por conta da grande melhora por parte das métricas de performance, estes *workflows* possuem um maior equilíbrio entre Performance e *Fairness* e acabam garantindo maiores pontuações na média, justificando as melhores pontuações nas primeiras execuções onde foram considerados todos os métodos. Fora isto, as demais observações anteriores também se aplicam nestas execuções.

Nos *workflows* sem algoritmos com redução de viés, é curioso notar que, ao comparar com *workflows* equivalentes mas com algoritmos usando redução de viés no dado, é possível notar que a hipótese principal se confirma em sua grande maioria: As pontuações em Performance são ligeiramente maiores e as pontuações em *Fairness* são ligeiramente menores. É possível notar uma exceção no *workflow* envolvendo o algoritmo *Random Forest*, mas nos outros casos a hipótese é verificada com sucesso. Ao comparar com *workflows* usando algoritmos com redução de viés no treinamento tal hipótese também se confirma, entretanto o uso de *Support Vector Machines* parece ser uma exceção a regra, implicando que o uso do algoritmo possibilita modelos mais justos para o conjunto de dados utilizado. Ao comparar com *workflows* usando algoritmos com redução de viés no resultado, a hipótese não se confirma devido a observação da grande melhora por parte das métricas de performance nos *workflows* usando algoritmos com redução de viés no resultado, mas ao verificar a pontuação em *Fairness* é possível notar uma ligeira melhora. Há a exceção de *workflows* envolvendo *Support Vector Machines* que são melhores nos *workflows* sem algorit-

mos com redução de viés, mas no uso de outros algoritmos ocorre melhora na pontuação em *Fairness*.

Em todas as execuções, foi possível perceber que a escolha do *workflow* com o melhor equilíbrio entre estes dois grupos de métricas com contextos completamente diferentes ainda se torna difuso diante da grande quantidade de métricas e conjuntos de algoritmos utilizados. Além disso, a diferença entre as métricas é extremamente pequena e dificulta ainda mais a escolha. Nesse contexto, a consolidação das métricas em grupos simplifica a visualização de quais *workflows* são mais equilibrados, e o uso de pesos para cada métrica e para cada grupo pode calibrar qual o melhor equilíbrio desejado para determinada situação. Deste modo, pode-se concluir também que, em um contexto de desenvolvimento, o processo simplifica a decisão do Cientista de Dados e reduz significantemente o tempo para obtenção e implantação de um modelo otimizado, pois não exigirá execuções em diversos algoritmos uma vez que já há uma base de conhecimento prévia. Além disso, poderá poupar processamento e custos para a resolução de diversos outros problemas, uma vez que as execuções economizadas pelas equipes que utilizariam esse processo abrem margem para que outras equipes utilizem esse processamento.

Tabela 4: Melhores opções escolhidas pelo modelo MAPE-K
Todos os métodos - 50% Performance/50% Fairness

Workflow				Pontuação		
Atributo protegido	Pré-processamento	Treinamento	Pós-processamento	Performance	Fairness	**Geral**
Renda	Learning Fair Representations	Random Forest	Nenhum	991	968	**979**
Renda	Nenhum	Gradient Boosting	Equalized Odds	988	969	**978**
Renda	Reweighing	Random Forest	Nenhum	991	963	**977**
Renda	Learning Fair Representations	Regressão Logística	Nenhum	981	973	**977**
Renda	Reweighing	Gradient Boosting	Nenhum	987	964	**976**

Tabela 5: Melhores opções escolhidas pelo modelo MAPE-K
Todos os métodos - 75% Performance/25% Fairness

Workflow				Pontuação		
Atributo protegido	Pré-processamento	Treinamento	Pós-processamento	Performance	Fairness	**Geral**
Renda	Nenhum	Regressão Logística	Equalized Odds	985	965	**980**
Renda	Learning Fair Representations	Gradient Boosting	Nenhum	987	960	**980**
Renda	Learning Fair Representations	Regressão Logística	Nenhum	981	973	**979**
Renda	Nenhum	Grid Search Reduction	Nenhum	989	950	**979**
Renda	Reweighing	Regressão Logística	Nenhum	981	965	**977**

Tabela 6: Melhores opções escolhidas pelo modelo MAPE-K
Todos os métodos - 25% Performance/75% Fairness

Workflow				Pontuação		
Atributo protegido	Pré-processamento	Treinamento	Pós-processamento	Performance	Fairness	**Geral**
Renda	Learning Fair Representations	Regressão Logística	Nenhum	981	973	**975**
Renda	Nenhum	Gradient Boosting	Equalized Odds	988	969	**974**
Renda	Learning Fair Representations	Random Forest	Nenhum	991	968	**973**
Renda	Nenhum	Exponentiated Gradient Reduction	Nenhum	986	966	**971**
Renda	Reweighing	Gradient Boosting	Nenhum	987	964	**970**

4.2 Estudo de Caso 2: Evolução do *Workflow* adicionando um novo conjunto de dados

Neste Estudo de Caso, foi realizada uma evolução do sistema adicionando um novo conjunto de dados mais próximo de conjuntos de dados reais. Além de reforçar a versatilidade do MAPE-K em diferentes contextos, um maior foco foi colocado na manutenção do sistema, discutindo se as etapas e arquiteturas escolhidas são viáveis para evoluir e manter o *Workflow* sem grandes deteriorações nas ideias originais de seu desenvolvimento, desta vez utilizando o Lendingclub Dataset [4] como conjunto de dados, Renda como atributo protegido e um intervalo de pontuação entre 500 e 980. Os resultados estão presentes abaixo nas Tabelas 4, 5 e 6:

Nessas execuções, a principal observação notada é que a relação de algoritmos com melhores desempenhos mudou completamente, com predominância do algoritmo *Learning Fair Representations* para redução de viés e Regressão Logística para algoritmo de treinamento. Também se nota que algoritmos com redução de viés no pós-processamento e algoritmos de treinamento como *Support Vector Machines* não foram tão eficientes. Com isso, pode-se concluir que, ao mudar o contexto do problema e os dados envolvidos, o MAPE-K pode ajudar a enxergar tais sutilezas e ajudar em uma decisão de forma mais eficiente e ágil. Entretanto, os dados e metadados obtidos não ajudaram a entender o porquê de tais sutilezas acontecerem.

Para processar o Lendingclub Dataset, foram necessárias modificações para realizar a evolução do sistema e adicionar este conjunto de dados como opção no *Workflow*. Estas foram contadas de acordo com

seus *commits* realizados no repositório e exibidos na Tabela 7:

Tabela 7: Quantidade de modificações realizadas ao adicionar um novo conjunto de dados ao *Workflow*

Parte do Sistema	Linhas alteradas	Total de linhas	Arquivos alterados	Total de arquivos	% linhas alteradas	% arquivos alterados
Engenharia de Dados	122	277	2	3	**44,04%**	**66,67%**
Workflow de IA	76	1982	5	38	**3,84%**	**13,16%**
Autonomia do *Workflow*	0	457	0	10	**0,00%**	**0,00%**
Interface Humano-Computador (*Frontend*)	13	2905	2	14	**0,45%**	**14,29%**
Interface Humano-Computador (*Backend*)	4	432	1	7	**0,93%**	**14,29%**
TOTAL	**215**	**6053**	**10**	**72**	**3,55%**	**13,89%**

A adição do conjunto de dados não exigiu modificações no Elemento Autônomo, mesmo os resultados sendo completamente diferentes do Estudo de Caso anterior. Isto reforça a autonomia proposta no MAPE-K, possibilitando escolhas diferentes baseadas nos metadados presentes no *Workflow* sem realizar modificações. O Elemento Autônomo só exigirá modificações se os metadados salvos do *Workflow* forem modificados, ou se modificar alguma configuração intrínseca ao próprio MAPE-K, que não possui nenhuma relação com o *Workflow*.

Os elementos na Interface Humano-Computador exigiram pouquíssimas modificações, podendo ser resumidos a simples adições para colocar a opção do novo conjunto de dados. As maiores modificações foram realizadas no *Workflow* de IA e nas Transformações do Conjunto de Dados, principalmente deste último. Das modificações no *Workflow* de IA, a grande parte (34 linhas, ou 44,74% das linhas) foi realizada no pré-processamento do dado para o cálculo dos algoritmos de treinamento. Embora a estruturação baseada na arquitetura *Pipe-and-Filter* traga algumas linhas a mais para as modificações devido a quantidade de classes criadas para o *workflow* (2 para pipes e 1 para filtro), etapas de processamento e transformação dos dados serão as partes de onde se consumirá mais tempo e modificações por parte dos Engenheiros e Cientistas de Dados.

Embora tenha a consequência de escrever algumas linhas a mais, o uso da arquitetura *Pipe-and-Filter* permite o encapsulamento dos algoritmos e a separação de interesses de forma simples, fazendo com que a parte de código existente para o *workflow* possa ter escolhas mais interessantes e elegantes para um bom *Design* do código. Dito isso, realizar a manutenção/evolução do sistema no *Workflow* e na Interface Humano-Computador é relativamente simples, desde que se saiba os arquivos onde as modificações serão realizadas. Por isso, a criação de uma documentação é extremamente importante para que um novo desenvolvedor entenda o todo do sistema e não adicione linhas em trechos desnecessários.

4.3 Estudo de Caso 3: Evolução do *Workflow* com desenvolvedor sem conhecimento anterior adicionando um novo algoritmo de treinamento

Neste Estudo de Caso, foi realizada uma evolução do sistema adicionando um novo algoritmo de classificação. O foco ainda é na manutenção do sistema, mas desta vez foi colocado para outros desenvolvedores desenvolverem a solução. A discussão se focará mais se as arquiteturas escolhidas são versáteis e simples o suficiente para que pessoas entendam o contexto do sistema e façam novas evoluções, novamente utilizando o Lendingclub Dataset [4] como conjunto de dados, Renda como atributo protegido e um intervalo de pontuação entre 500 e 980.

Foi realizada uma sessão de desenvolvimento com um outro desenvolvedor, durando um tempo entre 1 e 2 horas, onde este adicionou um novo algoritmo de classificação dentro da estrutura do *workflow* com a documentação montada durante o Estudo de caso anterior. Durante a sessão, alguns itens como erros na documentação e bugs foram notados e corrigidos, através de observação e *feedbacks* do desenvolvedor. Por causa destes erros, uma pequena parte da sessão foi gasto em auxílios e dúvidas para que o desenvolvedor pudesse realizar o desenvolvimento sem que precisasse gastar o tempo identificando problemas que não foram de responsabilidade dele. Depois da sessão, o desenvolvedor preencheu um questionário, refletindo um perfil com certa experiência na área de dados e desenvolvimento. No geral, as impressões foram positivas e a implementação ocorreu sem dificuldades. Embora a intenção inicial era que o desenvolvedor apenas se guiasse pela documentação e o auxílio durante a sessão acabou facilitado o entendimento por parte do

desenvolvedor, ele próprio considerou a adaptação a tal experimento rápida. Dada a afirmação, é possível concluir que as decisões arquiteturais tomadas foram corretas e facilitaram o desenvolvimento de evoluções.

O algoritmo de classificação escolhido para o desenvolvimento foi o Naive Bayes [23], que também é bastante difundido como classificador. Após o desenvolvimento, os resultados baseados nas pré-condições e restrições já comentadas na seção anterior estão presentes abaixo nas Tabelas 8, 9 e 10:

Tabela 8: Melhores opções escolhidas pelo modelo MAPE-K
Todos os métodos - 50% Performance/50% Fairness

Workflow				Pontuação		
Atributo protegido	Pré-processamento	Treinamento	Pós-processamento	Performance	Fairness	**Geral**
Renda	Learning Fair Representations	Random Forest	Nenhum	991	968	**979**
Renda	Nenhum	Gradient Boosting	Equalized Odds	988	969	**978**
Renda	Reweighing	Random Forest	Nenhum	991	963	**977**
Renda	Learning Fair Representations	Regressão Logística	Nenhum	981	973	**977**
Renda	Reweighing	Gradient Boosting	Nenhum	987	964	**976**

Tabela 9: Melhores opções escolhidas pelo modelo MAPE-K
Todos os métodos - 75% Performance/25% Fairness

Workflow				Pontuação		
Atributo protegido	Pré-processamento	Treinamento	Pós-processamento	Performance	Fairness	**Geral**
Renda	Nenhum	Regressão Logística	Equalized Odds	985	965	**980**
Renda	Learning Fair Representations	Gradient Boosting	Nenhum	987	960	**980**
Renda	Learning Fair Representations	Regressão Logística	Nenhum	981	973	**979**
Renda	Nenhum	Grid Search Reduction	Nenhum	989	950	**979**
Renda	Reweighing	Regressão Logística	Nenhum	981	965	**977**

Tabela 10: Melhores opções escolhidas pelo modelo MAPE-K
Todos os métodos - 25% Performance/75% Fairness

Workflow				Pontuação		
Atributo protegido	Pré-processamento	Treinamento	Pós-processamento	Performance	Fairness	**Geral**
Renda	Nenhum	Naive Bayes	Calibrated Equalized Odds	991	976	**980**
Renda	Learning Fair Representations	Regressão Logística	Nenhum	981	973	**975**
Renda	Nenhum	Gradient Boosting	Equalized Odds	988	969	**974**
Renda	Learning Fair Representations	Random Forest	Nenhum	991	968	**973**
Renda	Nenhum	Exponentiated Gradient Reduction	Nenhum	986	966	**971**

Para o contexto do Lendingclub Dataset a adição do Naive Bayes no *workflow* mostrou seu valor, apresentando uma pontuação acima do esperado para configurações que privilegiam justiça e só não foi mais destacada por causa do limiar máximo estabelecido de 980. Entretanto, como já visto em estudos de caso anteriores, pode não funcionar em outros contextos e os dados e metadados estabelecidos não definem explicações do porquê o Naive Bayes teve um comportamento positivo para este conjunto de dados.

As evoluções realizadas no *Workflow* para adicionar o Naive Bayes foram contadas de acordo com seus *commits* realizados no repositório e exibidos na Tabela 7:

Tabela 11: Quantidade de modificações realizadas ao adicionar um novo algoritmo ao *Workflow*

Parte do Sistema	Linhas alteradas	Total de linhas	Arquivos alterados	Total de arquivos	% linhas alteradas	% arquivos alterados
Engenharia de Dados	0	277	0	3	**0,00%**	**0,00%**
Workflow de IA	60	2042	5	39	**2,94%**	**12,82%**
Autonomia do *Workflow*	1	458	1	10	**0,22%**	**10,00%**
Interface Humano-Computador (*Frontend*)	71	2948	3	14	**2,41%**	**21,43%**
Interface Humano-Computador (*Backend*)	1	433	1	7	**0,23%**	**14,29%**
TOTAL	**132**	**6157**	**9**	**72**	**2,14%**	**12,50%**

Desta vez, a única parte do sistema sem necessidade de modificações foi a parte de Transformação do conjunto de dados. Entretanto, a única modificação necessária no componente MAPE-K e no *Backend* da Interface Humano-Computador foi a adição do Naive Bayes como parte de uma parametrização, que poderia ser transferida para um arquivo externo de configuração e não exigir modificações futuramente.

No geral, foram exigidas menos modificações que a evolução proposta no Estudo de Caso anterior. A única parte que exigiu mais modificações foi no *Frontend* da Interface Humano-Computador, em grande parte por causa de um único componente presente na tela de Configurações para Planejamento do MAPE-K. Fora esta exceção, adicionar um algoritmo é uma evolução mais simples de ser feita, e junto com a documentação criada um desenvolvedor com relativa experiência consegue executar essa tarefa sem grandes dificuldades.

5 Conclusões e Trabalhos Futuros

Pode-se dizer que o conjunto da arquitetura de *Pipe-And-Filter* com a arquitetura MAPE-K se adaptou muito bem na implementação dos objetivos principais. Com a arquitetura *Pipe-And-Filter*, foi possível encapsular todos os procedimentos presentes em um *workflow* de uma aplicação de IA em etapas coesas e

trocá-las caso haja a necessidade de teste com outro algoritmo, atributo protegido ou conjunto de dados. Com a proveniência dos dados, presente durante a avaliação dos modelos no *workflow*, foi possível coletar os dados e métricas para a elaboração de uma Base de Conhecimento. Com a arquitetura MAPE-K, foi possível realizar um fluxo para que os dados obtidos no processo fossem filtrados, analisados para que haja uma tomada de decisão automática. Com a junção destes três conceitos, foi possível estabelecer um *workflow* automatizado, dependendo apenas dos próprios dados obtidos em execuções anteriores para a automação ser executada. Entretanto, pode-se notar alguns pontos na implementação que não irão ser resolvidos apenas pela escolha da arquitetura.

Embora o *Pipe-And-Filter* seja um modelo que facilite o encapsulamento e a segmentação dos procedimentos do *workflow*, ele ainda é preso a limitações de fatores externos ao código, como o Hardware de onde será rodado e os dados a serem trabalhados. Desse modo, ao realizar a implementação, quem desenvolve terá de escolher um equilíbrio entre memória, armazenamento e performance. Por exemplo, como neste projeto os conjuntos de dados avaliados eram pequenos, eles puderam ser armazenados em memória e garantir performance máxima, mas conforme o número de dados avaliados for crescendo é necessário sacrificar performance e realizar as operações em dispositivos de armazenamento para manter a aplicação estável e com o mesmo custo, não necessitando da contratação de mais desenvolvedores e evitando gastos com infra-estrutura.

Embora olhar para a proveniência dos dados permita o armazenamento de dados importantes para estabelecer um elo entre o *workflow* e o componente MAPE-K, não foi possível garantir que ela substitua Explainable AI pois não foram encontradas evidências do porquê de determinados algoritmos funcionarem em determinados conjuntos de dados e não em outros com os metadados cadastrados na base de conhecimento. A qualidade dos dados também é um ponto que deverá ser observado conforme o número de dados for crescente: É possível ter resultados fora do comum (*outliers*), alguns atributos podem não permitir uma avaliação acurada, ou a quantidade de atributos pode não ser suficiente para avaliar. Antes da proveniência ser implementada, os próprios dados precisam ser modelados conforme o problema exige. Após a implementação, é preciso realizar processos de limpeza dos dados conforme execuções no *workflow* vão sendo realizadas.

Embora o MAPE-K permita o desenvolvimento de uma aplicação autônoma, isso não significa que ela seja seja completamente automatizada para qualquer problema relacionado a *Machine Learning*, necessitando de ação humana para funcionar. A principal limitação por parte do desenvolvimento é que a biblioteca AIF360 suporta apenas problemas de classificação binária, e para evoluções e novos métodos é provável que ocorram refatorações no *workflow*. Também há de se considerar que, mesmo que o *workflow* evolua para abrigar outros tipos de problemas, o contexto do problema é importante ao se avaliar se o modelo é considerado bom ou não. O uso de pesos para as métricas e diferentes estratégias nas fases de análise e planejamento do MAPE-K ajudam a definir o contexto para uma avaliação, mas ainda vai depender de um Cientista de Dados e/ou de um especialista de Domínio para entender quais as necessidades do problema analisado e se os resultados são aceitáveis para a publicação de um modelo otimizado.

Referências

[1] What's the difference between artificial intelligence, machine learning and deep learning? URL `https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learn`

[2] Machine learning: o que é e qual sua importância? URL `https://www.sas.com/pt_br/insights/analytics/machine-learning.html`.

[3] Aprendizado de máquina. URL `https://pt.wikipedia.org/wiki/Aprendizado_de_m%C3%A1quina`.

[4] Lendingclub dataset. URL `https://www.kaggle.com/datasets/wordsforthewise/lending-club`.

[5] Statlog (german credit data) data set. URL https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data).

[6] An architectural blueprint for autonomic computing. Technical report, IBM, 2005. URL https://www-03.ibm.com/autonomic/pdfs/AC%20Blueprint%20White%20Paper%20V7.pdf.

[7] Nadeem Abbas, Jesper Andersson, and Welf Löwe. Autonomic software product lines. pages 324–331, 2010.

[8] Tom Begley, Tobias Schwedes, Christopher Frye, and Ilya Feige. Explainability for fair machine learning, 2021.

[9] Sumon Biswas and Hridesh Rajan. Do the machine learning models on a crowd sourced platform exhibit bias? an empirical study on model fairness. page 642–653, 2020.

[10] Joy Buolamwini and Timnit Gebru. Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Conference on Fairness, Accountability and Transparency*, volume 81 of *Proceedings of Machine Learning Research*, pages 77–91, 2018.

[11] Moritz Hardt, Eric Price, Eric Price, and Nati Srebro. Equality of opportunity in supervised learning. In *Advances in Neural Information Processing Systems*, 2016. URL https://proceedings.neurips.cc/paper/2016/file/9d2682367c3935defcb1f9e247a97c0d-Paper.pdf.

[12] Faisal Kamiran and Toon Calders. Data pre-processing techniques for classification without discrimination. *Knowledge and Information Systems*, 2011. URL https://www.researchgate.net/publication/228975972_Data_Pre-Processing_Techniques_for_Classification_without_Discrimination.

[13] Faisal Kamiran, Asim Karim, and Xiangliang Zhang. Decision theory for discrimination-aware classification. In *2012 IEEE 12th International Conference on Data Mining*, pages 924–929, 2012.

[14] Andrei Kapishnikov, Tolga Bolukbasi, Fernanda B. Viégas, and Michael Terry. XRAI: better attributions through regions. In *IEEE/CVF International Conference on Computer Vision*, pages 4947–4956, 2019.

[15] Jeffrey Kephart and D.M. Chess. The vision of autonomic computing. pages 41 – 50, 2003.

[16] Sasan Maleki, Long Tran-Thanh, Greg Hines, Talal Rahwan, and Alex Rogers. Bounding the estimation error of sampling-based shapley value approximation with/without stratifying. 2013.

[17] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. A survey on bias and fairness in machine learning. *ACM Computing Surveys*, pages 1–35, 2021.

[18] Carlos Mougan, José Manuel Álvarez, Gourab K. Patro, Salvatore Ruggieri, and Steffen Staab. Fairness implications of encoding protected categorical attributes. 2022.

[19] Till Speicher, Hoda Heidari, Nina Grgic-Hlaca, Krishna P. Gummadi, Adish Singla, Adrian Weller, and Muhammad Bilal Zafar. A unified approach to quantifying algorithmic unfairness. 2018. URL http://dx.doi.org/10.1145/3219819.3220046.

[20] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3319–3328, 2017.

[21] Rich Zemel, Yu Wu, Kevin Swersky, Toni Pitassi, and Cynthia Dwork. Learning fair representations. In *Proceedings of the 30th International Conference on Machine Learning*, pages 325–333, 2013. URL https://proceedings.mlr.press/v28/zemel13.html.

[22] Brian Zhang, Blake Lemoine, and Margaret Mitchell. Mitigating unwanted biases with adversarial learning. pages 335–340, 2018. URL https://www.researchgate.net/publication/330299272_Mitigating_Unwanted_Biases_with_Adversarial_Learning.

[23] Harry Zhang. The optimality of naive bayes. 2004.