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

10 de março de 2023

Resumo

O uso de Aprendizado de Máquina (ML) 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 aos grupos que sofrem discriminações. Entretanto, com estas novas descobertas, aumenta-se a complexidade da análise do Cientista de Dados para obter modelos com os melhores resultados. Este artigo foca no tema de Aprendizado de Máquina do ponto de vista da Engenharia de Software, apresentando uma solução 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, utilizando um *Workflow* que possui uma diversa gama de algoritmos já implementados e utiliza a arquitetura MAPE-K para determinar qual possui o melhor balanceamento. Foram realizados diversos estudos de caso para determinar se o MAPE-K pode ser viável na resolução destes problemas e se a solução implementada é extensível sem grandes esforços, possibilitando um maior uso de algoritmos. Diante deste contexto, foi notado que a solução pode facilitar caminhos realizados pelo Especialista de ML, 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, um efeito colateral comum nestas soluções 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 [12], e uma vez que as métricas utilizadas para medir a qualidade de um modelo final são geralmente baseadas em acurácia, precisão e recall, discriminações não são facilmente percebidas por tais métricas.

Devido a esse problema, é possível estabelecer métricas para determinar o quão o modelo está preparado para dados sensíveis [10], 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* [17], *Adversarial Debiasing* [27] e *Reject Option Classification* [18]. 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.

Para este determinado contexto, foi desenvolvida uma estrutura de *Workflow* para aplicações de *Machine Learning* que é manuseada pelo Cientista de Dados de forma semi-autônoma, focada em dois objetivos 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.

Esta estrutura é dividida em 5 módulos: Engenharia de Dados, *Workflow* de ML, Gerenciador Autonômico, Backend e Interface. Para a Engenharia de Dados, foram realizadas modificações nos conjuntos de dados German Credit Dataset [5] e Lendingclub Dataset [4] para serem adequados ao uso do *workflow*. Para o desenvolvimento e estruturação do *workflow*, foi utilizada a arquitetura *Pipe-and-Filter*. Para o Gerenciador Autonômico, foi utilizada a arquitetura MAPE-K [6] para analisar uma base de conhecimento e prover o melhor *Workflow* seguindo regras pré-determinadas. Para a interface e para o backend, sua criação foi pensada nos moldes de uma aplicação web. O código deste desenvolvimento foi disponibilizado no GitHub[1] para avaliação e testes em estudos posteriores.

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 [21].

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,

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

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.

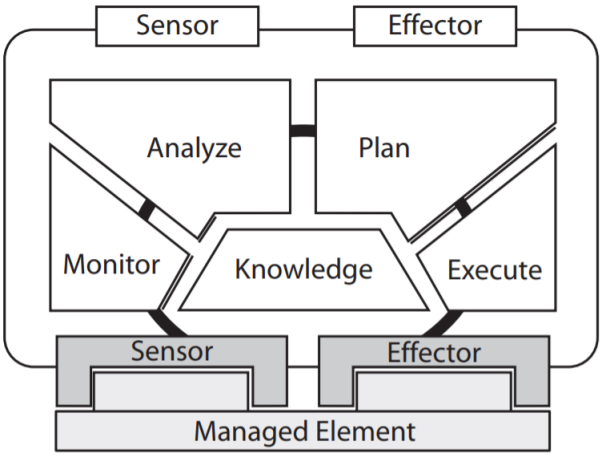

Figura 1: Diagrama de funcionamento da arquitetura MAPE-K [7].

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*

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 [10]. 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. Algoritmos de *Machine Learning* 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 [22]. 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 [23]. 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.

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 disparidade 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):

$$TPR = \frac{TP}{TP + FN} FPR = \frac{FP}{FP + TN} \quad (1)$$

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 [26]:** Esta métrica é baseada na seguinte fórmula:

$$Pr(Y = 1|Z = 0) - Pr(Y = 1|Z = 1) \quad (2)$$

  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*) [11]:** É a diferença entre a taxa positiva verdadeira do grupo não privilegiado e a taxa positiva verdadeira do grupo privilegiado:

$$TPR_{Z=0} - TPR_{Z=1} \quad (3)$$

  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 [16]

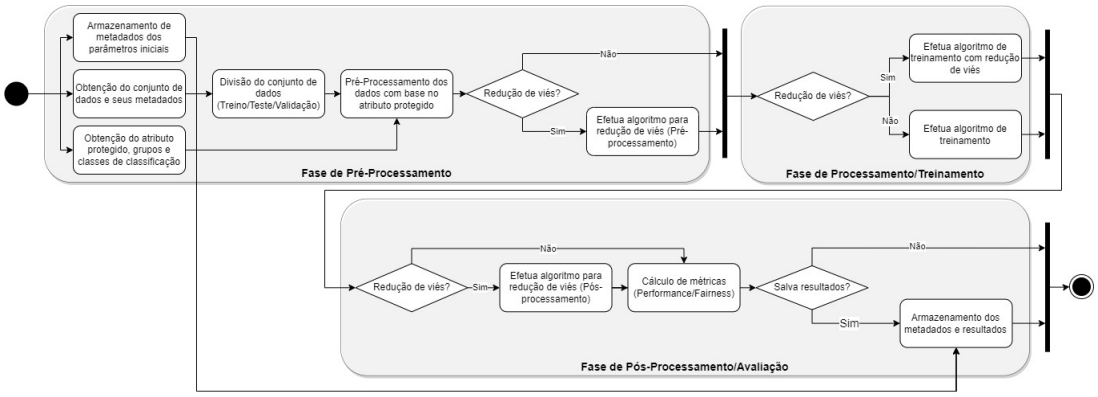

Figura 2: Diagrama de Atividades do *Workflow* de ML

- **Diferença de probabilidade média (*Average odds difference*) [11]:** 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*) [11]:** 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*) [25]:** Esta medida também é conhecida como índice de entropia generalizado, mas com α igual a 1 [25]. 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

O sistema implementado é dividido em 5 módulos:

- **Engenharia de Dados:** Módulo com o objetivo de executar processos de transformação e limpeza de dados.

- **Workflow de ML:** Módulo que executa um Pipeline de uma aplicação automatizada de ML, com etapas de preparação dos dados (Pré-processamento), treinamento (Processamento) e avaliação dos resultados (Pós-processamento) para a geração de um modelo final, cujas etapas estão ilustadas na figura 2.

- **Gerenciador Autonômico:** Módulo contendo o *loop* MAPE-K 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.

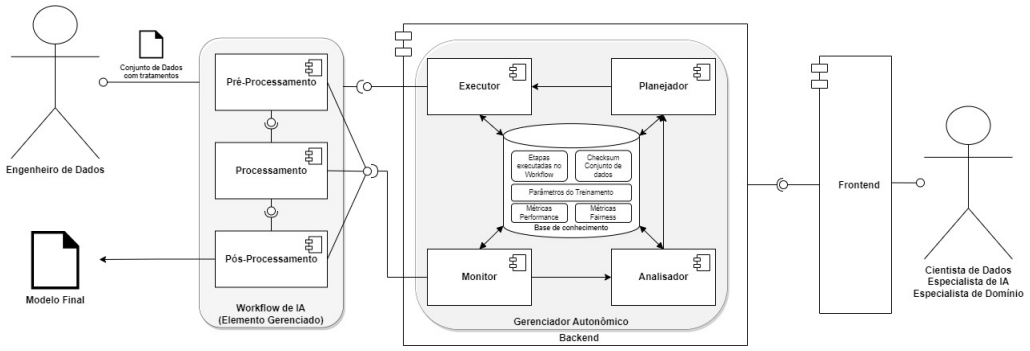

Figura 3: Comunicação entre Interface, gerente autonômico, Engenharia de Dados e *Workflow* de IA

- **Interface (Frontend):** Módulo criado com o objetivo de simular a etapa anterior, porém de modo a proporcionar uma experiência de usuário mais simples e intuitiva para o Cientista de Dados.

- **Backend:** Módulo criado com o objetivo de conectar o Frontend ao Gerenciador Autonômico.

A comunicação entre estes módulos está ilustrada na figura 3. O módulo de Engenharia de Dados seria utilizado pelo Engenheiro de Dados, que devolveria um Conjunto de Dados com os tratamentos necessários para execução dentro do *Workflow*. A interface seria utilizada pelo Cientista de Dados, que determinaria as configurações necessárias para execução dos melhores *Workflows* dependendo do contexto do problema.

3.2 Autonomia

Para determinar a autonomia deste workflow, o Monitor coleta os dados obtidos durante as execuções (métricas, parâmetros e etapas executadas) e os organiza para o Analisador realizar o processo de análise desses dados. Neste processo, 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 Performance 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.

Primeiramente, é necessário padronizar as métricas de Fairness m_{F_i} para m'_{F_i} em um intervalo de 0 a 1, conforme apresentado na Equação 5. Isso garante a uniformidade dos resultados e a aplicação correta dos pesos, evitando distorções no cálculo. Como as métricas de Fairness podem envolver tanto a razão entre dois valores diferentes, como no caso da *Disparate Impact*, quanto a diferença entre dois valores diferentes, como no caso da *Equal Opportunity Difference*, são necessários cálculos distintos para obter esse intervalo. Como as métricas de Performance avaliadas possuem o mesmo intervalo de 0 a 1, não há necessidade de tratamento adicional para as métricas m_{P_i}.

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.

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

$$S_F = \lfloor X \times \frac{\sum_{i=1}^{n_F} w_{m'_{F_i}} \times m'_{F_i}}{\sum_{i=1}^{n} w_{m'_{F_i}}} \rceil \tag{6}$$

$$S_P = \lfloor X \times \frac{\sum_{i=1}^{n_P} w_{m_{P_i}} \times m_{P_i}}{\sum_{i=1}^{n} w_{m_{P_i}}} \rceil$$

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

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 das melhores execuções possíveis pelo Workflow, algumas estratégias foram implantadas no Planejador:

- **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.
- **Intervalo de resultados**: Pelo mesmo motivo da estratégia anterior, métricas com resultados não confiáveis podem causar distorções a pontuação obtida pelo cálculo realizado no processo de análise. Para amenizar esas distorções, foi criado um intervalo de pontuação mínimo e máximo para determinar pontuações que podem ser consideradas confiáveis para avaliação.

Após a realização destas estratégias, o Planejador retornará as 5 melhores pontuações para o Cientista de Dados escolher, e solicitará ao Executor uma nova execução do Workflow após a escolha.

4 Avaliação da Solução

Foram realizados 3 Estudos de Caso para determinar a viabilidade da solução, manutenções futuras e viabilidade do Gerenciador Autonômico 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 contendo cerca de 650 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.

No *workflow*, foram implementados os algoritmos *Disparate Impact Remover* [15], *Learning Fair Representations* [26], *Reweighing* [17] e *Optimized Preprocessing* [13] na etapa de Pré-Processamento, os algoritmos Regressão Logística, *Gradient Boosting*, *Random Forest*, *Support Vector Machines*, *Adversarial Debiasing* [27], *Exponentiated Gradient Reduction* [8], *Grid Search Reduction* [9], *Meta Fair Classifier* [14], *Prejudice Remover* [19] e *Rich Subgroup Fairness* [20] na etapa de Processamento, e os algoritmos *Equalized Odds* [16], *Calibrated Equalized Odds* [24] e *Reject Option Classification* [18] na etapa de Pós-Processamento. Como não foram encontrados estudos onde são utilizados redução de viés em mais de uma etapa do processo de *Machine Learning*, é realizada a execução de apenas um algoritmo de redução de viés por *workflow*. Por apresentarem dados considerados não confiáveis, foram filtrados as execuções com os algoritmos *Optimized Preprocessing* e *Reject Option Classification*, que correspondiam cerca de 5% do total da base utilizada.

4.1 Estudo de Caso 1: Viabilidade e utilidade do Gerenciador Autonômico

Neste Estudo de Caso, o foco foi colocado em testar e verificar como o Gerenciador Autonômico 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 Gerenciador Autonômico
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 Gerenciador Autonômico
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 Gerenciador Autonômico
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.

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 sig-

nificantemente 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.

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 Gerenciador Autonômico 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:

Tabela 4: Melhores opções escolhidas pelo Gerenciador Autonômico
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 Gerenciador Autonômico
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 Gerenciador Autonômico
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**

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 Gerenciador Autonômico 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 ML	76	1982	5	38	**3,84%**	**13,16%**
Gerenciador Autonômico	0	457	0	10	**0,00%**	**0,00%**
Interface	13	2905	2	14	**0,45%**	**14,29%**
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 Gerenciador Autonômico, mesmo os resultados sendo completamente diferentes do Estudo de Caso anterior. Isto reforça a autonomia proposta no *loop* MAPE-K, possibilitando escolhas diferentes baseadas nos metadados presentes no *Workflow* sem realizar modificações. O Gerenciador Autonômico só

exigirá modificações se os metadados salvos do *Workflow* forem modificados, ou se modificar alguma configuração intrínseca ao próprio Gerenciador, que é desacoplado do *Workflow*.

Os elementos na Interface exigiram pouquíssimas modificações, podendo ser resumidos a simples adições para disponibilizar a opção do novo conjunto de dados ao Cientista de Dados. As maiores modificações foram realizadas no *Workflow* de ML e nas Transformações do Conjunto de Dados, principalmente deste último. Das modificações no *Workflow* de ML, 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* exija a criação de classes adicionais para o *workflow* (2 para pipes e 1 para filtro), é nas etapas de processamento e transformação dos dados que os Engenheiros e Cientistas de Dados vão gastar a maior parte do tempo e fazer mais modificações.

Embora o uso da arquitetura *Pipe-and-Filter* exija a escrita de algumas linhas a mais, ela permite encapsular os algoritmos e separar os interesses de forma simples, tornando possível ter um bom *Design* do código. Isso torna a manutenção/evolução do sistema no *Workflow* e na Interface relativamente simples, desde que o desenvolvedor saiba em quais arquivos as modificações serão realizadas. Por isso, a criação de uma documentação é extremamente importante para que um novo desenvolvedor entenda o sistema como um todo e não adicione linhas de forma desnecessária.

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

Neste Estudo de Caso, realizou-se uma evolução do sistema adicionando um novo algoritmo de classificação. Novamente é discutida a manutenção do sistema, desta vez com foco nas decisões de outros desenvolvedores. Para isso, foi verificado se as arquiteturas escolhidas são versáteis e simples o suficiente para que novos desenvolvedores possam entender o contexto do sistema e realizar novas evoluções com facilidade, novamente utilizando o Lendingclub Dataset [4] como conjunto de dados, Renda como atributo protegido e um intervalo de pontuação entre 500 e 980.

Durante uma sessão de desenvolvimento com um outro desenvolvedor, com duração de cerca de 1 e 2 horas, um novo algoritmo de classificação foi adicionado à estrutura do *workflow* por ele utilizando a documentação elaborada 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. Devido a esses problemas, uma pequena parte da sessão foi dedicada em auxílios e dúvidas para que o desenvolvedor evitasse perder tempo com eles e se concentrasse no desenvolvimento do *workflow*. Depois da sessão, o desenvolvedor preencheu um questionário indicando um perfil com certa experiência na área de dados e desenvolvimento, com impressões positivas da sessão e uma implementação sem dificuldades. Embora a intenção inicial era que o desenvolvedor apenas se guiasse pela documentação, ele próprio considerou a adaptação a tal experimento rápida, independente do auxílio fornecido. Isso sugere que as decisões iniciais tomadas na implementação foram corretas e facilitaram o desenvolvimento de evoluções.

O algoritmo de classificação escolhido para o desenvolvimento foi o Naive Bayes [28], que também é bastante difundido como classificador. Após o desenvolvimento, os resultados estão presentes abaixo nas Tabelas 8, 9 e 10:

Tabela 8: Melhores opções escolhidas pelo Gerenciador Autonômico
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 Gerenciador Autonômico
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 Gerenciador Autonômico
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 ML	60	2042	5	39	**2,94%**	**12,82%**
Gerenciador Autonômico	1	458	1	10	**0,22%**	**10,00%**
Interface	71	2948	3	14	**2,41%**	**21,43%**
Backend	1	433	1	7	**0,23%**	**14,29%**
TOTAL	**132**	**6157**	**9**	**72**	**2,14%**	**12,50%**

Desta vez, o único módulo sem necessidade de modificações foi Engenharia de Dados, por já terem sido implementadas anteriormente. Também, a única modificação necessária no Gerenciador Autonômico e no *Backend* 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 via código.

No geral, foram exigidas menos modificações que a evolução proposta no Estudo de Caso anterior. A única parte que exigiu mais modificações foi na Interface, em grande parte por causa de um único componente presente na tela de Configurações para Planejamento do Gerenciador Autonômico. 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

A literatura revisada sobre *Fairness* verifica apenas problemas de classificação binária, e para evoluções e novos métodos é provável que ocorram refatorações no *workflow*. Com a arquitetura *Pipe-And-Filter*, foi possível encapsular todos os procedimentos presentes em um *workflow* de uma aplicação de *Machine Learning* em etapas coesas e trocá-las caso haja a necessidade de teste com outro algoritmo, atributo protegido ou conjunto de dados, e facilitar o entendimento de outros desenvolvedores.

Com a arquitetura MAPE-K, foi possível realizar um fluxo para que os dados obtidos no *workflow* fossem analisados para facilitar uma tomada de decisão. Embora a autonomia seja viável, a supervisão humana é necessária devido a problemas ainda enfrentados pelo tema *Fairness*, onde o contexto social do problema é de extrema importância ao 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 Gerenciador Autonômico garantem o balanceamento performance/*Fairness* e ajudam a definir o contexto para uma avaliação, mas ainda depende de um Cientista de Dados e/ou de um especialista de Domínio entender quais as necessidades do problema analisado e se os resultados são aceitáveis para a publicação de um modelo otimizado e justo.

Dadas estas observações, pode-se dizer este conjunto se adaptou muito bem na implementação dos objetivos principais. Como trabalhos futuros, é possível trabalhar com algumas possibilidades. No Gerenciador Autonômico, o Analisador pode realizar uma análise mais profunda aumentando o número de indicadores e considerando grupos além de métricas de *Fairness* e métricas de Performance. No *workflow*, introduzir técnicas para melhora dos resultados como *Data Augmentation* e *K-Fold Cross-Validation*. Também é possível mudar o foco para solucionar problemas de MLOps, utilizando o sistema para determinar um melhor deploy em caso de piora nas métricas de um modelo já utilizado por clientes.

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-learning-deep-learning-ai/`.

[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] Alekh Agarwal, Alina Beygelzimer, Miroslav Dudík, John Langford, and Hanna Wallach. A reductions approach to fair classification. In *International Conference on Machine Learning*, pages 60–69, 2018.

[9] Alekh Agarwal, Miroslav Dudík, and Zhiwei Steven Wu. Fair regression: Quantitative definitions and reduction-based algorithms. In *International Conference on Machine Learning*, pages 120–129, 2019.

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

[11] 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.

[12] 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.

[13] Flavio Calmon, Dennis Wei, Bhanukiran Vinzamuri, Karthikeyan Natesan Ramamurthy, and Kush R Varshney. Optimized pre-processing for discrimination prevention. In *Advances in Neural Information Processing Systems*, 2017. URL `https://proceedings.neurips.cc/paper/2017/file/9a49a25d845a483fae4be7e341368e36-Paper.pdf`.

[14] L Elisa Celis, Lingxiao Huang, Vijay Keswani, and Nisheeth K Vishnoi. Classification with fairness constraints: A meta-algorithm with provable guarantees. In *Proceedings of the conference on fairness, accountability, and transparency*, pages 319–328, 2019.

[15] Michael Feldman, Sorelle A. Friedler, John Moeller, Carlos Scheidegger, and Suresh Venkatasubramanian. Certifying and removing disparate impact. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 259–268, 2015. URL https://doi.org/10.1145/2783258.2783311.

[16] 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.

[17] 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.

[18] 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.

[19] Toshihiro Kamishima, Shotaro Akaho, Hideki Asoh, and Jun Sakuma. Fairness-aware classifier with prejudice remover regularizer. pages 35–50, 2012. URL https://www.researchgate.net/publication/262176212_Fairness-Aware_Classifier_with_Prejudice_Remover_Regularizer.

[20] Michael Kearns, Seth Neel, Aaron Roth, and Zhiwei Steven Wu. Preventing fairness gerrymandering: Auditing and learning for subgroup fairness. In *International Conference on Machine Learning*, pages 2564–2572, 2018.

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

[22] 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.

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

[24] Geoff Pleiss, Manish Raghavan, Felix Wu, Jon Kleinberg, and Kilian Q Weinberger. On fairness and calibration. In *Advances in Neural Information Processing Systems*, 2017. URL https://proceedings.neurips.cc/paper/2017/file/b8b9c74ac526fffbeb2d39ab038d1cd7-Paper.pdf.

[25] 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.

[26] 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.

[27] 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.

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