



Universidade Estadual de Campinas
Instituto de Computação



Thales Eduardo Nazatto

Um arcabouço semi-autônomo para treino de modelos
de Aprendizado de Máquina com enfoque em métricas
de *Fairness*

CAMPINAS
2023

Thales Eduardo Nazatto

**Um arcabouço semi-autônomo para treino de modelos de
Aprendizado de Máquina com enfoque em métricas de *Fairness***

Dissertação apresentada ao Instituto de
Computação da Universidade Estadual de
Campinas como parte dos requisitos para a
obtenção do título de Mestre em Ciência da
Computação.

Orientadora: Profa. Dra. Cecília Mary Fischer Rubira
Coorientador: Prof. Dr. Leonardo Montecchi

Este exemplar corresponde à versão final da
Dissertação defendida por Thales Eduardo
Nazatto e orientada pela Profa. Dra.
Cecília Mary Fischer Rubira.

CAMPINAS
2023

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Ana Regina Machado - CRB 8/5467

N236a Nazatto, Thales Eduardo, 1989-
Um arcabouço semi-autônomo para treino de modelos de aprendizado de máquina com enfoque em métricas de Fairness / Thales Eduardo Nazatto. – Campinas, SP : [s.n.], 2023.

Orientador: Cecília Mary Fischer Rubira.
Coorientador: Leonardo Montecchi.
Dissertação (mestrado) – Universidade Estadual de Campinas, Instituto de Computação.

1. Aprendizado de máquina. 2. Inteligência artificial. 3. Responsabilidade social. 4. Computação autônoma. 5. Métricas de Fairness. 6. Ética. I. Rubira, Cecília Mary Fischer, 1964-. II. Montecchi, Leonardo, 1982-. III. Universidade Estadual de Campinas. Instituto de Computação. IV. Título.

Informações Complementares

Título em outro idioma: A semi-autonomic framework for developing machine learning-based applications using Fairness metrics

Palavras-chave em inglês:

Machine learning

Artificial intelligence

Social responsibility

Autonomic computing

Fairness metrics

Ethics

Área de concentração: Ciência da Computação

Titulação: Mestre em Ciência da Computação

Banca examinadora:

Cecília Mary Fischer Rubira [Orientador]

Fabio Kon

Marcos Medeiros Raimundo

Data de defesa: 08-11-2023

Programa de Pós-Graduação: Ciência da Computação

Identificação e informações acadêmicas do(a) aluno(a)

- ORCID do autor: <https://orcid.org/0000-0002-8138-3650>

- Currículo Lattes do autor: <http://lattes.cnpq.br/7857745016472913>



Universidade Estadual de Campinas
Instituto de Computação



Thales Eduardo Nazatto

Um arcabouço semi-autônomo para treino de modelos de
Aprendizado de Máquina com enfoque em métricas de *Fairness*

Banca Examinadora:

- Profa. Dra. Cecília Mary Fischer Rubira
IC/UNICAMP
- Prof. Dr. Fabio Kon
IME/USP
- Prof. Dr. Marcos Medeiros Raimundo
IC/UNICAMP

A ata da defesa, assinada pelos membros da Comissão Examinadora, consta no SIGA/Sistema de Fluxo de Dissertação/Tese e na Secretaria do Programa da Unidade.

Campinas, 08 de novembro de 2023

Você não consegue ligar os pontos olhando pra frente; você só consegue ligá-los olhando pra trás. Então você tem que confiar que os pontos se ligarão algum dia no futuro. Você tem que confiar em algo – seu instinto, destino, vida, carma, o que for. Esta abordagem nunca me desapontou, e fez toda diferença na minha vida.

(Steve Jobs)

Agradecimentos

Primeiramente, à minha família, pela dedicação que tiveram em me criar, pela liberdade de escolha para fazer o que gosto e pela compreensão atual de que hoje seguimos caminhos completamente distintos, apesar de mantermos contato constantemente.

A meus orientadores, Profa. Dra. Cecília Mary Fischer Rubira e Prof. Dr. Leonardo Montecchi, pelo desafio de orientar uma dissertação com temas fora de seus domínios de estudo, e também ao Prof. Dr. Gerberth Adín Ramírez Rivera, que me aceitou como orientador anteriormente, foi compreensivo no momento de minha desistência e que pude fazer reflexões com tal experiência que levei para esta dissertação final de alguma forma.

A todas as pessoas com quem morei em Campinas nesses anos desde que comecei a frequentar aulas, presentes nas Repúblicas Borritos, KioBio e Galinheiro, pelos momentos de lazer que me fizeram relaxar das tensões encaradas em um curso de Pós-Graduação.

A todas as pessoas que conheci na época em que eu estudei em Rio Claro e reencontrei em Campinas, e também a todas as pessoas que mantive contato de Rio Claro desde que comecei a frequentar aulas, principalmente por entender que as conexões e comunicações se mantêm mesmo quando um ciclo se fecha e um ciclo diferente é iniciado.

A todas as pessoas que trabalhei junto na CI&T, Dextra e Zup, pela compreensão de que o estudo também é essencial para a evolução profissional de uma pessoa.

E finalmente, a todas as pessoas que pude conhecer na Unicamp, pela troca de experiências e pelo contato com pessoas de altíssimo nível e acima de todas as minhas expectativas.

Resumo

O uso crescente de Aprendizado de Máquina em soluções digitais criou condições para o diálogo social sobre vieses e discriminações presentes nos dados, e requeriram a criação de novos algoritmos e métricas de *Fairness* para garantir decisões mais justas. Entretanto, a análise do Cientista de Dados para obter os melhores modelos é mais complexa pela necessidade de equilibrar o aprimoramento das métricas de *Fairness* com uma possível queda das métricas de avaliação tradicionais, considerando o contexto, e por uma maior variedade de algoritmos que podem ser utilizados isoladamente ou em conjunto durante o treinamento. Este trabalho apresenta uma solução para o treino destes modelos que é operada de forma semi-autônoma para encontrar configurações mais otimizadas em diferentes contextos, através de um módulo com variados algoritmos pré-implementados e a arquitetura MAPE-K para auxiliar na avaliação do equilíbrio ideal. Foram realizados diversos estudos de caso para determinar a viabilidade da solução proposta na resolução desses problemas e se a solução implementada é extensível a futuros algoritmos sem grandes esforços. Diante desses estudos, foi notado que a solução pode auxiliar o Cientista de Dados a ter melhor compreensão do processo de treinamento, possibilitando estudos de Engenharia de Software no auxílio ao treinamento de modelos confiáveis em Aprendizado de Máquina.

Palavras-chave — Aprendizado de Máquina, Inteligência Artificial Ética, Responsabilidade Social em Inteligência Artificial, Computação Autônoma, Métricas de *Fairness*

Abstract

The growing use of *Machine Learning* in digital solutions has created conditions for social dialogue on biases and discrimination in data, and has required the creation of new algorithms and *Fairness* metrics to ensure fairer decisions. However, the Data Scientist's analysis to obtain the best models is more complex due to the need to balance the improvement of *Fairness* metrics with a possible decline in traditional evaluation metrics considering the context, as well as a greater variety of algorithms that can be used independently or in combination during training. This work presents a solution for the training of these models that is operated in a semi-autonomous way to identify more optimized configurations across different contexts, through a module with several pre-implemented algorithms and the MAPE-K architecture to assist in evaluating the ideal balance. Several case studies were conducted to determine the feasibility of the proposed solution in solving these problems and if the implemented solution is extensible to future algorithms with minimal effort. In view of these studies, it was noted that the solution can help the Data Scientist in gaining a better understanding of the training process, enabling Software Engineering studies to help in the training of trustworthy *Machine Learning* models.

Keywords — Machine Learning, Artificial Intelligence Ethics, Social Responsibility in Artificial Intelligence, Autonomic Computing, Fairness Metrics

Lista de Figuras

2.1	Ciclo de vida e ecossistema do dado [25].	21
2.2	Exemplo de uma rede neural utilizada em <i>Deep Learning</i>	24
2.3	Processo padrão para aprendizado de máquina	35
2.4	IBM Analytics and AI Reference Architecture [5].	36
2.5	Diagrama de funcionamento da arquitetura MAPE-K [21].	38
2.6	Diagrama de uma arquitetura <i>Pipes-and-Filters</i>	39
2.7	Ciclo resumido do processo presente em MLOps [73]	40
2.8	Exemplo de Pipeline de MLOps com ambientes de Desenvolvimento, <i>Staging</i> e Produção [73]	43
3.1	Componentes arquiteturais da solução	46
3.2	Diagrama de sequência de uma execução manual no Módulo de ML	47
3.3	Diagrama de sequência de uma execução no Arcabouço Autônomo Proposto	48
3.4	Componentes e passos do Módulo de ML	50
3.5	Procedimento de análise para execuções no módulo de ML	52
3.6	Árvore de métricas a serem analisadas	53
3.7	Diagrama de atividades, com base na IBM AI Reference Architecture [5], indicando a subdivisão de cada papel no uso do Arcabouço Autônomo	57
4.1	Comportamento das opções de menu.	60
4.2	Configuração das métricas para o Analisador do Gerenciador Autônomo.	61
4.3	Cenários possíveis na configuração das métricas.	62
4.4	Configuração do Planejador do Gerenciador Autônomo.	63
4.5	Execução simples e manual do Módulo de ML.	64
4.6	Informações do resultado do Módulo de ML.	65
4.7	Métricas do resultado do Módulo de ML.	66
4.8	Execução autônoma do Módulo de ML gerenciada pelo Gerenciador Au- tônomo.	67
4.9	Seleção da configuração arquitetural após análise.	68
4.10	Execução do Módulo de ML após seleção.	69

Lista de Tabelas

2.1	Matriz de confusão	25
2.2	Categorização dos trabalhos relacionados	45
5.1	Melhores configurações escolhidas pelo Gerenciador Autônomico Uso dos algoritmos implementados - 50% Avaliação/50% <i>Fairness</i>	72
5.2	Melhores configurações escolhidas pelo Gerenciador Autônomico Uso dos algoritmos implementados - 75% Avaliação/25% <i>Fairness</i>	73
5.3	Melhores configurações escolhidas pelo Gerenciador Autônomico Uso dos algoritmos implementados - 25% Avaliação/75% <i>Fairness</i>	73
5.4	Melhores configurações escolhidas pelo Gerenciador Autônomico Algoritmos para redução de viés de pré-processamento - 50% Avaliação/50% <i>Fairness</i>	74
5.5	Melhores configurações escolhidas pelo Gerenciador Autônomico Algoritmos para redução de viés de pré-processamento - 75% Avaliação/25% <i>Fairness</i>	74
5.6	Melhores configurações escolhidas pelo Gerenciador Autônomico Algoritmos para redução de viés de pré-processamento - 25% Avaliação/75% <i>Fairness</i>	74
5.7	Melhores configurações escolhidas pelo Gerenciador Autônomico Algoritmos para redução de viés de processamento - 50% Avaliação/50% <i>Fairness</i>	75
5.8	Melhores configurações escolhidas pelo Gerenciador Autônomico Algoritmos para redução de viés de processamento - 75% Avaliação/25% <i>Fairness</i>	75
5.9	Melhores configurações escolhidas pelo Gerenciador Autônomico Algoritmos para redução de viés de processamento - 25% Avaliação/75% <i>Fairness</i>	75
5.10	Melhores configurações escolhidas pelo Gerenciador Autônomico Algoritmos para redução de viés de pós-processamento - 50% Avaliação/50% <i>Fairness</i>	76
5.11	Melhores configurações escolhidas pelo Gerenciador Autônomico Algoritmos para redução de viés de pós-processamento - 75% Avaliação/25% <i>Fairness</i>	76
5.12	Melhores configurações escolhidas pelo Gerenciador Autônomico Algoritmos para redução de viés de pós-processamento - 25% Avaliação/75% <i>Fairness</i>	76
5.13	Melhores configurações escolhidas pelo Gerenciador Autônomico Sem uso de algoritmos para redução de viés - 50% Avaliação/50% <i>Fairness</i>	77

5.14	Melhores configurações escolhidas pelo Gerenciador Autônomico Sem uso de algoritmos para redução de viés - 75% Avaliação/25% <i>Fairness</i>	77
5.15	Melhores configurações escolhidas pelo Gerenciador Autônomico Sem uso de algoritmos para redução de viés - 25% Avaliação/75% <i>Fairness</i>	77
5.16	Métricas de Avaliação das execuções no Módulo de ML	79
5.17	Métricas de <i>Fairness</i> das execuções no Módulo de ML	80
6.1	Melhores configurações escolhidas pelo Gerenciador Autônomico Uso dos algoritmos implementados - 50% Avaliação/50% <i>Fairness</i>	83
6.2	Melhores configurações escolhidas pelo Gerenciador Autônomico Uso dos algoritmos implementados - 75% Avaliação/25% <i>Fairness</i>	84
6.3	Melhores configurações escolhidas pelo Gerenciador Autônomico Uso dos algoritmos implementados - 25% Avaliação/75% <i>Fairness</i>	84
6.4	Quantidade de modificações realizadas ao adicionar um novo conjunto de dados ao Módulo de ML	84
7.1	Melhores configurações escolhidas pelo Gerenciador Autônomico Uso dos algoritmos implementados - 50% Avaliação/50% <i>Fairness</i>	89
7.2	Melhores configurações escolhidas pelo Gerenciador Autônomico Uso dos algoritmos implementados - 75% Avaliação/25% <i>Fairness</i>	89
7.3	Melhores configurações escolhidas pelo Gerenciador Autônomico Uso dos algoritmos implementados - 25% Avaliação/75% <i>Fairness</i>	89
7.4	Quantidade de modificações realizadas ao adicionar um novo algoritmo ao Módulo de ML	90

Sumário

1	Introdução	15
1.1	Contexto	15
1.2	Problemas encontrados	16
1.3	Objetivo	16
1.4	Solução Proposta	17
1.5	Resultados Obtidos	17
1.6	Organização	17
2	Background e Trabalhos Relacionados	18
2.1	Ciência de Dados e Engenharia de Dados	18
2.1.1	Engenharia de Dados	18
2.1.2	Ciência de Dados	20
2.1.3	O ciclo do dado: Semelhanças e diferenças entre Engenharia de Dados e Ciência de Dados	20
2.2	Machine Learning	23
2.2.1	Métricas de Avaliação	24
2.2.2	Tipos de Algoritmos de Machine Learning	25
2.3	<i>Fairness</i> em Algoritmos de Aprendizado de Máquina	30
2.3.1	Métricas de <i>Fairness</i>	31
2.3.2	Algoritmos para redução de vieses	33
2.4	Engenharia de Software para Aplicações de ML	34
2.4.1	Arquitetura de Software	35
2.4.2	Arquitetura de referência para IA	35
2.4.3	Arquitetura MAPE-K	36
2.4.4	Arquitetura <i>Pipes-and-Filters</i>	38
2.5	MLOps	40
2.6	Trabalhos Relacionados	44
3	Arcabouço Autônomo Proposto	46
3.1	Arquitetura da solução	46
3.2	Módulo de ML	49
3.3	Gerenciador Autônomo	51
3.3.1	Monitor	52
3.3.2	Analisador	52
3.3.3	Planejador	55
3.3.4	Executor	56
3.4	Atuação do Arcabouço Autônomo dentro do processo	56

4	Implementação do Arcabouço	58
4.1	Visão geral da implementação	58
4.2	Interface	59
4.2.1	Opções para parametrização do Gerenciador Autônomo	60
4.2.2	Opções para execução do Módulo de ML	63
5	Estudo de Caso 1: Classificação de Crédito (German Credit Dataset)	70
5.1	Contexto e limitações	70
5.2	Resultados e Discussões	72
6	Estudo de Caso 2: Classificação de Crédito (Lendingclub Dataset) e evolução do sistema	82
6.1	Contexto e limitações	82
6.2	Resultados e Discussões	83
7	Estudo de Caso 3: Evolução do sistema com outros desenvolvedores	87
7.1	Contexto e limitações	87
7.2	Resultados e Discussões	88
8	Conclusões	91
8.1	Limitações	92
8.2	Trabalhos Futuros	93
	Referências Bibliográficas	94
A	Documentação de Instalação	100
A.1	Introdução	100
A.2	Programas necessários para instalação	100
A.3	Instalação do sistema	101
A.3.1	Obtenção do código-fonte	101
A.3.2	Montagem de ambiente	102
A.3.3	Instalação das bibliotecas	103
A.4	Execução do sistema	104
A.4.1	Engenharia de dados	104
A.4.2	Workflow de IA	104
A.4.3	Autonomia do Workflow	105
A.4.4	Interface	105
B	Documentação de Manutenção	107
B.1	Introdução	107
B.2	Arquitetura do Workflow e Framework	107
B.2.1	Estrutura	108
B.2.2	Operações	109
B.2.3	Ligação de Pipe com Filter	109
B.2.4	Ligação de Filter com Pipe	109
B.2.5	Seleção parcial de dados presentes no Pipe	109
B.2.6	Junção de Pipes	109
B.3	Incrementos no Workflow	109
B.3.1	Adicionando um novo Conjunto de Dados	109

B.3.2 Adicionando um novo Algoritmo	115
C Questionário Aplicado ao Desenvolvedor	122

Capítulo 1

Introdução

Técnicas de Inteligência Artificial e Aprendizado de Máquina (*Machine Learning*, ou ML) 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 mas que são difíceis de serem traduzidos em código. Entretanto, nos últimos anos ocorreu um crescimento no uso dessas tecnologias em aplicações tradicionais, com estimativa de US\$ 57 bilhões em investimentos em 2021, 480% maior em relação a 2017 [18]. No Brasil, o número de empresas de IA aumentou de 120 em 2018 para 206 em 2020 [19].

Isso se mostra possível devido a grande quantidade de dados processada diariamente pelas empresas, que coletam estatísticas toda vez que um usuário acessa suas aplicações. Com esses dados, podem traçar diferentes perfis e usar soluções de ML para ter tomadas de decisão mais assertivas com o objetivo de melhorar a experiência de usuário e corrigir problemas. Porém, muitas dessas soluções foram projetadas sem pensar em governança de dados como requisito de projeto, e se mostram ineficientes quando ela é tida em consideração.

1.1 Contexto

Governança de dados é um tema que entrou em evidência recentemente em países como o Brasil: Iniciativas como a LGPD - Lei Geral de Proteção de Dados [20], de 14 de agosto de 2018, mostram como as aplicações e seus dados possuem cada vez mais influência na sociedade moderna. E nesse ponto muitas aplicações baseadas em Aprendizado de Máquina falham: muitas implementações são baseadas em caixas pretas, onde o determinante para estabelecer a confiança no modelo implementado é sua entrada e sua saída. Um efeito colateral dessa estratégia é a exposição de vieses que, embora sejam vistos como não-intencionais pelos desenvolvedores, refletem preconceitos existentes da sociedade atual. Uma entrada de dados enviesada resulta em um modelo de ML que realiza discriminações em sua classificação [29], e considerando que as métricas de avaliação tradicionais avaliam a totalidade do conjunto de dados em vez de grupos específicos, discriminações não são facilmente percebidas por elas.

Devido a esse problema, existem métricas para determinar o quão o modelo está

preparado para dados sensíveis, passíveis de discriminações, termo que é conhecido como *Fairness* [24]. Com a evolução das pesquisas na comunidade acadêmica, também foram criados algoritmos para redução dos vieses presentes nos conjuntos de dados, como por exemplo *Reweighting* [47], *Adversarial Debiasing* [76] e *Reject Option Classification* [48].

1.2 Problemas encontrados

Embora suas premissas tenham bastante utilidade no cenário do mundo de hoje, uma solução voltada ao treino de modelos de ML mais justos adicionam alguns problemas em relação a um modelo de ML tradicional:

- **Aumento da complexidade na configuração arquitetural utilizada:** Em uma solução voltada ao treino de modelos de ML tradicionais, os algoritmos utilizados são executados, em geral, durante a fase de treinamento. Embora haja técnicas e algoritmos que podem ser aplicados antes deste processo para melhorar a modelagem do problema, ao adicionar o cálculo de métricas de *Fairness* como requisito desta aplicação aumentamos a gama de algoritmos a se testar para obter melhores resultados, que podem ser executados antes, durante e após a fase de treinamento.
- **Aumento da complexidade na avaliação e calibração de um modelo de ML:** Para medir o cálculo de métricas de *Fairness* são necessárias novas métricas especificamente para esse fim, que aumentam o tempo para análise e validação se o modelo final é bom o suficiente ou não. Além do maior número de métricas a se avaliar, é provável que, quando um modelo de ML mais justo é considerado como requisito, um modelo treinado com algoritmos voltados para reduzir vieses e com um determinado conjunto de dados seja avaliado com métricas de avaliação piores do que um modelo treinado com algoritmos tradicionais, uma vez que o algoritmo voltado para reduzir vieses precisa fazer concessões para que grupos considerados com privilégios e grupos considerados sem privilégios tenham os resultados mais semelhantes possíveis. Esse compromisso precisa ser considerado nos resultados finais para que o aprimoramento das métricas de *Fairness* compense uma hipotética queda das métricas de avaliação tradicionais.

1.3 Objetivo

O objetivo desta dissertação de mestrado é desenvolver uma estrutura para o treinamento de modelos de *Machine Learning* de forma semi-autônoma, que permita que o cientista de dados obtenha a melhor configuração arquitetural do processo, resolvendo os dois problemas principais:

- Facilitar a criação de modelos justos e confiáveis com a automatização do seguinte processo: Preparação dos dados, treinamento e avaliação.
- Estabelecer um balanceamento entre métricas de avaliação tradicionais com métricas de *Fairness* para obter modelos mais justos com eficiência.

1.4 Solução Proposta

A solução é dividida em 4 módulos principais: Engenharia de dados, Módulo de ML, Gerenciador Autônomo e uma Interface Humano-Computador. Tais módulos sintetizam os processos de Organização e Análise da IBM Analytics and AI Reference Architecture [5].

Para o desenvolvimento do Módulo de ML, será utilizado o padrão de arquitetura *Pipes-and-Filters* [43]. Para o Gerenciador Autônomo, será utilizada a arquitetura MAPE-K [15] para analisar uma base de conhecimento e prover o melhor pipeline seguindo regras pré-determinadas. A Interface Humano-Computador foi criada nos moldes de uma aplicação web. O código deste desenvolvimento foi disponibilizado no GitHub¹ para avaliação e testes em estudos posteriores.

1.5 Resultados Obtidos

A escolha do padrão de arquitetura *Pipes-and-Filters* se mostrou favorável para o desenvolvimento de um *workflow* para aplicações envolvendo IA, permitindo que ele seja modular e que sejam feitas evoluções sem exigir grandes esforços. O uso da arquitetura MAPE-K também se mostrou favorável, permitindo diversos resultados para diferentes contextos de problema e uma simplificação da análise realizada pelo Cientista de Dados, podendo resultar em economia de tempo. Ela também possibilitou um balanço entre métricas de avaliação e métricas de *Fairness* através da adição de pesos para cada métrica na parte de análise. Embora os pesos não sejam parte da arquitetura, a divisão presente na arquitetura permite que o desenvolvimento seja pensado de maneira mais clara.

A obtenção de metadados do Módulo de ML se mostrou essencial para alimentar o Gerenciador Autônomo e possibilitou a análise e tomadas de decisão baseadas em dados. É possível realizar análises mais detalhadas conforme novas execuções forem realizadas, consequentemente podendo resultar em melhores configurações arquiteturais para contextos distintos. Posteriormente, é possível focar em melhorar a análise do Gerenciador Autônomo adicionando novos indicadores, expandir o Módulo de ML com técnicas não exploradas e explorar processos de operação e infra-estrutura, também conhecidos como MLOps, para melhorar a confiabilidade dos modelos por meio de mecanismos para implantação dos modelos de ML e notificações.

1.6 Organização

O restante da dissertação se organizará da seguinte forma: o Capítulo 2 descreve os conceitos que irão ser abordados neste projeto e trabalhos relacionados; o Capítulo 3 mostra como a solução é organizada e como ela se encaixaria em um processo de desenvolvimento; o Capítulo 4 mostra detalhes de implementação e seu uso; os Capítulos 5, 6 e 7 discutem os resultados obtidos através dos Estudos de Caso e, finalmente, o Capítulo 8 estabelece as conclusões, considerações finais e sugestões de trabalhos futuros e evoluções.

¹Repositório Git contendo os códigos deste projeto: <https://github.com/tenazatto/MsC>

Capítulo 2

Background e Trabalhos Relacionados

Este capítulo fornece uma visão geral dos principais conceitos e dos trabalhos relacionados neste trabalho. Os tópicos que exploram Ciência de Dados e Engenharia de Dados abordarão seus conceitos, semelhanças e diferenças. Os tópicos que exploram *Machine Learning* e *Fairness* abordarão seus conceitos, as métricas utilizadas, algoritmos principais e algoritmos focados para redução de vieses. Os tópicos que exploram Engenharia de Software abordarão arquiteturas e padrões utilizados neste trabalho, como MAPE-K e *Pipes-and-Filters*, e conceitos explorados em aplicações de ML, como AI Reference Architecture e MLOps. Estes conhecimentos são necessários para entender a abordagem que esta dissertação apresenta, propondo uma integração entre Dados, Engenharia de Software e Machine Learning de maneira coesa.

2.1 Ciência de Dados e Engenharia de Dados

2.1.1 Engenharia de Dados

A engenharia de dados é o meio para entender um processo. Os dados podem ser gerados de várias maneiras, ou um subconjunto dos dados disponíveis pode usar técnicas de análise de dados de estatísticas, aprendizado de máquina, reconhecimento de padrões ou redes neurais, juntamente com outras tecnologias, como visualização, otimização, sistemas de banco de dados. Dados, ferramentas de prototipagem e elicitação de conhecimento. O objetivo é usar os dados disponíveis ou gerar mais dados e assim entender o processo que está sendo investigado. O processo de analisar os dados, criar novas ferramentas de análise especificamente para a tarefa e trabalhar com especialistas do domínio é um aspecto fundamental dessa tarefa de engenharia. Atualmente engenharia de dados é muito utilizada em conjunto com o termo *Big Data*, para a limpeza, tratamento e estabelecimento de processos para governança de grandes volumes de dados.

O termo *Big Data* apareceu uma vez como conceito em 1974 e novamente em editoriais em 2006 e 2007, e somente em 2008 seu uso como conceito começou a aparecer regularmente em artigos científicos, mas implementações desse conceito começaram a partir de 2010 [59]. Quanto a engenharia de dados, embora periódicos como o IEEE Transactions on Knowledge and Data Engineering, cuja primeira edição foi lançada em 1989, e conferências como a IEEE International Conference on Data Engineering (ICDE), cuja primeira

edição foi realizada em 1984, possam ser consideradas pontapés iniciais para discussões e artigos acadêmicos, o embrião da engenharia de dados vem do *paper "A Business Intelligence System"*, de 1958 [17]. Nele, é mencionada a ideia de sistemas rodados por máquinas para abstrair e padronizar informações de vários setores da sociedade, como industrial, científico e de organizações governamentais, e como estas podem disseminar informação de uma maneira mais eficiente. Embora os anos 50/60 foram o embrião para o conceito, os anos 70/80 o amadureceram e construíram a base para a estrutura de engenharia de dados [17].

Nos anos 70/80, os problemas em engenharia de dados eram classificados de acordo com 3 atributos [60]:

- **Completude do conhecimento e dos dados:** Os dados e o conhecimento disponíveis no ambiente poderiam ser considerados como *completos* ou *incompletos*. Se estiverem *completos*, não seria necessário conhecimento adicional para a resolução do problema. Se estiverem *incompletos*, como no caso de grandes volumes de dados onde é exigida uma sumarização mais detalhada, seria necessário determinar heurísticas para encontrar um conjunto de dados mais específico para a resolução do problema.
- **Exatidão do conhecimento e dos dados:** Os dados e o conhecimento disponíveis no ambiente poderiam ser considerados como exatos ou inexatos. Se estiverem exatos, poderiam ser representados em uma forma numérica ou lógica, como datas e coordenadas. Se estiverem inexatos, o número de casos possíveis seria infinito e seria impossível enumerar ou representar todos eles, sendo necessário determinar heurísticas para definir um número finito de possibilidades ou redefinir o significado de exatidão para que o que está disponível possa ser tratado como exato, como o reconhecimento de um objeto em uma imagem ou a definição do menor custo em uma determinada rota.
- **Conhecimento sobre o objetivo e especificações do problema:** Um objetivo de um problema pode ser bem ou mal definido. Um objetivo bem definido podia ser medido e representado em termos de parâmetros para que seja possível comparar a qualidade de uma solução com outra, enquanto um objetivo mal definido envolvia parâmetros que não podem ser mensurados ou não poder ser medido, sendo impossível comparar a qualidade de soluções alternativas e necessitando de tratamentos adicionais para que o objetivo deixe de ser mal definido e passe a ser bem definido.

Dado estas categorias, é possível pensar em soluções que conversam com os objetivos da Engenharia de Software: Abordar diversos tipos de conceitos; como teoria, projeto, desenvolvimento, avaliação e manutenção de novos dados e técnicas, metodologias e sistemas de gerenciamento de conhecimento; em prol do desenvolvimento e manutenção de sistemas e soluções com qualidade e com o custo mais viável possível. Estes sistemas podem ter questões relacionadas ao cumprimento desses objetivos incluindo estudos sobre os aspectos teóricos, ferramentas e metodologias de projeto, tradeoffs de projeto, representação e programabilidade, algoritmos e controle, confiabilidade e tolerância a falhas

e projetos usando tecnologias existentes e emergentes. É tarefa do engenheiro de dados elaborar processos que reflitam tais questões em um sistema com objetivos definidos.

2.1.2 Ciência de Dados

Ciência de dados como um conceito precedeu o *Big Data*, sendo um conjunto de princípios fundamentais que apoiam e orientam a extração baseada em princípios de informação e conhecimento de dados [59]. Essa definição enfatiza a estreita relação de ciência de dados com a mineração de dados e, conseqüentemente, com a engenharia de dados. O termo foi cunhado nos anos 60 para descrever uma nova profissão que daria suporte à compreensão e interpretação da grande quantidade de dados que estavam sendo acumulados na época [41]. Na academia, o termo começou a ser utilizado de modo mais formal a partir de 2001, quando os títulos dos artigos científicos começaram a usar, embora artigos já estavam focando em ciência de dados e usando o termo desde os anos 60 [59] [41].

A estatística e o uso de modelos estatísticos estão profundamente enraizados no campo da ciência de dados, pois começou com estatísticas e evoluiu para incluir conceitos e práticas principalmente para aplicações de IA. À medida que mais e mais dados se tornam disponíveis, por meio de comportamentos e tendências registradas em softwares espalhados pela Internet ou mesmo por aplicações empresariais, as empresas os coletam e armazenam em quantidades cada vez maiores. Uma vez que as portas foram abertas por empresas que buscam aumentar os lucros e impulsionar melhores tomadas de decisão, seu uso, em conjunto com *Big Data*, começou a ser aplicado a outros campos, como medicina, engenharia e ciências sociais.

Um cientista de dados funcional, ao contrário de um estatístico geral, tem compreensão da arquitetura de software e entende várias linguagens de programação. O cientista de dados define o problema, identifica as principais fontes de informação e projeta a estrutura para coletar e rastrear os dados necessários. O software é normalmente responsável por coletar, processar e modelar os dados. Ele usa os princípios da ciência de dados e toda sua visão multidisciplinar para obter conhecimentos mais profundos. A ciência de dados continua a evoluir como uma disciplina que usa ciência da computação e metodologia estatística para fazer previsões úteis e obter insights em uma ampla gama de campos. Embora seja usada em áreas como astronomia e medicina, também é usada nos negócios para ajudar a tomar decisões mais inteligentes.

2.1.3 O ciclo do dado: Semelhanças e diferenças entre Engenharia de Dados e Ciência de Dados

Nos tempos atuais, empresas e instituições acadêmicas utilizam Engenharia de Dados e Ciência de Dados para otimizar as suas aplicações de IA. Embora esses dois termos se refiram a áreas distintas, com profissionais que possuem funções distintas, na prática é frequente que cientistas de dados também realizem funções de engenharia de dados e vice-versa por ambas se complementarem na criação de aplicações de IA. Um bom exemplo de como estas áreas se conversam está no *paper Concise Survey of Computer Methods*, de 1974 [17]. Nele, Peter Naur detalha diversos métodos de processamento de dados e

aplicações, o que poderia definir como um, mas a citação mais importante de seu *paper* está justamente no termo ciência de dados. Também define que a utilidade de um dado e de seus processos deriva de sua aplicação no desenvolvimento e manutenção de modelos da realidade [41]. Por isso, a existência de um processo que trate os dados é tão importante quanto a construção de um modelo através de seus dados. Ao entender como os dados devem ser usados, é possível criar processos que auxiliem a otimizar os resultados obtidos e criar modelos cada vez mais próximos da realidade.

A principal razão para ambas as áreas se conversarem está no ciclo de vida do dado, fundamental para entender as oportunidades e os desafios de aproveitar ao máximo os dados digitais. Como um ser vivo, os dados têm um ciclo de vida, desde o nascimento, passando por uma vida ativa até alguma forma de expiração, podendo ser "imortalizado" dependendo de sua importância. Também como um organismo vivo e inteligente, sobrevive em um ambiente que fornece suporte físico, contexto social e significado existencial [25].

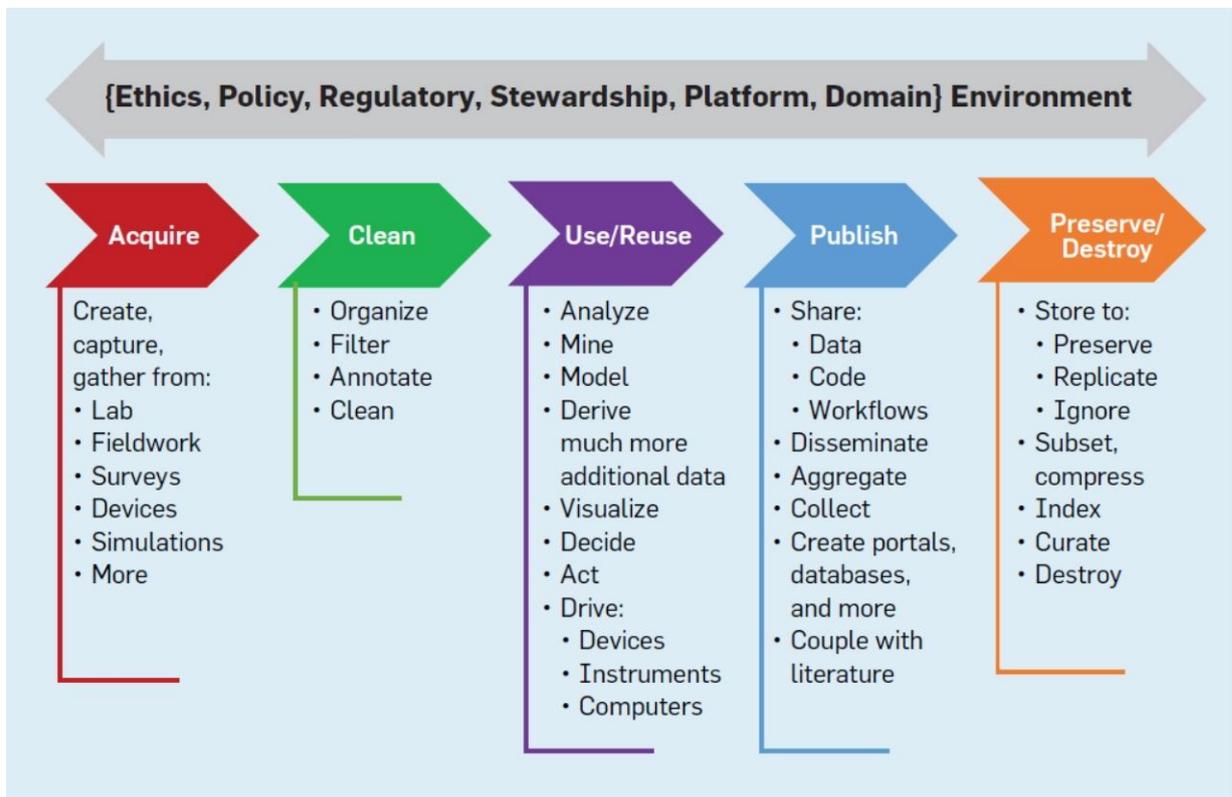


Figura 2.1: Ciclo de vida e ecossistema do dado [25].

Este ciclo, ilustrado na Figura 2.1, é dividido em 5 etapas:

- **Aquisição:** Refere-se a processos de criação e coleta de dados, através de experimentos, sensores, pesquisas, sistemas, simulações, entre outros processos
- **Limpeza:** Refere-se a processos de limpeza, organização e filtragem dos dados adquiridos.

- **Uso/Reuso:** Refere-se a aplicações que os dados podem ter para a aquisição de novos conhecimentos, como análise, mineração, modelagem, enriquecimento, síntese de novos dados, visualização e tomadas de decisão.
- **Publicação:** Refere-se ao compartilhamento destes dados após seu uso, podendo ter como meio alguma plataforma de compartilhamento, bases de dados ou artigos acadêmicos.
- **Preservação/Destruição:** Refere-se a processos de armazenamento, curadoria e validação do dado após seu uso e publicação. No contexto de validação, se o mesmo ainda continua útil para o contexto em que foi coletado, podendo ser atualizado ou destruído em caso negativo. Também pode ser comprimido ou indexado caso espaço ou desempenho sejam considerados gargalos para os usuários que forem usar os dados publicados.

Como exemplo, é possível citar os dados para experimentos do Grande Colisor de Hádrons (LHC), representando colisões de partículas dentro de um túnel com 27 km de circunferência para testar as previsões de várias teorias da física de partículas e da física de alta energia [25]. A maioria dos dados gerados é tecnicamente irrelevante e são descartados, mas isso não impede que uma enorme quantidade de dados seja influente e continue a ser analisada e preservada. Em 2012, dados sobre experimentos do LHC forneceram fortes evidências para o bóson de Higgs, apoiando a veracidade do Modelo Padrão da Física. Esta descoberta científica foi a "Descoberta do Ano" de 2012 da revista Science e o Prêmio Nobel de Física em 2013.

As estimativas são de que, em 2040, haverá de 10 exabytes a 100 exabytes de dados influentes produzidos pelo LHC. Os dados retidos do LHC são anotados, preparados para preservação e arquivados em mais de uma dúzia de locais físicos. O resultado desse processo é divulgado à comunidade para análise e uso em mais de 100 outros locais de pesquisa. Além do desenvolvimento de protocolos de administração, disseminação e uso de dados, o ecossistema de dados do LHC também fornece um modelo econômico que suporta de forma sustentável os dados e sua infraestrutura. Essa combinação entre administração dos dados e administração econômica permitem que os dados sejam mantidos.

O diagrama do ciclo de vida dos dados descrito na figura e o exemplo do LHC sugerem um conjunto contínuo de ações e transformações nos dados, mas em muitas comunidades científicas e disciplinas hoje essas etapas são isoladas. Os cientistas de domínio se concentram em gerar e usar dados. Cientistas da computação podem se concentrar em questões de plataforma e desempenho, incluindo mineração, organização, modelagem e visualização, bem como os mecanismos para extrair significado dos dados por meio de aprendizado de máquina e outras abordagens. Estatísticos podem se concentrar na matemática dos modelos de risco e inferência [25]. Engenheiros de dados podem se concentrar na administração e preservação de dados gerados pelo cientista de domínio e no *backend* do pipeline, seguindo a aquisição, decisões e ações no domínio da publicação, arquivamento e curadoria. Cientistas de dados podem unir o trabalho dos cientistas da computação e dos estatísticos para extrair novos conhecimentos e conhecer tomadas de decisão mais eficientes.

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. Então, 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 a ela a habilidade de aprender como executar a tarefa” [14]. Com isso, o computador consegue a habilidade de realizar determinado cálculo ou tarefa sem que necessite de programação adicional ou interferência humana para isso.

O Aprendizado de Máquina é fortemente relacionado com a Estatística, uma vez que seus métodos e parte de seus algoritmos, como regressões, tiveram como base modelos estatísticos e a análise de seus dados. As tarefas de aprendizado podem ser classificadas em três categorias básicas [3] [7]:

- **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 valores de rótulos em dados que não estão rotulados. 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.

O termo se tornou muito mais evidente com a possibilidade da implementação do *Deep Learning*, que é uma técnica que utiliza Redes Neurais Artificiais para atingir seus resultados. Redes Neurais Artificiais são modelos computacionais inspirados no sistema nervoso do cérebro, onde temos neurônios divididos em camadas e conectados entre si, podendo ser abstraído conforme ilustração na Figura 2.2. Dependendo da tarefa a ser realizada, cada neurônio atribui uma ponderação para os dados que entram e a saída final é determinada pelo total dessas ponderações [14]. As redes neurais utilizadas em *Deep*

Learning possuem, ao menos, duas camadas de neurônios entre a camada que recebe os dados de entrada e a camada final que faz o tratamento final dos dados de saída.

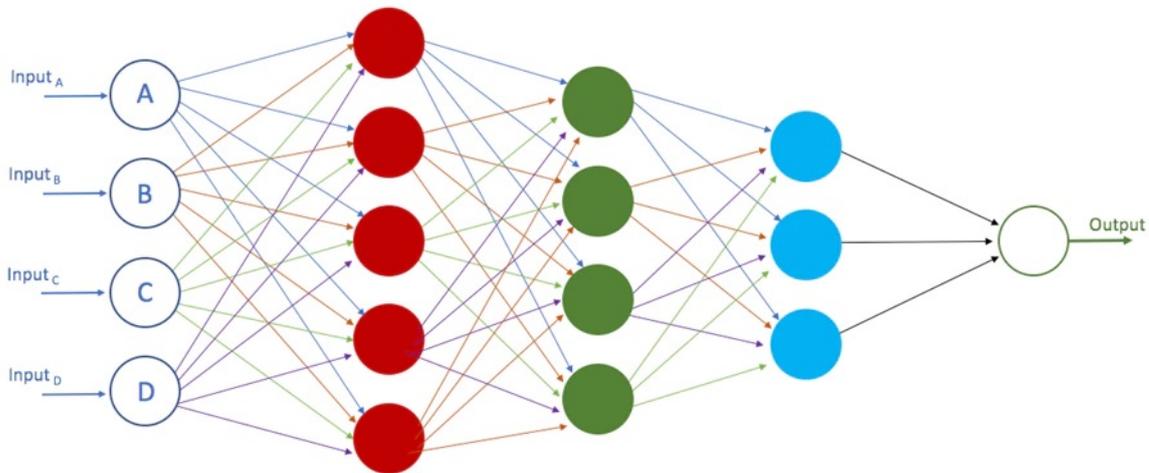


Figura 2.2: Exemplo de uma rede neural utilizada em *Deep Learning*.

Com a evolução da computação, o treino de uma tarefa passou a ser cada vez mais viável, uma vez que a execução de algoritmos de Aprendizado de Máquina é computacionalmente muito custosa, especialmente quando redes neurais são utilizadas. Sua viabilidade também se prova quando tais tarefas se tornam efetivas no mundo real: Como exemplo, reconhecimento de imagens por máquinas treinadas através de deep learning em alguns cenários possuem uma taxa de acerto maior que a de humanos [14].

2.2.1 Métricas de Avaliação

Tradicionalmente em um problema de classificação binária, uma previsão do classificador pode ter 4 tipos de resultados em uma matriz de confusão, presente na Tabela 2.2.1. Os Verdadeiros Positivos (VP, ou TP pelo termo em inglês *True Positives*) e Verdadeiros Negativos (VN, ou TN pelo termo em inglês *True Negatives*) são classificações corretamente previstas pelo classificador. Um Falso Positivo (FP) ocorre quando dados previstos para estar na classe positiva pertencem à classe negativa em seu resultado real, e um Falso Negativo (FN) ocorre quando dados previstos para estar na classe negativa pertencem à classe positiva em seu resultado real.

A taxa de sucesso, também conhecida como **acurácia**, é o número de previsões corretas dividido pelo número total de resultados:

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \quad (2.1)$$

Ela é considerada um indicador da realização de um bom treinamento e de um bom funcionamento do modelo obtido. Entretanto, não é a melhor métrica para interpretar situações onde a qualidade dos dados utilizados interfere no resultado final, como classes desequilibradas.

Tabela 2.1: Matriz de confusão

		Classe prevista	
		Positiva	Negativa
Classe atual	Positiva	Verdadeiros Positivos (VP/TP)	Falsos Positivos (FP)
	Negativa	Falso Negativo (FN)	Verdadeiros Negativos (VN/TN)

A **precisão** enfatiza situações quando o número de falsos positivos é alto e mostra a precisão das previsões positivas. É a fração de casos positivos previstos corretamente para estar na classe positiva de todos os casos positivos previstos no modelo:

$$P = \frac{TP}{TP + FP} \quad (2.2)$$

O **recall**, ou sensibilidade, enfatiza situações quando o número de falsos negativos é alto. É a fração de casos positivos previstos corretamente para estar na classe positiva de todos os casos positivos reais:

$$R = \frac{TP}{TP + FN} \quad (2.3)$$

O **F1-score** é um caso especial de F-score, uma medida geral da acurácia de um modelo que combina precisão e recall. Para tal, usa-se a média harmônica entre ambos para realizar a medição:

$$F1 = 2 \times \frac{P \times R}{P + R} \quad (2.4)$$

O **AUC** (*Area under the ROC Curve*), também chamada de precisão balanceada, pode interpretar a capacidade do classificador de evitar falsos positivos e falsos negativos. Se sai melhor que a acurácia em situações que ela não é apropriada, como o desequilíbrio entre classes já citado anteriormente:

$$AUC = \frac{1}{2} \left(\frac{TP}{TP + FN} + \frac{TN}{TN + FP} \right) \quad (2.5)$$

2.2.2 Tipos de Algoritmos de Machine Learning

Existem diversos algoritmos de *Machine Learning* utilizados para resolver os problemas de aprendizado supervisionado, não-supervisionado e por reforço. Nesta seção serão abordados apenas os algoritmos utilizados por este trabalho de Mestrado.

Regressão Logística

A Regressão Logística é um método de classificação baseado na aplicação da técnica de Regressão Linear. Como na Regressão Linear, é assumida uma relação linear entre os recursos e calcula a soma ponderada dos recursos mais um termo de viés. Neste método, o resultado não é utilizado diretamente, mas calcula a logística do resultado. Quando peso das *features* é w e o termo de viés é b , a probabilidade estimada do modelo de regressão logística é a seguinte:

$$\hat{p} = \sigma(w^T x + b) \quad (2.6)$$

Logo após, a função logística σ é calculada:

$$\sigma(z) = \frac{1}{1 + \exp(-z)} \quad (2.7)$$

Uma vez que o modelo de Regressão Logística estimou a probabilidade \hat{p} , a previsão de classificação é feita considerando a seguinte regra:

$$\hat{p} = \begin{cases} 1 & \text{Se } \hat{p} \geq 0,5 \\ 0 & \text{Se } \hat{p} < 0,5 \end{cases} \quad (2.8)$$

O objetivo do treinamento de um modelo de classificação é de minimizar a diferença entre o resultado real y e o resultado previsto \hat{y} para as m amostras presentes para treinamento. Para medir o quão próximo a função resultante do treinamento se aproxima do conjunto de dados, a seguinte função de custo é calculada conforme a equação abaixo. A função $L(\hat{y}, y)$ utilizada pode variar, tendo como exemplos a função por entropia cruzada ou a função por diferença de quadrados.

$$J(w, b) = \frac{1}{m} \sum_{i=1}^m L(\hat{y}^{(i)}, y^{(i)}) \quad (2.9)$$

Durante o treinamento do modelo de Regressão Logística, é preciso encontrar os parâmetros w e b que minimizem a função de custos globais. Como a função de custo é convexa, diferentes métodos de otimização, como o gradiente descendente, garantem encontrar o mínimo global.

Para lidar com problemas com múltiplas classes para classificação, é possível calcular uma função de custo separada para cada rótulo de classe por observação e somar os resultados, técnica conhecida como *One-Vs-All* [61]. Outra técnica que pode generalizar o método de regressão logística para suportar várias classes diretamente é chamada de Regressão Softmax, ou Regressão Logística Multinomial, utilizando a função Softmax para substituir a função de probabilidade da Regressão Logística convencional:

$$\hat{p}(Y_i = n) = \frac{e^{\beta_n \cdot \mathbf{X}_i}}{1 + \sum_{k=1}^{K-1} e^{\beta_k \cdot \mathbf{X}_i}} \quad (2.10)$$

Support Vector Machines (SVM)

Support Vector Machines (SVM) é um algoritmo de aprendizado de máquina que pode ser usado para detecção linear, não linear, de classificação, regressão e até mesmo detecção de anomalias (*Outliers*). A ideia fundamental por trás dessa técnica é que as classes de saída são separadas com um hiperplano maximizando a margem (distância máxima entre os pontos de dados de ambas as classes) [70]. As *Support Vector Machines* criam um ou vários hiperplanos em um espaço de n dimensões. A dimensão dos hiperplanos depende do número de *features*. Quando há duas *features*, é apenas uma linha, ou quando há três *features*, é um plano bidimensional.

Para lidar com conjuntos de dados não lineares, uma abordagem é adicionar mais *features*, como um recurso polinomial, ou a outra abordagem é usar *kernels*, mapeando os dados de entrada de n dimensões para um espaço de dimensão superior, onde os dados podem ser separados linearmente.

Regressão Kernel Ridge

A Regressão Kernel Ridge (*Kernel Ridge Regression/KRR*) é um algoritmo de aprendizado de máquina proveniente da combinação de duas operações: A Regressão Ridge com o que é conhecido como "truque do kernel" [74]. A Regressão Ridge substitui a função de custo tradicional por uma com um termo de penalização incluído, conforme ilustrado na Equação 2.11, utilizando o método dos mínimos quadrados. O "truque do kernel" é uma técnica matemática que permite exemplificar problemas não-lineares de forma linear utilizando kernels, reduzindo a complexidade para uma simples operação matricial.

$$\sum_i (y_i - w^T x_i)^2 - \lambda \|w\|^2 \quad (2.11)$$

Por causa de tais operações, A Regressão Kernel Ridge exige mais processamento do que uma regressão tradicional. No entanto, é vantajoso usá-la em casos que um ajuste não-linear é desejado ou onde há mais atributos do que elementos no conjunto de treinamento. Em casos onde há mais elementos no conjunto de treinamento do que atributos, a Regressão Kernel Ridge peca por não abranger o conceito utilizado nas *Support Vector Machines*, onde é necessário somar apenas o conjunto de vetores de suporte ao invés de todo o conjunto de treinamento. No entanto, por causa deste mesmo conceito as *Support Vector Machines* exigem mais processamento.

Árvores de Decisão (*Decision Trees*)

Árvores de decisão (*Decision Trees*) são um grupo de algoritmos de aprendizado de máquina que podem ser usados para classificação e regressão. Eles são os métodos de aprendizado mais comuns que são muito poderosos e capazes de ajustar conjuntos de dados complexos. Os algoritmos de Árvore de Decisão são baseados em uma abordagem de dividir e conquistar para os problemas de classificação [74]. Uma Árvore de Decisão é feita pelo processo contínuo de dividir o conjunto de dados nos atributos da melhor maneira possível em diferentes classes até que um critério de parada específico seja alcançado. Nas

Árvores de Decisão, as observações sobre os itens são mostradas em ramificações e as conclusões das observações são mostradas nos nós. Existem três tipos diferentes de nós: os nós raiz que indicam o início do processo de decisão e não têm arestas de entrada, os nós internos que têm exatamente uma entrada e pelo menos duas arestas de saída e os nós finais (ou as folhas).

Sempre que o rótulo de classificação de destino assume valores discretos, a Árvore de Decisão é chamada de árvore de classificação e, sempre que recebe valores contínuos, é chamada de árvore de regressão. Um dos méritos do uso de Árvores de Decisão é que elas exigem pouco pré-processamento de dados e não há necessidade de escala. Seu treinamento geralmente é realizado com uma árvore menos complicada e mais abrangente. A complexidade de uma Árvore de Decisão pode ser controlada usando critérios de parada e métodos de poda [62], existindo quatro métricas diferentes para poder medi-la: o número total de nós, o número total de folhas, a profundidade da árvore e o número de atributos usados.

Os algoritmos usados para construir uma Árvore de Decisão a partir de um conjunto de dados são chamados de indutores. Normalmente, o objetivo desses algoritmos é encontrar a Árvore de Decisão ótima minimizando o erro de generalização, considerando o número mínimo de nós e a profundidade mínima da árvore. Os algoritmos da Árvore de Decisão funcionam de forma *top-down*, pois escolhem a melhor variável em cada estágio que pode dividir o conjunto de dados em um atributo específico. Diferentes indutores usam critérios diferentes para encontrar a melhor variável.

Random Forest

Random Forest é um algoritmo de aprendizado de máquina que combina a simplicidade das Árvores de Decisão com a flexibilidade, resultando em melhorias na acurácia [28]. Neste algoritmo, o *Bagging*, uma técnica que gera uma coleção de classificadores introduzindo subconjuntos randômicos na entrada do algoritmo [74], é utilizado para gerar uma coleção de Árvores de Decisão simplificadas com a finalidade de generalizar o resultado no conjunto da obra.

No final, a classe selecionada é a que for mais citada dentre essa coleção. Esta simplificação, em geral, otimiza o treinamento em relação a uma Árvore de Decisão tradicional.

Gradient Boosting

Gradient Boosting, também conhecido como *Gradient Boosting Machine* (GBM) ou *Gradient Boosted Regression Tree* (GBRT) [33], é um algoritmo de aprendizado de máquina que faz a classificação através da composição de pequenos modelos pela utilização do *Boosting* [42] [16]. *Gradient Boosting* faz uso de *Boosting* para gradualmente aproximar um melhor modelo, de modo a somar submodelos ao modelo composto. Árvores de Decisão tendem a gerar *overfitting*, e o *Gradient Boosting* é uma possibilidade para solucionar este problema.

Boosting é uma combinação de modelos simples, chamados de modelos fracos, onde tipicamente estes modelos são Árvores de Decisão. O algoritmo combina classificadores fracos com intuito de produzir um classificador forte. Diferente do *Bagging*, a obtenção

de subconjuntos vindos do conjunto de dados não é feita de maneira aleatória, e sim feita priorizando a substituição de dados mal classificados, que ganham um peso maior para serem substituídos nas iterações seguintes [16].

Redes Neurais Artificiais

As Redes Neurais Artificiais, muitas vezes chamadas apenas de Redes Neurais, são um grupo de métodos de modelagem de dados estatísticos não lineares, que a princípio foram inspirados nos cérebros e nas estruturas dos neurônios biológicos. Elas são a base do aprendizado profundo, sendo escaláveis e capazes de trabalhar com grandes tarefas complexas de aprendizado de máquina, como serviços de reconhecimento de imagem e reconhecimento de fala. O processo simplificado para treinamento de uma Rede Neural é o seguinte:

1. Os dados de entrada são fornecidos à rede, eles se propagam pelas camadas e o processo de encaminhamento produz a previsão.
2. Calcula-se o erro entre o produto previsto e o produto real (função de custo).
3. A Rede Neural usa um algoritmo de otimização para ajustar os pesos de forma a reduzir a função custo.
4. O processo de encaminhamento inicia novamente e continua até que a taxa de erro seja minimizada a um determinado valor, ou até um certo número de iterações.

As Redes Neurais Artificiais não são novas para os sistemas de computação, pois foram introduzidas pela primeira vez por Warren McCulloch e Walter Pitts em 1943 [54]. Desde então, o desenvolvimento dessas técnicas passou por altos e baixos até ter uma adoção considerável graças aos avanços significativos tanto no poder computacional dos hardwares quanto nas otimizações de algoritmos implementadas nos softwares. Atualmente, é possível treinar Redes Neurais de grande complexidade, e há uma enorme quantidade de dados disponíveis para uso em bancos de dados. Elas podem ser divididas em dois grupos principais [68]:

- **Redes retroalimentadas (*Feedforward*):** Nessas redes, o fluxo de dados se move apenas na direção direta da camada de entrada para os nós de saída. Não há alimentação ou loop no sistema.
- **Redes recorrentes:** Este tipo de rede pode ter a opção de feedback e reutiliza os dados dos estágios posteriores para os estágios anteriores.

Algoritmos de otimização são métodos usados para minimizar o valor da função de custo ajustando os parâmetros internos do modelo. Algumas das técnicas de otimização mais comuns nas estruturas de aprendizado profundo são as seguintes: *Stochastic Gradient Descent* (SGD) [65], *Momentum* [58], *Nesterov Accelerated Gradient* (NAG) [71], *Adaptive Gradient* (AdaGrad) [36], *Root mean square prop* (RMSprop) [45], *Adaptive moment estimation*, (Adam) [52], *Nesterov and Adam optimizer* (Nadam) [35].

Toda Rede Neural consiste em alguns nós (neurônios), conexões ponderadas entre os nós e uma abordagem computacional chamada função de ativação usada para definir a saída de cada neurônio. Diferentes tipos de funções de ativação podem ser usados com esta técnica, como sigmóide, tanh (tangente hiperbólica) ou ReLU (sigla de *Rectified Linear Unit*).

Redes Neurais consistem em diferentes camadas. O tipo mais simples de Rede Neural inclui uma camada de entrada que recebe informações de fontes externas, como valores de atributos do conjunto de dados de entrada. A camada de saída gera a saída da rede e as camadas ocultas conectam a camada de entrada e a camada de saída entre si. O valor de entrada de cada nó em cada camada é calculado pela soma de todos os nós de entrada multiplicada pelo respectivo peso da interconexão entre os nós [39].

2.3 *Fairness* em Algoritmos de Aprendizado de Máquina

Como a coleta de dados está presente atualmente no dia-a-dia de variados setores da sociedade, o uso de Aprendizado de Máquina é extremamente versátil para tomadas de decisão, podendo ser utilizados em problemas como admissão de universidades, contratações, análise de crédito e reconhecimento de doenças. Com o aumento dessa influência, 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 [55].

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 Aprendizado de Máquina, determinarão como o algoritmo se comportará e, conseqüentemente, afetará suas métricas [56]. 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 [24]. 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.

Geralmente, as descrições de justiça se dividem em dois grupos principais: justiça individual e justiça de grupo. O objetivo da justiça individual é que indivíduos semelhantes devem obter resultados semelhantes, enquanto na justiça de grupo, cada um dos grupos

definidos pelo atributo protegido devem ser tratados igualmente. No geral, os estudos atuais costumam realizar seus experimentos em casos de justiça de grupo, uma vez que o escopo de justiça de grupo é muito mais amplo e tende a exemplificar melhor a relação entre dados, relações sociais e vieses do mundo atual.

2.3.1 Métricas de *Fairness*

Para avaliar a justiça de um modelo, as métricas utilizadas diferem das métricas utilizadas para avaliação do modelo, que possuem o propósito de verificar se um modelo tem previsões confiáveis ou não. As métricas de *Fairness* possuem um propósito diferente, pois verificam os dados de forma mais intimista. Elas não medem o modelo como um todo, mas o quanto os grupos e registros avaliados estão próximos dos outros. 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.

Assim como algumas métricas utilizadas para avaliação, muitas métricas utilizam Verdadeiros Positivos, Verdadeiros Negativos, Falsos positivos e Falsos Negativos para analisar o quão justo o modelo é. Entretanto, diferente da acurácia, precisão e recall utilizados anteriormente, a medição das discriminações utiliza outras métricas, utilizadas ou não para avaliar a performance, para estabelecer novas métricas mais adequadas para a sua finalidade.

Exemplos de métricas utilizadas para isso são a Taxa de Verdadeiros Positivos e a Taxa de Falsos Positivos. Enquanto a **Taxa de Verdadeiros Positivos** (ou TPR, pelo termo em inglês **True Positive Rate**) é outro termo para denominar o recall, a **Taxa de Falsos Positivos** (ou FPR, pelo termo em inglês **False Positive Rate**) é a fração de casos negativos previstos incorretamente como estando na classe positiva de todos os casos positivos reais:

$$FPR = \frac{FP}{FP + TN} \quad (2.12)$$

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 [75]:** Esta métrica determina a diferença de chances entre o grupo privilegiado e o grupo não-privilegiado de receber uma previsão favorável. É baseada na seguinte fórmula:

$$Pr(Y = 1|Z = 0) - Pr(Y = 1|Z = 1) \quad (2.13)$$

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*) [26]:** Esta métrica determina a diferença de chances entre o grupo privilegiado e o grupo não-privilegiado de receber uma previsão favorável que corresponda a realidade. É a diferença entre a taxa de verdadeiros positivos do grupo não-privilegiado e a taxa de verdadeiros positivos do grupo privilegiado:

$$TPR_{Z=0} - TPR_{Z=1} \quad (2.14)$$

Um valor próximo de 0 é considerado justo. Um classificador binário satisfaz a igualdade de oportunidades quando a taxa de verdadeiros positivos de ambos os grupos são iguais [46]

- **Diferença de probabilidade média (*Average odds difference*) [26]:** Esta métrica determina a diferença de previsões incorretas entre o grupo privilegiado e o grupo não-privilegiado. Ela usa a taxa de falsos positivos e a taxa de verdadeiros positivos para calcular a tendência, calculando a igualdade de probabilidades com a fórmula:

$$\frac{1}{2}(|FPR_{Z=0} - FPR_{Z=1}| + |TPR_{Z=0} - TPR_{Z=1}|) \quad (2.15)$$

O valor precisa ser próximo a 0 para ser considerado justo.

- **Impacto de disparidade (*Disparate impact*) [26]:** Para esta métrica, é usada a seguinte fórmula:

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

Usa as probabilidades condicionais utilizadas na diferença de paridade estatística, mas como é calculada como uma proporção o valor esta métrica possui um comportamento não-linear. Desta forma, um valor próximo de 1 é considerado justo. Um valor que não esteja no intervalo $[0, 8; 1, 25]$ indica um grande desbalanceamento do modelo treinado [40]

- **Índice de Theil (*Theil index*) [69]:** Esta métrica é utilizada não apenas para determinar *Fairness* entre o grupo privilegiado e o grupo não-privilegiado, mas também para determinar *Fairness* entre indivíduos. Em outras palavras, que indivíduos similares recebam previsões similares. Também é conhecida como índice de entropia generalizado, mas com o valor de α usado neste índice igual a 1 [69]. É calculado com a seguinte fórmula:

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

Onde n é a quantidade de elementos, b_i é obtido com a fórmula $b_i = \hat{y}_i - y_i + 1$ e μ é a média de todos os elementos b_i . Nesta fórmula, y_i é o conjunto de saídas, \hat{y}_i é

o conjunto de previsões dadas pelo modelo. Também precisa ser próximo a 0 para ser considerado justo.

2.3.2 Algoritmos para redução de vieses

Há diversos tipos de algoritmos diferentes na inteligência artificial para redução de vieses, a fim de garantir *Fairness* nos projetos de Aprendizado de Máquina. É possível classificá-los em três categorias diferentes: Algoritmos de pré-processamento, de processamento e de pós-processamento.

Os algoritmos de **pré-processamento** tentam eliminar a discriminação transformando os dados, antes de executar o algoritmo de treinamento. Tais algoritmos podem ser usados caso seja permitida a modificação dos dados de treinamento [34], e a ideia por trás de tais algoritmos é que suas previsões serão mais balanceadas se o classificador for treinado com os dados já balanceados [30]. Nesta categoria se enquadram os seguintes algoritmos:

- **Reposição (*Reweighting*) [47]:** Pondera os exemplos em cada combinação de grupo e rótulo de maneira diferente para garantir a justiça antes da classificação.
- **Removedor de impacto de disparidade (*Disparate impact remover*) [40]:** Edita valores de *features* para otimizar *Fairness* em cada grupo enquanto preserva a ordem de classificação dentro dos mesmos.
- **Aprendizado de representações justas (LFR, ou *Learning fair representations*) [75]:** Encontra uma representação latente que codifica os dados, mas ofusca informações dos atributos protegidos.
- **Pré-processamento otimizado (*Optimized pre-processing*) [31]:** Aprende uma transformação probabilística que edita *features* e rótulos nos dados abordando primariamente *Fairness* em cada grupo, e garante a fidelidade de dados através de restrições para controle de distorção.

Os algoritmos de **processamento** tentam realizar modificações nos algoritmos de treinamento para mitigar a discriminação durante o processo de treinamento do modelo. Se for permitido fazer mudanças no processo de treinamento, então os algoritmos podem ser usados incorporando mudanças na função de custo ou impondo restrições [55]. Nesta categoria se enquadram os seguintes algoritmos:

- **Remoção de viés adversário (*Adversarial debiasing*) [76]:** Aprende um classificador que maximiza a precisão e reduz a capacidade de um adversário de depender do atributo protegido nas previsões. Essa abordagem leva a um classificador justo, pois as previsões realizadas pelo classificador não possuem nenhuma informação de discriminação nos grupos.
- **Removedor de preconceito (*Prejudice remover*) [40]:** Técnica que adiciona no algoritmo escolhido um termo de regularização baseado na discriminação.

- **Meta-Algoritmo para classificações justas (Meta-Algorithm for Fair Classification) [32]:** Aprende um classificador compatível com uma gama grande de métricas de Fairness, sendo prático o suficiente para abrangê-las sem grande perda de performance.
- **Justiça de Subgrupo Diversificado (Rich Subgroup Fairness) [49]:** Aprende um classificador que procura equalizar as taxas de falsos positivos e falsos negativos entre os dados que envolvem atributos protegidos, considerados como subgrupos.
- **Redução por Gradiente Exponencial (Exponentiated Gradient Reduction) [22]:** Aprende um classificador baseado em Gradiente Exponencial que tende a minimizar o erro de uma classificação ponderada.
- **Redução por busca em grade (Grid Search Reduction) [22] [23]:** Aprende um classificador baseado na busca em uma grade de valores que tende a minimizar o erro de uma classificação ponderada. É mais simples e impreciso que a Redução por Gradiente Exponencial, mas sua escolha pode ser razoável se a quantidade de métricas de Fairness a serem consideradas for pequena.

Os algoritmos de **pós-processamento** utilizam um conjunto de validação, que não foi envolvido no processo de treinamento para melhorar a imparcialidade das previsões [34]. Quando não há possibilidade de fazer alterações nos dados de treinamento ou no treinamento do modelo, apenas algoritmos de pós-processamento podem ser usados. Nesta categoria se enquadram os seguintes algoritmos:

- **Pós-processamento para igualdade de probabilidade (Equalized Odds Post-processing) [46]:** Resolve um problema linear cuja saída é um conjunto de probabilidades e altera os rótulos de saída entre o grupo privilegiado e o grupo não-privilegiado para equalizar estas probabilidades.
- **Pós-processamento calibrado para igualdade de probabilidade (Calibrated Equalized Odds Postprocessing) [57]:** Otimiza um conjunto de probabilidades através de uma calibração das previsões do classificador obtido e altera os rótulos de saída entre o grupo privilegiado e o grupo não-privilegiado para equalizar estas probabilidades.
- **Classificação baseada em Rejeição de Opções (Reject Option-based Classification) [48]:** Dá resultados favoráveis para grupos não privilegiados e resultados desfavoráveis para grupos privilegiados de acordo com uma faixa de confiança.

2.4 Engenharia de Software para Aplicações de ML

Quando se fala de Arquitetura e Engenharia de Software, se fala da definição dos componentes de software, suas propriedades externas, e seus relacionamentos com outros softwares para fazer com que um sistema seja documentável, reusável e testável. A preocupação está em como um sistema deve ser organizado e com a estrutura geral desse

sistema. Dado as definições sobre IA já detalhadas, é possível encaixar Aprendizado de Máquina na forma de um processo bem definido, de forma que é possível sistematizar todo esse processo na forma de componentes e definir formas em que o modelo resultante deste processo é disponibilizado para aplicações externas.

No processo, ilustrado na Figura 2.3, o conjunto de dados passa por um pré-processamento e é dividido em dois conjuntos, um para treinamento e outro para teste. O conjunto de treino é utilizado para o algoritmo realizar o processo de treinamento, obtendo um modelo após o término desse processo. O conjunto de testes é utilizado para mensurar se o modelo obtido no processo de treinamento realiza previsões confiáveis ou não.

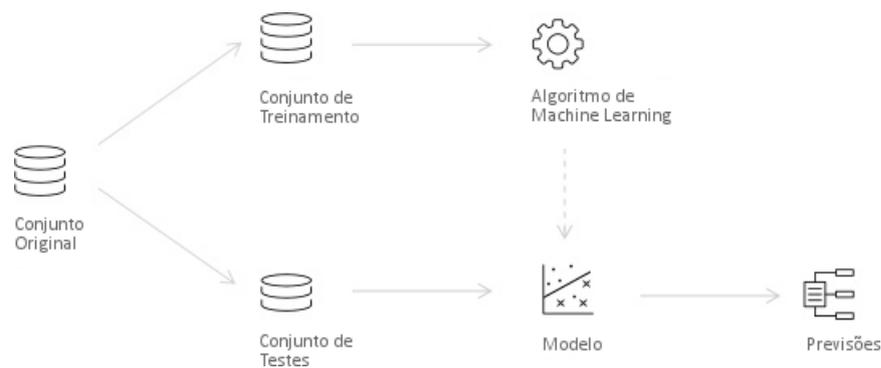


Figura 2.3: Processo padrão para aprendizado de máquina

2.4.1 Arquitetura de Software

O consenso sobre a definição do termo arquitetura de software foi atingido com a adoção do padrão IEEE 1471, que define arquitetura de software como *"a organização fundamental de um sistema incorporado em seus componentes, seus relacionamentos entre si e com o meio ambiente e os princípios que orientam seu projeto e evolução"*. Com esta definição, o componente e o conector são reforçados como conceitos centrais da arquitetura de software [27].

O nível de projeto da arquitetura de software no desenvolvimento de uma aplicação vai além dos algoritmos e estruturas de dados da computação. Incluem fatores como organização, protocolos de comunicação, acesso a dados, atribuição de funcionalidades, escalabilidade, performance, composição e seleção do *design* ideal [43]. É possível tratar uma arquitetura de um sistema específico como uma coleção de componentes juntamente com uma descrição dos conectores, que define as interações entre os componentes.

2.4.2 Arquitetura de referência para IA

Um estilo de arquitetura define uma família de tais sistemas em termos de um padrão de organização estrutural, e determina o vocabulário de componentes e conectores que podem ser usados em instâncias desse estilo, juntamente com um conjunto de restrições sobre como eles podem ser combinados [43]. A decisão sobre tal estilo depende da solução e dos requisitos de um sistema. Ela pode adicionar novos componentes, incrementá-los com novos requisitos ou adicionar restrições sobre eles [27].

Um exemplo de arquitetura que generaliza todo o processo, desde a necessidade de negócio até a implantação do modelo de IA, é a IBM Analytics and AI Reference Architecture [5], ilustrada na Figura 2.4. Nela, são definidos os seguintes requisitos não-funcionais: performance, estabilidade, segurança, escalabilidade, manutenibilidade e regulamentações de privacidade/*compliance*, e pode ser classificada em 4 grupos principais envolvendo diversos tipos de processos e componentes:

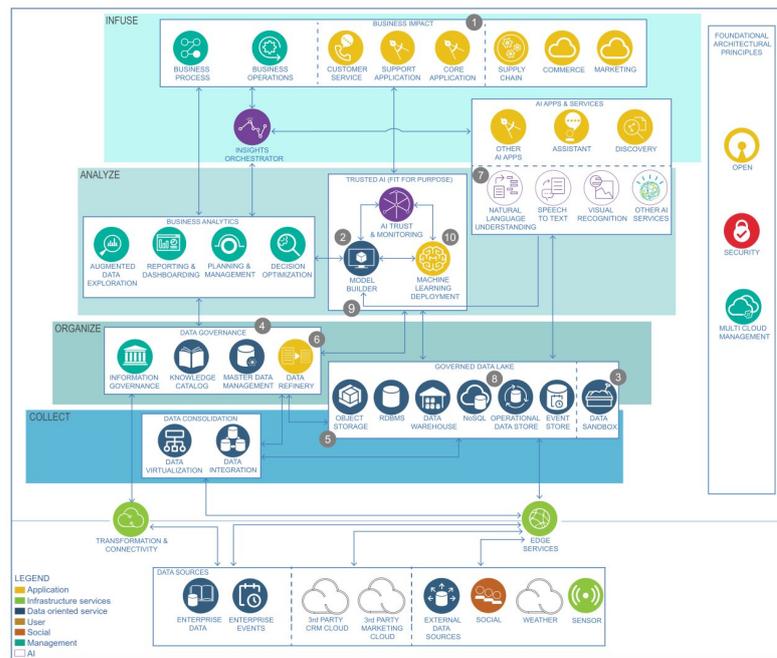


Figura 2.4: IBM Analytics and AI Reference Architecture [5].

- **Coleta:** Relaciona os processos de coleta, armazenamento e transformação de diversas fontes de dados, estruturadas ou não estruturadas, para determinados repositórios (*Data Lakes*).
- **Organização:** Relaciona os processos de organização e estruturação dos dados nos presentes nos *Data Lakes* e necessários para o uso das aplicações que envolvem a análise dos dados. Dependendo do uso pode-se aplicar processos de Governança.
- **Análise:** Relaciona os processos para desenvolvimento de aplicações de IA e relatórios após a organização dos dados para tomadas de decisão e uso de aplicações externas.
- **Infusão:** Relaciona os processos de disponibilização desses dados e conhecimento obtidos na fase de análise para aplicações externas.

2.4.3 Arquitetura MAPE-K

Em 2001, Paul Horn introduziu o conceito de Computação Autônoma como possível 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 [51].

Em um ambiente autônomo e autogerenciado, os componentes de sistema podem incorporar como funcionalidade um *loop* de controle, que podem ser categorizados em 4 categorias principais. Essas categorias são consideradas atributos dos componentes do sistema e são definidas como [15]:

- **Auto-configuração:** Pode se adaptar dinamicamente a mudanças no ambiente. Um componente autoconfigurável realiza esta adaptação usando políticas fornecidas pelo profissional. Tais mudanças podem incluir a implantação de novos componentes ou a remoção dos existentes, ou mudanças drásticas nas características do sistema. A adaptação dinâmica ajuda a garantir força e produtividade contínuas da infraestrutura, resultando em crescimento e flexibilidade dos negócios.
- **Auto-cura:** Pode descobrir, diagnosticar e reagir a interrupções. Um componente auto-curável pode detectar falhas no sistema e iniciar ações corretivas baseadas em políticas sem interromper o ambiente. A ação corretiva pode envolver um produto alterando seu próprio estado ou efetuando mudanças em outros componentes do ambiente. Com isso, o sistema se torna mais resiliente porque as operações cotidianas possuem menos probabilidade de falhar.
- **Auto-otimização:** Pode monitorar e ajustar recursos automaticamente. Um componente auto-otimizável pode se ajustar para atender às necessidades do usuário. As ações de ajuste podem significar realocar recursos para melhorar a utilização geral, como em resposta a cargas de trabalho que mudam dinamicamente, ou garantir que processamentos possam ser concluídos em tempo hábil. A auto-otimização ajuda a fornecer um alto padrão de serviço para quem vai utilizar o sistema. Sem funções de auto-otimização, não há uma maneira fácil de re-escalonar os recursos de infraestrutura quando um aplicativo não os usa totalmente.
- **Auto-proteção:** Pode antecipar, detectar, identificar e proteger contra ameaças de qualquer lugar. Um componente de autoproteção pode detectar comportamentos hostis à medida que ocorrem e tomar ações corretivas para se tornarem menos vulneráveis. Os comportamentos hostis podem incluir acesso e uso não autorizados, infecção e proliferação de vírus e ataques de negação de serviço. Os recursos de autoproteção permitem que as empresas apliquem consistentemente políticas de segurança e privacidade.

Para a Computação Autônoma acontecer, é implementado um Elemento Autônomo [21], 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 2.5. 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 um controlador de uma linha de produção, um software, como um gerenciador de banco de dados, outro Elemento Autônomo ou funções específicas, como balanceamento de carga. Um *loop* de controle MAPE-K é dividido da seguinte forma:

- **Monitoramento (*Monitor*):** Esta parte é 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 efector (*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.

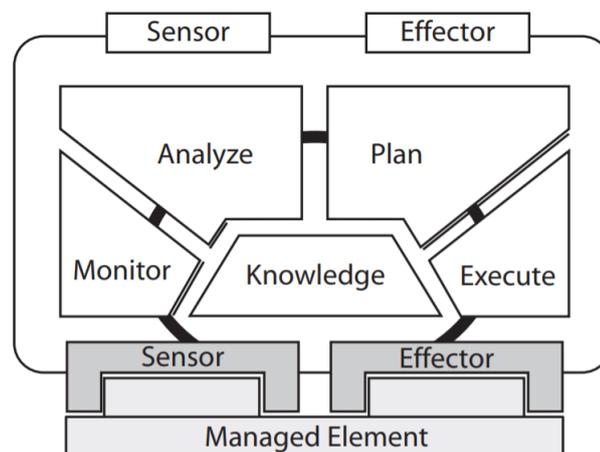


Figura 2.5: Diagrama de funcionamento da arquitetura MAPE-K [21].

2.4.4 Arquitetura *Pipes-and-Filters*

Em uma arquitetura *Pipes-and-Filters*, ilustrada na Figura 2.6, cada componente tem um conjunto de entradas e um conjunto de saídas. Um componente lê *streams* (ou fluxos) de dados em suas entradas e produz *streams* de dados em suas saídas, abstraindo a entrega de um resultado como um todo. O *stream* de entrada é transformado de modo que a

saída comece a ser produzida antes da entrada ser completamente consumida. Por isso, os componentes são chamados de filtros (*filters*). Os conectores deste estilo servem como condutores para os *streams*, transmitindo as saídas de um filtro para as entradas de outro. Por isso, os conectores são chamados de tubos (*pipes*) [43].



Figura 2.6: Diagrama de uma arquitetura *Pipes-and-Filters*.

Os filtros devem ser independentes, isto é, não devem compartilhar estado com outros filtros e, durante a programação, o ideal é que os filtros independam da ordem de processamento. Suas especificações podem restringir os dados transportados nos *pipes* de entrada e nos *pipes* de saída, mas não conhecem quaisquer outros filtros, ou componentes, conectados por seus *pipes*. Especializações comuns desse estilo incluem *pipelines*, que restringem as topologias a sequências lineares de filtros; *Pipes* limitados, que restringem a quantidade de dados que podem ser passados em um *pipe*, e *pipes* tipados, que exigem que os dados passados entre dois filtros tenham um tipo bem definido.

O uso da arquitetura *Pipes-and-Filters* possui como vantagens:

- Permite que o Arquiteto de Software/Desenvolvedor entendam o comportamento geral de entrada/saída de um sistema como uma composição simples dos comportamentos dos filtros individuais.
- Suporta a reutilização: quaisquer dois filtros podem ser conectados, desde que concordem com os dados que estão sendo transmitidos entre eles.
- Os sistemas podem ser facilmente mantidos e aprimorados: novos filtros podem ser adicionados a sistemas existentes e filtros antigos podem ser substituídos por outros melhorados.
- Permitem certos tipos de análise especializada, como análise de rendimento e de impasse.
- Naturalmente suportam a execução simultânea: Cada filtro pode ser implementado como uma tarefa separada e potencialmente executado em paralelo com outros filtros.

Como desvantagens, é possível citar:

- Geralmente levam a uma organização de processamento em lote. Embora os filtros possam processar dados de forma incremental, uma vez que os filtros são inerentemente independentes, o Arquiteto de Software é forçado a pensar em cada filtro como fornecendo uma transformação completa dos dados de entrada em dados de saída.

- Por sua natureza de transformação de dados, os sistemas que usam a arquitetura *Pipes-and-Filters* normalmente não são bons para lidar com aplicativos interativos. Esse problema é mais grave quando são necessárias atualizações de exibição incrementais, porque o padrão de saída para atualizações incrementais é radicalmente diferente do padrão para saída de filtro.
- Podem ser prejudicados por terem que manter correspondências entre dois *streams* separados, mas relacionados.
- Podem forçar um resultado médio na transmissão de dados em situações onde muitos filtros sejam encadeados de uma vez, resultando em trabalho adicional para cada filtro separar seus dados e analisar o que for necessário. Isso, por sua vez, pode levar tanto à perda de desempenho quanto ao aumento da complexidade na escrita dos próprios filtros.

2.5 MLOps

No caso de MLOps, ele pode ser visto como o processo de implantar os melhores modelos de *Machine Learning* para o ambiente de produção, unindo os campos de *Machine Learning*, DevOps e Engenharia de Dados. O termo MLOps é usado para definir uma junção entre modelos de Aprendizado de máquina (ML) e DevOps, focado principalmente na governança e gerenciamento do ciclo de vida de modelos de ML e de decisão operacionalizados. Os principais recursos incluem integração contínua/entrega contínua, ou *continuous integration/continuous delivery* (CI/CD), ambientes de desenvolvimento, sistema de controle de versão, armazenamento e reversão de modelos. O termo pode ser visto como o processo de implantar os melhores modelos de ML para o ambiente de produção, definindo uma estrutura e plataforma para gerenciamento completo do ciclo de vida de artefatos de aplicativos de Aprendizado de Máquina, conforme ilustração na Figura 2.7 e unindo os campos de Aprendizado de Máquina, DevOps e Engenharia de Dados.

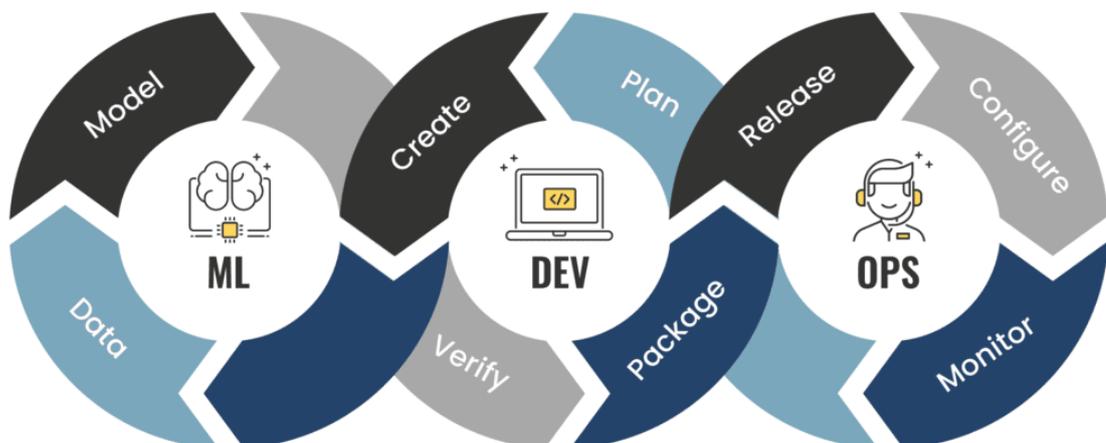


Figura 2.7: Ciclo resumido do processo presente em MLOps [73]

A operacionalização dos procedimentos em Software já é uma prática comum de ser

adotada, mas em aplicações de ML sua adoção é mais recente, possuindo desafios e estilos de dívidas técnicas diferentes de uma aplicação tradicional de Software [66]:

- **Complexidade dos modelos implantados:** A prática tradicional de Engenharia de Software mostrou que o uso de encapsulamento e design modular ajudam a criar código sustentável no qual é fácil de fazer isoladamente mudanças e melhorias. Tais práticas ajudam a expressar a consistência das entradas e saídas de informações de um determinado componente. Infelizmente, é difícil traduzir essas abstrações para sistemas de ML prescrevendo um comportamento pretendido específico. De fato, ML é necessária exatamente nos casos em que o comportamento desejado não pode ser efetivamente expresso na lógica do Software sem dependência de dados externos. O mundo real não se encaixa totalmente de forma encapsulada.
- **Dependência dos dados:** Dívidas técnicas relativas a dependências no código são apontadas como uma dos principais contribuintes para a complexidade do código e dívida técnica em configurações clássicas de Engenharia de Software. Entretanto, as dependências de dados em sistemas de ML possuem uma capacidade semelhante para aumentar a dívida, mas podem ser mais difíceis de detectar. As dependências no código podem ser identificadas por meio de análise estática por compiladores e ligadores. Sem ferramentas semelhantes para dependências de dados, não é difícil construir grandes cadeias de dependências de dados que podem ser difíceis de arrumar.
- **Ciclos de *Feedback*:** Uma das principais características dos sistemas de ML é que eles geralmente acabam influenciando seu próprio comportamento se forem atualizados ao longo do tempo. Isso leva a uma forma de dívida técnica de análise, na qual é difícil prever o comportamento de um determinado modelo antes de ele ser lançado. Esses ciclos de *feedback* podem assumir diferentes formas, mas são mais difíceis de detectar e resolver se ocorrerem gradualmente ao longo do tempo, como pode ser o caso quando os modelos são atualizados com pouca frequência.
- **Anti-padrões nas aplicações:** É comum que sistemas de ML acabem com grandes dívidas em padrões de design. Dívidas como *glue code* (integração de sistemas teoricamente incompatíveis), *pipeline jungles* (*pipelines* de grande complexidade e com difícil rastreamento de problemas, geralmente com etapas fortemente acopladas), código não utilizado, dívidas de abstração, *code smells* (códigos que podem indicar problemas subjacentes em um componente ou sistema) devem ser evitados ou refatorados sempre que possível.
- **Dívidas de configuração:** Qualquer sistema grande tem uma ampla gama de opções configuráveis, incluindo quais *features* são usadas, como os dados são selecionados, uma ampla variedade de configurações de aprendizado específicas de algoritmos, potencial pré ou pós-processamento, métodos de verificação, entre outras. A grande quantidade de opções e a relação que essas tem umas com as outras no sistema torna a configuração difícil de modificar corretamente e difícil de raciocinar.

nar. No entanto, erros na configuração podem custar caro, levando a sérias perdas de tempo, desperdício de recursos de computação ou problemas de produção.

- **Mudanças do mundo externo:** Sistemas de ML geralmente interagem diretamente com o mundo externo, e este raramente é estável, criando um custo de manutenção contínuo. Para isso, é importante prever os vieses e determinar limiares para as previsões que estes sistemas irão realizar.
- **Testagem e fidelidade dos dados:** Embora estejam relacionados a itens já mencionados (dependência dos dados e mudanças do mundo externo), a garantia de que estes desafios estejam bem controlados começa durante o desenvolvimento dos modelos. Dados de entrada testados e fidedignos às situações presentes do mundo garantem este controle, e procedimentos para realizar esta tarefa constituem em outro desafio para garantia da qualidade da aplicação.
- **Processos e cultura do time:** É importante criar equipes que se importem na melhora do modelo e do seu próprio processo de implantação. Sistemas maduros podem ter dezenas ou centenas de modelos rodando simultaneamente, tornando-se complexos ao longo do tempo e necessitando de melhoria contínua dos processos para manter a qualidade.

O processo de um projeto de MLOps, conforme ilustrado na Figura 2.7, deve dar suporte à automação, integração e monitoramento em todas as etapas da construção de um modelo de ML, incluindo treinamento, integração, teste, lançamento, implantação e gerenciamento de infraestrutura [72]. Com tal processo, projeta-se um pipeline automatizado utilizando o recurso de CI/CD, conforme exemplo mostrado na Figura 2.8. CI deve permitir a quem desenvolve experimentos orquestrados para análise, validação e preparação de dados, treinamento, avaliação e validação de novos modelos de ML, e CD deve permitir um pipeline que implante um modelo de ML definido como estável em produção, realize o monitoramento deste e implante outro modelo de ML caso as métricas deste monitoramento sejam consideradas insatisfatórias.

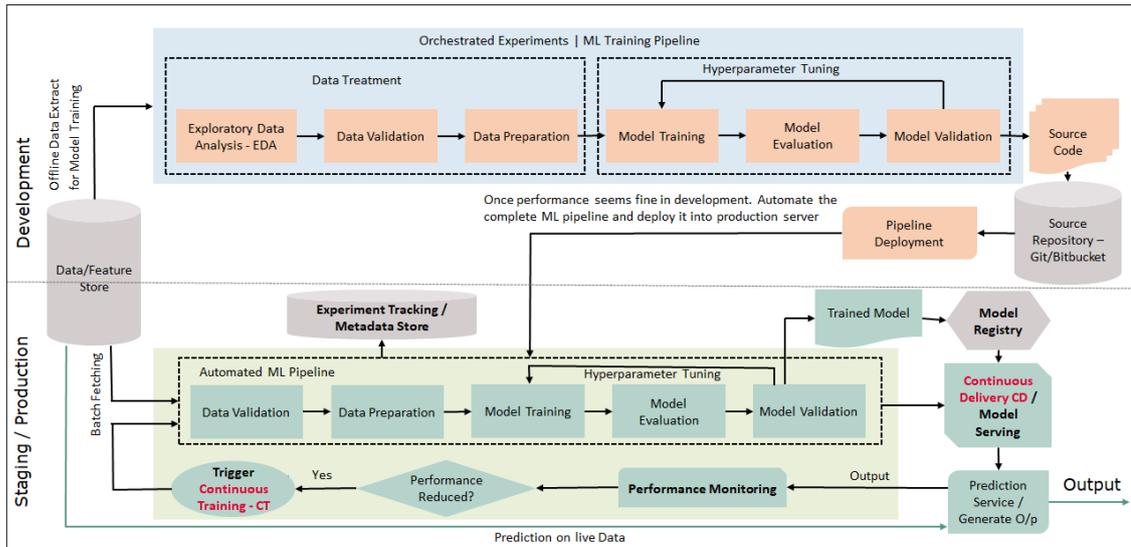


Figura 2.8: Exemplo de Pipeline de MLOps com ambientes de Desenvolvimento, *Staging* e Produção [73]

O ciclo de vida de um modelo de ML tem metodologias diferentes para se adequar a diferentes cenários e tipos de dados. A abordagem mais utilizada por cientistas de dados é o Cross-Industry Standard Process for Data Mining (CRISP-DM) [67], introduzido em 1996 pela Daimler Chrysler. Especialistas podem emprestar as metodologias CRISP-DM padrão e tentar aplicá-las ao pipeline de MLOps, envolvendo dois papéis: O Cientista de Dados, responsável pelo treinamento e teste do modelo, e o Engenheiro de *Machine Learning*, responsável pela produção e implantação. Tais papéis trabalhariam em conjunto em tarefas envolvendo as seguintes etapas:

- Análise do Problema do Negócio
- Definição e entendimento dos dados
- Metodologia analítica de ML
- Componentes de um Pipeline de CI/CD
- Acionamento automatizado do Pipeline
- Armazenamento de registro de modelo
- Monitoramento e desempenho
- Serviço para implantação de modelos ML em Produção

Isso pode ser feito por uma aplicação própria para realizar este gerenciamento, por funcionalidades implementadas na própria aplicação de ML ou através de *frameworks* de automação como o Amazon SageMaker [2], que podem poupar tempo para desenvolvimento de uma nova aplicação a troco de um pagamento de acordo com a demanda utilizada. Estes frameworks podem cuidar de toda a parte de gerenciamento dos dados,

modelagem/treinamento ou operação/implantação, e como são modulares e cuidam apenas de uma destas três categorias do processo, podem ser escolhidas ou não de acordo com a necessidade do projeto e conhecimento da equipe.

2.6 Trabalhos Relacionados

Os principais trabalhos encontrados envolvendo métricas de *Fairness* e arquitetura de sistemas pensando em *Fairness* como requisito vêm do time de engenharia do LinkedIn. [44] descreve quais métricas foram utilizadas e quais algoritmos foram implementados para ranqueamento dos dados. [50] descreve como o conceito de *Fairness* pode ser implementado em uma aplicação de IA, e mostra a arquitetura sobre ela.

Quanto ao processo de desenvolvimento e arquiteturas utilizadas para definir a estrutura final, a IBM AI Reference Architecture [5] foi utilizada no processo como modelo para desenvolvimento de aplicações. Uma arquitetura comparável à arquitetura *Pipes-and-Filters* é a FBFlow do Facebook [38], onde o uso de DAGs (*Direct Acyclic Graphs*) para a execução de um determinado processo ou *workflow* é uma alternativa simples de ser implementada e interpretada pelos desenvolvedores, definindo possível conectar o processo de ML e realizar o cálculo de métricas de forma automatizada. Há diferenças entre DAGs e *Pipes-and-Filters*, mas o objetivo de ambos acaba sendo o mesmo: Conectar diversos processos menores para formar um sistema coeso.

Para o desenvolvimento da parte de análise realizada na arquitetura MAPE-K, uma abordagem similar à aplicada é a *Logic Scoring of Preference* (LSP), utilizada em trabalhos como "*LSP method and its use for evaluation of Java IDEs*" [37], que foi utilizada primariamente para a avaliação de aplicações e define pontuações através de agregações e critérios bem definidos para a avaliação das mesmas. Métodos de *Multi-Criteria Decision Making* (MCDM) como o *Analytic Hierarchy Process* (AHP), presente no livro "*The analytic hierarchy process : planning, priority setting, resource allocation*" [64], também possuem uma abordagem bastante semelhante a realizada e podem ser relacionados, embora não tenham nenhuma relação com os conceitos abordados. Um outro trabalho que pode ser mencionado é o artigo "*The VADA Architecture for Cost-Effective Data Wrangling*" [53], que utiliza pesos para dar contexto aos dados medidos. Embora não seja o principal tema, indicar a importância dos dados analisados garantiu a autonomia necessária sem exigir grandes modificações.

A tabela 2.2 categoriza tais trabalhos de acordo com os conceitos relacionados neste capítulo, divididos nas seguintes colunas: "Ciência/Eng. Dados" para a seção 2.1, "ML" para a seção 2.2, "Fairness" para a seção 2.3, "Eng. Software" para a seção 2.4 e "MLOps" para a seção 2.5.

Tabela 2.2: Categorização dos trabalhos relacionados

Referências	Descrição	Conceitos				
		Ciência/Eng. Dados	ML	Fairness	Eng. Software	MLOps
Geyik et al., 2019 [44]	Métricas utilizadas Ranqueamento dos dados	✓	✓	✓		
Kenthapadi et al., 2019 [50]	Arquitetura/implementação de métricas de Fairness	✓	✓	✓	✓	✓
IBM Analytics and AI Reference Architecture [5]	Arquitetura de referência para Dados e ML	✓	✓		✓	
FBFlow - Facebook (Dunn, 2016) [38]	Uso de DAGs para determinar <i>workflow</i> Execução/interpretação de processos de ML	✓	✓			✓
Konstantinou et al., 2017 [53]	Arquitetura para entrelaçamento de dados Utilização de pesos para determinar contexto aos dados			✓	✓	
Saaty, 1980 [64]	Hierarquização de múltiplos critérios Definição de pesos, ranqueamento e tomada de decisão					
Dujmovic et al., 2006 [37]	Uso de <i>Logic Scoring of Preference</i> Definição de pontuações para avaliação de aplicações				✓	

Capítulo 3

Arcabouço Autônomo Proposto

Neste capítulo, será abordado todo o processo para obtenção da solução e o detalhamento de 2 dos 4 módulos apresentados na Introdução: Módulo de ML e Gerenciador Autônomo. Com esta estrutura foi possível exibir os resultados presentes nos capítulos que detalham os Estudos de Caso, onde diferentes configurações arquiteturais puderam ser exibidas dependendo dos dados treinados e da calibração aplicada.

3.1 Arquitetura da solução

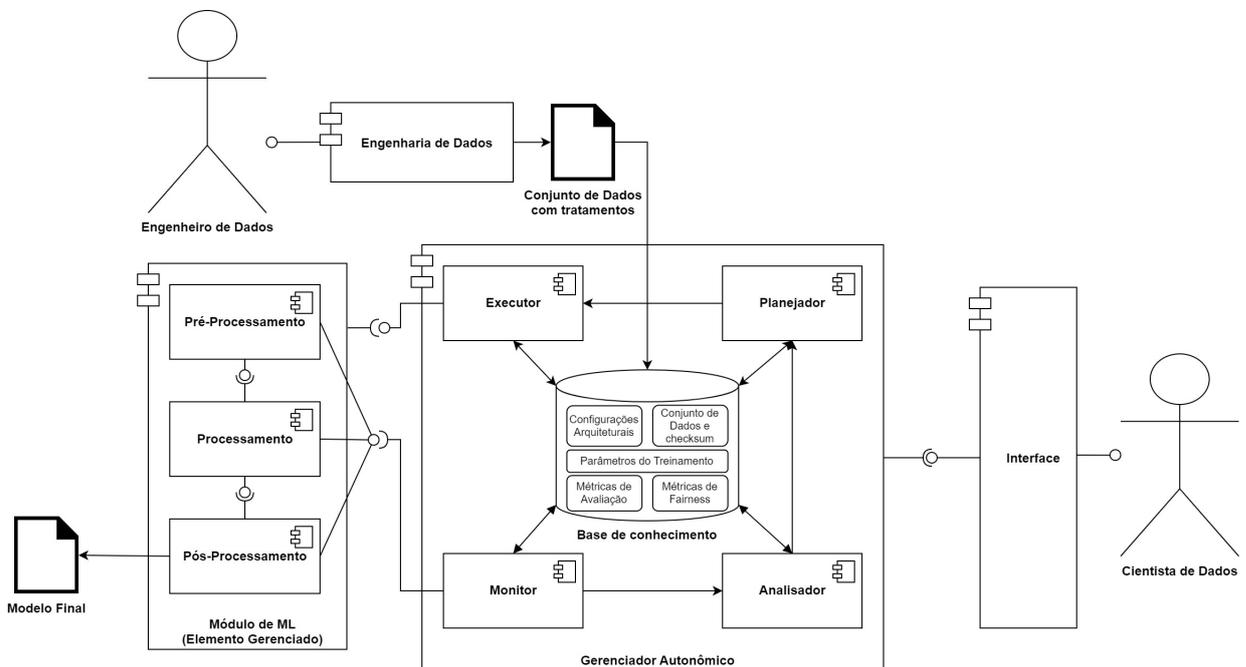


Figura 3.1: Componentes arquiteturais da solução

A arquitetura da solução é composta por 4 módulos:

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

- **Módulo de ML:** Módulo que executa um *Pipeline* capaz de automatizar uma aplicação de ML, com estágios de preparação de dados (Pré-processamento), treinamento (Processamento) e avaliação dos resultados (Pós-processamento) para a geração de um modelo final.
- **Gerenciador Autônomo:** Módulo contendo um *loop* MAPE-K que controla o Módulo de ML como um Elemento Gerenciado para automatizar parte das atividades a serem executadas.
- **Interface:** Módulo cujo objetivo é prover uma experiência de usuário mais simples e intuitiva para o Cientista de Dados para configurar e iniciar o Gerenciador Autônomo.

A integração entre esses módulos é ilustrada na Figura 3.1. O módulo de Engenharia de Dados será utilizado pelo Engenheiro de Dados e retorna um conjunto de dados com os tratamentos necessários. A Interface será utilizada pelo Cientista de Dados, que determinaria os parâmetros necessários dependendo do contexto do problema.

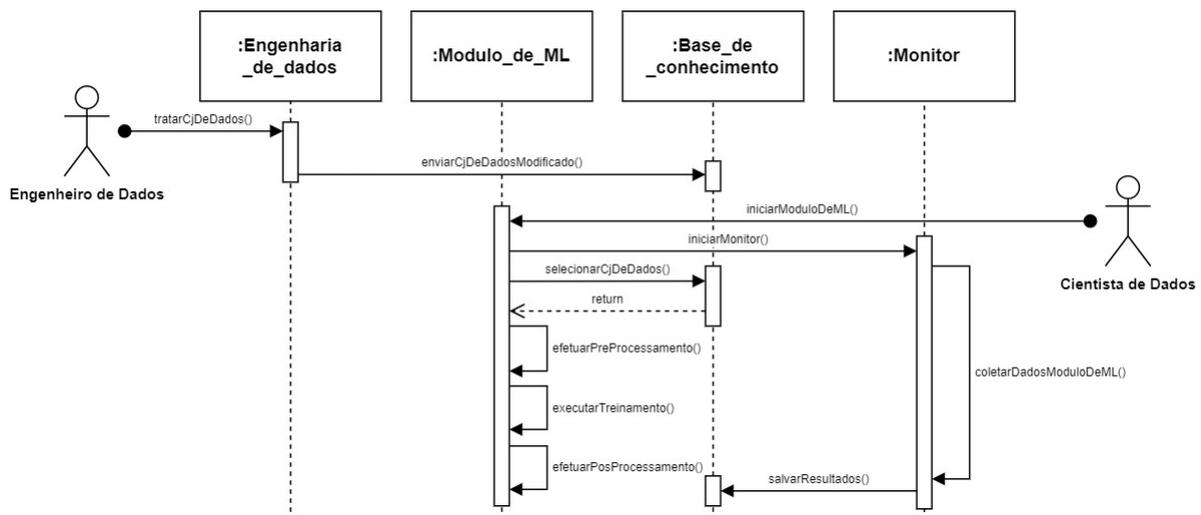


Figura 3.2: Diagrama de sequência de uma execução manual no Módulo de ML

É possível executar o Módulo de ML de forma manual ou auxiliado pelo Gerenciador Autônomo, e o processo de sua execução manual pode ser visualizada no Diagrama de Sequência presente na Figura 3.2. Antes da execução, o Engenheiro de Dados manipula os conjuntos de dados com os processos de limpeza e transformação implementados no módulo de Engenharia de Dados, e os armazena na Base de Conhecimento. Desta forma, o Cientista de Dados pode iniciar o Módulo de ML ao passar os parâmetros necessários para a execução. Durante a execução, o Módulo de ML inicia o Monitor do Gerenciador Autônomo para coletar dados durante a execução, recupera o conjunto de dados armazenado na Base de Conhecimento, e executa as etapas de Pré-processamento, Processamento e Pós-processamento definidas pelos parâmetros passados para a geração do modelo final. Ao finalizar a execução, o Monitor salva todos os dados coletados na Base de Conhecimento para usos futuros.

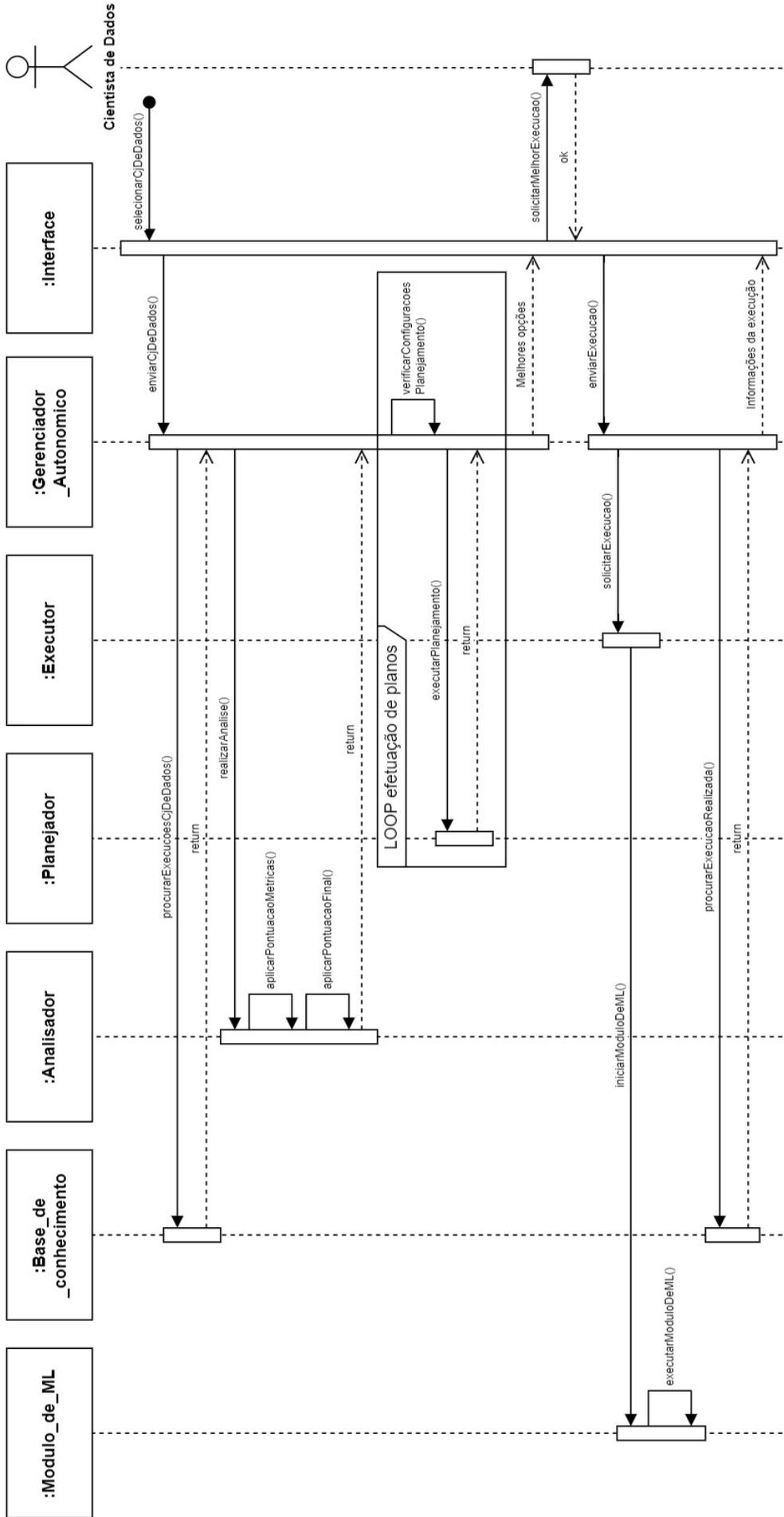


Figura 3.3: Diagrama de sequência de uma execução no Arcabouço Autônomo Proposto

O processo de execução auxiliado pelo Gerenciador Autônomo pode ser visualizado no Diagrama de Sequência presente na Figura 3.3. Através do módulo de Interface, o Cientista de Dados escolhe um Conjunto de Dados e solicita a execução do Gerenciador Autônomo. Este, por sua vez, verifica todas as execuções passadas no Módulo de Machine Learning que utilizaram o conjunto de dados modificado e estão armazenadas na Base de Conhecimento. Em seguida, realiza uma análise detalhada com cálculos (que serão descritos posteriormente) e, em um loop, verifica as configurações de planos de execução já implementados. O Gerenciador Autônomo retorna as melhores opções de execução, permitindo que o Cientista de Dados escolha a mais adequada e solicite uma nova execução para o Módulo de ML. Esta execução é equivalente a já explicada na Figura 3.2, e o Gerenciador Autônomo busca os dados desta execução para exibição na Interface ao Cientista de Dados.

3.2 Módulo de ML

O Módulo de ML segue uma estrutura de passos, ilustrados na Figura 3.4, que é dividida em 3 componentes principais:

- **Pré-Processamento:** Abrange os passos de preparação do dado necessários para a aplicação de um algoritmo para treinamento. Quando *Fairness* entra no escopo de requisitos, também se aplica preparações baseadas no atributo protegido e pode-se aplicar ou não algoritmos para redução de vieses de pré-processamento.
- **Processamento/Treinamento:** Abrange o treinamento do dado preparado no componente de Pré-Processamento através de um algoritmo específico. Quando *Fairness* entra no escopo de requisitos, tal algoritmo pode conter a redução de vieses em seu treinamento ou não.
- **Pós-Processamento/Validação:** Abrange o cálculo de métricas e a verificação se o modelo gerado nesta execução atingiu os valores adequados. Quando *Fairness* entra no escopo de requisitos, pode-se aplicar ou não algoritmos para redução de vieses de pós-processamento.

O módulo é parametrizado para proporcionar ao Cientista de Dados controle sobre a configuração arquitetural utilizada em sua execução, considerando o conjunto de dados, o atributo protegido e os algoritmos empregados. Essa parametrização não apenas facilita a programação de planos de execução no Gerenciador Autônomo, mas também permite a flexibilidade necessária. Ao adotar o padrão de arquitetura *Pipes-and-Filters*, cada etapa, conforme ilustrado na figura, é composta por diversos *Pipes* e Filtros intercambiáveis, dependendo da parametrização escolhida.

Os algoritmos utilizados por este módulo estão presentes nas seções 2.2.2 e 2.3.2, e as métricas calculadas neste módulo estão presentes nas seções 2.2.1 e 2.3.1.

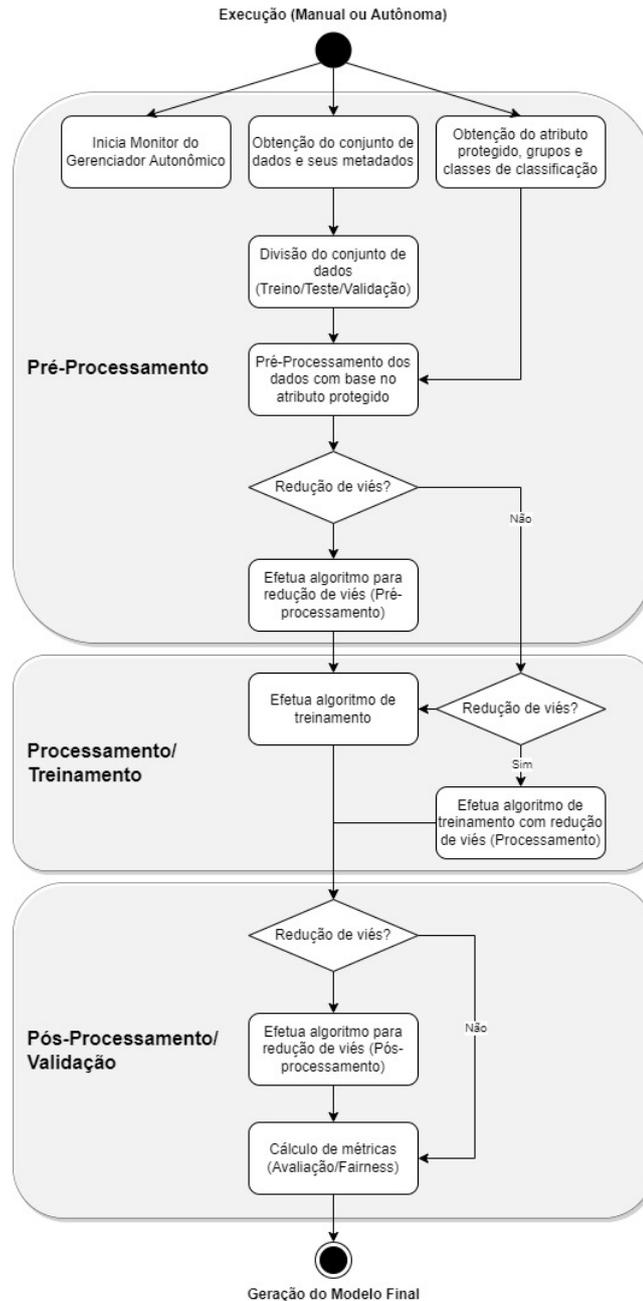


Figura 3.4: Componentes e passos do Módulo de ML

Cada passo está presente em um processo padrão para desenvolvimento de um modelo de Aprendizado de Máquina, e se comportam da seguinte maneira:

- **Pré-Processamento:**

- **Inicia Monitor do Gerenciador Autônomo:** Ao iniciar uma execução, o Monitor do Gerenciador Autônomo é iniciado para efetuar a coleta de metadados presentes durante toda a execução.
- **Obtenção do conjunto de dados e seus metadados:** De acordo com o parâmetro passado para o Módulo de ML, um *Pipe* relativo ao conjunto de

dados em questão é selecionado, e os metadados presentes neste Pipe podem ser utilizados para uma gravação em arquivo caso necessário.

- **Obtenção do atributo protegido, grupos e classes de classificação:** De acordo com o parâmetro passado para o Módulo de ML, um *Pipe* relativo ao atributo protegido em questão é selecionado, e informações de grupos privilegiados e não-privilegiados e classes para classificação também estão presentes neste Pipe.
 - **Divisão do conjunto de dados (Treino/Validação/Testes):** Um Filtro relativo a divisão do conjunto de dados (entre conjunto de treino, conjunto de validação e conjunto de testes) é executado e suas divisões resultantes são colocadas em um *Pipe*.
 - **Pré-Processamento dos dados com base no atributo protegido:** As informações de atributo protegido são passadas para um Filtro para realizar um pré-processamento no conjunto de dados, preparando o conjunto de dados para usos em algoritmos para redução de vieses.
 - **Execução(ou não) de algoritmo para redução de viés:** De acordo com o parâmetro passado para o Módulo de ML, um algoritmo para redução de viés na fase de Pré-Processamento é colocado ou não para ser executado no *Workflow*.
- **Processamento/Treinamento:**
 - **Execução de algoritmo de treinamento com(ou sem) redução de viés:** De acordo com o parâmetro passado para o Módulo de ML, um algoritmo de treinamento é colocado ou não para ser executado.
 - **Pós-Processamento/Validação:**
 - **Execução(ou não) de algoritmo para redução de viés:** De acordo com o parâmetro passado para o Módulo de ML, um algoritmo para redução de viés na fase de Pós-Processamento é colocado ou não para ser executado no módulo.
 - **Cálculo de métricas (Avaliação/Fairness):** É realizado uma predição do algoritmo treinado com o conjunto de teste, e as predições são comparadas com os valores verdadeiros para obtenção das métricas.

3.3 Gerenciador Autônomo

Os dados armazenados na Base de Conhecimento a cada execução do Módulo de ML vão ser utilizados como insumos para análise no Gerenciador Autônomo, seguindo o modelo de arquitetura MAPE-K. Após algumas execuções do Módulo de ML, a Base de Conhecimento possui informação suficiente para a realização da análise do Gerenciador Autônomo, que ajudaria a selecionar a melhor configuração arquitetural para o módulo gerar melhores modelos. Após a obtenção de tal Base de Conhecimento, é possível verificar

como cada componente do Gerenciador Autônomo se comportará, conforme já ilustrado na Figura 3.1.

3.3.1 Monitor

O Monitor é iniciado junto com a execução do Módulo de ML, conforme já ilustrado na Figura 3.4. Os metadados que podem ser guardados são os seguintes:

- **Pré-Processamento:** Parâmetros referentes a configuração arquitetural executada pelo módulo (Conjunto de dados, Atributo protegido e Algoritmo de pré-processamento) e "assinatura"/checksum do conjunto de dados utilizado.
- **Processamento:** Parâmetros referentes a configuração arquitetural executada (Algoritmo de treinamento) e parâmetros utilizados para a aplicação do treinamento.
- **Pós-Processamento:** Parâmetros referentes a configuração arquitetural executada (Algoritmo de pós-processamento) e Métricas do modelo resultante (Métricas de Avaliação e métricas de *Fairness*).

Os dados coletados por execução são organizados em dois conjuntos diferentes: Um contendo os metadados e informações referentes a configuração arquitetural executada pelo Módulo de ML, e outro contendo as métricas obtidas pelo modelo resultante dessa execução, ambos possuindo um identificador da execução para estabelecer consistência entre os dados dos dois conjuntos. Com tal organização, é facilitado o desenvolvimento do Analisador.

3.3.2 Analisador

A Figura 3.5 exemplifica o procedimento de análise para todas as configurações arquiteturais. As execuções realizadas no Módulo de ML que foram coletadas pelo Monitor e armazenadas na Base de Conhecimento são extraídas e analisadas por suas métricas. Nesta análise, cada execução recebe pontuações consolidando os valores de suas métricas de Avaliação e suas métricas de *Fairness*. Posteriormente, as análises realizadas são agrupadas em suas respectivas configurações arquiteturais, que recebem uma pontuação definitiva.

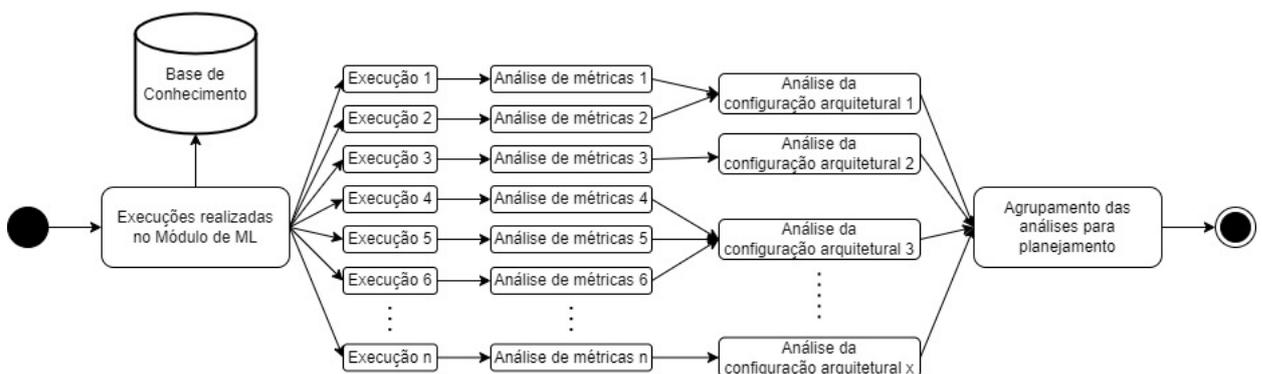


Figura 3.5: Procedimento de análise para execuções no módulo de ML

As pontuações de cada configuração arquitetural são agrupadas e colocadas em um conjunto de dados. Para o desenvolvimento de melhores estratégias, alguns metadados, como data de execução, são obtidos e agrupados junto às pontuações. O conjunto resultante deste processo é passado ao Planejador.

Análise de métricas

Para a análise de cada execução feita pelo Módulo de ML, é realizado um cálculo de pesos 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 consolidar as métricas de Avaliação e métricas de *Fairness* para simplificar as estratégias de planejamento.

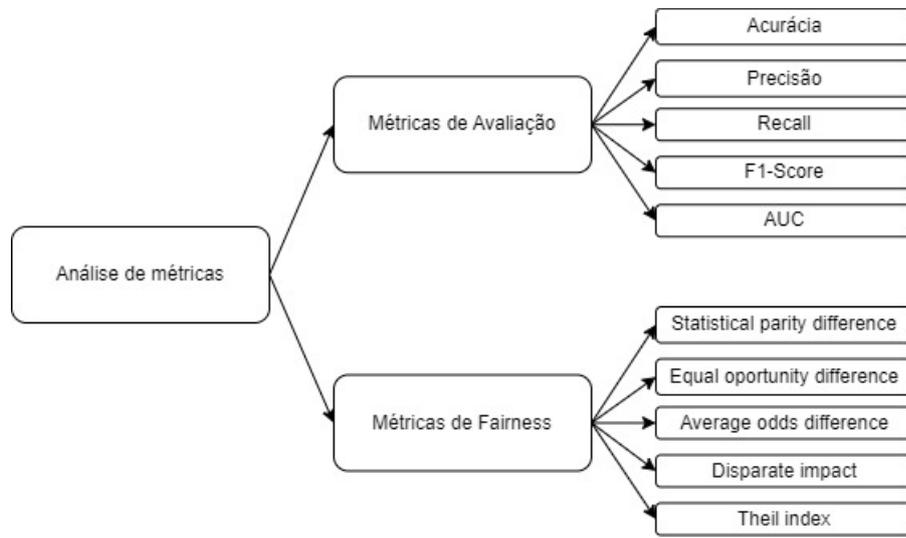


Figura 3.6: Árvore de métricas a serem analisadas

Para um melhor isolamento dos contextos, foi estabelecida para este cálculo uma árvore de métricas, ilustrada na Figura 3.6, onde as mesmas foram divididas em dois grupos (Métricas de Avaliação e Métricas de *Fairness*), e foram atribuídos as métricas referentes a cada grupo, presentes nas seções 2.2.1 e 2.3.1. Para cada grupo, dependendo do contexto do problema, são atribuídos os pesos w_E para o grupo de Métricas de Avaliação e w_F para o grupo Métricas de *Fairness*. Com isso, a forma utilizada para consolidar todas as métricas foi através de uma pontuação S definida por uma média ponderada, exibida na equação 3.1, entre as pontuações S_E e S_F , que são calculadas isoladamente.

$$S = \frac{w_F \times S_F + w_E \times S_E}{w_F + w_E} \quad (3.1)$$

Para o cálculo de S_E e S_F , que são as pontuações respectivas referentes ao grupo de Métricas de Avaliação e ao grupo de Métricas de *Fairness*, também é utilizado o cálculo de média ponderada e são atribuídos pesos distintos para cada métrica, sendo $w_{m_{E_i}}$ para o grupo de Métricas de Avaliação e $w_{m_{F_i}}$ para o grupo Métricas de *Fairness*. Entretanto, há algumas diferenças a se considerar dado a natureza das métricas presente nos grupos. Para o caso de S_E , todas as métricas calculadas estão presentes no intervalo $[0, 1]$ e seus

resultados são diretamente proporcionais, indicando que os melhores resultados também são os que possuem maiores valores. Isto implica no uso de uma média ponderada sem adaptações.

$$S_E = \frac{\sum_{i=1}^{n_{m_E}} w_{m_{E_i}} \times m_{E_i}}{\sum_{i=1}^{n_{m_E}} w_{m_{E_i}}} \quad (3.2)$$

Nela, n_{m_E} é o número de métricas de avaliação utilizadas (para a árvore definida na Figura 3.6, $n_{m_E} = 5$), $w_{m_{E_i}}$ é o peso dado para uma determinada métrica de avaliação e m_{E_i} é o valor de uma determinada métrica de avaliação na execução analisada.

Entretanto, para o caso de S_F , ocorrem 3 intervalos e situações distintas para as métricas de *Fairness* presentes no grupo:

- Para o caso das métricas *Statistical parity difference*, *Equal opportunity difference* e *Average odds difference*, estas métricas envolvem a diferença entre 2 ou mais indicadores presentes no intervalo $[0, 1]$, estando no intervalo $[-1, 1]$ e sendo 0 considerado o melhor resultado.
- Para o caso da métrica *Theil index*, o resultado está no intervalo $[0, 1]$, porém é inversamente proporcional: 0 é considerado o melhor resultado e 1 o pior
- Para o caso da métrica *Disparate impact*, esta é a única métrica que envolve a razão entre dois indicadores, estando no intervalo $[0, +\infty[$ e sendo 1 considerado o melhor resultado.

Para normalizar as métricas m_{F_i} a uma métrica m'_{F_i} no intervalo $[0, 1]$, cada caso foi tratado isoladamente. Para o primeiro caso, foi utilizado $m'_{F_i} = 1 - |m_{F_i}|$. Para o segundo caso, poderia-se utilizar $m'_{F_i} = 1 - m_{F_i}$, mas o cálculo utilizado no primeiro caso também satisfaz o mesmo intervalo e produz os mesmos resultados. Para o terceiro caso, por se tratar de uma razão e pelo melhor resultado ser outro valor, foi utilizado outro cálculo:

$$m'_{F_i} = \begin{cases} 1 - \left| \frac{1}{m_{F_i}} - 1 \right| & \text{caso } m_{F_i} > 1 \\ 1 - |m_{F_i} - 1| & \text{caso } 0 \leq m_{F_i} \leq 1 \end{cases} \quad (3.3)$$

Consolidando todos os casos, temos o cálculo para adequar as métricas m_{F_i} a nova métrica m'_{F_i} :

$$m'_{F_i} = \begin{cases} 1 - \left| \frac{1}{m_{F_i}} - 1 \right| & \text{caso } m_{F_i} \text{ seja } \textit{Disparate impact} \text{ e } m_{F_i} > 1 \\ 1 - |m_{F_i} - 1| & \text{caso } m_{F_i} \text{ seja } \textit{Disparate impact} \text{ e } 0 \leq m_{F_i} \leq 1 \\ 1 - |m_{F_i}| & \text{caso contrário} \end{cases} \quad (3.4)$$

Após obter m'_{F_i} , é possível obter S_F com o uso de uma média ponderada:

$$S_F = \frac{\sum_{i=1}^{n_{m_F}} w_{m_{F_i}} \times m'_{F_i}}{\sum_{i=1}^{n_{m_F}} w_{m_{F_i}}} \quad (3.5)$$

Nela, n_{m_F} é o número de métricas de *Fairness* utilizadas (para a árvore definida na Figura 3.6, $n_{m_F} = 5$), $w_{m_{F_i}}$ é o peso dado para uma determinada métrica de *Fairness* e m'_{F_i} é o valor de uma determinada métrica de *Fairness* na execução analisada que foi normalizada no intervalo $[0, 1]$.

Análise da configuração arquitetural

Para consolidar as pontuações definitivas S'_E , S'_F e S' para cada configuração arquitetural, cada execução é agrupada de acordo com a configuração arquitetural utilizada e é realizada uma média aritmética simples de acordo com as n' execuções encontradas para tal configuração.

$$S'_F = \frac{\sum_{i=1}^{n'} S_{F_i}}{n'} \quad S'_E = \frac{\sum_{i=1}^{n'} S_{E_i}}{n'} \quad S' = \frac{\sum_{i=1}^{n'} S_i}{n'} \quad (3.6)$$

Para facilitar a visualização das pontuações pelo Cientista de Dados, multiplica-se as pontuações S'_E , S'_F e S' por um fator arbitrário $X = 1000$ e arredonda-se em seguida, conforme exibido na equação 3.7.

$$S'_F = \left\lceil X \times \frac{\sum_{i=1}^{n'} S_{F_i}}{n'} \right\rceil \quad S'_E = \left\lceil X \times \frac{\sum_{i=1}^{n'} S_{E_i}}{n'} \right\rceil \quad S' = \left\lceil X \times \frac{\sum_{i=1}^{n'} S_i}{n'} \right\rceil \quad (3.7)$$

3.3.3 Planejador

Para o Planejador, foram desenvolvidas as seguintes estratégias, cujo cada funcionamento é explicado abaixo, para a definição da melhor configuração arquitetural a ser executada pelo Módulo de ML em determinado contexto:

- **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.
- **Limiar de resultados da análise:** Pelo mesmo motivo da estratégia anterior, métricas não confiáveis significam que podem existir distorções na pontuação final determinada pelo Analisador. Para esse caso, foi criado um limiar de pontuação mínimo e máximo para determinar pontuações que podem ser consideradas confiáveis para avaliação.
- **Obtenção das melhores configurações arquiteturais para execução:** As configurações arquiteturais que não foram filtradas nas estratégias anteriores são selecionadas de acordo com as maiores pontuações e são selecionados os parâmetros necessários para executar o Módulo de ML.

O Planejador verifica as configurações de cada estratégia e as executa. Após a execução de todas as estratégias, os parâmetros selecionados são passados para o Executor.

3.3.4 Executor

Para o Executor, os parâmetros selecionados na fase de planejamento são configurados como parâmetros para a execução do Módulo de ML, que executa os procedimentos detalhados na seção 3.2.

3.4 Atuação do Arcabouço Autônomo dentro do processo

Com base nas etapas da AI Reference Architecture [5], foi desenhado um diagrama de atividades, presente na Figura 3.7, determinando como o Arcabouço Autônomo Proposto se encaixaria no processo. É possível definir os seguintes papéis presentes no desenvolvimento de aplicações de *Machine Learning* e como se encaixariam em um contexto onde o arcabouço se faria presente:

- **Especialista de Domínio:** É a pessoa que detém o conhecimento de todo o conjunto de regras do qual a aplicação deve respeitar. Seu conhecimento é atuante na comunicação das regras com os demais papéis, independente de seu conhecimento técnico. Está presente nas fases de Coleta, Organização e Infusão do ciclo.
- **Engenheiro de Dados:** É a pessoa responsável pelos processos de coleta e transformação dos dados para o uso em outros processos, sejam eles de Softwares tradicionais ou aplicações de *Machine Learning*. Pode aplicar processos de governança antes de definir que o dado esteja pronto para ser usado por outras pessoas. Embora não esteja colocado nas fases atuantes do Arcabouço Autônomo Proposto, sua participação é vital para que o arcabouço funcione corretamente, uma vez que os dados manipulados por ele são transportados para a Base de Conhecimento do mesmo. Está presente nas fases de Coleta e Organização do ciclo.
- **Cientista de Dados:** É a pessoa responsável pela análise dos dados e do desenvolvimento do processo de *Machine Learning* após a transformação e tratamento dos dados. Pode realizar tratamentos próprios antes do treinamento, como *Encoding (Label Encoding/One-hot Encoding)*, normalização, processos de regularização como aumento de dados, para melhorar a performance do mesmo. Por esta gama de funções, o Arcabouço Autônomo Proposto atua exatamente nas fases onde o mesmo atua, sendo parte vital do seu uso. Está presente nas fases de Organização e Análise do ciclo.
- **Engenheiro de Software:** É a pessoa responsável por usar o modelo de *Machine Learning* obtido na fase de Análise em aplicações que façam sentido para o uso do mesmo, como assistentes, automações, dashboards e relatórios. Está presente apenas na fase de Infusão, mas pode ser considerado na fase de Análise para verificar com o Cientista de Dados como está o andamento dos modelos desenvolvidos e desenhar alternativas caso os mesmos não estejam prontos para uso.

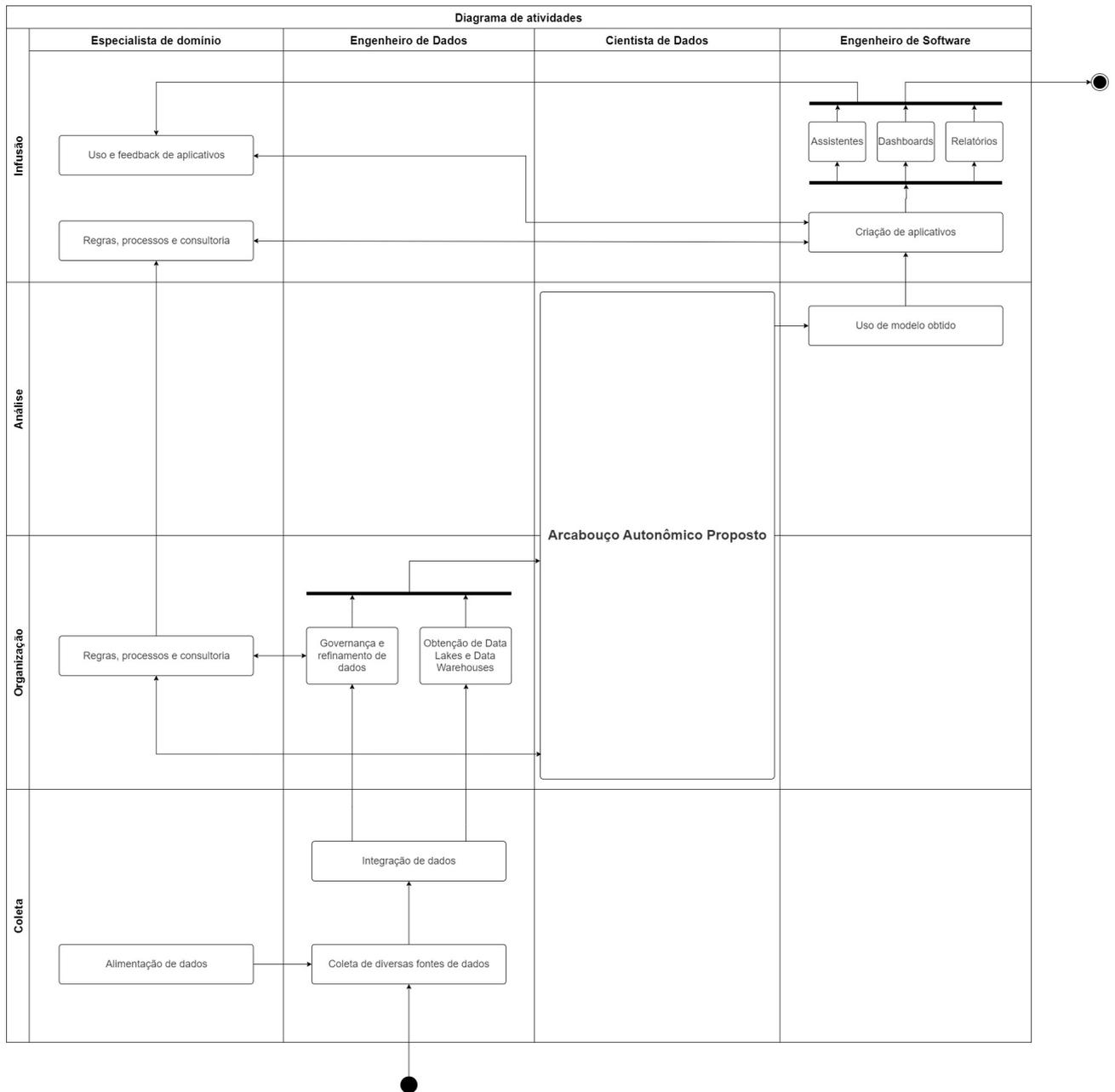


Figura 3.7: Diagrama de atividades, com base na IBM AI Reference Architecture [5], indicando a subdivisão de cada papel no uso do Arcabouço Autônomo

Capítulo 4

Implementação do Arcabouço

Este capítulo apresenta a ferramenta FairPEK que implementa a solução descrita no Capítulo 3. A ferramenta utiliza um arcabouço cliente-servidor sendo que a seção 4.1 descreve brevemente detalhes de implementação presentes nos módulos apresentados e a seção 4.2 descreve a Interface Humano-Computador implementada, referente ao módulo de Interface.

4.1 Visão geral da implementação

O Módulo de ML, o Gerenciador Autônomo e o módulo de Engenharia de Dados foram desenvolvidos em Python pelo motivo de que grande parte dos algoritmos e bibliotecas utilizados já estão implementados com essa linguagem. Para a execução dos algoritmos com redução de viés no Módulo de ML, é utilizada a biblioteca AI Fairness 360, ou AIF360 [1], biblioteca da IBM que compila diversos algoritmos para este fim e facilita o cálculo das métricas de Fairness. Para os algoritmos sem redução de viés, é usado o scikit-learn [12]. Um pequeno *framework* também foi desenvolvido no Módulo de ML para facilitar o desenvolvimento com o padrão de arquitetura *Pipes-and-Filters*. O Gerenciador Autônomo e o módulo de Engenharia de Dados usam a biblioteca Pandas [9] para auxiliar nos processos de transformação de dados e na organização dos dados presentes na Base de Conhecimento.

Durante o desenvolvimento do Módulo de ML e do Gerenciador Autônomo, foi notado que o número de configurações e a complexidade das mesmas era muito grande, ocasionando problemas na hora de documentar e detalhar todo o processo executado. Para facilitar tais configurações, foi criada uma Interface Humano-Computador onde é condensada toda a organização dos parâmetros, dos arquivos utilizados e da realização das execuções simples e autônoma do Módulo de ML. Também foi criado um Backend dentro do Gerenciador Autônomo para fazer a intermediação entre a interface e o mesmo.

O Backend também foi desenvolvido em Python para reusar códigos já desenvolvidos em etapas anteriores, usando o framework Flask [4] para construir as requisições web e foi dividido em 3 camadas. A camada *web* corresponde às requisições que constroem a ponte entre Frontend e Backend, a camada *service* corresponde às funcionalidades e casos de uso que serão chamados pelas requisições, e a camada *repo* corresponde às operações

de leitura e escrita que serão realizadas nos arquivos do Módulo de ML.

Após a implementação e durante a elaboração dos experimentos, foi notado a necessidade de desenvolver uma documentação, presente nos Apêndices A e B. Nela estão explicadas as tecnologias utilizadas para instalação e execução do sistema, além de passos para manutenção caso novos algoritmos e conjuntos de dados sejam adicionados a implementação do arcabouço.

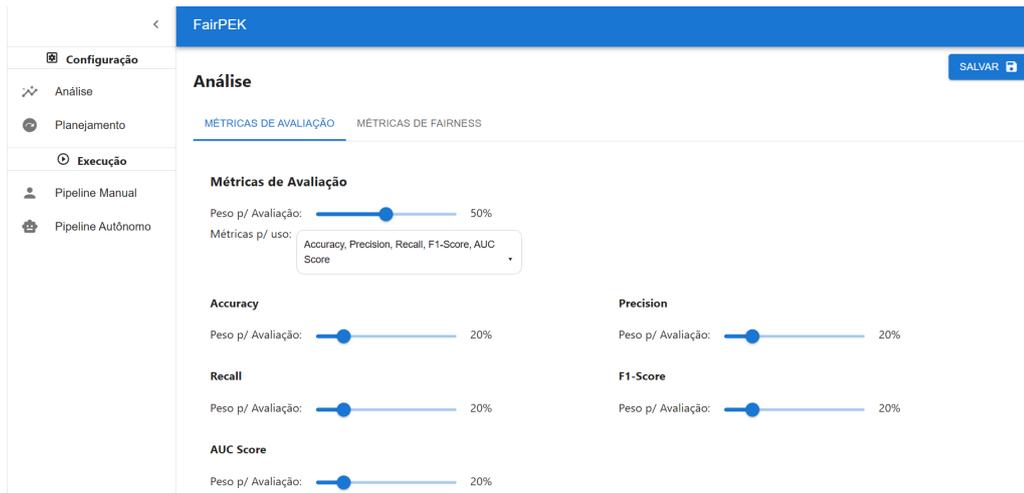
4.2 Interface

A interface foi desenvolvida em JavaScript, aproveitando a facilidade e robustez dessa linguagem, juntamente com o extenso conjunto de ferramentas disponíveis para facilitar o desenvolvimento. Também foram utilizadas as bibliotecas React [10] e Redux [11], além da biblioteca de componentes Material UI [8]. A escolha da Material UI proporciona economia de tempo ao aproveitar componentes prontos, enquanto oferece um *look-and-feel* semelhante aos aplicativos do Sistema Operacional Android, seguindo as diretrizes do Material Design desenvolvido pelo Google. Isso é particularmente relevante, uma vez que o foco deste trabalho não demanda componentes específicos para a interface.

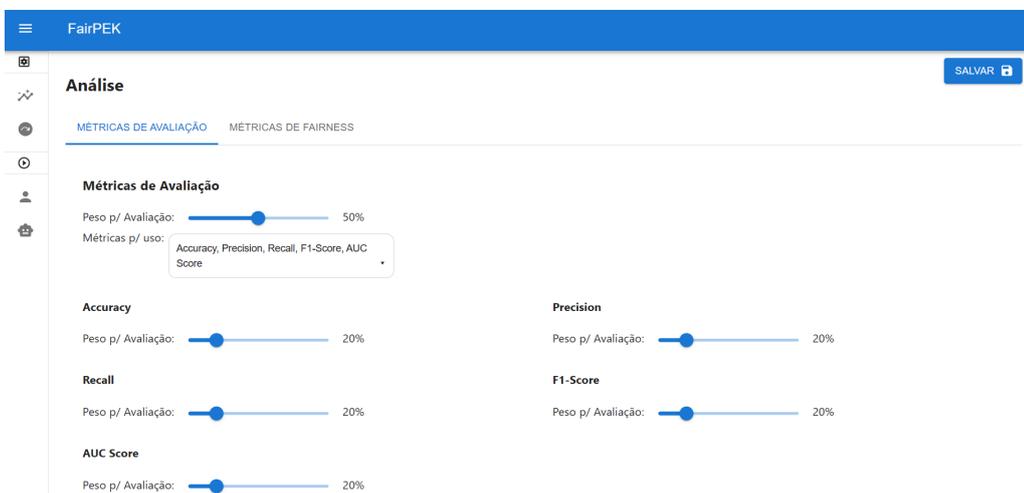
A interface foi nomeada de FairPEK, junção dos termos Fairness e MAPE-K, e possui os seguintes detalhes:

Opções do Menu

Conforme ilustrado na Figura 4.1, o menu pode ser expandido e recolhido, onde suas opções são selecionáveis independente da configuração, e foram colocados ícones ao lado do nome de sua opção para que a localização das opções seja acessível mesmo com o menu recolhido.



(a) Opções de seleção de menu expandidas.



(b) Opções de seleção de menu recolhidas.

Figura 4.1: Comportamento das opções de menu.

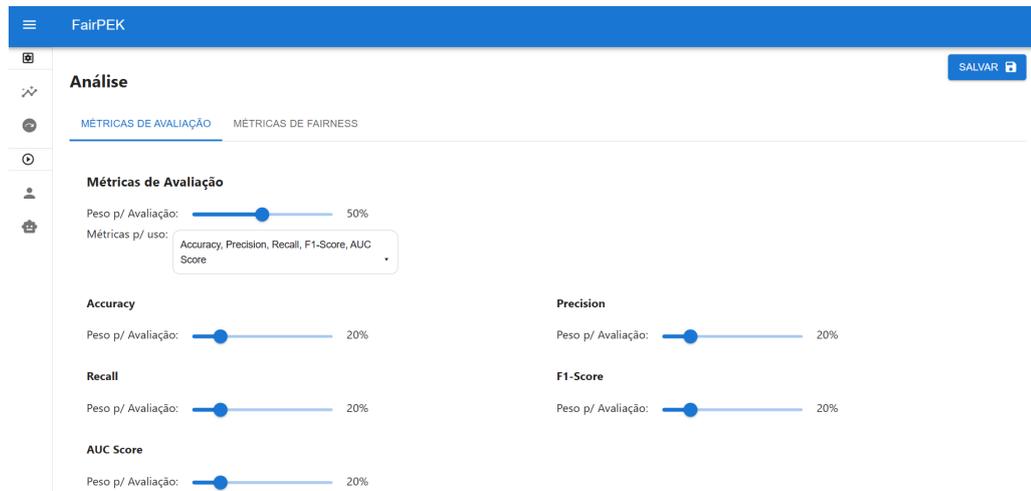
No menu, as opções são divididas em Configuração e Execução. Em Configuração, há as opções Análise e Planejamento relativas às configurações presentes no Gerenciador Autônomo. Em Execução, há as opções Pipeline Manual e Pipeline Autônomo relativas às maneiras de como realizar uma execução do Módulo de ML.

4.2.1 Opções para parametrização do Gerenciador Autônomo

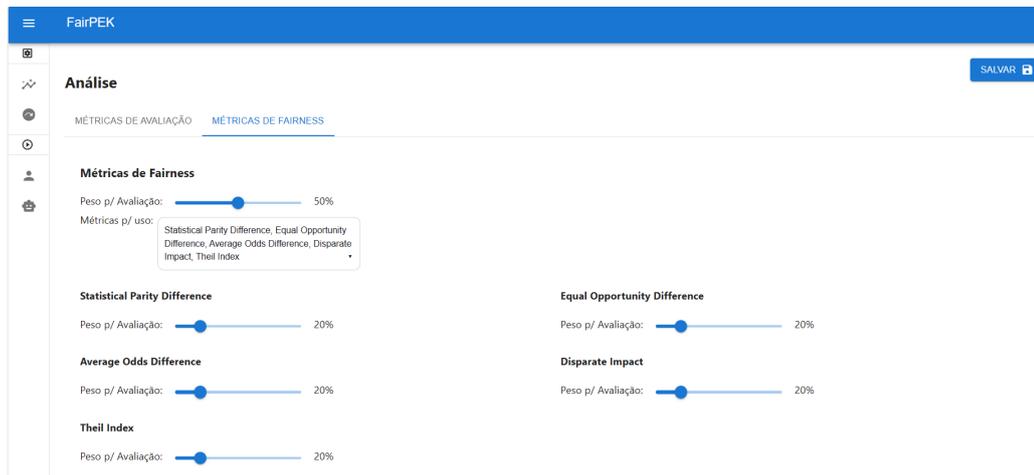
Configurações para Análise

Ao clicar a opção do menu "Análise", é exibida a tela ilustrada na Figura 4.2. Ela é dividida em duas abas: Métricas de Avaliação e Métricas de Fairness, relativas aos grupos de métricas divididos no Gerenciador Autônomo. Em ambas as abas, há os campos de peso para avaliação, que simboliza o peso no cálculo da pontuação final, e o campo de métricas para uso, que determina quais métricas serão utilizadas para o cálculo da pontuação de cada grupo. Uma vez que há apenas duas abas, a alteração do campo de peso para avaliação de uma aba automaticamente alterará o campo de peso para avaliação

da outra aba para complementar a soma de 100% sem precisar de validações.



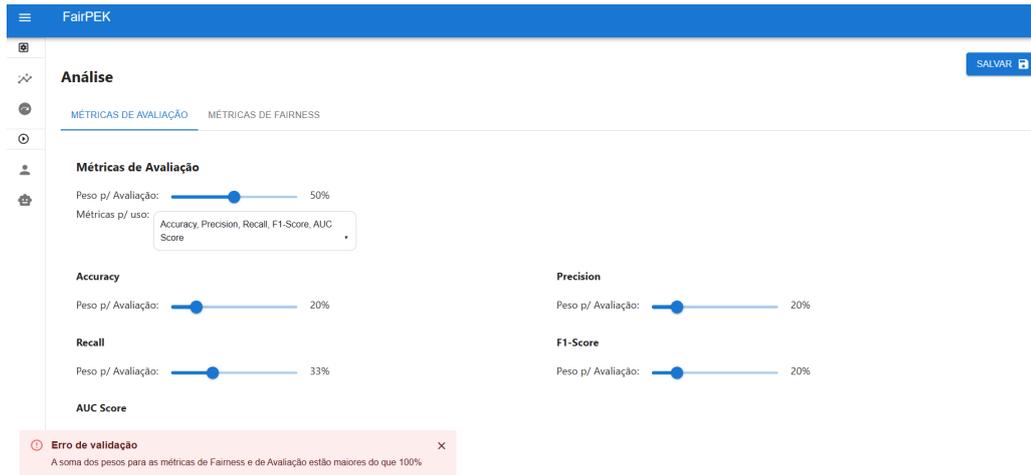
(a) Configuração para Métricas de Avaliação.



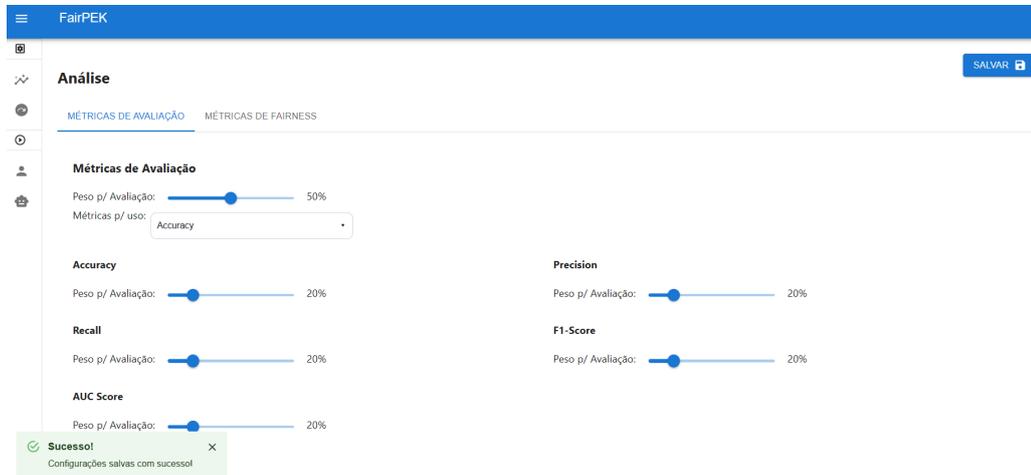
(b) Configuração para Métricas de Fairness.

Figura 4.2: Configuração das métricas para o Analisador do Gerenciador Autônomo.

A alteração do campo de métricas para uso pode resultar em uma mudança no número de métricas para as quais os pesos devem ser ajustados. Dado que é possível ter mais de duas métricas, a tarefa de assegurar automaticamente que a soma total das métricas atinja 100% torna-se mais complexa, sendo substituída por um processo de validação conforme ilustrado na Figura 4.3. Ao clicar no botão "Salvar" localizado no canto superior direito, é realizada a validação da soma de todas as métricas e, em caso positivo, chamada uma requisição que salva o arquivo com as opções selecionadas e seus respectivos valores. Após este procedimento, será exibida uma indicação de sucesso ou erro de validação no canto inferior esquerdo indicando se o arquivo foi salvo ou não.



(a) Erros de validação ao concluir a operação.



(b) Indicação de sucesso da operação.

Figura 4.3: Cenários possíveis na configuração das métricas.

Configurações para Planejamento

Ao clicar a opção do menu "Planejamento", é exibida a tela ilustrada na Figura 4.4. Ela possui três opções, relativas às estratégias desenvolvidas para o Planejador do Gerenciador Autônomo. Ao clicar no botão "Salvar" localizado no canto superior direito, são chamadas requisições que salvam os arquivos necessários e é exibida uma indicação de sucesso no canto inferior esquerdo.



(a) Opções para configurar o Planejador do Gerenciador Autônomo.



(b) Indicação de sucesso da operação.

Figura 4.4: Configuração do Planejador do Gerenciador Autônomo.

Se a estratégia de restrição de algoritmos for selecionada, é exibido um submenu contendo as configurações arquiteturais que podem ser selecionadas, e que são salvas em um arquivo separado para serem filtrados na etapa de planejamento. Se a estratégia de restrição por um limiar de pontuação for selecionada, é exibido um slider com uma pontuação mínima e uma pontuação máxima, e ambas as pontuações são salvas em um arquivo separado para serem filtrados na etapa de planejamento.

4.2.2 Opções para execução do Módulo de ML

Execução manual

Ao clicar a opção do menu "Pipeline Manual", é exibida a tela ilustrada na Figura 4.5. Ela possui indicações de etapas divididas em Parametrização, Execução e Resultados, campos para selecionar o conjunto de dados e o atributo protegido e opções para selecionar onde a redução de viés será executada. Dependendo da opção selecionada, aparecem campos para selecionar o algoritmo de treinamento e o algoritmo de redução de viés. Ao clicar no botão "Executar" localizado no canto superior direito, é feita uma requisição para executar o

Módulo de ML com as opções selecionadas, é exibida uma indicação de sucesso no canto inferior esquerdo e a indicação de etapa é atualizada para a etapa de execução.

(a) Opções para executar o Módulo de ML manualmente.

(b) Módulo de ML em execução.

Figura 4.5: Execução simples e manual do Módulo de ML.

Após a execução ser concluída, a indicação de etapa é atualizada para a etapa de resultados, conforme ilustração nas Figuras 4.6 e 4.7. Nela, os parâmetros gravados são organizados em 4 grupos: Execução, relativos aos parâmetros utilizados e estatísticas da execução, Métricas de Avaliação, relativas aos resultados das métricas de avaliação, Métricas de Fairness, relativas aos resultados das métricas de Fairness, e Pontuação, relativas ao cálculo realizado com as configurações utilizadas na parte de métricas.

Pipeline Manual

Parametrização Execução Resultado

Execução	Parâmetros utilizados e estatísticas da execução
Checksum do conjunto de dados	fcd189dd32a1631a474e1087a7f4835525194741309493e0475bce7cb3451d0a499c70007db6c4b5e33cb4ecb2da9feb859b57cfa21255573103d92e6b497d
Conjunto de dados utilizado	Datasets.GERMAN_CREDIT
Atributo protegido	Preprocessors.AGE
Algoritmo de redução de viés no pré-processamento	UnbiasDataAlgorithms.REWEIGHING
Algoritmo de treinamento (redução de viés: Não)	Algorithms.LOGISTIC_REGRESSION
Algoritmo de redução de viés no pós-processamento	UnbiasPostProcAlgorithms.NOTHING
Data de início da execução	11/09/2023 03:38:35.138495
Data de fim da execução	11/09/2023 03:38:39.104185
Tempo de Execução	3965 ms

Métricas de Avaliação Métricas relacionadas a qualidade do modelo

Métricas de Fairness Métricas relacionadas a justiça do modelo

Pontuação Pontuação dada de acordo com os pesos e métricas utilizadas para a execução

(a) Exibição dos parâmetros no resultado.

Pipeline Manual

Parametrização Execução Resultado

Execução	Parâmetros utilizados e estatísticas da execução
Métricas de Avaliação	Métricas relacionadas a qualidade do modelo
Métricas de Fairness	Métricas relacionadas a justiça do modelo

Pontuação Pontuação dada de acordo com os pesos e métricas utilizadas para a execução

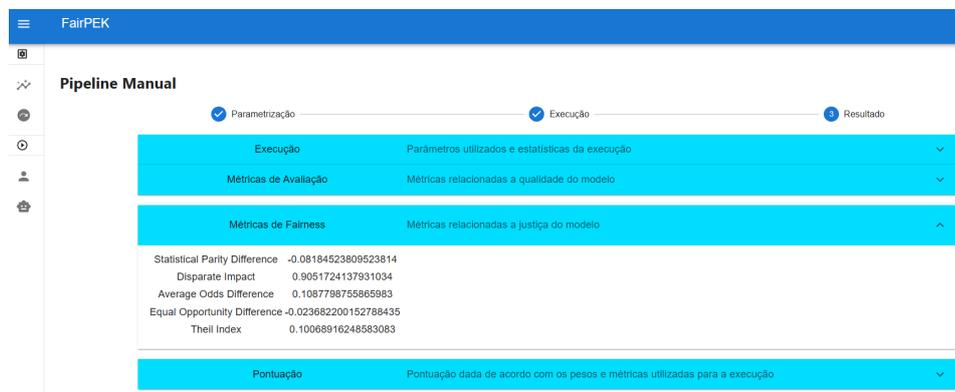
Pontuação das métricas de avaliação	786
Pontuação das métricas de fairness	918
Pontuação geral	852

(b) Exibição das pontuações no resultado.

Figura 4.6: Informações do resultado do Módulo de ML.



(a) Exibição das métricas de avaliação no resultado.

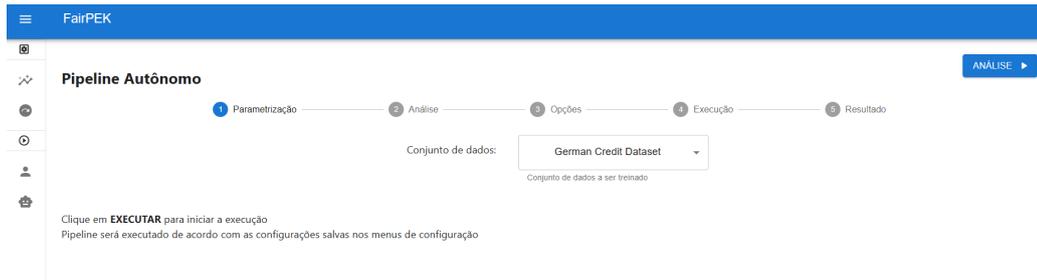


(b) Exibição das métricas de fairness no resultado.

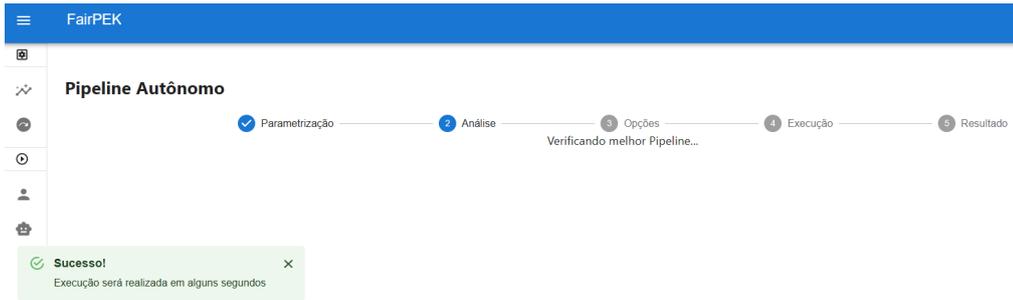
Figura 4.7: Métricas do resultado do Módulo de ML.

Execução autônoma

Ao clicar a opção do menu "Pipeline Autônomo", é exibida a tela ilustrada na Figura 4.8. Ela possui indicações de etapas divididas em Parametrização, Análise, Opções, Execução e Resultados e um campo para selecionar o conjunto de dados. Ao clicar no botão "Executar" localizado no canto superior direito, é feita uma requisição para escolher o melhor conjunto de parâmetros baseado em execuções anteriores, é exibida uma indicação de sucesso no canto inferior esquerdo e a indicação de etapa é atualizada para a etapa de análise.



(a) Opções para configurar o Módulo de ML de forma autônoma.



(b) Análise do Gerenciador Autônomo em execução.

Figura 4.8: Execução autônoma do Módulo de ML gerenciada pelo Gerenciador Autônomo.

Após a etapa de análise ser concluída, a indicação de etapa é atualizada para a etapa de opções, conforme ilustração na Figura 4.9. Nela, os parâmetros sugeridos são organizados nos mesmos grupos presentes nas Figuras 4.6 e 4.7 e podem ser consultados para selecionar a melhor escolha possível das 5 melhores sugestões de acordo com a pontuação calculada, podendo contestar ou não a melhor escolha sugerida pelo Gerenciador Autônomo.



(a) Opções para seleção da configuração arquitetural para execução.



(b) Parâmetros a serem utilizados para execução.

Figura 4.9: Seleção da configuração arquitetural após análise.

Ao clicar novamente no botão "Executar" localizado no canto superior direito, é feita uma requisição para executar o Módulo de ML com a opção selecionada, e a indicação de etapa é atualizada para a etapa de execução. Após a execução ser concluída, a indicação de etapa é atualizada para a etapa de resultados, conforme ilustração na Figura 4.10. Nela, os parâmetros sugeridos são organizados nos mesmos grupos presentes na Figura 4.9, mas desta vez refletem as métricas e pontuação da execução realizada pelo Módulo de ML.



(a) Módulo de ML em execução.



(b) Resultados da execução realizada.

Figura 4.10: Execução do Módulo de ML após seleção.

Após a implementação de todos os módulos presentes no arcabouço, foi possível realizar a execução de 680 treinamentos para armazenamento na base de conhecimento e obter a medição da pontuação de todas as configurações arquiteturais calculadas pelo Gerenciador Autônomo. Com isso, foi possível elaborar Estudos de Caso para verificar a viabilidade da solução proposta e a escolha das arquiteturas utilizadas.

Capítulo 5

Estudo de Caso 1: Classificação de Crédito (German Credit Dataset)

Este capítulo apresenta um Estudo de Caso executado pelo autor para verificar a viabilidade da solução proposta. Nele, ela foi usada para construir um modelo de classificação de crédito utilizando o conjunto de dados German Credit Dataset, que é um conjunto de dados de classificação binária com 1000 amostras. O conjunto de dados inclui variáveis como idade, emprego, histórico de crédito e renda. Os resultados obtidos e as discussões sobre esses resultados são analisados, permitindo verificar se a solução consegue extrair um equilíbrio entre qualidade e justiça.

5.1 Contexto e limitações

Neste Estudo de Caso, o real foco foi colocado em testar e verificar como o Gerenciador Autônomo se comporta em um cenário padrão de uso, com diversas execuções prévias do Módulo de ML. Como execuções anteriores estão armazenadas na Base de Conhecimento, o Gerenciador Autônomo pode executar uma análise através destes dados e determinar um plano indicando as configurações arquiteturais dos processos de treinamento de um modelo com melhores resultados para um determinado conjunto de dados. O Estudo de Caso foi realizado com o seguinte contexto:

- **Contexto:** Obter classificação de crédito de uma pessoa (apto ou não apto), por meio de uma série de *features*.
- **Conjunto de dados:** German Credit Dataset [13].
- **Transformações realizadas no conjunto de dados:** Mudanças nos valores dos atributos para valores de interpretação mais simples, para que o Cientista de Dados possa categorizar o dado de forma mais fácil no Módulo de ML.
- **Atributos protegidos:** Idade ou Nacionalidade
- **Grupo privilegiado:** Idade: Maior ou igual a 25 anos; Nacionalidade: Alemã

- **Grupo não-privilegiado:** Idade: Menor que 25 anos; Nacionalidade: Diferente da Alemã (Estrangeiro)

Para realizar este experimento, foi considerado o seguinte objetivo e consideradas as seguintes limitações:

- **Experimento:** Execução do Módulo de ML para obtenção de dados iniciais na base de conhecimento e discussão do Gerenciador Autônomo como facilitador da escolha das configurações arquiteturais para execução.
- **Medição:** Serão coletadas as pontuações de cada grupo de métricas (Avaliação e Fairness) após uma execução do Gerenciador Autônomo, cujos cálculos foram explicados anteriormente na seção 3.3. Tais pontuações também serão comparadas com os valores originais, presentes na Base de Conhecimento, para verificar a validade da pontuação.
- **Obtenção dos dados:** A execução do Módulo de ML utilizando o Gerenciador Autônomo foi realizada com 3 pesagens diferentes na pontuação geral:
 - 50% para métricas de Avaliação e 50% para métricas de Fairness, para uma configuração equilibrada.
 - 75% para métricas de Avaliação e 25% para métricas de Fairness, para uma configuração que prioriza a qualidade em detrimento da justiça.
 - 25% para métricas de Avaliação e 75% para métricas de Fairness, para uma configuração que prioriza a justiça em detrimento da qualidade.

São utilizadas as métricas Acurácia, Precisão, *Recall*, *F1-Score* e AUC como métricas de Avaliação 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 seus respectivos grupos.

- **Pré-condição 1:** Execuções anteriores foram realizadas e já formaram uma Base de Conhecimento. Para o experimento, foi considerado ao menos 1 execução para cada configuração arquitetural, composta por conjunto de dados, atributo protegido e algoritmos utilizados nos componentes do Módulo de ML (Pré-processamento, Processamento/Treinamento e Pós-processamento/Validação).
- **Pré-condição 2:** Cada execução do Módulo de ML gravada na Base de Conhecimento gerada para uso do Gerenciador Autônomo terá todas as métricas de Avaliação e métricas de *Fairness* mencionadas, além de sua configuração arquitetural utilizada.
- **Pré-condição 3:** Podem existir ruídos nos resultados finais devido ao German Credit Dataset ser uma base de dados com poucas amostras (apenas 1000), mas eles serão desconsiderados uma vez que o conjunto de dados ainda é considerado como *benchmark* em alguns trabalhos acadêmicos [47] [40] [32] e seus dados exemplificam

muito bem uma situação real onde dados sensíveis podem ser utilizados e são possíveis de afetar a decisão de um modelo e, conseqüentemente, a situação de vida de uma pessoa.

- **Restrição 1:** O algoritmo para redução de vieses é executado em apenas um dos componentes (Pré-processamento, Processamento/Treinamento ou Pós-processamento/Validação). Isto foi decidido pois não foram encontradas referências onde a aplicação desses algoritmos em duas ou em todas as três etapas impacta em melhora nas métricas de *Fairness*.
- **Restrição 2:** Por não conseguir rodar com sucesso, foram removidas as configurações arquiteturais que rodaram o algoritmo *Optimized Preprocessing*.
- **Restrição 3:** Configurações arquiteturais também foram removidas da análise do Gerenciador Autônomo por apresentarem métricas com valores máximos, para evitar análises caso exista alguma falha não detectada na implementação. Como exemplo, as que rodaram o algoritmo *Reject Option Classification*.
- **Restrição 4:** Para evitar configurações arquiteturais com métricas ruins e também pelo mesmo motivo da restrição anterior, o intervalo de pontuação para análise foi limitado de 500 a 950.

5.2 Resultados e Discussões

Os resultados baseados nas pré-condições e restrições já comentadas na seção anterior estão presentes abaixo nas Tabelas 5.1, 5.2 e 5.3:

Tabela 5.1: Melhores configurações escolhidas pelo Gerenciador Autônomo
Uso dos algoritmos implementados - 50% Avaliação/50% *Fairness*

Atributo protegido	Configuração arquitetural			Pontuação		Geral
	Pré-processamento	Treinamento	Pós-processamento	Avaliação	Fairness	
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	Reweighting	Gradient Boosting	Nenhum	804	931	868

Tabela 5.2: Melhores configurações escolhidas pelo Gerenciador Autônomo
 Uso dos algoritmos implementados - 75% Avaliação/25% *Fairness*

Atributo protegido	Configuração arquitetural			Pontuação		
	Pré-processamento	Treinamento	Pós-processamento	Avaliação	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 5.3: Melhores configurações escolhidas pelo Gerenciador Autônomo
 Uso dos algoritmos implementados - 25% Avaliação/75% *Fairness*

Atributo protegido	Configuração arquitetural			Pontuação		
	Pré-processamento	Treinamento	Pós-processamento	Avaliação	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	Reweighting	Support Vector Machines	Nenhum	755	972	918
Nacionalidade	Learning Fair Representations	Support Vector Machines	Nenhum	755	972	918

Nessas execuções, 2 observações são destacadas. A primeira é o fato da predominância de algoritmos com redução de viés para o componente de pós-processamento especialmente em configurações que priorizavam qualidade, contrariando o esperado de que os algoritmos com redução de viés aumentavam justiça em detrimento da qualidade. A segunda é a predominância de algoritmos com redução de viés para o componente de pré-processamento em configurações que priorizavam justiça, principalmente pois todas as execuções usavam *Support Vector Machines* como algoritmo de treinamento.

Diante destas 2 predominâncias envolvendo todas as configurações arquiteturais executadas, novos experimentos com restrições adicionais foram realizados para obter observações mais detalhadas a respeito dos resultados:

- Uso apenas de configurações arquiteturais com o uso de algoritmos com redução de viés para o componente de pré-processamento.
- Uso apenas de configurações arquiteturais com o uso de algoritmos com redução de viés para o componente de processamento.
- Uso apenas de configurações arquiteturais com o uso de algoritmos com redução de viés para o componente de pós-processamento.
- Uso apenas de configurações arquiteturais sem o uso de algoritmos com redução de viés.

As pré-condições, restrições e pesagens nas pontuações usadas anteriormente foram mantidas e seus resultados estão presentes abaixo nas Tabelas 5.4, 5.5, 5.6, 5.7, 5.8, 5.9, 5.10, 5.11, 5.12, 5.13, 5.14 e 5.15:

Tabela 5.4: Melhores configurações escolhidas pelo Gerenciador Autônomo
Algoritmos para redução de viés de pré-processamento - 50% Avaliação/50% *Fairness*

Atributo protegido	Configuração arquitetural			Pontuação		
	Pré-processamento	Treinamento	Pós-processamento	Avaliação	Fairness	Geral
Idade	Reweighting	Gradient Boosting	Nenhum	804	931	868
Idade	Learning Fair Representations	Gradient Boosting	Nenhum	804	931	868
Idade	Disparate Impact Remover	Support Vector Machines	Nenhum	747	989	868
Nacionalidade	Disparate Impact Remover	Support Vector Machines	Nenhum	747	989	868
Nacionalidade	Learning Fair Representations	Support Vector Machines	Nenhum	755	972	864

Tabela 5.5: Melhores configurações escolhidas pelo Gerenciador Autônomo
Algoritmos para redução de viés de pré-processamento - 75% Avaliação/25% *Fairness*

Atributo protegido	Configuração arquitetural			Pontuação		
	Pré-processamento	Treinamento	Pós-processamento	Avaliação	Fairness	Geral
Idade	Reweighting	Gradient Boosting	Nenhum	804	931	836
Idade	Learning Fair Representations	Gradient Boosting	Nenhum	804	931	836
Nacionalidade	Learning Fair Representations	Gradient Boosting	Nenhum	811	878	828
Nacionalidade	Reweighting	Gradient Boosting	Nenhum	811	878	828
Nacionalidade	Learning Fair Representations	Random Forest	Nenhum	801	883	821

Tabela 5.6: Melhores configurações escolhidas pelo Gerenciador Autônomo
Algoritmos para redução de viés de pré-processamento - 25% Avaliação/75% *Fairness*

Atributo protegido	Configuração arquitetural			Pontuação		
	Pré-processamento	Treinamento	Pós-processamento	Avaliação	Fairness	Geral
Idade	Disparate Impact Remover	Support Vector Machines	Nenhum	747	989	928
Nacionalidade	Disparate Impact Remover	Support Vector Machines	Nenhum	747	989	928
Nacionalidade	Learning Fair Representations	Support Vector Machines	Nenhum	755	972	918
Nacionalidade	Reweighting	Support Vector Machines	Nenhum	755	972	918
Idade	Learning Fair Representations	Support Vector Machines	Nenhum	755	969	916

Nas configurações arquiteturais utilizando apenas algoritmos para redução de viés de pré-processamento, percebe-se a predominância dos algoritmos de treinamento *Gradient Boosting* e *Support Vector Machines*, sendo o *Gradient Boosting* predominante em configurações priorizando qualidade 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 qualidade ou apenas em justiça, uma vez que há exemplos de conjuntos de algoritmos com pontuações ligeiramente maiores em Avaliação 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.

Tabela 5.7: Melhores configurações escolhidas pelo Gerenciador Autônomo
Algoritmos para redução de viés de processamento - 50% Avaliação/50% *Fairness*

Atributo protegido	Configuração arquitetural			Pontuação		Geral
	Pré-processamento	Treinamento	Pós-processamento	Avaliação	Fairness	
Idade	Nenhum	Adversarial Debiasing	Nenhum	742	979	860
Nacionalidade	Nenhum	Grid Search Reduction	Nenhum	789	895	845
Idade	Nenhum	Meta Fair Classifier	Nenhum	776	910	843
Idade	Nenhum	Exponentiated Gradient Reduction	Nenhum	811	869	840
Nacionalidade	Nenhum	Rich Subgroup Fairness	Nenhum	791	856	824

Tabela 5.8: Melhores configurações escolhidas pelo Gerenciador Autônomo
Algoritmos para redução de viés de processamento - 75% Avaliação/25% *Fairness*

Atributo protegido	Configuração arquitetural			Pontuação		Geral
	Pré-processamento	Treinamento	Pós-processamento	Avaliação	Fairness	
Idade	Nenhum	Exponentiated Gradient Reduction	Nenhum	811	869	826
Nacionalidade	Nenhum	Grid Search Reduction	Nenhum	789	895	820
Idade	Nenhum	Meta Fair Classifier	Nenhum	776	910	809
Nacionalidade	Nenhum	Exponentiated Gradient Reduction	Nenhum	807	810	808
Nacionalidade	Nenhum	Rich Subgroup Fairness	Nenhum	791	856	807

Tabela 5.9: Melhores configurações escolhidas pelo Gerenciador Autônomo
Algoritmos para redução de viés de processamento - 25% Avaliação/75% *Fairness*

Atributo protegido	Configuração arquitetural			Pontuação		Geral
	Pré-processamento	Treinamento	Pós-processamento	Avaliação	Fairness	
Idade	Nenhum	Adversarial Debiasing	Nenhum	742	979	920
Idade	Nenhum	Meta Fair Classifier	Nenhum	776	910	876
Nacionalidade	Nenhum	Grid Search Reduction	Nenhum	795	895	870
Idade	Nenhum	Exponentiated Gradient Reduction	Nenhum	811	869	854
Nacionalidade	Nenhum	Prejudice Remover	Nenhum	770	874	848

Nas configurações arquiteturais utilizando apenas algoritmos para redução de viés de processamento, 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*. Nelas, as 2 observações percebidas em configurações arquiteturais 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 5.8, onde o uso da Nacionalidade como atributo protegido possui pontuações de Avaliação e *Fairness* piores que a Idade e concluindo que o processamento utilizado no atributo protegido pode afetar todas as métricas.

Tabela 5.10: Melhores configurações escolhidas pelo Gerenciador Autônomo
Algoritmos para redução de viés de pós-processamento - 50% Avaliação/50% *Fairness*

Atributo protegido	Configuração arquitetural			Pontuação		
	Pré-processamento	Treinamento	Pós-processamento	Avaliação	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	Nenhum	Random Forest	Equalized Odds	898	799	849

Tabela 5.11: Melhores configurações escolhidas pelo Gerenciador Autônomo
Algoritmos para redução de viés de pós-processamento - 75% Avaliação/25% *Fairness*

Atributo protegido	Configuração arquitetural			Pontuação		
	Pré-processamento	Treinamento	Pós-processamento	Avaliação	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 5.12: Melhores configurações escolhidas pelo Gerenciador Autônomo
Algoritmos para redução de viés de pós-processamento - 25% Avaliação/75% *Fairness*

Atributo protegido	Configuração arquitetural			Pontuação		
	Pré-processamento	Treinamento	Pós-processamento	Avaliação	Fairness	Geral
Nacionalidade	Nenhum	Random Forest	Calibrated Equalized Odds	902	922	917
Nacionalidade	Nenhum	Gradient Boosting	Calibrated Equalized Odds	870	925	911
Idade	Nenhum	Regressão Logística	Equalized Odds	968	860	887
Idade	Nenhum	Gradient Boosting	Equalized Odds	927	862	878
Idade	Nenhum	Random Forest	Equalized Odds	898	799	824

Nas configurações arquiteturais utilizando apenas algoritmos para redução de viés de pós-processamento, surpreende o fato de que as configurações arquiteturais obtiveram as melhores pontuações em Avaliação 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 Avaliação, tais configurações arquiteturais possuem um maior equilíbrio entre Avaliação e *Fairness* e acabam garantindo maiores pontuações na média, justificando as melhores pontuações nas primeiras execuções onde foram considerados Uso dos algoritmos implementados. Fora isto, as demais observações anteriores também se aplicam nestas execuções.

Tabela 5.13: Melhores configurações escolhidas pelo Gerenciador Autônomo
Sem uso de algoritmos para redução de viés - 50% Avaliação/50% *Fairness*

Atributo protegido	Configuração arquitetural			Pontuação		
	Pré-processamento	Treinamento	Pós-processamento	Avaliação	Fairness	Geral
Nacionalidade	Nenhum	Support Vector Machines	Nenhum	755	972	864
Idade	Nenhum	Support Vector Machines	Nenhum	755	969	862
Nacionalidade	Nenhum	Random Forest	Nenhum	802	885	844
Nacionalidade	Nenhum	Regressão Logística	Nenhum	782	865	824
Nacionalidade	Nenhum	Gradient Boosting	Nenhum	817	784	800

Tabela 5.14: Melhores configurações escolhidas pelo Gerenciador Autônomo
Sem uso de algoritmos para redução de viés - 75% Avaliação/25% *Fairness*

Atributo protegido	Configuração arquitetural			Pontuação		
	Pré-processamento	Treinamento	Pós-processamento	Avaliação	Fairness	Geral
Nacionalidade	Nenhum	Random Forest	Nenhum	802	885	823
Nacionalidade	Nenhum	Gradient Boosting	Nenhum	817	784	809
Nacionalidade	Nenhum	Support Vector Machines	Nenhum	755	972	809
Idade	Nenhum	Support Vector Machines	Nenhum	755	969	808
Idade	Nenhum	Gradient Boosting	Nenhum	817	778	807

Tabela 5.15: Melhores configurações escolhidas pelo Gerenciador Autônomo
Sem uso de algoritmos para redução de viés - 25% Avaliação/75% *Fairness*

Atributo protegido	Configuração arquitetural			Pontuação		
	Pré-processamento	Treinamento	Pós-processamento	Avaliação	Fairness	Geral
Nacionalidade	Nenhum	Support Vector Machines	Nenhum	755	972	918
Idade	Nenhum	Support Vector Machines	Nenhum	755	969	916
Nacionalidade	Nenhum	Random Forest	Nenhum	802	885	864
Nacionalidade	Nenhum	Regressão Logística	Nenhum	782	865	844
Idade	Nenhum	Regressão Logística	Nenhum	782	802	797

Olhando as configurações arquiteturais sem algoritmos para redução de viés, é curioso notar que, ao comparar com configurações arquiteturais 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 Avaliação são ligeiramente maiores e as pontuações em *Fairness* são ligeiramente menores. É possível notar uma exceção na configuração arquitetural envolvendo o algoritmo *Random Forest*, mas nos outros casos a hipótese é verificada com sucesso. Ao comparar com configurações arquiteturais 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 configurações arquiteturais usando algoritmos com redução de viés no resultado, a hipótese não se confirma devido a observação da grande melhora por parte das métricas

de Avaliação nas configurações arquiteturais 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 configurações arquiteturais envolvendo *Support Vector Machines* que são melhores nas configurações arquiteturais sem algoritmos com redução de viés, mas no uso de outros algoritmos ocorre melhora na pontuação em *Fairness*.

Após a verificação das pontuações, foram catalogadas todas as métricas de todos os conjuntos de algoritmos em seus valores máximo, mínimo e médio para verificar a eficácia destas pontuações, definidas nas Tabelas 5.16 e 5.17 exibidas abaixo.

Tabela 5.16: Métricas de Avaliação das execuções no Módulo de ML

Atributo protegido	Configuração arquitetural			Métricas									
	Pré-processamento	Treinamento	Pos-processamento	Acurácia		Precisão		Recall		F1-Score		AUC	
Nacionalidade	Nenhum	Support Vector Machines	Nenhum	Mínimo	0.715	Mínimo	0.7143	Mínimo	0.9929	Mínimo	0.8309	Mínimo	0.5219
				Máximo	0.715	Máximo	0.7143	Máximo	0.9929	Máximo	0.8309	Máximo	0.5219
				Média	0.715	Média	0.7143	Média	0.9929	Média	0.8309	Média	0.5219
Idade	Nenhum	Support Vector Machines	Nenhum	Mínimo	0.715	Mínimo	0.7143	Mínimo	0.9929	Mínimo	0.8309	Mínimo	0.5219
				Máximo	0.715	Máximo	0.7143	Máximo	0.9929	Máximo	0.8309	Máximo	0.5219
				Média	0.715	Média	0.7143	Média	0.9929	Média	0.8309	Média	0.5219
Nacionalidade	Nenhum	Random Forest	Nenhum	Mínimo	0.76	Mínimo	0.7784	Mínimo	0.9007	Mínimo	0.8442	Mínimo	0.6474
				Máximo	0.79	Máximo	0.8037	Máximo	0.9291	Máximo	0.8618	Máximo	0.6933
				Média	0.774	Média	0.7933	Média	0.9192	Média	0.8515	Média	0.6731
Nacionalidade	Nenhum	Regressão Logística	Nenhum	Mínimo	0.75	Mínimo	0.7661	Mínimo	0.9291	Mínimo	0.8397	Mínimo	0.6256
				Máximo	0.75	Máximo	0.7661	Máximo	0.9291	Máximo	0.8397	Máximo	0.6256
				Média	0.75	Média	0.7661	Média	0.9291	Média	0.8397	Média	0.6256
Nacionalidade	Nenhum	Gradient Boosting	Nenhum	Mínimo	0.79	Mínimo	0.8194	Mínimo	0.9007	Mínimo	0.8581	Mínimo	0.7131
				Máximo	0.79	Máximo	0.8194	Máximo	0.9007	Máximo	0.8581	Máximo	0.7131
				Média	0.79	Média	0.8194	Média	0.9007	Média	0.8581	Média	0.7131
Idade	Nenhum	Gradient Boosting	Nenhum	Mínimo	0.79	Mínimo	0.8235	Mínimo	0.8936	Mínimo	0.8571	Mínimo	0.718
				Máximo	0.79	Máximo	0.8235	Máximo	0.8936	Máximo	0.8571	Máximo	0.718
				Média	0.79	Média	0.8235	Média	0.8936	Média	0.8571	Média	0.718
Idade	Nenhum	Regressão Logística	Nenhum	Mínimo	0.75	Mínimo	0.7661	Mínimo	0.9291	Mínimo	0.8397	Mínimo	0.6256
				Máximo	0.75	Máximo	0.7661	Máximo	0.9291	Máximo	0.8397	Máximo	0.6256
				Média	0.75	Média	0.7661	Média	0.9291	Média	0.8397	Média	0.6256
Idade	Reweighing	Gradient Boosting	Nenhum	Mínimo	0.775	Mínimo	0.8	Mínimo	0.9078	Mínimo	0.8505	Mínimo	0.6827
				Máximo	0.775	Máximo	0.8	Máximo	0.9078	Máximo	0.8505	Máximo	0.6827
				Média	0.775	Média	0.8	Média	0.9078	Média	0.8505	Média	0.6827
Idade	Learning Fair Representations	Gradient Boosting	Nenhum	Mínimo	0.775	Mínimo	0.8	Mínimo	0.9078	Mínimo	0.8505	Mínimo	0.6827
				Máximo	0.775	Máximo	0.8	Máximo	0.9078	Máximo	0.8505	Máximo	0.6827
				Média	0.775	Média	0.8	Média	0.9078	Média	0.8505	Média	0.6827
Idade	Disparate Impact Remover	Support Vector Machines	Nenhum	Mínimo	0.705	Mínimo	0.705	Mínimo	1	Mínimo	0.827	Mínimo	0.5
				Máximo	0.705	Máximo	0.705	Máximo	1	Máximo	0.827	Máximo	0.5
				Média	0.705	Média	0.705	Média	1	Média	0.827	Média	0.5
Nacionalidade	Disparate Impact Remover	Support Vector Machines	Nenhum	Mínimo	0.705	Mínimo	0.705	Mínimo	1	Mínimo	0.827	Mínimo	0.5
				Máximo	0.705	Máximo	0.705	Máximo	1	Máximo	0.827	Máximo	0.5
				Média	0.705	Média	0.705	Média	1	Média	0.827	Média	0.5
Nacionalidade	Learning Fair Representations	Support Vector Machines	Nenhum	Mínimo	0.715	Mínimo	0.7143	Mínimo	0.9929	Mínimo	0.8309	Mínimo	0.5219
				Máximo	0.715	Máximo	0.7143	Máximo	0.9929	Máximo	0.8309	Máximo	0.5219
				Média	0.715	Média	0.7143	Média	0.9929	Média	0.8309	Média	0.5219
Nacionalidade	Learning Fair Representations	Gradient Boosting	Nenhum	Mínimo	0.785	Mínimo	0.8063	Mínimo	0.9149	Mínimo	0.8571	Mínimo	0.6947
				Máximo	0.785	Máximo	0.8063	Máximo	0.9149	Máximo	0.8571	Máximo	0.6947
				Média	0.785	Média	0.8063	Média	0.9149	Média	0.8571	Média	0.6947
Nacionalidade	Reweighing	Gradient Boosting	Nenhum	Mínimo	0.785	Mínimo	0.8063	Mínimo	0.9149	Mínimo	0.8571	Mínimo	0.6947
				Máximo	0.785	Máximo	0.8063	Máximo	0.9149	Máximo	0.8571	Máximo	0.6947
				Média	0.785	Média	0.8063	Média	0.9149	Média	0.8571	Média	0.6947
Nacionalidade	Learning Fair Representations	Random Forest	Nenhum	Mínimo	0.76	Mínimo	0.7831	Mínimo	0.8936	Mínimo	0.84	Mínimo	0.6559
				Máximo	0.79	Máximo	0.8113	Máximo	0.922	Máximo	0.86	Máximo	0.7032
				Média	0.772	Média	0.7949	Média	0.9121	Média	0.8494	Média	0.6746
Nacionalidade	Reweighing	Support Vector Machines	Nenhum	Mínimo	0.715	Mínimo	0.7143	Mínimo	0.9929	Mínimo	0.8309	Mínimo	0.5219
				Máximo	0.715	Máximo	0.7143	Máximo	0.9929	Máximo	0.8309	Máximo	0.5219
				Média	0.715	Média	0.7143	Média	0.9929	Média	0.8309	Média	0.5219
Idade	Learning Fair Representations	Support Vector Machines	Nenhum	Mínimo	0.715	Mínimo	0.7143	Mínimo	0.9929	Mínimo	0.8309	Mínimo	0.5219
				Máximo	0.715	Máximo	0.7143	Máximo	0.9929	Máximo	0.8309	Máximo	0.5219
				Média	0.715	Média	0.7143	Média	0.9929	Média	0.8309	Média	0.5219
Idade	Nenhum	Adversarial Debiasing	Nenhum	Mínimo	0.295	Mínimo	0	Mínimo	0	Mínimo	0	Mínimo	0.5
				Máximo	0.705	Máximo	0.7097	Máximo	1	Máximo	0.827	Máximo	0.5105
				Média	0.6317	Média	0.5888	Média	0.8168	Média	0.6812	Média	0.503
Nacionalidade	Nenhum	Grid Search Reduction	Nenhum	Mínimo	0.75	Mínimo	0.7791	Mínimo	0.9007	Mínimo	0.8355	Mínimo	0.6453
				Máximo	0.78	Máximo	0.7939	Máximo	0.9362	Máximo	0.8562	Máximo	0.6764
				Média	0.765	Média	0.7857	Média	0.9167	Média	0.8161	Média	0.6596
Idade	Nenhum	Meta Fair Classifier	Nenhum	Mínimo	0.73	Mínimo	0.7278	Mínimo	0.9291	Mínimo	0.8374	Mínimo	0.5522
				Máximo	0.755	Máximo	0.7661	Máximo	0.9858	Máximo	0.8474	Máximo	0.6256
				Média	0.742	Média	0.7466	Média	0.9617	Média	0.8402	Média	0.5893
Idade	Nenhum	Exponentiated Gradient Reduction	Nenhum	Mínimo	0.785	Mínimo	0.8063	Mínimo	0.9149	Mínimo	0.8571	Mínimo	0.6947
				Máximo	0.785	Máximo	0.8063	Máximo	0.9149	Máximo	0.8571	Máximo	0.6947
				Média	0.785	Média	0.8063	Média	0.9149	Média	0.8571	Média	0.6947
Nacionalidade	Nenhum	Rich Subgroup Fairness	Nenhum	Mínimo	0.76	Mínimo	0.808	Mínimo	0.8653	Mínimo	0.8356	Mínimo	0.6869
				Máximo	0.76	Máximo	0.808	Máximo	0.8653	Máximo	0.8356	Máximo	0.6869
				Média	0.76	Média	0.808	Média	0.8653	Média	0.8356	Média	0.6869
Nacionalidade	Nenhum	Exponentiated Gradient Reduction	Nenhum	Mínimo	0.775	Mínimo	0.8038	Mínimo	0.9007	Mínimo	0.8495	Mínimo	0.6876
				Máximo	0.785	Máximo	0.8101	Máximo	0.9078	Máximo	0.8562	Máximo	0.6997
				Média	0.7788	Média	0.8057	Média	0.9043	Média	0.8521	Média	0.6915
Nacionalidade	Nenhum	Prejudice Remover	Nenhum	Mínimo	0.735	Mínimo	0.7683	Mínimo	0.8936	Mínimo	0.8262	Mínimo	0.6248
				Máximo	0.735	Máximo	0.7683	Máximo	0.8936	Máximo	0.8262	Máximo	0.6248
				Média	0.735	Média	0.7683	Média	0.8936	Média	0.8262	Média	0.6248
Idade	Nenhum	Regressão Logística	Equalized Odds	Mínimo	0.965	Mínimo	0.9589	Mínimo	0.9929	Mínimo	0.9756	Mínimo	0.9456
				Máximo	0.965	Máximo	0.9589	Máximo	0.9929	Máximo	0.9756	Máximo	0.9456
				Média	0.965	Média	0.9589	Média	0.9929	Média	0.9756	Média	0.9456
Nacionalidade	Nenhum	Random Forest	Calibrated Equalized Odds	Mínimo	0.82	Mínimo	0.7966	Mínimo	1	Mínimo	0.8868	Mínimo	0.6919
				Máximo	0.93	Máximo	0.9097	Máximo	1	Máximo	0.9527	Máximo	0.8814
				Média	0.8925	Média	0.87	Média	1	Média	0.9299	Média	0.8178
Nacionalidade	Nenhum	Gradient Boosting	Calibrated Equalized Odds	Mínimo	0.84	Mínimo	0.8150	Mínimo	1	Mínimo	0.8981	Mínimo	0.7288
				Máximo	0.875	Máximo	0.8494	Máximo	1	Máximo	0.9186	Máximo	0.7881
				Média	0.855	Média	0.8296	Média	1	Média	0.9068	Média	0.7542
Idade	Nenhum	Gradient Boosting	Equalized Odds	Mínimo	0.905	Mínimo	0.9769	Mínimo	0.8723	Mínimo	0.9283	Mínimo	0.9249
				Máximo	0.915	Máximo	0.9919	Máximo	0.9007	Máximo	0.9373	Máximo	0.9277
				Média	0.9083	Média	0.9869	Média	0.8818	Média	0.9313	Média	0.9268
Idade	Nenhum	Random Forest	Equalized Odds	Mínimo	0.8	Mínimo	0.9697	Mínimo	0.7234	Mínimo	0.8361	Mínimo	0.8532
				Máximo	0.915	Máximo	0.9903	Máximo	0.9078	Máximo	0.9377	Máximo	0.92
				Média	0.8733	Média	0.9789	Média	0.8392	Média	0.9011	Média	0.897

independente de utilizar algoritmos para redução de viés ou não em sua configuração arquitetural.

A análise dos resultados das Tabelas 5.16 e 5.17 confirma que o cálculo utilizado no Analisador do Gerenciador Autônomo foi capaz de consolidar as métricas existentes de forma eficiente, proporcionando uma visão geral das configurações arquiteturais utilizadas sem comprometer a interpretação dos resultados. Analisando as métricas isoladamente, é difícil identificar qual configuração arquitetural oferece o melhor equilíbrio entre dois grupos de métricas com contextos completamente diferentes, diante da grande quantidade de métricas, a diferença extremamente pequena entre os resultados e a grande quantidade de configurações arquiteturais. Nesse contexto, a consolidação das métricas em grupos simplifica a visualização de quais configurações arquiteturais são mais equilibradas, 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 significativamente 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.

Capítulo 6

Estudo de Caso 2: Classificação de Crédito (Lendingclub Dataset) e evolução do sistema

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 Autônomo 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 Módulo de ML sem grandes deteriorações nas ideias originais de seu desenvolvimento.

6.1 Contexto e limitações

O experimento foi realizado com o seguinte conjunto de dados e dado o seguinte contexto:

- **Contexto:** Obter classificação de crédito de uma pessoa (apto ou não apto), por meio de uma série de *features*.
- **Conjunto de dados:** Lendingclub Dataset [6].
- **Transformações realizadas no conjunto de dados:** Filtragem de linhas que não definiam classificação boa ou ruim de crédito; seleção de 20 *features* utilizando algoritmo de informação mútua [63], mais uma *feature* para caracterizar dado sensível (atributo protegido) e *feature* de classificação de crédito, totalizando 22 *features* no total.
- **Atributos protegidos:** Renda
- **Grupo privilegiado:** Renda de 1 ou mais salários mínimos
- **Grupo não-privilegiado:** Renda de menos de 1 salário mínimo

Para realizar este Estudo de Caso, foi considerado o seguinte objetivo e consideradas as seguintes limitações:

- **Experimento:** Realização das mesmas condições do Estudo de Caso 1 para discussão da manutenção do sistema e reforço da viabilidade do Gerenciador Autônomo em casos mais próximos do mundo real.
- **Medição:** Além das medições do Estudo de Caso 1, serão medidas a quantidade de linhas modificadas e sua relação com a quantidade de linhas escritas.
- **Obtenção dos dados:** A execução do Módulo de ML utilizando o Gerenciador Autônomo foi realizada nas mesmas condições do Estudo de Caso 1. A contagem de linhas e arquivos foi realizada executando o comando `find ./src -name '*.py' | xargs wc -l` para os arquivos Python e o comando `find ./ml-ui/src -name '*.js' | xargs wc -l` para os arquivos Javascript presentes no projeto, excluindo-se os arquivos `__init__.py` que não possuem linhas de código e são criados apenas para o Python utilizar códigos de arquivos que estão dentro de outras pastas.
- **Pré-condições:** Por se tratar de um outro conjunto de dados, a pré-condição 3 foi desconsiderada. as pré-condições 1 e 2 do Estudo de Caso 1 foram mantidas.
- **Restrições:** As 4 restrições determinadas no Estudo de Caso 1 foram mantidas, com uma diferença: Na Restrição 4, como o *Lendingclub Dataset* possui métricas com valores melhores do que as apresentadas no *German Credit Dataset*, o intervalo de pontuação para análise foi ampliado de 500 a 980, mas as motivações para esse valor permanecem as mesmas.

6.2 Resultados e Discussões

Os resultados baseados nas pré-condições e restrições já comentadas na seção anterior estão presentes abaixo nas Tabelas 6.1, 6.2 e 6.3:

Tabela 6.1: Melhores configurações escolhidas pelo Gerenciador Autônomo
Uso dos algoritmos implementados - 50% Avaliação/50% *Fairness*

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

Tabela 6.2: Melhores configurações escolhidas pelo Gerenciador Autônomo
 Uso dos algoritmos implementados - 75% Avaliação/25% *Fairness*

Atributo protegido	Configuração arquitetural			Pontuação		
	Pré-processamento	Treinamento	Pós-processamento	Avaliação	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	Reweighting	Regressão Logística	Nenhum	981	965	977

Tabela 6.3: Melhores configurações escolhidas pelo Gerenciador Autônomo
 Uso dos algoritmos implementados - 25% Avaliação/75% *Fairness*

Atributo protegido	Configuração arquitetural			Pontuação		
	Pré-processamento	Treinamento	Pós-processamento	Avaliação	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	Reweighting	Gradient Boosting	Nenhum	987	964	970

Nessas execuções, a principal observação notada foi a mudança de configurações arquiteturais com melhores desempenhos, com predominância do algoritmo *Learning Fair Representations* para redução de viés e Regressão Logística para algoritmo de treinamento. Também se nota que algoritmos para redução de viés de pós-processamento e algoritmos de treinamento como *Support Vector Machines* não foram tão eficientes quanto no Estudo de Caso anterior. Com isso, pode-se concluir que, ao mudar o contexto do problema e os dados envolvidos, o Gerenciador Autônomo 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 adicionar este conjunto de dados como opção no Módulo de ML e realizar a evolução do sistema com base nesta adição. Estas foram contadas de acordo com seus *commits* realizados no repositório e exibidos na Tabela 6.4:

Tabela 6.4: Quantidade de modificações realizadas ao adicionar um novo conjunto de dados ao Módulo de ML

Módulo	Quantidade de linhas alteradas	Total de linhas	Quantidade de arquivos alterados	Total de arquivos	Porcentagem de linhas alteradas	Porcentagem de arquivos alterados
Engenharia de Dados	122	277	2	3	44,04%	66,67%
Módulo de ML	76	1982	5	38	3,84%	13,16%
Gerenciador Autônomo	4	889	1	17	0,45%	5,88%
Interface	13	2905	2	14	0,45%	14,29%
TOTAL	215	6053	10	72	3,55%	13,89%

A primeira conclusão que é possível discutir é relativa à modificação no Gerenciador Autônomo, que se baseou em parametrizações adicionais para a integração entre o Gerenciador Autônomo e a Interface e não afetou os componentes baseados na arquitetura MAPE-K. Este fato e a grande diferença entre os resultados dos Estudo de Caso 1 e 2 reforçam a autonomia proposta neste módulo, possibilitando configurações diferentes baseadas nos metadados capturados no Módulo de ML.

Os elementos na Interface exigiram poucas modificações, podendo ser resumidos a simples adições para colocar a opção do novo conjunto de dados. As maiores modificações foram realizadas no Módulo de ML e, principalmente, no módulo de Engenharia de Dados. Das modificações no Módulo 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 arquitetura *Pipes-and-Filters* tenha a consequência de escrever algumas linhas a mais para elas devido a quantidade de classes criadas no Módulo de ML (2 para pipes e 1 para filtro), etapas de processamento e transformação dos dados serão o foco de tempo e modificações por parte dos Engenheiros e Cientistas de Dados.

O uso da arquitetura *Pipes-and-Filters* permite o encapsulamento dos algoritmos e a separação de interesses de forma simples, fazendo com que o código presente no Módulo de ML possa ter escolhas mais elegantes para um bom *Design* do código. Como exemplo disso há o Trecho 6.1, onde há grande flexibilidade para configurar o método de pós-processamento, podendo ser expandido conforme novas classes de filtro forem implementadas com apenas a adição de um novo item no array **unbias_postproc_options**.

```

1  def data_postprocess(self, unbias_postproc_algorithm,
2                          test_pipe, prediction_pipe, fairness_pipe):
3      unbias_postproc_options = [
4          (UnbiasPostProcAlgorithms.EQUALIZED_ODDS,
5           EqualizedOddsFilter()),
6          (UnbiasPostProcAlgorithms.CALIBRATED_EQUALIZED_ODDS,
7           CalibratedEqualizedOddsFilter()),
8          (UnbiasPostProcAlgorithms.REJECT_OPTION_CLASSIFICATION,
9           RejectOptionClassificationFilter())
10     ]
11
12     for option, filter in unbias_postproc_options:
13         if unbias_postproc_algorithm == option:
14             init_pipe = test_pipe + prediction_pipe +
15                 fairness_pipe[
16                     'unprivileged_group',
17                     'privileged_group'
18                 ]
19             init_pipe >= filter == prediction_pipe
20             break
21
22     return prediction_pipe

```

Código 6.1: Método para seleção de algoritmo para redução de viés de pós-processamento

Dito isso, realizar a manutenção/evolução do sistema 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.

Capítulo 7

Estudo de Caso 3: Evolução do sistema com outros desenvolvedores

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 concentra na avaliação da versatilidade e simplicidade das arquiteturas escolhidas para permitir que outros desenvolvedores entendam o contexto do sistema e façam novas evoluções.

7.1 Contexto e limitações

O experimento foi realizado com o seguinte conjunto de dados e dado o seguinte contexto:

- **Contexto:** Obter classificação de crédito de uma pessoa (apto ou não apto), por meio de uma série de *features*.
- **Conjunto de dados:** Lendingclub Dataset [6].
- **Transformações realizadas no conjunto de dados:** Filtragem de linhas que não definiam classificação boa ou ruim de crédito; seleção de 20 *features* utilizando algoritmo de informação mútua [63], mais uma *feature* para caracterizar dado sensível (atributo protegido) e *feature* de classificação de crédito, totalizando 22 *features* no total.
- **Atributos protegidos:** Renda
- **Grupo privilegiado:** Renda de 1 ou mais salários mínimos
- **Grupo não-privilegiado:** Renda de menos de 1 salário mínimo

Para realizar este Estudo de Caso, foi considerado o seguinte objetivo e consideradas as seguintes limitações:

- **Experimento:** Realização das mesmas condições do Estudo de Caso 2 para discussão da manutenção do sistema, porém com outro desenvolvedor adicionando um novo algoritmo.

- **Medição:** As mesmas medições do Estudo de Caso 2, atualizadas para o contexto deste Estudo de Caso.
- **Obtenção dos dados:** Mesmas condições do Estudo de Caso 2, observações do desenvolvedor através de um questionário, e pontos de melhoria obtidos com o mesmo durante a sessão de desenvolvimento.
- **Pré-condições:** Mesmas pré-condições do Estudo de Caso 2.
- **Restrições:** Mesmas restrições do Estudo de Caso 2.

7.2 Resultados e Discussões

Foi realizada uma sessão de desenvolvimento com um outro desenvolvedor, com a duração de aproximadamente 1 hora e 40 minutos, onde este adicionou um novo algoritmo de classificação dentro do Módulo de ML e realizou a evolução do sistema com a documentação montada durante o Estudo de Caso anterior. Durante a sessão, foram notados e corrigidos, através de observação e *feedbacks* do desenvolvedor, os seguintes itens:

- **Adaptações para Linux** - Como uma das máquinas que testamos rodava o sistema operacional Linux, certas estruturas de pastas mudam em relação ao Windows e tais mudanças foram atualizadas na documentação.
- **Fixar versão de biblioteca** - Durante o período de testes, uma biblioteca foi atualizada e isto causou incompatibilidades com outras. A solução foi fixá-la no arquivo onde elas são informadas.
- **Instalação em PCs sem GPU Nvidia e com GPU Nvidia** - Como uma das máquinas que testamos não tinha GPU Nvidia, foi colocado um modo apenas para uso de CPU. Para máquinas com GPUs Nvidia, foi colocada na documentação a orientação para instalar o CUDA Toolkit.
- **Correções de bugs** - Foram encontrados 2 bugs durante o desenvolvimento, 1 na Interface e outro no Gerenciador Autônomo, por falta de detalhes na documentação. Foram detectados rapidamente, corrigidos e suas causas foram adicionadas na documentação.
- **Correções na documentação** - Foram colocados modificações em métodos que faltaram constar na documentação para obter certas parametrizações dos módulos presentes na solução, além de deixar mais explícito onde cada arquivo se encontra.

Por causa de alguns itens presentes acima, 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 presente no Apêndice C, 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.

Outro item foi notado durante todo o desenvolvimento do sistema: alertas de vulnerabilidades das bibliotecas utilizadas no projeto e disponibilizados no GitHub. Isso reforça o fato de que o processo de Engenharia de Software para evolução do sistema é contínuo, mas que existem certas situações que estão fora do seu escopo. A própria AIF360 é uma biblioteca que implementa vários algoritmos com redução de viés baseadas em artigos disponibilizados na comunidade acadêmica mas ainda peca pela não atualização dos mesmos, fazendo com que muitos algoritmos utilizados necessitem de bibliotecas desatualizadas para funcionar.

O algoritmo de classificação escolhido para o desenvolvimento foi o Naive Bayes [77], 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 7.1, 7.2 e 7.3:

Tabela 7.1: Melhores configurações escolhidas pelo Gerenciador Autônomo
Uso dos algoritmos implementados - 50% Avaliação/50% *Fairness*

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

Tabela 7.2: Melhores configurações escolhidas pelo Gerenciador Autônomo
Uso dos algoritmos implementados - 75% Avaliação/25% *Fairness*

Atributo protegido	Configuração arquitetural			Pontuação		Geral
	Pré-processamento	Treinamento	Pós-processamento	Avaliação	Fairness	
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	Reweighting	Regressão Logística	Nenhum	981	965	977

Tabela 7.3: Melhores configurações escolhidas pelo Gerenciador Autônomo
Uso dos algoritmos implementados - 25% Avaliação/75% *Fairness*

Atributo protegido	Configuração arquitetural			Pontuação		Geral
	Pré-processamento	Treinamento	Pós-processamento	Avaliação	Fairness	
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 Módulo de ML 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 mostraram evidências do porquê o Naive Bayes teve um comportamento positivo para este conjunto de dados.

As evoluções realizadas no Módulo de ML para adicionar o Naive Bayes foram contadas de acordo com seus *commits* realizados no repositório e exibidos na Tabela 6.4:

Tabela 7.4: Quantidade de modificações realizadas ao adicionar um novo algoritmo ao Módulo de ML

Módulo	Quantidade de linhas alteradas	Total de linhas	Quantidade de arquivos alterados	Total de arquivos	Porcentagem de linhas alteradas	Porcentagem de arquivos alterados
Engenharia de Dados	0	277	0	3	0,00%	0,00%
Módulo de ML	60	2042	5	39	2,94%	12,82%
Gerenciador Autônomo	2	891	2	17	0,23%	11,77%
Interface	71	2948	3	14	2,41%	21,43%
TOTAL	132	6157	9	72	2,14%	12,50%

Desta vez, o único módulo do sistema sem necessidade de modificações foi o módulo de Engenharia de Dados. Entretanto, a modificação necessária no Gerenciador Autônomo foi a adição do Naive Bayes como 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. O único módulo que fugiu de tal observação que exigiu mais modificações foi a Interface, em grande parte por causa de um único componente presente na tela de Configurações para Planejamento do Gerenciador Autônomo. 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.

Capítulo 8

Conclusões

Neste projeto, a AI Reference Architecture permitiu visualizar com clareza quais pessoas, quais etapas e projetos para desenvolvimento de funcionalidades e aplicações são necessários para ir da obtenção dos dados, passando pela implantação do modelo até chegar ao consumo pelo cliente final. Deste modo, é possível traçar melhores planejamentos para desenvolvimento de uma aplicação baseada em Inteligência Artificial.

Pode-se dizer também que o conjunto da arquitetura de *Pipes-and-Filters* com a arquitetura MAPE-K se adaptou muito bem na implementação dos objetivos principais. Com a arquitetura *Pipes-and-Filters*, foi possível encapsular todos os procedimentos presentes na elaboração de um modelo de ML em etapas coesas e trocá-las caso haja a necessidade de teste com outro algoritmo, atributo protegido ou conjunto de dados. 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 conceitos, foi possível estabelecer um processo automatizado, dependendo apenas dos próprios dados obtidos em execuções anteriores para a automação ser executada.

Ao usar a arquitetura MAPE-K para possibilitar uma escolha autônoma de algoritmos e processamento do conjunto de dados foram notadas algumas vantagens. É um modelo de organização conhecido, o que facilita a manutenção do desenvolvedor que já possui conhecimento desta arquitetura. Ela permite separar muito bem as etapas para executar um plano e, com isso, reconfigurar o Gerenciador Autônomo para executar as opções com melhores resultados. Esta separação também auxilia na manutenção, uma vez que o ciclo de monitoria, análise, planejamento, execução e obtenção de conhecimento é um formato muito bem definido para análise de dados e execução de ações, facilitando o entendimento de seu funcionamento e, conseqüentemente, de seu código.

Ao usar uma interface humano-computador para intermediar as interações entre o Gerenciador Autônomo e as escolhas de um usuário, foi possível obter uma melhor compreensão do funcionamento do sistema. A interface permitiu explicar melhor como funcionam as etapas, como funciona o cálculo para análise e como as configurações presentes para a etapa de planejamento afetam o resultado final. Tal compreensão do funcionamento do sistema é importante para a sustentabilidade do mesmo, sendo possível treinar novos usuários com mais eficiência e contribuir para a expansão de seu uso. O uso da interface também levou a uma conclusão inesperada: Com alguns ajustes, é possível adaptar o sistema para processos de MLOps, uma vez que a base de conhecimento gerada pode ajudar

na decisão de retornar modelos mais antigos, porém com menos ruídos em seus dados e, conseqüentemente, melhores métricas no geral.

8.1 Limitações

Pode-se notar alguns pontos na implementação que não irão ser resolvidos apenas pela escolha da arquitetura. Embora o *Pipes-and-Filters* seja um padrão que facilite o encapsulamento e a modularização, durante o processo de desenvolvimento da solução quem desenvolve terá de equilibrar memória, armazenamento e performance. Os conjuntos de dados avaliados nos Estudos de Caso puderam ser armazenados em memória e garantir performance máxima, mas em um contexto onde *gigabytes/terabytes* de dados são consumidos diariamente é necessário sacrificar performance e realizar as operações em dispositivos de armazenamento para manter a aplicação estável e com baixo custo, evitando gastos com infra-estrutura.

Ainda não é recomendável uma autonomia completa devido aos problemas ainda enfrentados pelo tema *Fairness*. A literatura revisada sobre *Fairness* verifica apenas problemas de classificação binária, e para evoluções e novos métodos é provável que ocorram refatorações no arcabouço. Também há de se considerar que, mesmo que o arcabouço 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 Gerenciador Autônomo 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 e justo. A avaliação por seres humanos ainda garantiria a segurança e privacidade necessárias para entrar em conformidade com a lei, uma vez que há o envolvimento de dados considerados sensíveis.

A qualidade dos dados coletados pelo Gerenciador Autônomo também é um ponto que deverá ser observado conforme o uso do sistema 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 após evoluções no processo de análise. É preciso realizar processos de limpeza dos dados conforme novas execuções no Módulo de ML e atualizações no Gerenciador Autônomo forem realizadas.

Apesar da autonomia tornar o processo de decisão por uma gama de algoritmos mais simples e rápido para o Cientista de Dados, o número de configurações necessárias para dar autonomia ao arcabouço pode ser um entrave para que este opte por este tipo de abordagem e fique com a abordagem mais tradicional. Este fato, além de mostrar a importância de uma boa documentação para facilitar o entendimento da pessoa que irá usar a solução, mostra como a interface simplifica o entendimento do processo utilizado para a obtenção da configuração arquitetural e pode ser um caminho mais interessante para uma maior adoção do que a implementação de configurações no próprio projeto, que são mais simples de serem implementadas mas podem ser consideradas frustrantes para uma pessoa fora deste processo de desenvolvimento.

8.2 Trabalhos Futuros

Como trabalhos futuros, é possível determinar algumas possibilidades para os módulos mais importantes e para o arcabouço em si:

- No Gerenciador Autônomo, o Analisador pode realizar uma análise mais profunda aumentando o número de indicadores, considerando grupos além de métricas de *Fairness* e métricas de Avaliação e adaptações para abordagens baseadas em *Logic Scoring of Preference* (LSP) e métodos de *Multi-Criteria Decision Making* (MCDM).
- No Módulo de ML, é possível introduzir técnicas para melhora dos resultados como *Data Augmentation* e *K-Fold Cross-Validation* e introduzir soluções para *AI Explainability* como forma de garantir mais transparência.
- Também é possível mudar o foco para solucionar problemas de MLOps, utilizando o Gerenciador Autônomo para determinar uma melhor implantação em caso de piora nas métricas de um modelo já utilizado por clientes, além de funcionalidades como notificação para tais casos de diminuição de métricas, mecanismos para implantação dos modelos obtidos, opções de visualização dos dados, um sistema para versionamento dos conjuntos de dados para armazenamento e economia de espaço.

Referências Bibliográficas

- [1] Ai fairness 360. <https://aif360.mybluemix.net>.
- [2] Amazon sagemaker. <https://aws.amazon.com/pt/sagemaker/>.
- [3] Aprendizado de máquina. https://pt.wikipedia.org/wiki/Aprendizado_de_máquina.
- [4] Flask. <https://flask.palletsprojects.com/>.
- [5] IBM - Analytics and AI architecture. <https://www.ibm.com/cloud/architecture/architectures/aiAnalyticsArchitecture/reference-architecture/>.
- [6] Lendingclub dataset. <https://www.kaggle.com/datasets/wordsforthewise/lending-club>.
- [7] Machine learning: o que é e qual sua importância? https://www.sas.com/pt_br/insights/analytics/machine-learning.html.
- [8] Material ui. <https://mui.com/>.
- [9] Pandas. <https://pandas.pydata.org/>.
- [10] React. <https://react.dev/>.
- [11] Redux. <https://redux.js.org/>.
- [12] scikit-learn. <https://scikit-learn.org>.
- [13] Statlog (german credit data) data set. [https://archive.ics.uci.edu/ml/datasets/statlog+\(german+credit+data\)](https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data)).
- [14] What's the difference between artificial intelligence, machine learning and deep learning? <https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/>.
- [15] An architectural blueprint for autonomic computing. Technical report, IBM, 2005. <https://www-03.ibm.com/autonomic/pdfs/AC%20Blueprint%20White%20Paper%20V7.pdf>.
- [16] *The elements of statistical learning: data mining, inference and prediction*. Springer, 2009. <https://hastie.su.domains/Papers/ESLII.pdf>.

- [17] Data engineering - introduction and epochs. Technical report, panoply.io, 2017. <https://www-03.ibm.com/autonomic/pdfs/AC%20Blueprint%20White%20Paper%20V7.pdf>.
- [18] Machine learning: things are getting intense, 2018. <https://www2.deloitte.com/content/dam/Deloitte/global/Images/infographics/technologymediatelecommunications/gx-deloitte-tmt-2018-intense-machine-learning-report.pdf>.
- [19] Brasil se destaca com 42% das iniciativas de IA na América Latina, em 2020, 2021. <https://cio.com.br/tendencias/brasil-se-destaca-com-42-das-iniciativas-de-ia-na-america-latina-em-2020/>.
- [20] Proteção de Dados - LGPD, 2021. <https://www.gov.br/defesa/pt-br/aceso-a-informacao/lei-geral-de-protecao-de-dados-pessoais-lgpd>.
- [21] Nadeem Abbas, Jesper Andersson, and Welf Löwe. Autonomic software product lines. pages 324–331, 2010.
- [22] 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.
- [23] 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.
- [24] Tom Begley, Tobias Schwedes, Christopher Frye, and Ilya Feige. Explainability for fair machine learning, 2021.
- [25] Francine Berman, Rob Rutenbar, Brent Hailpern, Henrik Christensen, Susan Davidson, Deborah Estrin, Michael Franklin, Margaret Martonosi, Padma Raghavan, Victoria Stodden, and Alexander S. Szalay. Realizing the potential of data science. *Communications of the ACM*, 61:67–72, 2018.
- [26] Sumon Biswas and Hriday Rajan. Do the machine learning models on a crowd sourced platform exhibit bias? an empirical study on model fairness. page 642–653, 2020.
- [27] Jan Bosch. Software architecture: The next step. pages 194–199, 2004.
- [28] L Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.
- [29] 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.
- [30] Toon Calders, Faisal Kamiran, and Mykola Pechenizkiy. Building classifiers with independency constraints. In *2009 IEEE International Conference on Data Mining Workshops*, pages 13–18, 2009.

- [31] 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. <https://proceedings.neurips.cc/paper/2017/file/9a49a25d845a483fae4be7e341368e36-Paper.pdf>.
- [32] 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.
- [33] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *22nd ACM SIGKDD International Conference*, pages 785–794, 2016.
- [34] Brian d’Alessandro, Cathy O’Neil, and Tom LaGatta. Conscientious classification: A data scientist’s guide to discrimination-aware classification. *Big Data*, pages 120–134, 2017.
- [35] Timothy Dozat. Incorporating nesterov momentum into adam. 2016.
- [36] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011.
- [37] Jozo J. Dujmović and Hajime Nagashima. Lsp method and its use for evaluation of java ides. *International Journal of Approximate Reasoning*, 41(1):3–22, 2006.
- [38] Jeffrey Dunn. Introducing FBLeaRner Flow: Facebook’s AI backbone. <https://engineering.fb.com/2016/05/09/core-data/introducing-fblearner-flow-facebook-s-ai-backbone/>.
- [39] R. J. Erb. Introduction to backpropagation neural network computation. *Pharmaceutical Research*, 10:165–170, 1993.
- [40] 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. <https://doi.org/10.1145/2783258.2783311>.
- [41] Keith D. Foote. A brief history of data science, 2021.
- [42] Jerome Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29, 2000.
- [43] David Garlan and Mary Shaw. An introduction to software architecture. In *Advances in Software Engineering and Knowledge Engineering*, 1993.
- [44] Sahin Cem Geyik, Stuart Ambler, and Krishnaram Kenthapadi. Fairness-Aware Ranking in Search and Recommendation Systems with Application to LinkedIn Talent Search. In *Proceedings of the 25th International Conference on Knowledge Discovery and Data Mining*, pages 2221–2231, 2019.

- [45] Alex Graves. Generating sequences with recurrent neural networks. 2013.
- [46] Moritz Hardt, Eric Price, Eric Price, and Nati Srebro. Equality of opportunity in supervised learning. In *Advances in Neural Information Processing Systems*, 2016. <https://proceedings.neurips.cc/paper/2016/file/9d2682367c3935defcb1f9e247a97c0d-Paper.pdf>.
- [47] Faisal Kamiran and Toon Calders. Data pre-processing techniques for classification without discrimination. *Knowledge and Information Systems*, 2011. https://www.researchgate.net/publication/228975972_Data_Pre-Processing_Techniques_for_Classification_without_Discrimination.
- [48] 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.
- [49] 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.
- [50] Krishnaram Kenthapadi, Stuart Ambler, Ahsan Chudhary, Mark Dietz, Sahin Cem Geyik, Ian Koeppel, Varun Mithal, Guillaume Saint-Jacques, Amir Sepehri, Thanh Tran, and Sriram Vasudevan. Fairness, privacy, and transparency by design in ai/ml systems, 2019. <https://engineering.linkedin.com/blog/2019/fairness-privacy-transparency-by-design>.
- [51] Jeffrey Kephart and D.M. Chess. The vision of autonomic computing. pages 41 – 50, 2003.
- [52] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2014.
- [53] Nikolaos Konstantinou, Martin Koehler, Edward Abel, Cristina Civili, Bernd Neumayr, Emanuel Sallinger, Alvaro A.A. Fernandes, Georg Gottlob, John A. Keane, Leonid Libkin, and Norman W. Paton. The vada architecture for cost-effective data wrangling. In *Proceedings of the 2017 ACM International Conference on Management of Data*, page 1599–1602, 2017. <https://doi.org/10.1145/3035918.3058730>.
- [54] Warren Mcculloch and Walter Pitts. A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:127–147, 1943.
- [55] 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.
- [56] Carlos Mougán, José Manuel Álvarez, Gourab K. Patro, Salvatore Ruggieri, and Steffen Staab. Fairness implications of encoding protected categorical attributes. 2022.

- [57] Geoff Pleiss, Manish Raghavan, Felix Wu, Jon Kleinberg, and Kilian Q Weinberger. On fairness and calibration. In *Advances in Neural Information Processing Systems*, 2017. <https://proceedings.neurips.cc/paper/2017/file/b8b9c74ac526ffffbe b2d39ab038d1cd7-Paper.pdf>.
- [58] Boris Polyak. Some methods of speeding up the convergence of iteration methods. *Ussr Computational Mathematics and Mathematical Physics*, 4:1–17, 1964.
- [59] Daphne Raban and Avishag Gordon. The evolution of data science and big data research: A bibliometric analysis. *Scientometrics*, 122:1563–1581, 2020.
- [60] C.v Ramamoorthy and Benjamin Wah. Knowledge and data engineering. *IEEE Transactions on Knowledge and Data Engineering*, 1:9 – 16, 1989.
- [61] Ryan Rifkin and Aldebaro Klautau. In defense of one-vs-all classification. *Journal of Machine Learning Research*, 5:101–141, 12 2004.
- [62] Lior Rokach and Oded Maimon. Top-down induction of decision trees classifiers—a survey. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 35:476 – 487, 2005.
- [63] Brian C. Ross. Mutual information between discrete and continuous data sets. *PLoS ONE*, 2014. <http://dx.plos.org/10.1371/journal.pone.0087357>.
- [64] Thomas L. Saaty. *The analytic hierarchy process : planning, priority setting, resource allocation*. 1980.
- [65] Mark Schmidt, Nicolas Roux, and Francis Bach. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 162, 2013.
- [66] D. Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-François Crespo, and Dan Dennison. Hidden technical debt in machine learning systems. In *Advances in Neural Information Processing Systems*, 2015. <https://proceedings.neurips.cc/paper/2015/file/86df7dcfd896fcdf2674f757a2463eba-Paper.pdf>.
- [67] Colin Shearer. The crisp-dm model: The new blueprint for data mining. *Journal of Data Warehousing*, 2000.
- [68] Yashpal Singh and Alok Singh Chauhan. Neural networks in data mining. *Journal of Theoretical & Applied Information Technology*, 5, 2009.
- [69] 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. <http://dx.doi.org/10.1145/3219819.3220046>.
- [70] Ingo Steinwart and Andreas Christmann. *Support Vector Machines*. Springer Science & Business Media, 2008.

- [71] I. Sutskever, J. Martens, G. Dahl, and G. Hinton. On the importance of initialization and momentum in deep learning. *30th International Conference on Machine Learning, ICML 2013*, pages 1139–1147, 01 2013.
- [72] Matteo Testi, Matteo Ballabio, Emanuele Frontoni, Giulio Iannello, Sara Moccia, Paolo Soda, and Gennaro Vessio. Mlops: A taxonomy and a methodology. *IEEE Access*, pages 63606–63618, 2022.
- [73] Ashutosh Tripathi. Mlops: A complete guide to machine learning operations | mlops vs devops. <https://ashutoshtripathi.com/2021/08/18/mlops-a-complete-guide-to-machine-learning-operations-mlops-vs-devops/>.
- [74] Ian H. Witten, Eibe Frank, and Mark A. Hall. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016. <https://www.wi.hs-wismar.de/~cleve/vorl/projects/dm/ss13/HierarClustern/Literatur/WittenFrank-DM-3rd.pdf>.
- [75] 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. <https://proceedings.mlr.press/v28/zemel13.html>.
- [76] Brian Zhang, Blake Lemoine, and Margaret Mitchell. Mitigating unwanted biases with adversarial learning. pages 335–340, 2018. https://www.researchgate.net/publication/330299272_Mitigating_Unwanted_Biases_with_Adversarial_Learning.
- [77] Harry Zhang. The optimality of naive bayes. 2004.

Apêndice A

Documentação de Instalação

A.1 Introdução

Esta documentação foi criada com o objetivo de guiar o desenvolvedor de Software a entender, configurar e manter este sistema, que é dividido em 5 módulos 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, desde uma categorização dos dados mais específica do que na etapa anterior, passando pelo algoritmo utilizado e finalizando obtendo métricas para determinar qualidade do resultado final.
- **Gerenciador Autônomo:** Etapa que executa um componente para automatizar todas as etapas do Workflow, 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. 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 Gerenciador Autônomo.

A.2 Programas necessários para instalação

- **Python:** É a linguagem de programação utilizada para montar e executar todas as etapas com a exceção do Frontend da interface. É disponível no site <https://www.python.org/> e é necessária a versão **3.8**, **3.9** ou **3.10**. A versão **3.11** apresentou problemas de compatibilidade devido a mudanças em seu funcionamento.

- **Node.js:** É o programa necessário para montar o Frontend da interface. É disponível no site <https://nodejs.org/> e foi testado na versão **16.14.2**, embora outras versões podem ser executadas sem problemas de compatibilidade.
- **Git:** É o programa necessário para realizar o download do código-fonte e realizar atualizações no mesmo. É disponível no site <https://git-scm.com/> e foi testado na versão **2.35.1**, embora outras versões podem ser executadas sem problemas de compatibilidade.
- **CUDA Toolkit:** É a biblioteca necessária para rodar alguns dos algoritmos presentes no Workflow. É disponível no site <https://developer.nvidia.com/cuda-toolkit-archives> e foi testado na versão **11**, mas não houve testes se versões posteriores são compatíveis. É compatível apenas para GPUs Nvidia, caso não tiver há uma versão apenas para CPUs disponível.

A.3 Instalação do sistema

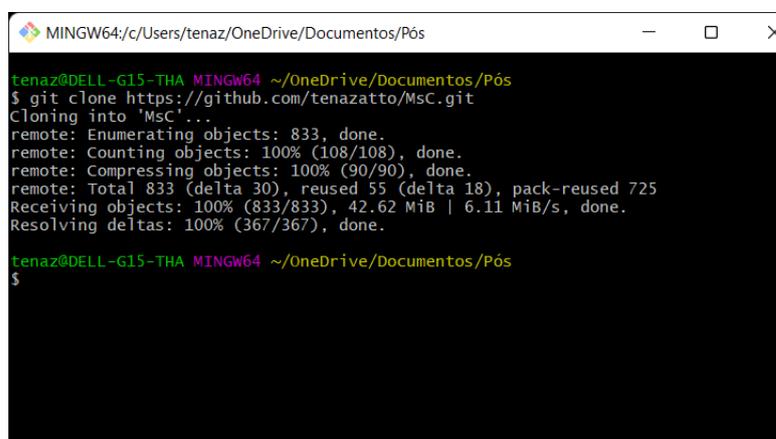
A partir desta parte, os exemplos serão realizados utilizando o Git Bash no Sistema Operacional Windows. Entretanto, no Linux e no Mac os passos são semelhantes por ambos também utilizarem esta linha de comando.

A.3.1 Obtenção do código-fonte

O sistema se encontra no repositório <https://github.com/tenazatto/MsC>. Para obter seu código-fonte, basta digitar o seguinte comando:

```
git clone https://github.com/tenazatto/MsC.git
```

O Git baixará todos os arquivos e após o download é possível ver a pasta e seus arquivos na pasta **MsC**



```
MINGW64;c:/Users/tenaz/OneDrive/Documentos/Pós
tenaz@DELL-G15-THA MINGW64 ~/OneDrive/Documentos/Pós
$ git clone https://github.com/tenazatto/MsC.git
Cloning into 'MsC'...
remote: Enumerating objects: 833, done.
remote: Counting objects: 100% (108/108), done.
remote: Compressing objects: 100% (90/90), done.
remote: Total 833 (delta 30), reused 55 (delta 18), pack-reused 725
Receiving objects: 100% (833/833), 42.62 MiB | 6.11 MiB/s, done.
Resolving deltas: 100% (367/367), done.
tenaz@DELL-G15-THA MINGW64 ~/OneDrive/Documentos/Pós
$
```

A.3.2 Montagem de ambiente

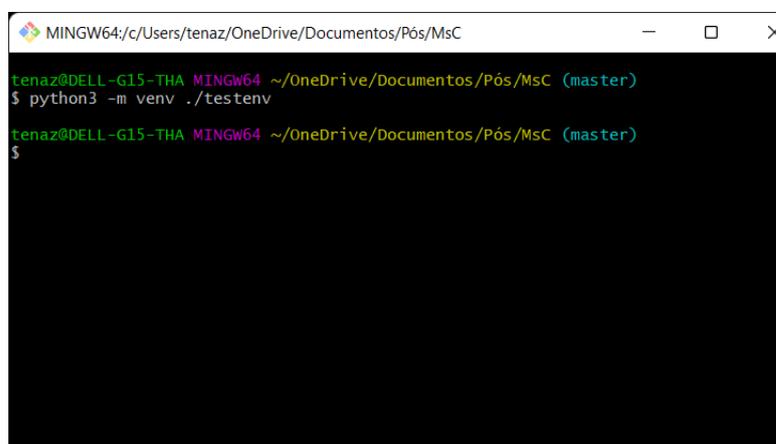
Para evitar problemas de versão com bibliotecas de outros projetos instalados, é possível criar um ambiente virtual para realizar a instalação das bibliotecas separadamente. Para criar, é necessário o **virtualenv** instalado no Python. Caso ele não esteja instalado, ele é obtido através do comando:

```
pip install virtualenv
```

Para criar um novo ambiente virtual, é preciso digitar o comando

```
python3 -m venv ./(nome do ambiente)
```

Como exemplo, nesta documentação foi criado o documento **testenv**

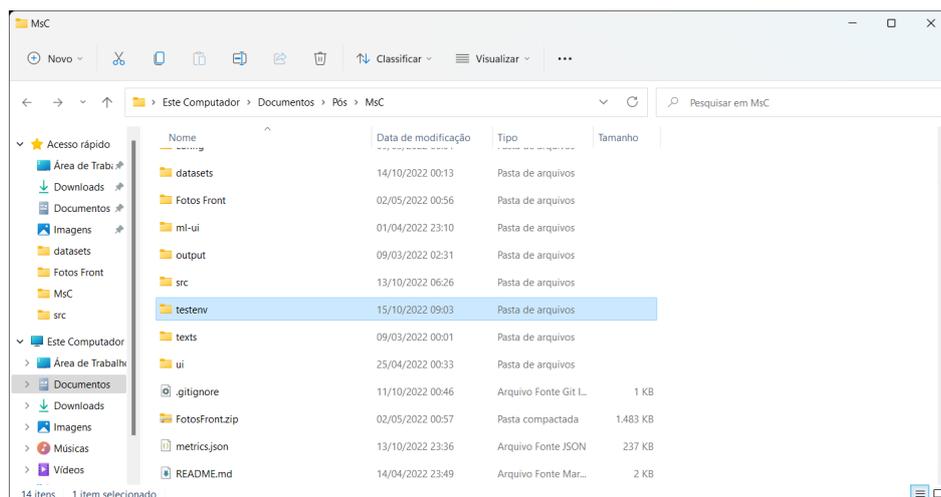


```

MINGW64/c/Users/tenaz/OneDrive/Documentos/Pós/MsC
tenaz@DELL-G15-THA MINGW64 ~/OneDrive/Documentos/Pós/MsC (master)
$ python3 -m venv ./testenv
tenaz@DELL-G15-THA MINGW64 ~/OneDrive/Documentos/Pós/MsC (master)
$

```

Após o término, aparecerá uma nova pasta de mesmo nome



Após criar o ambiente virtual, é preciso ativá-lo para utilizar

```
.\(nome do ambiente)\Scripts\activate.bat (Windows - Prompt de Comando)
```

```
source ./(nome do ambiente)/Scripts/activate (Windows - Bash)
```

source ./(nome do ambiente)/bin/activate (Linux)

Para verificar se o ambiente foi ativado, é possível verificar, ao digitar qualquer comando no bash, que o nome do ambiente virtual aparece logo abaixo.

```

MINGW64~/c/Users/tenaz/OneDrive/Documentos/Pós/MsC
tenaz@DELL-G15-THA MINGW64 ~/OneDrive/Documentos/Pós/MsC (master)
$ python3 -m venv ./testenv
tenaz@DELL-G15-THA MINGW64 ~/OneDrive/Documentos/Pós/MsC (master)
$ source ./testenv/Scripts/activate
(testenv)
tenaz@DELL-G15-THA MINGW64 ~/OneDrive/Documentos/Pós/MsC (master)
$ python --version
Python 3.8.10
(testenv)
tenaz@DELL-G15-THA MINGW64 ~/OneDrive/Documentos/Pós/MsC (master)
$ echo $VIRTUAL_ENV
C:\Users\tenaz\OneDrive\Documentos\Pós\MsC\testenv
(testenv)
tenaz@DELL-G15-THA MINGW64 ~/OneDrive/Documentos/Pós/MsC (master)
$ ls
'Fotos Front'/  README.md  datasets/  ml-ui/  src/  texts/
FotosFront.zip  config/    metrics.json  output/  testenv/  ui/
(testenv)
tenaz@DELL-G15-THA MINGW64 ~/OneDrive/Documentos/Pós/MsC (master)
$

```

Para desativar o ambiente virtual, é preciso digitar o comando

deactivate

Para verificar se o ambiente foi desativado, é possível verificar, ao digitar qualquer comando no bash, que o nome do ambiente virtual não irá mais aparecer até ser ativado novamente.

No caso do Node.js não é necessário realizar tais etapas, pois a instalação das bibliotecas nesta documentação é realizada de maneira local

A.3.3 Instalação das bibliotecas

Python

Com o ambiente virtual criado e ativado, é possível utilizar os arquivos **src/requirements.txt**, dependendo das configurações da sua máquina, para instalar todas as bibliotecas necessárias através do comando

pip install -r ./src/requirements-nvidia.txt (GPU Nvidia)

pip install -r ./src/requirements-cpu.txt (CPU)

Estes arquivos foram preparados apenas para rodar de acordo com o hardware apresentado. A execução de ambos os comandos não é necessária e nem recomendável.

Node.js

Para o Node.js, como o arquivo **package.json** está dentro da pasta **ml-ui**, é possível acessar essa pasta e digitar o comando

npm install

A.4 Execução do sistema

A.4.1 Engenharia de dados

Dentro da pasta **MsC** e com o ambiente virtual criado e ativado, para rodar a etapa de Engenharia de dados basta digitar o seguinte comando

```
python -m src.data_engineering.data_engineering_start --data (Opção)
```

No momento, há 3 opções disponíveis:

- **GERMAN_CREDIT:** Manipula o German Credit Dataset, cujo arquivo está na localização **datasets/german.data**, para utilização no Workflow.
- **LENDINGCLUB:** Baixa e manipula o Lendingclub Dataset para utilização no Workflow.
- **METRICS:** Obtém o maior valor, menor valor e a média de cada métrica para cada Workflow já executado.

A.4.2 Workflow de IA

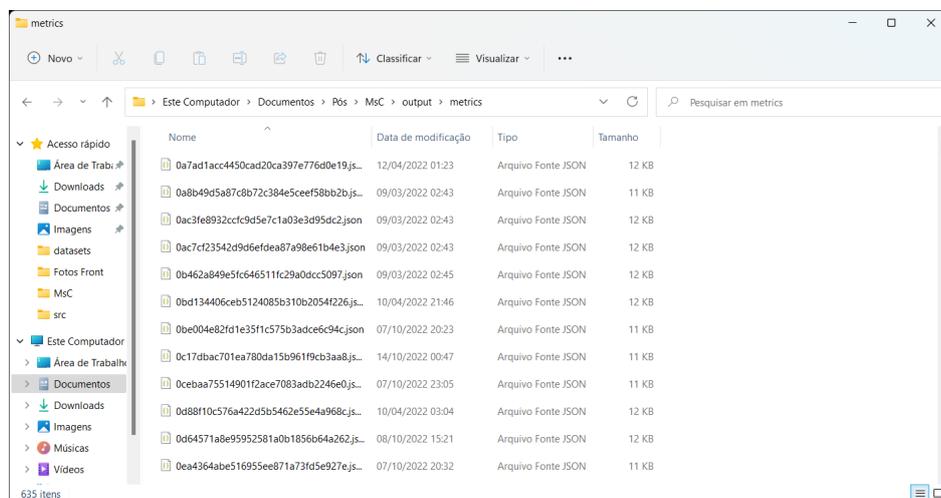
Dentro da pasta **MsC** e com o ambiente virtual criado e ativado, para rodar todos os Workflows possíveis basta digitar o seguinte comando

```
python -m src.pipeline.pipeline_start --dataset (Opção)
```

No momento, há 4 opções disponíveis:

- **ADULT_INCOME_SEX:** Executa os Workflows para o Adult Income Dataset, cujo arquivo está na localização **datasets/adult.csv**, utilizando Sexo (Masculino/Feminino) como atributo protegido.
- **GERMAN_CREDIT_FOREIGN:** Executa os Workflows para o German Credit Dataset, cujo arquivo é manipulado na etapa anterior, utilizando Nacionalidade (Alemão/Estrangeiro) como atributo protegido.
- **GERMAN_CREDIT_AGE:** Executa os Workflows para o German Credit Dataset, cujo arquivo é manipulado na etapa anterior, utilizando Idade (-25 anos/25 ou + anos) como atributo protegido.
- **LENDINGCLUB_INCOME:** Executa os Workflows para o Lendingclub Dataset, cujo arquivo é manipulado na etapa anterior, utilizando Renda (-1 salário mínimo/1 ou + salários mínimos) como atributo protegido.

Após a execução, é possível ver a geração das métricas dentro da pasta **output/metrics**, necessárias para a execução da próxima etapa.



A.4.3 Autonomia do Workflow

Dentro da pasta **MsC**, com o ambiente virtual criado e ativado e com pelo menos um Workflow executado, é possível verificar como a etapa de autonomia funciona com o seguinte comando

```
python -m src.mapek.mapek_start
```

Nesta etapa, ele escolhe o Workflow que apresentou as melhores métricas, porém a filtragem por conjunto de dados está desenvolvida até o momento apenas na próxima etapa. Como ele roda ininterruptamente, é preciso interromper sua execução.

A.4.4 Interface

Backend

Dentro da pasta **MsC**, com o ambiente virtual criado e ativado e com pelo menos um Workflow executado, é possível rodar o Backend da interface com o seguinte comando

```
python -m src.api.flask_start
```

Ele vai iniciar um servidor na porta 8080, necessário para rodar as requisições que o Frontend vai solicitar

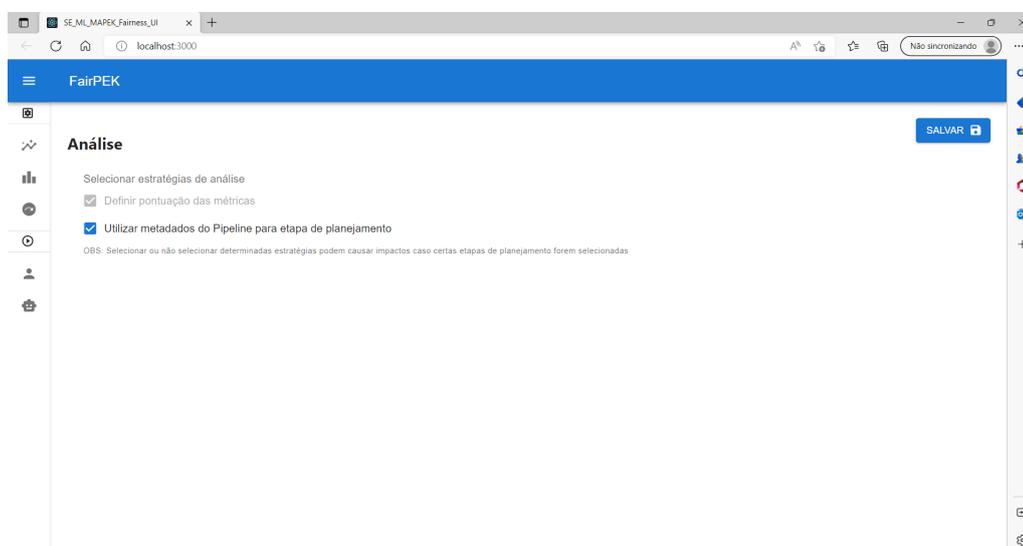
```
MINGW64:/c:/Users/tenaz/OneDrive/Documentos/Pós/MsC
(testenv)
tenaz@DELL-G15-THA MINGW64 ~/OneDrive/Documentos/Pós/MsC (master)
$ python -m src.api.flask_start
2022-10-15 10:27:17.437341: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cudart64_110.dll'; dlderror: cudart64_110.dll not found
2022-10-15 10:27:17.437371: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your machine.
* Serving Flask app 'flask_start' (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Running on http://127.0.0.1:8080
Press CTRL+C to quit
```

Frontend

Dentro da pasta **MsC/ml-ui**, é possível rodar o Frontend da interface com o seguinte comando

```
npm start
```

Ele vai iniciar o navegador acessando um servidor na porta 3000, e deverá iniciar a tela no menu de Análise



Apêndice B

Documentação de Manutenção

B.1 Introdução

Esta documentação foi criada com o objetivo de guiar o desenvolvedor de Software a entender, configurar e manter este sistema, que é dividido em 4 módulos 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, desde uma categorização dos dados mais específica do que na etapa anterior, passando pelo algoritmo utilizado e finalizando obtendo métricas para determinar qualidade do resultado final.
- **Autonomia do Workflow (Gerenciador Autônomo):** Etapa que executa um componente para automatizar todas as etapas do Workflow, 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. 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 Gerenciador Autônomo.

B.2 Arquitetura do Workflow e Framework

O Workflow usa uma arquitetura chamada de Pipes-and-Filters, onde Pipes, que transportam os dados, são ligados por Filters, que realizam as manipulações desses dados. Pipes e Filters se encadeiam para formar uma sequência de operações, caracterizando todo o Pipeline. Para auxiliar no desenvolvimento, um pequeno framework foi criado, para facilitar as operações necessárias entre Pipes e Filters.

B.2.1 Estrutura

Pipes herdam a classe **BasePipe**. Essa classe possui o atributo **value**, que caracteriza os dados transformados, em formato de um dicionário Python

```

1 class FairnessPipe(BasePipe):
2     privileged_group = []
3     unprivileged_group = []
4
5     label_names = []
6     protected_attribute_names = []
7
8     optim_options = {}
9
10    def __init__(self):
11        self.value = {
12            'privileged_group': self.privileged_group,
13            'unprivileged_group': self.unprivileged_group,
14            'label_names': self.label_names,
15            'protected_attribute_names': self.protected_attribute_names,
16            'optim_options': self.optim_options
17        }

```

Filters herdam a classe **BaseFilter**. Essa classe possui dois Pipes (input e output) e um método chamado **execute**, que é onde as operações de transformação do Pipe de input são executadas para serem colocadas no Pipe de output

```

1 from sklearn.model_selection import train_test_split
2
3 from src.pipeline.pipe_filter.pipe import BaseFilter
4
5
6 class TrainTestSplit(BaseFilter):
7     test_size = 0.2
8
9     def __init__(self, test_size=0.2):
10        self.test_size = test_size
11
12    def execute(self):
13        df_x = self.input['df_x']
14        df_y = self.input['df_y']
15
16        x_train, x_test, y_train, y_test = train_test_split(df_x, df_y,
17            test_size=self.test_size, random_state=42)
18
19        self.output = {
20            'x_train': x_train,
21            'x_test': x_test,
22            'y_train': y_train,
23            'y_test': y_test,
24            'checksum': self.input['checksum']
25        }

```

B.2.2 Operações

No Framework, estão presentes as seguintes operações:

B.2.3 Ligação de Pipe com Filter

Para juntar um Pipe com um Filter, basta realizar o comando `>=`, caracterizando o transporte a um Filter.

```
1 Pipe1() >= Filter1()
```

Se o Pipe estiver carregando os dados corretos, o Filter será automaticamente executado.

B.2.4 Ligação de Filter com Pipe

Para juntar um Filter com um Pipe, basta realizar o comando `==`, caracterizando o transporte a um Pipe.

```
1 Filter1() == Pipe1()
```

Se o Filter estiver corretamente implementado, o Pipe será caracterizado como a saída desse mesmo Filter.

B.2.5 Seleção parcial de dados presentes no Pipe

Para selecionar apenas alguns atributos presentes no Pipe, basta adicionar colchetes e colocar os campos desejados.

```
1 pipe1['campo1', 'campo2', 'campo3']
```

B.2.6 Junção de Pipes

Para juntar os dados de dois pipes em um só, basta realizar o comando `+`, caracterizando uma junção de Pipes.

```
1 pipe1 + pipe2
```

B.3 Incrementos no Workflow

Aqui estão dois exemplos de como é possível realizar a manutenção do Workflow

B.3.1 Adicionando um novo Conjunto de Dados

Engenharia de dados

1. Geralmente conjuntos de dados não vem filtrados, e é preciso um trabalho de Engenharia de dados para realizar experimentos com melhores resultados. Neste sistema, a parte de Engenharia de dados se encontra na pasta `src/data_engineering`.

- Os métodos onde os processamentos são realizados ficam no arquivo `data_engineering.py`. Para adicionar um novo processamento, basta adicionar um novo método neste arquivo.
- Importar o novo método no arquivo `data_engineering_start.py`.
- Adicionar uma nova opção no parâmetro `choices` presente no método `parser.add_argument`

```

1 parser.add_argument("--data", help="Selecao do gerador do
2     conjunto de dados tratado",
3     choices=['GERMAN_CREDIT', 'LENDINGCLUB', '
4     METRICS'])

```

- Adicionar uma nova condição com a opção adicionada

Workflow

- Neste sistema, a parte do Workflow se encontra na pasta `src/pipeline`. Dentro dela, os Pipes que armazenam os conjuntos de dados se encontram na pasta `processors/preprocessors/data`. Dentro dela, no arquivo `dataset.py`, criar uma classe que herda a classe `Dataset`, preenchendo os atributos `dataset_path` com o caminho do arquivo.

```

1 class LendingclubDataset(Dataset):
2     dataset_path = 'datasets/lendingclub_dataset.csv'
3
4     def __init__(self):
5         super().__init__()

```

- Criar um novo arquivo.
- Criar uma classe que herda a classe `FairnessPipe`, preenchendo os atributos `privileged_group`, `unprivileged_group`, `label_names`, `protected_attribute_names` e `optim_options`.

```

1 class LendingclubIncomeFairnessPipe(FairnessPipe):
2     privileged_group = [{'annual_inc': 1}]
3     unprivileged_group = [{'annual_inc': 0}]
4
5     label_names = ['loan_status']
6     protected_attribute_names = ['annual_inc']
7
8     optim_options = {
9         "distortion_fun": get_distortion_german,
10        "epsilon": 0.05,
11        "clist": [0.99, 1.99, 2.99],
12        "dlist": [.1, 0.05, 0]
13    }
14
15    def __init__(self):
16        super().__init__()

```

4. Criar uma classe que herda a classe **FairnessPreprocessor**, preenchendo o método **dataset_preprocess** e retornando um DataFrame Pandas de dados de entrada e um DataFrame de dados de saída.

```

1 class LendingclubIncomePreprocessor(FairnessPreprocessor):
2     def dataset_preprocess(self, df):
3         df.info()
4
5         SAMPLE_PERCENTAGE = 100
6         df_sample_nok = df[df['loan_status'] == 'Charged Off'].
sample(frac=SAMPLE_PERCENTAGE/100)
7         df_sample_ok = df[df['loan_status'] == 'Fully Paid'].sample
(frac=SAMPLE_PERCENTAGE / 100)
8         df_sample = pd.concat([df_sample_ok, df_sample_nok])
9
10        df_x = df_sample.drop('loan_status', axis=1)
11        df_y = pd.DataFrame(df_sample.loan_status)
12
13        return df_x, df_y

```

5. Na pasta **src/pipeline/processors** e dentro do arquivo **enums.py**, colocar novas opções nas classes **Datasets** e **Preprocessors**.

```

1 class Datasets(ExtendedEnum):
2     ADULT_INCOME = 1
3     GERMAN_CREDIT = 2
4     LENDINGCLUB = 3
5
6
7 class Preprocessors(ExtendedEnum):
8     SEX = 1
9     AGE = 2
10    FOREIGN = 3
11    INCOME = 4

```

6. Na pasta **src/pipeline** e dentro do arquivo **validation.py**, atualizar a variável **existant_preprocessors** com as novas opções colocadas no item anterior.

```

1     existant_preprocessors = \
2         (dataset == Datasets.ADULT_INCOME and preprocessor ==
Preprocessors.SEX) or \
3         (dataset == Datasets.GERMAN_CREDIT and preprocessor ==
Preprocessors.AGE) or \
4         (dataset == Datasets.GERMAN_CREDIT and preprocessor ==
Preprocessors.FOREIGN) or \
5         (dataset == Datasets.LENDINGCLUB and preprocessor ==
Preprocessors.INCOME)

```

7. Na pasta **src/pipeline** e dentro do arquivo **pipeline.py**, encontrar o método **select_data_preprocessor**, atualizar a variável **options** com as novas opções e classes implementadas nos itens anteriores.

```

1  def select_data_preprocessor(self, dataset, preprocessor):
2      choice = [dataset, preprocessor]
3      options = [
4          ([Datasets.ADULT_INCOME, Preprocessors.SEX], (
5              AdultDataset(), AdultSexPreprocessor(), AdultSexFairnessPipe()))
6          ,
7          ([Datasets.GERMAN_CREDIT, Preprocessors.AGE], (
8              GermanDataset(), GermanAgePreprocessor(), GermanAgeFairnessPipe
9              ())),
10         ([Datasets.GERMAN_CREDIT, Preprocessors.FOREIGN], (
11             GermanDataset(), GermanForeignPreprocessor(),
12             GermanForeignFairnessPipe()))),
13         ([Datasets.LENDINGCLUB, Preprocessors.INCOME], (
14             LendingclubDataset(), LendingclubIncomePreprocessor(),
15             LendingclubIncomeFairnessPipe()))),
16     ]
17
18     for option, pipe_filter in options:
19         if choice == option:
20             dataset_pipe, data_preprocessor_filter,
21             fairness_pipe = pipe_filter
22             preprocessed_data_pipe = dataset_pipe >=
23             data_preprocessor_filter == self.new_pipe()
24             break
25
26     return preprocessed_data_pipe, fairness_pipe

```

8. Na pasta **src/pipeline** e dentro do arquivo **pipeline_start.py**, adicionar uma nova opção no parâmetro **choices** presente no método **parser.add_argument**.

```

1  parser.add_argument("--dataset", help="Conjunto de dados
2      tratado com atributo protegido",
3                      choices=['ADULT_INCOME_SEX',
4                              'GERMAN_CREDIT_FOREIGN', '
5      GERMAN_CREDIT_AGE',
6                              'LENDINGCLUB_INCOME'])

```

9. No mesmo arquivo, adicionar uma nova condição com a opção adicionada.

```

1  if args.dataset == 'ADULT_INCOME_SEX':
2      datasets.append((Datasets.ADULT_INCOME, Preprocessors.SEX))
3  elif args.dataset == 'ADULT_INCOME_FOREIGN':
4      datasets.append((Datasets.GERMAN_CREDIT, Preprocessors.
5      FOREIGN))
6  elif args.dataset == 'GERMAN_CREDIT_AGE':
7      datasets.append((Datasets.GERMAN_CREDIT, Preprocessors.AGE)
8  )
9  elif args.dataset == 'LENDINGCLUB_INCOME':
10     datasets.append((Datasets.LENDINGCLUB, Preprocessors.INCOME
11     ))]

```

Interface (Backend)

1. Neste sistema, a parte do Backend se encontra na pasta **src/api**. Dentro dela, no arquivo **repo/pipeline.py**, adicionar a opção no método **get_dataset**.

```

1     def get_dataset(self, dataset):
2         indexes = {
3             'Datasets.ADULT_INCOME': Datasets.ADULT_INCOME,
4             'Datasets.GERMAN_CREDIT': Datasets.GERMAN_CREDIT,
5             'Datasets.LENDINGCLUB': Datasets.LENDINGCLUB,
6         }
7
8         return next(filter(lambda a: a[0] == dataset, indexes.items
          ())), [1]

```

2. Na pasta **src/api** e dentro do arquivo **repo/pipeline.py**, adicionar a opção no método **get_preprocessor**.

```

1     def get_preprocessor(self, preprocessor):
2         indexes = {
3             'Preprocessors.SEX': Preprocessors.SEX,
4             'Preprocessors.AGE': Preprocessors.AGE,
5             'Preprocessors.FOREIGN': Preprocessors.FOREIGN
6         }
7
8         return next(filter(lambda a: a[0] == preprocessor, indexes.
          items()), [1]

```

Interface (Frontend)

1. Neste sistema, a parte do Frontend se encontra na pasta **ml-ui/src**. Dentro dela, no arquivo **Auto-Pipeline-Menu.js**, adicionar a opção colocada na etapa de Workflow no componente Select onde estão as outras opções de conjunto de dados.

```

1 <Select
2   sx={{fontSize: '14px'}}
3   value={dataset}
4   onChange={handleDatasetChange}
5   displayEmpty
6   inputProps={{ 'aria-label': 'Without label' }}
7 >
8   <MenuItem value={'Datasets.ADULT_INCOME'}>Adult Income Dataset</
9     MenuItem>
10  <MenuItem value={'Datasets.GERMAN_CREDIT'}>German Credit Dataset
11  </MenuItem>
12  <MenuItem value={'Datasets.LENDINGCLUB'}>Lendingclub Dataset</
13    MenuItem>
14 </Select>

```

2. Na pasta **ml-ui/src** e dentro do arquivo **Manual-Pipeline-Menu.js**, adicionar a opção colocada na etapa de Workflow no componente Select onde estão as outras opções de conjunto de dados.

```

1 <Select
2   sx={{fontSize: '14px'}}
3   value={dataset}
4   onChange={handleDatasetChange}
5   displayEmpty
6   inputProps={{ 'aria-label': 'Without label' }}
7 >
8   <MenuItem value={'Datasets.ADULT_INCOME'}>Adult Income Dataset</
  MenuItem>
9   <MenuItem value={'Datasets.GERMAN_CREDIT'}>German Credit Dataset
  </MenuItem>
10  <MenuItem value={'Datasets.LENDINGCLUB'}>Lendingclub Dataset</
  MenuItem>
11 </Select>

```

3. Na pasta **ml-ui/src** e dentro do arquivo **Manual-Pipeline-Menu.js**, adicionar a opção colocada na etapa de Workflow no componente Select onde estão as outras opções de pré-processadores.

```

1 <FormControl sx={{ m: 1, width: 300, marginLeft: '35px' }}>
2   {dataset === 'Datasets.ADULT_INCOME' ?
3     <Select
4       sx={{fontSize: '14px'}}
5       value={protectedAtt}
6       onChange={handleProtectedAttChange}
7       displayEmpty
8       inputProps={{ 'aria-label': 'Without label' }}
9     >
10      <MenuItem value={'Preprocessors.SEX'}>Sexo (Masculino/
  Feminino)</MenuItem>
11    </Select>
12  : dataset === 'Datasets.ADULT_INCOME' ?
13    <Select
14      sx={{fontSize: '14px'}}
15      value={protectedAtt}
16      onChange={handleProtectedAttChange}
17      displayEmpty
18      inputProps={{ 'aria-label': 'Without label' }}
19    >
20      <MenuItem value={'Preprocessors.AGE'}>Idade (-25 anos/+25
  anos)</MenuItem>
21      <MenuItem value={'Preprocessors.FOREIGN'}>Nacionalidade (
  Local/Estrangeiro)</MenuItem>
22    </Select>
23  :
24    <Select
25      sx={{fontSize: '14px'}}
26      value={protectedAtt}
27      onChange={handleProtectedAttChange}
28      displayEmpty
29      inputProps={{ 'aria-label': 'Without label' }}
30    >

```

```

31     <MenuItem value={ 'Preprocessors.INCOME' }>Renda (-1 Salario
      Minimo/1+ Salarios Minimios)</MenuItem>
32   </Select>
33 }
34 <FormHelperText>Atributo protegido para medir justicia</
      FormHelperText>
35 </FormControl>

```

- Na pasta **ml-ui/src** e dentro do arquivo **Manual-Pipeline-Menu.js** adicionar a condição para a opção colocada na etapa de Workflow no método **handleDatasetChange**.

```

1  const handleDatasetChange = (event) => {
2    setDataset(event.target.value);
3
4    if (event.target.value === 'Datasets.ADULT_INCOME') {
5      setProtectedAtt('Preprocessors.SEX');
6    } else if (event.target.value === 'Datasets.GERMAN_CREDIT') {
7      setProtectedAtt('Preprocessors.AGE');
8    } else {
9      setProtectedAtt('Preprocessors.INCOME');
10   }
11 }

```

B.3.2 Adicionando um novo Algoritmo

Workflow

- Neste sistema, a parte do Workflow se encontra na pasta **src/pipeline**. Dentro dela, os Filters que executam os algoritmos ficam dentro da pasta **processors**. Dentro dela, no arquivo **enums.py**, colocar novas opções na classe **Algorithms**.

```

1  class Algorithms:
2    LOGISTIC_REGRESSION = 1
3    RANDOM_FOREST = 2
4    GRADIENT_BOOST = 3
5    SUPPORT_VECTOR_MACHINES = 4
6    LINEAR_REGRESSION = 901
7    DECISION_TREE = 902
8    KERNEL_RIDGE = 903

```

A classe **Algorithms** serve para este exemplo em questão, mas para outros tipos de algoritmos as classes **UnbiasDataAlgorithms**, **UnbiasInProcAlgorithms** ou **UnbiasPostProcAlgorithms** podem ser mais adequadas.

- Na pasta **src/pipeline** e dentro do arquivo **validation.py**, adicionar uma nova condição para a variável **existant_algorithms** no método **validate_params**.

```

1    existant_algorithms = \
2      (algorithm == Algorithms.LOGISTIC_REGRESSION and
      unbias_data_algorithm == UnbiasDataAlgorithms.NOTHING and
      unbias_postproc_algorithm == UnbiasPostProcAlgorithms.NOTHING)
      or \

```

```

3         (...)
4         (algorithm == UnbiasInProcAlgorithms.
GRID_SEARCH_REDUCTION and unbias_data_algorithm ==
UnbiasDataAlgorithms.NOTHING and unbias_postproc_algorithm ==
UnbiasPostProcAlgorithms.NOTHING) or \
5         (algorithm == Algorithms.NOVA_OPCAO and
unbias_data_algorithm == UnbiasDataAlgorithms.NOTHING and
unbias_postproc_algorithm == UnbiasPostProcAlgorithms.NOTHING)
or \
6         (algorithm == Algorithms.NOVA_OPCAO and
unbias_data_algorithm == UnbiasDataAlgorithms.NOTHING and
unbias_postproc_algorithm == UnbiasPostProcAlgorithms.
EQUALIZED_ODDS) or \
7         (algorithm == Algorithms.NOVA_OPCAO and
unbias_data_algorithm == UnbiasDataAlgorithms.NOTHING and
unbias_postproc_algorithm == UnbiasPostProcAlgorithms.
CALIBRATED_EQUALIZED_ODDS) or \
8         (algorithm == Algorithms.NOVA_OPCAO and
unbias_data_algorithm == UnbiasDataAlgorithms.NOTHING and
unbias_postproc_algorithm == UnbiasPostProcAlgorithms.
REJECT_OPTION_CLASSIFICATION) or \
9         (algorithm == Algorithms.NOVA_OPCAO and
unbias_data_algorithm == UnbiasDataAlgorithms.REWEIGHING and
unbias_postproc_algorithm == UnbiasPostProcAlgorithms.NOTHING)
or \
10        (algorithm == Algorithms.NOVA_OPCAO and
unbias_data_algorithm == UnbiasDataAlgorithms.
DISPARATE_IMPACT_REMOVER and unbias_postproc_algorithm ==
UnbiasPostProcAlgorithms.NOTHING) or \
11        (algorithm == Algorithms.NOVA_OPCAO and
unbias_data_algorithm == UnbiasDataAlgorithms.
OPTIMIZED_PREPROCESSING and unbias_postproc_algorithm ==
UnbiasPostProcAlgorithms.NOTHING) or \
12        (algorithm == Algorithms.NOVA_OPCAO and
unbias_data_algorithm == UnbiasDataAlgorithms.
LEARNING_FAIR_REPRESENTATIONS and unbias_postproc_algorithm ==
UnbiasPostProcAlgorithms.NOTHING)

```

3. Na pasta **src/pipeline** e dentro do arquivo **pipeline.py**, colocar as novas opções colocadas no item anterior no método **find_algorithm**.

```

1     def find_algorithm(self, algorithm):
2         indexes = {
3             'Algorithms.LOGISTIC_REGRESSION': 1,
4             'Algorithms.RANDOM_FOREST': 2,
5             'Algorithms.GRAIDENT_BOOST': 3,
6             'Algorithms.SUPPORT_VECTOR_MACHINES': 4,
7             'Algorithms.LINEAR_REGRESSION': 901,
8             'Algorithms.DECISION_TREE': 902,
9             'Algorithms.KERNEL_RIDGE': 903,
10            'UnbiasInProcAlgorithms.PREJUDICE_REMOVER': 101,
11            'UnbiasInProcAlgorithms.ADVERSARIAL_DEBIASING': 102,
12            'UnbiasInProcAlgorithms.

```

```

13     'EXPONENTIATED_GRADIENT_REDUCTION': 103,
14         'UnbiasInProcAlgorithms.RICH_SUBGROUP_FAIRNESS': 104,
15         'UnbiasInProcAlgorithms.GRID_SEARCH_REDUCTION': 105,
16         'UnbiasInProcAlgorithms.META_FAIR_CLASSIFIER': 106,
17         'UnbiasInProcAlgorithms.ART_CLASSIFIER': 107
18     }
19     return next(filter(lambda a: a[1] == algorithm, indexes.
    items()))[0]

```

4. Criar um novo arquivo na pasta categorizada pelo algoritmo a ser implementado. Para o exemplo exemplo ilustrado abaixo, como o Gradient Boosting é um algoritmo de treinamento e não foi projetado para redução de viés, os Filters que executam os algoritmos se encontram na pasta **processors/inprocessors/inproc_algorithms**. Seguem as pastas onde os respectivos Filters se encontram:

- **Algoritmo de treinamento sem redução de viés:** processors/inprocessors/inproc_algorithms
- **Algoritmo com redução de viés no dado (Pré-processamento):** processors/preprocessors/unbias_algorithms
- **Algoritmo com redução de viés no treinamento (Processamento):** processors/inprocessors/unbias_algorithms
- **Algoritmo com redução de viés no resultado (Pós-processamento):** processors/postprocessors

5. No arquivo criado, criar uma classe que herda a classe **BaseFilter**.

```

1 class GradientBoostFilter(BaseFilter):
2     weighed = False
3
4     def __init__(self, weighed=False):
5         self.weighed = weighed

```

6. Implementar nesta classe o método **execute**, atribuindo em **self.output** um dicionário Python com os atributos **y_pred** e **scores**.

```

1     def execute(self):
2         y_pred, scores = self.gradient_boost_weighed() if self.
3         weighed else self.gradient_boost()
4
5         self.output = {
6             'y_pred': y_pred,
7             'scores': scores
8         }

```

7. Na pasta **src/pipeline** e dentro do arquivo **pipeline.py**, encontrar o método **process**, atualizar a variável **process_options** com as novas opções e classes implementadas nos itens anteriores.

```

1     def process(self, process_pipe, algorithm,
2       unbiased_data_algorithm):
3         weighed_algorithm = unbiased_data_algorithm ==
4         UnbiasDataAlgorithms.REWEIGHING or \
5         unbiased_data_algorithm ==
6         UnbiasDataAlgorithms.LEARNING_FAIR_REPRESENTATIONS
7
8         process_options = [
9             (Algorithms.LOGISTIC_REGRESSION,
10            LogisticRegressionFilter(weighed=weighed_algorithm)),
11            (Algorithms.RANDOM_FOREST, RandomForestFilter(weighed=
12            weighed_algorithm)),
13            (Algorithms.GRAIDENT_BOOST, GradientBoostFilter(weighed
14            =weighed_algorithm)),
15            (Algorithms.SUPPORT_VECTOR_MACHINES, SVMFilter(weighed=
16            weighed_algorithm)),
17            (UnbiasInProcAlgorithms.PREJUDICE_REMOVER,
18            PrejudiceRemoverFilter()),
19            (UnbiasInProcAlgorithms.ADVERSARIAL_DEBIASING,
20            AdversarialDebiasingFilter()),
21            (UnbiasInProcAlgorithms.
22            EXPONENTIATED_GRADIENT_REDUCTION,
23            ExponentiatedGradientReductionFilter(algorithm=Algorithms.
24            GRAIDENT_BOOST)),
25            (UnbiasInProcAlgorithms.RICH_SUBGROUP_FAIRNESS,
26            RichSubgroupFairnessFilter(algorithm=Algorithms.DECISION_TREE)),
27            (UnbiasInProcAlgorithms.META_FAIR_CLASSIFIER,
28            MetaFairClassifierFilter()),
29            (UnbiasInProcAlgorithms.GRID_SEARCH_REDUCTION,
30            GridSearchReductionFilter(algorithm=Algorithms.RANDOM_FOREST))
31        ]
32
33        for option, filter in process_options:
34            if algorithm == option:
35                prediction_pipe = process_pipe >= filter == self.
36                new_pipe()
37                break
38
39        return prediction_pipe

```

O método `process` serve para este exemplo em questão, mas para outros tipos de algoritmos os métodos `unbias_data_preprocessor` ou `data_postprocess` podem ser mais adequados.

- Na pasta `src/pipeline` dentro do arquivo `pipeline_start.py`, adicionar as opções (adaptadas a opção corrente) na variável `process_options`.

```

1     process_options = [
2         (Algorithms.NOVA_OP CAO, UnbiasDataAlgorithms.NOTHING,
3         UnbiasPostProcAlgorithms.NOTHING),
4         (Algorithms.NOVA_OP CAO, UnbiasDataAlgorithms.NOTHING,
5         UnbiasPostProcAlgorithms.EQUALIZED_ODDS),
6         (Algorithms.NOVA_OP CAO, UnbiasDataAlgorithms.NOTHING,

```

```

5         UnbiasPostProcAlgorithms.CALIBRATED_EQUALIZED_ODDS),
6         (Algorithms.NOVA_OP CAO, UnbiasDataAlgorithms.NOTHING,
7         UnbiasPostProcAlgorithms.REJECT_OPTION_CLASSIFICATION),
8         (Algorithms.NOVA_OP CAO, UnbiasDataAlgorithms.REWEIGHING,
UnbiasPostProcAlgorithms.NOTHING),
9         (Algorithms.NOVA_OP CAO, UnbiasDataAlgorithms.
DISPARATE_IMPACT_REMOVER,
10        UnbiasPostProcAlgorithms.NOTHING),
11        # (Algorithms.NOVA_OP CAO, UnbiasDataAlgorithms.
OPTIMIZED_PREPROCESSING, UnbiasPostProcAlgorithms.NOTHING),
12        (Algorithms.NOVA_OP CAO, UnbiasDataAlgorithms.
LEARNING_FAIR_REPRESENTATIONS,
13        UnbiasPostProcAlgorithms.NOTHING)
14    ]

```

MAPE-K

1. Neste sistema, a parte do Backend se encontra na pasta `src/mapek`. Dentro dela, no arquivo `ml/planner.py`, colocar as novas opções colocadas no item anterior no método `find_inproc_algorithm`.

```

1     def find_inproc_algorithm(self, algorithm):
2         indexes = {
3             'Algorithms.LOGISTIC_REGRESSION': 1,
4             'Algorithms.RANDOM_FOREST': 2,
5             'Algorithms.GRAIDENT_BOOST': 3,
6             'Algorithms.SUPPORT_VECTOR_MACHINES': 4,
7             'Algorithms.LINEAR_REGRESSION': 901,
8             'Algorithms.DECISION_TREE': 902,
9             'Algorithms.KERNEL_RIDGE': 903,
10            'UnbiasInProcAlgorithms.PREJUDICE_REMOVER': 101,
11            'UnbiasInProcAlgorithms.ADVERSARIAL_DEBIASING': 102,
12            'UnbiasInProcAlgorithms.
EXPONENTIATED_GRADIENT_REDUCTION': 103,
13            'UnbiasInProcAlgorithms.RICH_SUBGROUP_FAIRNESS': 104,
14            'UnbiasInProcAlgorithms.GRID_SEARCH_REDUCTION': 105,
15            'UnbiasInProcAlgorithms.META_FAIR_CLASSIFIER': 106,
16            'UnbiasInProcAlgorithms.ART_CLASSIFIER': 107
17        }
18
19        return next(filter(lambda a: a[0] == algorithm, indexes.
items()))[1]

```

Interface (Backend)

1. Neste sistema, a parte do Backend se encontra na pasta `src/api`. Dentro dela, no arquivo `repo/pipeline.py`, colocar as novas opções colocadas no item anterior no método `get_inproc_algorithm`.

```

1     def get_inproc_algorithm(self, algorithm):
2         indexes = {
3             'Algorithms.LOGISTIC_REGRESSION': 1,

```

```

4         'Algorithms.RANDOM_FOREST': 2,
5         'Algorithms.GRAIDENT_BOOST': 3,
6         'Algorithms.SUPPORT_VECTOR_MACHINES': 4,
7         'Algorithms.LINEAR_REGRESSION': 901,
8         'Algorithms.DECISION_TREE': 902,
9         'Algorithms.KERNEL_RIDGE': 903,
10        'UnbiasInProcAlgorithms.PREJUDICE_REMOVER': 101,
11        'UnbiasInProcAlgorithms.ADVERSARIAL_DEBIASING': 102,
12        'UnbiasInProcAlgorithms.
EXPONENTIATED_GRADIENT_REDUCTION': 103,
13        'UnbiasInProcAlgorithms.RICH_SUBGROUP_FAIRNESS': 104,
14        'UnbiasInProcAlgorithms.GRID_SEARCH_REDUCTION': 105,
15        'UnbiasInProcAlgorithms.META_FAIR_CLASSIFIER': 106,
16        'UnbiasInProcAlgorithms.ART_CLASSIFIER': 107
17    }
18
19    return next(filter(lambda a: a[0] == algorithm, indexes.
items()))[1]

```

Interface (Frontend)

1. Neste sistema, a parte do Frontend se encontra na pasta **ml-ui/src**. Dentro dela, no arquivo **Manual-Pipeline-Menu.js**, adicionar a opção colocada na etapa de Workflow no componente Select onde estão as outras opções de algoritmos.

```

1 <Select
2   sx={{fontSize: '14px'}}
3   value={trainAlgorithm}
4   onChange={handleTrainAlgorithmChange}
5   displayEmpty
6   inputProps={{ 'aria-label': 'Without label' }}
7 >
8   <MenuItem value={'Algorithms.LOGISTIC_REGRESSION'}>Logistic
   Regression</MenuItem>
9   <MenuItem value={'Algorithms.RANDOM_FOREST'}>Random Forest</
   MenuItem>
10  <MenuItem value={'Algorithms.GRAIDENT_BOOST'}>Gradient Boost</
   MenuItem>
11  <MenuItem value={'Algorithms.SUPPORT_VECTOR_MACHINES'}>Support
   Vector Machines</MenuItem>
12 </Select>

```

2. Na pasta **ml-ui/src** e dentro do arquivo **Planning-Menu.js**, adicionar as opções (adaptadas a opção corrente) na variável **validAlgorithms**.

```

1   {
2     options: ["Algorithms.NOVA_OPcao", "UnbiasDataAlgorithms.NOTHING",
   "UnbiasPostProcAlgorithms.NOTHING"],
3     labels: ["Nome da nova opcao", "Sem metodo", "Sem metodo"],
4     selected: true
5   },
6   {

```

```
7     options: ["Algorithms.NOVA_OP CAO", "UnbiasDataAlgorithms.NOTHING",
8     "UnbiasPostProcAlgorithms.EQUALIZED_ODDS"],
9     labels: ["Nome da nova opcao", "Sem metodo", "Equalized Odds"],
10    selected: true
11  },
12  {
13    options: ["Algorithms.NOVA_OP CAO", "UnbiasDataAlgorithms.NOTHING",
14    "UnbiasPostProcAlgorithms.CALIBRATED_EQUALIZED_ODDS"],
15    labels: ["Nome da nova opcao", "Sem metodo", "Calibrated Equalized
16    Odds"],
17    selected: true
18  },
19  {
20    options: ["Algorithms.NOVA_OP CAO", "UnbiasDataAlgorithms.NOTHING",
21    "UnbiasPostProcAlgorithms.REJECT_OPTION_CLASSIFICATION"],
22    labels: ["Nome da nova opcao", "Sem metodo", "Reject Option
23    Classification"],
24    selected: true
25  },
26  {
27    options: ["Algorithms.NOVA_OP CAO", "UnbiasDataAlgorithms.
28    REWEIGHING", "UnbiasPostProcAlgorithms.NOTHING"],
29    labels: ["Nome da nova opcao", "Reweighting", "Sem metodo"],
30    selected: true
31  },
32  {
33    options: ["Algorithms.NOVA_OP CAO", "UnbiasDataAlgorithms.
34    DISPARATE_IMPACT_REMOVER", "UnbiasPostProcAlgorithms.NOTHING"],
35    labels: ["Nome da nova opcao", "Disparate Impact Remover", "Sem
36    metodo"],
37    selected: true
38  },
39  {
40    options: ["Algorithms.NOVA_OP CAO", "UnbiasDataAlgorithms.
    OPTIMIZED_PREPROCESSING", "UnbiasPostProcAlgorithms.NOTHING"],
    labels: ["Nome da nova opcao", "Optimized Preprocessing", "Sem
    metodo"],
    selected: true
  },
  {
    options: ["Algorithms.NOVA_OP CAO", "UnbiasDataAlgorithms.
    LEARNING_FAIR_REPRESENTATIONS", "UnbiasPostProcAlgorithms.NOTHING"],
    labels: ["Nome da nova opcao", "Learning Fair Representations", "
    Sem metodo"],
    selected: true
  }
}
```

Apêndice C

Questionário Aplicado ao Desenvolvedor

Observação: Nome e e-mail não disponibilizados para manter o anonimato.

Nome: *****

E-mail: *****

Anos de experiência em Desenvolvimento de Aplicações:

- Não tive experiência
- Até 2 Anos
- 2 a 5 Anos
- Mais de 5 Anos

Anos de experiência em Engenharia de Dados:

- Não tive experiência
- Até 2 Anos
- 2 a 5 Anos
- Mais de 5 Anos

Anos de experiência em Ciência de Dados:

- Não tive experiência
- Até 2 Anos
- 2 a 5 Anos
- Mais de 5 Anos

Quais os pontos positivos ao realizar o desenvolvimento?

A documentação possui informações detalhadas sobre o que é e como funciona o Framework, explicando as tecnologias usadas, como instalar e usar.

O que poderia melhorar a experiência do desenvolvimento?

Alguns pontos que levantei na documentação sobre a referência dos diretórios.

Comentários/Sugestões adicionais a serem feitas?

Consegui criar uma função nova com o método Naive Bayes com certa facilidade e gostei bastante do trabalho.