



Universidade Estadual de Campinas
Instituto de Computação



Thales Eduardo Nazatto

Workflow autônomo para aplicações de Machine Learning utilizando métricas de Fairness

CAMPINAS
2022

Thales Eduardo Nazatto

***Workflow autônomo para aplicações de Machine Learning
utilizando 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 da Dissertação entregue à banca antes da defesa.

CAMPINAS
2022

Na versão final, esta página será substituída pela ficha catalográfica.

De acordo com o padrão da CCPG: “Quando se tratar de Teses e Dissertações financiadas por agências de fomento, os beneficiados deverão fazer referência ao apoio recebido e inserir esta informação na ficha catalográfica, além do nome da agência, o número do processo pelo qual recebeu o auxílio.”

e

“caso a tese de doutorado seja feita em Cotutela, será necessário informar na ficha catalográfica o fato, a Universidade convenente, o país e o nome do orientador.”

Na versão final, esta página será substituída por outra informando a composição da banca e que a ata de defesa está arquivada pela Unicamp.

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

Palavras-chave — *Workflow*, *Machine Learning*, Inteligência Artificial, Computação Autônoma, Métricas de *Fairness*

Abstract

The use of Artificial Intelligence (AI) involving big data volumes has been growing as our society migrates manual work processes to digital solutions and requires faster and more assertive decision-making, but due to ethical and legal barriers, metrics initially used to defining the effectiveness of an algorithm proved to be limited in measuring biases that reflect society in scenarios that were not expected by the developers of the solution. To solve this problem, new algorithms were developed and a new set of metrics, called Fairness metrics, is used to determine a balance between groups that suffer discrimination. With the introduction of this new set of algorithms and metrics, new problems arise, increasing the complexity of the Data Scientist's analysis to obtain the best models possible to these contexts. This Master's thesis aims to contribute to the topic of Artificial Intelligence (AI) from the point of view of Software Engineering, presenting a *Workflow* for *Machine Learning* applications that can be run autonomously without the need to do trial-and-error cycles with a lot of techniques and can be more assertive in finding options that are more optimized for different contexts. For this, the Pipe-and-Filter architecture is used to develop the *Workflow*, using data provenance to record metadata, and the MAPE-K architecture to determine this autonomy. Several case studies were carried out to determine if MAPE-K can be viable in solving these problems, and if *Workflow* can be evolved without great complexity, and it was possible to notice that the proposed architectures managed to be robust and modular, enabling Software Engineering studies in applications with the responsible use of data.

Keywords — *Workflow*, *Machine Learning*, Artificial Intelligence, Autonomic Computing, *Fairness Metrics*

Listas de Figuras

2.1	Ciclo de vida e ecossistema do dado [23].	20
2.2	Exemplo de uma rede neural utilizada em <i>Deep Learning</i>	23
2.3	Processo padrão para aprendizado de máquina	33
2.4	IBM Analytics and AI Reference Architecture.	34
2.5	Ciclo de vida da proveniência. [66]	36
2.6	Diagrama de funcionamento da arquitetura MAPE-K [18].	39
2.7	Diagrama de uma arquitetura <i>Pipe-and-Filter</i>	40
3.1	Diagrama de atividades com subdivisão de cada papel em uma aplicação de IA utilizando métricas de Fairness, com base na IBM AI Reference Architecture [5]	44
3.2	<i>Assurance Cases</i> feitos para detalhar os objetivos necessários para executar o Workflow mais adequado	46
3.3	Diagrama de classes do <i>Framework</i> baseado na arquitetura <i>Pipe-and-Filter</i>	48
3.4	Fases e etapas do Workflow implementado	51
3.5	Adequação do workflow ao gerenciador MAPE-K	53
3.6	Aplicação inteira modificada e dividida em Backend e Frontend	58
3.7	Comportamento das opções de menu.	59
3.8	Configuração da etapa de análise para o workflow autônomo.	60
3.9	Configuração das métricas para etapa de análise do workflow autônomo.	61
3.10	Cenários possíveis na configuração das métricas.	62
3.11	Configuração da etapa de planejamento para o workflow autônomo.	63
3.12	Execução simples e manual do Workflow.	64
3.13	Informações do resultado do workflow.	65
3.14	Métricas do resultado do workflow.	66
3.15	Execução autônoma do Workflow.	67
3.16	Seleção do workflow após análise.	68
3.17	Execução do workflow após seleção.	69
A.1	Fases de um processo baseado em IA explicável.	101
A.2	Ciclo resumido do processo presente em MLOps	106
A.3	Exemplo de Pipeline de MLOps com ambientes de Desenvolvimento e Produção	108

Lista de Tabelas

2.1	Matriz de confusão	23
4.1	Melhores opções escolhidas pelo modelo MAPE-K Todos os métodos - 50% Performance/50% Fairness	72
4.2	Melhores opções escolhidas pelo modelo MAPE-K Todos os métodos - 75% Performance/25% Fairness	72
4.3	Melhores opções escolhidas pelo modelo MAPE-K Todos os métodos - 25% Performance/75% Fairness	72
4.4	Melhores opções escolhidas pelo modelo MAPE-K Apenas com redução de viés no dado - 50% Performance/50% Fairness . .	73
4.5	Melhores opções escolhidas pelo modelo MAPE-K Apenas com redução de viés no dado - 75% Performance/25% Fairness . .	73
4.6	Melhores opções escolhidas pelo modelo MAPE-K Apenas com redução de viés no dado - 25% Performance/75% Fairness . .	74
4.7	Melhores opções escolhidas pelo modelo MAPE-K Apenas com redução de viés no treinamento - 50% Performance/50% Fairness	74
4.8	Melhores opções escolhidas pelo modelo MAPE-K Apenas com redução de viés no treinamento - 75% Performance/25% Fairness	74
4.9	Melhores opções escolhidas pelo modelo MAPE-K Apenas com redução de viés no treinamento - 25% Performance/75% Fairness	75
4.10	Melhores opções escolhidas pelo modelo MAPE-K Apenas com redução de viés no resultado - 50% Performance/50% Fairness	75
4.11	Melhores opções escolhidas pelo modelo MAPE-K Apenas com redução de viés no resultado - 75% Performance/25% Fairness	75
4.12	Melhores opções escolhidas pelo modelo MAPE-K Apenas com redução de viés no resultado - 25% Performance/75% Fairness	76
4.13	Melhores opções escolhidas pelo modelo MAPE-K Apenas sem redução de viés - 50% Performance/50% Fairness	76
4.14	Melhores opções escolhidas pelo modelo MAPE-K Apenas sem redução de viés - 75% Performance/25% Fairness	76
4.15	Melhores opções escolhidas pelo modelo MAPE-K Apenas sem redução de viés - 25% Performance/75% Fairness	77
4.16	Métricas de performance das execuções de Workflows	78
4.17	Métricas de Fairness das execuções de Workflows	79
5.1	Melhores opções escolhidas pelo modelo MAPE-K Todos os métodos - 50% Performance/50% Fairness	83
5.2	Melhores opções escolhidas pelo modelo MAPE-K Todos os métodos - 75% Performance/25% Fairness	84

5.3	Melhores opções escolhidas pelo modelo MAPE-K Todos os métodos - 25% Performance/75% Fairness	84
5.4	Quantidade de modificações realizadas ao adicionar um novo conjunto de dados ao <i>Workflow</i>	84
6.1	Melhores opções escolhidas pelo modelo MAPE-K Todos os métodos - 50% Performance/50% Fairness	89
6.2	Melhores opções escolhidas pelo modelo MAPE-K Todos os métodos - 75% Performance/25% Fairness	89
6.3	Melhores opções escolhidas pelo modelo MAPE-K Todos os métodos - 25% Performance/75% Fairness	89
6.4	Quantidade de modificações realizadas ao adicionar um novo algoritmo ao <i>Workflow</i>	90

Sumário

Agradecimentos	6
Resumo	7
Abstract	8
1 Introdução	14
1.1 Motivação	14
1.2 Objetivo	15
1.3 Solução Proposta	15
1.4 Resultados Obtidos	16
1.5 Organização	16
2 Conceitos Relacionados	17
2.1 Ciência de Dados e Engenharia de Dados	17
2.1.1 Engenharia de Dados	17
2.1.2 Ciência de Dados	18
2.1.3 O ciclo do dado: Semelhanças e diferenças entre Engenharia de Dados	19
2.2 Machine Learning	21
2.2.1 Métricas de Avaliação	23
2.2.2 Algoritmos	24
2.3 Fairness em Machine Learning	29
2.3.1 Métricas de Fairness	29
2.3.2 Algoritmos para redução de vieses	31
2.4 Engenharia de Software	33
2.4.1 Engenharia de Software para Aplicações de IA	33
2.5 Proveniência de Dados	34
2.6 Arquitetura de Software	37
2.6.1 Arquitetura MAPE-K	38
2.6.2 Arquitetura <i>Pipe-and-Filter</i>	39
2.7 Estado da Arte	41
3 Metodologia	43
3.1 Detalhamento do processo	43
3.2 Arquitetura do código	46
3.2.1 Transformação dos conjuntos de dados	46
3.2.2 Workflow	48
3.2.3 Componente MAPE-K	53
3.2.4 Interface Humano-Computador	57

4 Estudo de Caso 1: Classificação de Crédito (German Credit Dataset)	70
4.1 Contexto e limitações	70
4.2 Resultados e Discussões	72
5 Estudo de Caso 2: Classificação de Crédito (Lendingclub Dataset) e evolução do sistema	82
5.1 Contexto e limitações	82
5.2 Resultados e Discussões	83
6 Estudo de Caso 3: Evolução do sistema com outros desenvolvedores	87
6.1 Contexto e limitações	87
6.2 Resultados e Discussões	88
7 Conclusões	91
A Conceitos complementares	101
A.1 AI Explainability	101
A.2 <i>Fluent API</i>	103
A.3 <i>Feature Toggles</i>	104
A.4 <i>ModelOps</i>	105
A Documentação de Instalação	110
A.1 Introdução	110
A.2 Programas necessários para instalação	110
A.3 Instalação do sistema	111
A.3.1 Obtenção do código-fonte	111
A.3.2 Montagem de ambiente	112
A.3.3 Instalação das bibliotecas	113
A.4 Execução do sistema	114
A.4.1 Engenharia de dados	114
A.4.2 Workflow de IA	114
A.4.3 Autonomia do Workflow	115
A.4.4 Interface	115
B Documentação de Manutenção	117
B.1 Introdução	117
B.2 Arquitetura do Workflow e Framework	117
B.2.1 Estrutura	118
B.2.2 Operações	119
B.2.3 Ligação de Pipe com Filter	119
B.2.4 Ligação de Filter com Pipe	119
B.2.5 Seleção parcial de dados presentes no Pipe	119
B.2.6 Junção de Pipes	119
B.3 Incrementos no Workflow	119
B.3.1 Adicionando um novo Conjunto de Dados	119
B.3.2 Adicionando um novo Algoritmo	125
C Questionário Aplicado ao Desenvolvedor	132

Capítulo 1

Introdução

1.1 Motivação

Técnicas de Inteligência Artificial e Aprendizado de Máquina já são utilizadas há bastante tempo no ramo da Computação. Ramos como robótica e jogos são grandes exemplos, dada a necessidade nos mesmos de automatizar comportamentos que seriam tidos como triviais para um ser humano. Entretanto, nos últimos anos ocorreu um crescimento no uso dessas tecnologias em aplicações tradicionais, com previsão de US\$ 57 bilhões em investimentos em 2021, 480% maior em relação a 2017 [15]. No Brasil, o número de empresas de IA aumentou de 120 em 2018 para 206 em 2020 [16].

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 IA 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 tratada em consideração.

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 [17], 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 de IA falham: muitas implementações foram implementadas como *black boxes*, onde o determinante para estabelecer a confiança no modelo implementado é sua entrada e sua saída.

Um outro efeito colateral dessa estratégia é a exposição de vieses que, embora sejam vistos como não-intencionais pelos desenvolvedores por ter a possibilidade de ser um *outlier* no modelo treinado, refletem preconceitos escancarados da sociedade atual. Uma entrada de dados enviesada resulta em um algoritmo que realiza discriminações em sua classificação [29], e uma vez que as métricas utilizadas para medir a qualidade de um modelo *black box* são geralmente baseadas em acurácia, precisão e recall, discriminações não são facilmente percebidas por tais métricas. Ao mesmo tempo, um modelo *black box* pode ter uma alta dependência de poucos dados, determinando problemas de acoplamento. Para resolver este problema, é possível que a criação de um novo modelo seja uma melhor opção que realizar a correção em apenas parte dele.

Devido a esses tipos de problemas, o termo *Explainable AI* (XAI) ganha força para envolver o desenvolvimento de uma IA que seja acurada e simultaneamente transparente. Como IA possui diversos tipos de métodos diferentes para enquadrar diversos tipos de dados, o mesmo acaba se aplicando em Explainable AI, podendo enquadrar em diversos tipos de dados [82], ou dados específicos como imagens [52] e tabelas [62]. Como o objetivo em XAI é fazer com que os resultados alcançados pela solução de IA sejam compreendidos por humanos, é possível considerar este fato como requisito no design de uma solução de IA, fazendo com que a mesma seja reusável e testável.

No mesmo tema, é possível estabelecer métricas para determinar o quanto o modelo está preparado para dados sensíveis [22], termo que é conhecido como *Fairness*. Com a evolução das pesquisas na comunidade acadêmica, foram descobertos algoritmos para redução dos vieses presentes nos conjuntos de dados, como *Reweighting* [50], *Adversarial Debiasing* [89] e *Reject Option Classification* [51]. Por consequência, ocorre melhora nas métricas em questão, mas pode desfavorecer métricas que já são amplamente utilizadas como garantia de um bom modelo desenvolvido com técnicas de Aprendizado de Máquina.

1.2 Objetivo

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

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

1.3 Solução Proposta

Foi desenvolvido um sistema, que pode ser dividido em 4 etapas principais: Engenharia de dados, *Workflow* de IA, Autonomia do *Workflow* e Interface Humano-Computador, que serão detalhados no Capítulo onde se discute sobre a metodologia adotada.

Para o desenvolvimento do *workflow*, será utilizada a arquitetura *Pipe-and-Filter*. Para a autonomia deste, será criado um componente utilizando a arquitetura MAPE-K [12] para analisar uma base de conhecimento e prover o melhor pipeline seguindo regras pré-determinadas. Para a interface, ela foi criada nos moldes de uma aplicação web. O código deste desenvolvimento foi disponibilizado no GitHub¹ para avaliação e testes em estudos posteriores.

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

1.4 Resultados Obtidos

Esta pesquisa mostra que a escolha da arquitetura *Pipe-and-Filter* se mostra 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 performance e justiça 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.

Embora a proveniência de dados não teve o mesmo efeito da aplicação de um método feito para XAI, a obtenção de metadados do *workflow* se mostrou essencial para alimentar o componente baseado na arquitetura MAPE-K 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 escolhas de algoritmos para contextos distintos, e até considerar novas aplicações de IA caso o volume de dados seja consideravelmente grande. Posteriormente, é possível também rever quais dados podem ser adicionados para que o objetivo de XAI possa ser alcançado de maneira satisfatória.

1.5 Organização

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

Capítulo 2

Conceitos Relacionados

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 é muito utilizado 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 [71]. 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 [14]. 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 [14].

Nos anos 70/80, os problemas em engenharia de dados eram classificados de acordo com 3 atributos [73]:

- **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 grandes volumes de dados, 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 incluem 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 refletem 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 [71]. Essa definição enfatiza a estreita relação de ciência de dados com a mineração de dados e, consequentemente, 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 [42]. 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 [71] [42].

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. Eles usam 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

Nos tempos atuais, empresas e instituições acadêmicas utilizam Engenharia de Dados e Ciência de Dados para otimizar as suas aplicações de IA. Embora tais termos sejam distintos e seus profissionais performam tarefas distintas, na prática é frequente a função de engenharia de dados ser feita por cientistas de dados e vice-versa pois ambas se complementam muito 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 [14]. 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 [42]. Em outras palavras, 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 [23].

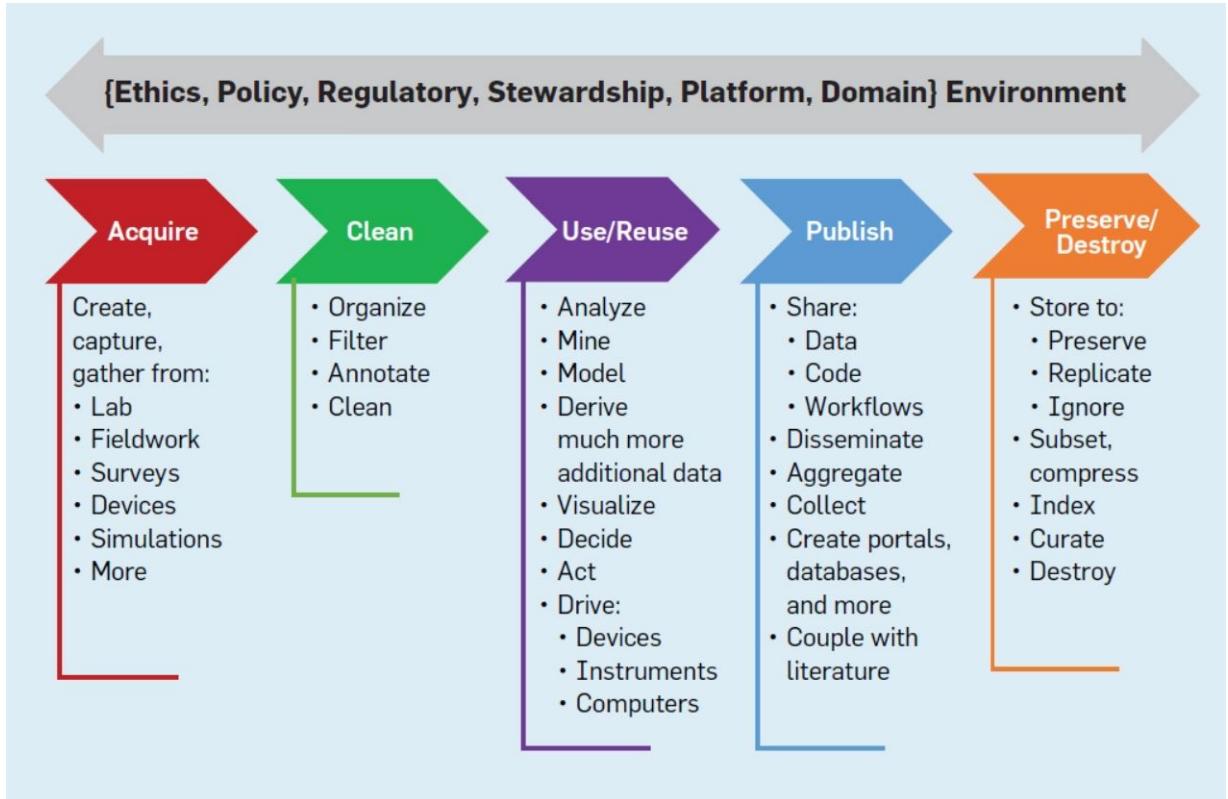


Figura 2.1: Ciclo de vida e ecossistema do dado [23].

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, sintetização 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 H adrons (LHC), representando colisões de part culas dentro de um t nel de 17 milhas para testar as previsões de v rias teorias da f sica de part culas e da f sica de alta energia [23]. A maioria dos dados gerados   tecnicamente irrelevante e s o descartados, mas isso n o impede que uma enorme quantidade de dados seja influente e continua a ser analisada e preservada. Em 2012, dados sobre experimentos do LHC forneceram fortes evid ncias para o b son de Higgs, apoiando a veracidade do Modelo Padr o da F sica. Esta descoberta cient fica foi a "Descoberta do Ano" de 2012 da revista *Science* e o Pr mio Nobel de F sica em 2013.

As estimativas s o de que, em 2040, haver  de 10 exabytes a 100 exabytes de dados influentes produzidos pelo LHC. Os dados retidos do LHC s o anotados, preparados para preserv o e arquivados em mais de uma d zia de locais f sicos. O resultado desse processo   divulgado   comunidade para an lise e uso em mais de 100 outros locais de pesquisa. Al m do desenvolvimento de protocolos de administra o, dissemin o e uso de dados, o ecossistema de dados do LHC tamb m fornece um modelo econ mico que suporta de forma sustent vel os dados e sua infraestrutura. Essa combina o entre administra o dos dados e administra o econ mica 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 cont nua de a es e transforma es nos dados, mas em muitas comunidades cient ficas e disciplinas hoje essas etapas s o isoladas. Os cientistas de dom nio se concentram em gerar e usar dados. Cientistas da computa o podem se concentrar em quest es de plataforma e desempenho, incluindo minera o, organiz o, modelagem e visualiza o, bem como os mecanismos para extrair significado dos dados por meio de aprendizado de m quina e outras abordagens. Estat sticos podem se concentrar na matem tica dos modelos de risco e infer ncia [23]. Engenheiros de dados podem se concentrar na administra o e preserv o de dados gerados pelo cientista de dom nio e no *backend* do pipeline, seguindo a aquisi o, decis es e a es no dom nio da publica o, arquivamento e curadoria. Cientistas de dados podem unir o trabalho dos cientistas da computa o e dos estat sticos para extrair novos conhecimentos e conhecer tomadas de decis o mais eficientes.

2.2 Machine Learning

Aprendizado de M quina (*Machine Learning*, em ingl s) pode ser definido como "a pr tica de usar algoritmos para coletar dados, aprender com eles, e ent o fazer uma determina o ou predi o sobre alguma coisa no mundo. Ent o em vez de implementar as rotinas de software manualmente, com um gama espec fica de instru es para completar uma tarefa em particular, a m quina   'treinada' usando uma quantidade grande de dados e algoritmos que d o e ela a habilidade de aprender como executar a tarefa" [3]. Com isso, o computador consegue a habilidade de realizar determinado c lculo ou tarefa sem que necessite de programaci o adicional ou interfer ncia humana para isso.

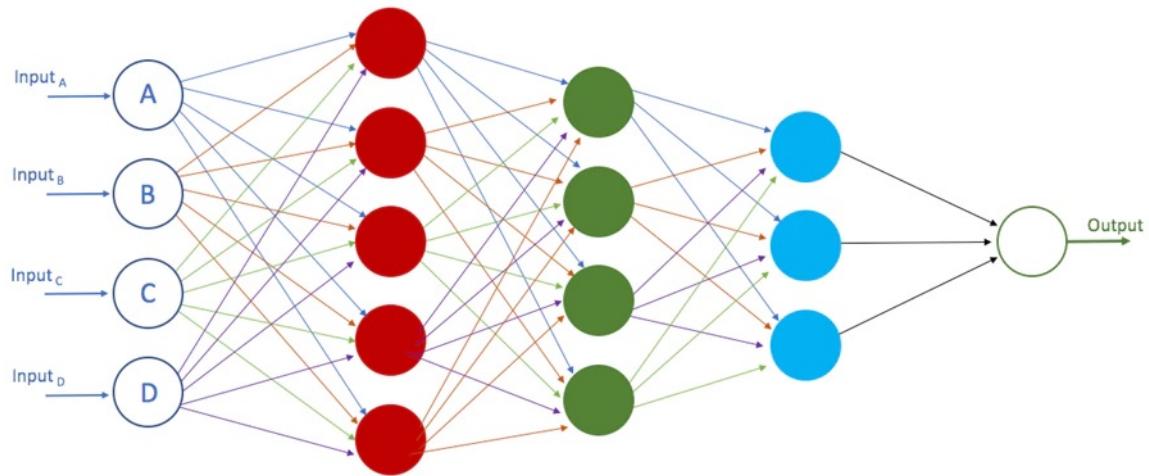
O *Machine Learning*   fortemente relacionado com a Estat stica, uma vez que seus m todos e parte de seus algoritmos, como regress es, tiveram como base modelos estat st icos.

ticos e a análise de seus dados. As tarefas de aprendizado podem ser classificadas em três categorias básicas [7] [6]:

- **Aprendizado supervisionado:** O treinamento é realizado por meio de exemplos rotulados, como uma entrada na qual a saída desejada é conhecida. Através de métodos como classificação, regressão e *gradient boosting*, o aprendizado supervisionado utiliza padrões para prever os valores de rótulos em dados não-rotulados adicionais. O aprendizado supervisionado é comumente empregado em aplicações nas quais dados históricos preveem eventos futuros prováveis.
- **Aprendizado não-supervisionado:** É utilizado em dados que não possuem rótulos históricos. A “resposta certa” não é informada ao sistema, o algoritmo deve descobrir o que está sendo mostrado. O objetivo é explorar os dados e encontrar alguma estrutura dentro deles. Técnicas populares incluem mapas auto-organizáveis, mapeamento por proximidade, agrupamento *k-means* e decomposição em valores singulares. Esses algoritmos também são utilizados para segmentar tópicos de texto, recomendar itens e identificar pontos discrepantes nos dados.
- **Aprendizado por reforço:** O algoritmo descobre através de testes do tipo “tentativa e erro” quais ações rendem as maiores recompensas. Este tipo de aprendizado possui três componentes principais: o agente (o aprendiz ou tomador de decisão), o ambiente (tudo com que o agente interage) e ações (o que o agente pode fazer). O objetivo é que o agente escolha ações que maximizem a recompensa esperada em um período de tempo determinado. O agente atingirá o objetivo muito mais rápido se seguir uma boa política, então o foco do aprendizado por reforço é descobrir a melhor política.

O termo *Machine Learning* 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 abstruído conforme ilustração na Figura 2.2. Dependendo da tarefa a ser realizada, cada neurônio atribui um peso para os dados que entram e a saída final é determinada pelo total desses pesos [3]. 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.

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 *Machine Learning* é computacionalmente muito custosa, especialmente quando redes neurais são utilizadas. E sua viabilidade é acompanhada de efetividade: 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 [3].

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

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 um caso previsto para estar na classe positiva quando o resultado real pertence à classe negativa, e um Falso Negativo (FN) ocorre quando um caso previsto para estar na classe negativa quando o resultado real pertence à classe positiva.

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 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 quanto a aplicação do modelo ao problema, como classes desequilibradas.

A **precisão** ajuda quando os custos de falsos positivos são altos 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, ajuda quando o custo 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 uma medida geral da acurácia de um modelo que combina precisão e recall. Pode ser interpretado como uma ponderação entre ambos e usa a média harmônica 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 uma classificação falsa. 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 Algoritmos

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.

Régressão Logística

A Régressão Logística é um método de classificação baseado na aplicação da técnica de Régressão Linear. Como na Régressão Linear, ele assume 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 w no peso das *features* e o termo de viés é b , a probabilidade estimada do modelo de régressão 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 facilmente 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 atual e o resultado previsto. Para medir o quanto próximo a função resultante do treinamento se aproxima do conjunto de dados, a seguinte função de custo é calculada conforme equação abaixo. A função 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*. 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) [81]. 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" [86]. 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 "vetores de suporte" 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, 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 [86]. 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 dimensionamento ou centralização de dados. Treinando uma Árvore de Decisão, geralmente, é realizada 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 [74], 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 [27]. A principal ideia que o diferencia de uma Árvore de Decisão é o melhorar a redução de variância do *Bagging* por diminuição de correlação entre as árvores sem aumentar muito a variância, pois se considera apenas um subconjunto aleatório de variáveis a cada passo [27].

Bagging é uma técnica que gera uma coleção de classificadores introduzindo randomização na entrada do algoritmo, geralmente com excelentes resultados [86]. No *Random Forest*, ela é utilizada para gerar uma coleção de Árvores de Decisão simplificadas com a finalidade de generalizar o resultado no conjunto da obra.

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* [44] [13]. *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 criação de subconjuntos não é feita de maneira aleatória, e sim feita priorizando subconjuntos mal classificados [13].

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. Eles são poderosos, escaláveis e capazes de trabalhar com grandes tarefas complexas de aprendizado de máquina, como serviços de reconhecimento de imagem e reconhecimento de fala.

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 [63].

Desde então, o desenvolvimento dessas técnicas passou por altos e baixos, até ter uma adoção considerável graças aos avanços significativos no poder computacional tanto em hardware quanto em software, é possível treinar Redes Neurais complexas, e há uma enorme quantidade de dados disponíveis para uso em bancos de dados. As Redes Neurais podem ser divididas em dois grupos principais [79]:

- **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.

O processo simplificado 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.

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) [76], *Momentum* [70], *Nesterov Accelerated Gradient* (NAG) [83], *Adaptive Gradient* (AdaGrad) [37], *Root mean square prop* (RMSprop) [47], *Adaptive moment estimation*, (Adam) [59], *Nesterov and Adam optimizer* (Nadam) [36].

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 que 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 [40].

2.3 Fairness em Machine Learning

Como a coleta de dados está presente atualmente no dia-a-dia de variados setores da sociedade, o uso de *Machine Learning* 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 [64].

Estes dados sensíveis, tendo como exemplos cor de pele, raça, sexo, idade e altura, são considerados atributos protegidos, que precisam ser classificados e processados antes da execução de um algoritmo de *Machine Learning*, determinarão como o algoritmo se comportará e, consequentemente, afetará suas métricas [67]. 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 [22]. 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 quanto 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** (TVP, ou TPR) pelo termo em inglês **True Positive Rate**) é outro termo para denominar o recall, a **Taxa de Falsos Positivos** (TFP, ou FPR pelo termo em inglês **False Positive Rate**) é a fração de casos negativos previstos incorretamente como estando na classe positiva 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 [88]:** Esta métrica 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*) [24]:** É a diferença entre a taxa positiva verdadeira do grupo não privilegiado e a taxa positiva verdadeira do grupo privilegiado:

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

Um valor próximo de 0 é considerado justo. Um classificador binário satisfaz a igualdade de oportunidades quando a taxa positiva verdadeira de ambos os grupos são iguais [48]

- **Diferença de probabilidade média (*Average odds difference*) [24]:** Essa métrica usa a taxa de falsos positivos e a taxa positiva verdadeira para calcular a tendência, calculando a igualdade de probabilidades com a fórmula:

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

Precisa ser próximo a 0 para ser considerado justo.

- **Impacto de disparidade (*Disparate impact*) [24]:** Para esta métrica, é usada a seguinte fórmula:

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

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

- **Índice de Theil (*Theil index*) [80]:** Esta medida também é conhecida como índice de entropia generalizado, mas com α igual a 1 [80]. É calculado com a seguinte fórmula:

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

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

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, 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平衡adas se o classificador for treinado com os dados já balanceados [30]. Nesta categoria se enquadram os seguintes algoritmos:

- **Reposição (*Reweighting*) [50]:** 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*) [41]:** Edita valores de *features* aumentando a justiça de cada grupo enquanto preserva a ordem de classificação dentro dos mesmos.
- **Aprendizado de representações justas (LFR, ou *Learning fair representations*) [88]:** 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 efetuando justiça em cada grupo, distorção individual, garantindo a fidelidade de dados através de restrições.

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 [?]. Nesta categoria se enquadram os seguintes algoritmos:

- **Remoção de viés adversário (Adversarial debiasing) [89]:** 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) [41]:** Técnica que adiciona no algoritmo escolhido um termo de regularização baseado na discriminação (No caso da biblioteca AI Fairness 360 é usado o programa da publicação, que usa a regressão logística como algoritmo base).
- **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 por Subgrupos Ricos (Rich Subgroup Fairness) [53]:** 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) [19]:** Aprende um classificador baseado em Gradiente Exponencial que tende a minimizar o erro de uma classificação ponderada.
- **Redução por busca em grid (Grid Search Reduction) [19] [20]:** Aprende um classificador baseado na busca em um grid 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:

- **Igualdade de probabilidade calibrada (Calibrated Equalized odds) [69]:** Otimiza as previsões do classificador obtido, calibrando para alterar os rótulos de saída e obter probabilidades igualadas entre os grupos.
- **Igualdade de probabilidade (Equalized odds) [48]:** Resolve um problema linear para alterar os rótulos de saída e obter probabilidades igualadas entre os grupos.
- **Classificação baseada em Rejeição de Opções (Reject Option-based Classification) [51]:** 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

2.4.1 Engenharia de Software para Aplicações de IA

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 Machine Learning 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 do mesmo é disponibilizado para aplicações externas.

No processo, ilustrado na Figura 2.3, o conjunto de dados passa por um pré-processamento e 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.

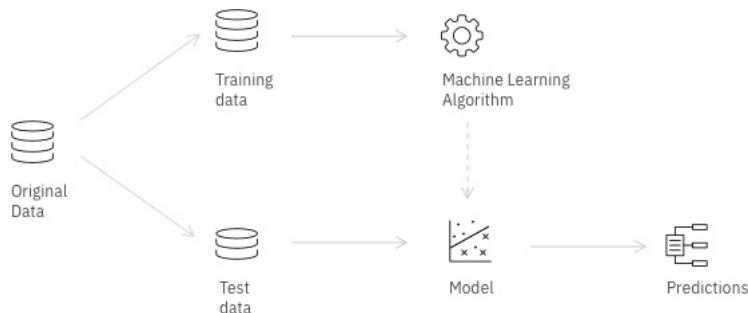


Figura 2.3: Processo padrão para aprendizado de máquina

Um exemplo de arquitetura que generaliza todo o processo, desde a necessidade de negócio até o deploy 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:

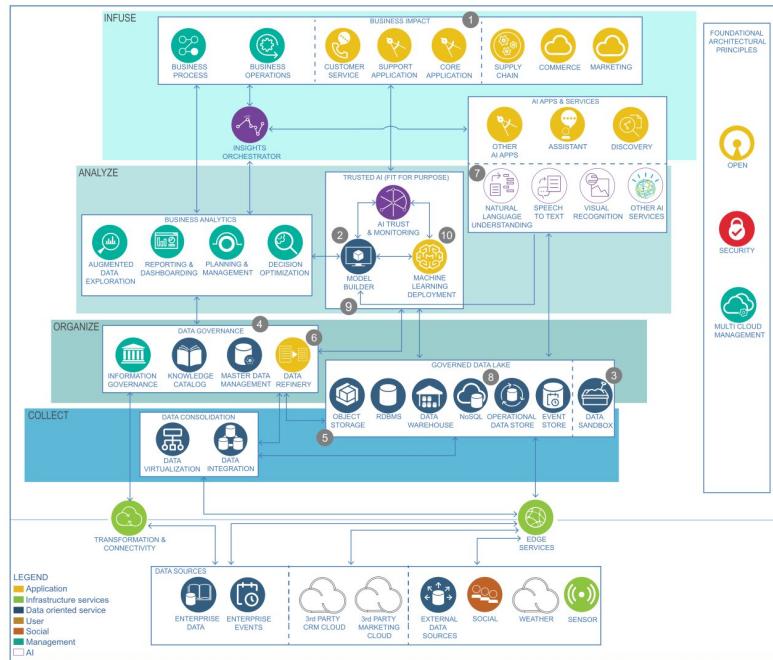


Figura 2.4: IBM Analytics and AI Reference Architecture.

- **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.

2.5 Proveniência de Dados

O termo proveniência é comumente usado no contexto da arte para denotar a história documentada ou a cadeia de propriedade de um objeto de arte [65] [66] [39] [84]. A proveniência ajuda a determinar a autenticidade e, portanto, o valor dos objetos de arte. Passando para o contexto da computação e do processamento de dados, a proveniência dos dados, às vezes chamada de linhagem ou *pedigree*, é a descrição das origens de um dado e o processo pelo qual ele chegou a um banco de dados [28], contendo metadados informando "como", "quando", "onde", "por que" ele foi obtido e "quem" o obteve. Em outras palavras, o conceito, não somente inclui a origem do dado (identificação, responsável pelo dado, data de criação), mas também os processos aplicados a ele (algoritmos e os

parâmetros utilizados para executá-lo) [87]. Os principais benefícios da proveniência para a qualidade de dados são [26]:

- **Comunica a qualidade de dados:** confiabilidade, adequação, acurácia, atualidade, redundância;
- Melhora a interpretação do dado: em relação a função do reconhecimento da fonte e na utilização do dado para um aspecto de tomada de decisão.
- **Justificativa do uso de um determinado dado:** em relação as limitações e intenções originais do uso de um determinado dado de conjuntos de dados ambientais.
- **Redução de erros:** no quesito do juízo da precisão do dado, no acompanhamento preciso da linhagem de conjuntos de dados científicos.
- **Passos do processamento:** permite que usuários não especialistas em dados entendam a capacidade de recuperar e entender os relacionamentos entre produtos de dados, scripts ou dados gerados por programas.
- **Criação de dados científicos:** permite identificar o processo utilizado para ajudar a identificar e avaliar os componentes básicos dos sistemas que fornecem a recuperação de linhagem para produtos de dados científicos.
- **Atualização de dados:** permite a partir do desenvolvimento de estudos formais para executar rastreamento de linhagem de dados em visões relacionais.
- **Modificação de *schemas* de visões relacionais:** modelos gráficos e estudos experimentais.
- **Fontes de dados históricas:** permite identificar a origem e o subsequente histórico de processamento.

Tanto a comunidade científica quanto a empresarial adotaram o estilo de arquitetura orientada a serviços (SOA), que permite que os serviços sejam descobertos e compostos dinamicamente [66]. Os aplicativos baseados em SOA tornam-se mais dinâmicos e abertos, mas também precisam atender a novos requisitos. É preciso verificar se o processo que os trouxe resultados está em conformidade com regulamentações ou metodologias específicas, provar que os resultados são derivados independentemente de serviços ou bancos de dados com determinadas restrições de licença, e estabelecer que os dados foram capturados na fonte por instrumentos que ofereçam confiabilidade.

Como tais verificações não são automatizadas, há sempre a possibilidade de erro humano: uma etapa pode não ser feita ou feita parcialmente, por uma falha em alguma verificação, ou mesmo por falta de suporte. Os dados podem não conter as informações históricas necessárias para fazerem as verificações. Portanto, há a necessidade de capturar informações extras (documentação do processo) que descrevam o que realmente ocorreu durante a execução. A documentação do processo é para os dados o que um registro de propriedade é para uma obra de arte. Os aplicativos capazes de realizar proveniência criam a documentação do processo e a armazenam em uma base de proveniência, cujo

papel é oferecer um armazenamento seguro e persistente de longo prazo da documentação do processo, conforme ilustrado na Figura 2.5. Esse papel acomoda várias implantações: por exemplo, uma base de proveniência pode ser um serviço único e autônomo ou, para ser mais escalável, pode ser uma coleção de bases distribuídas [66].

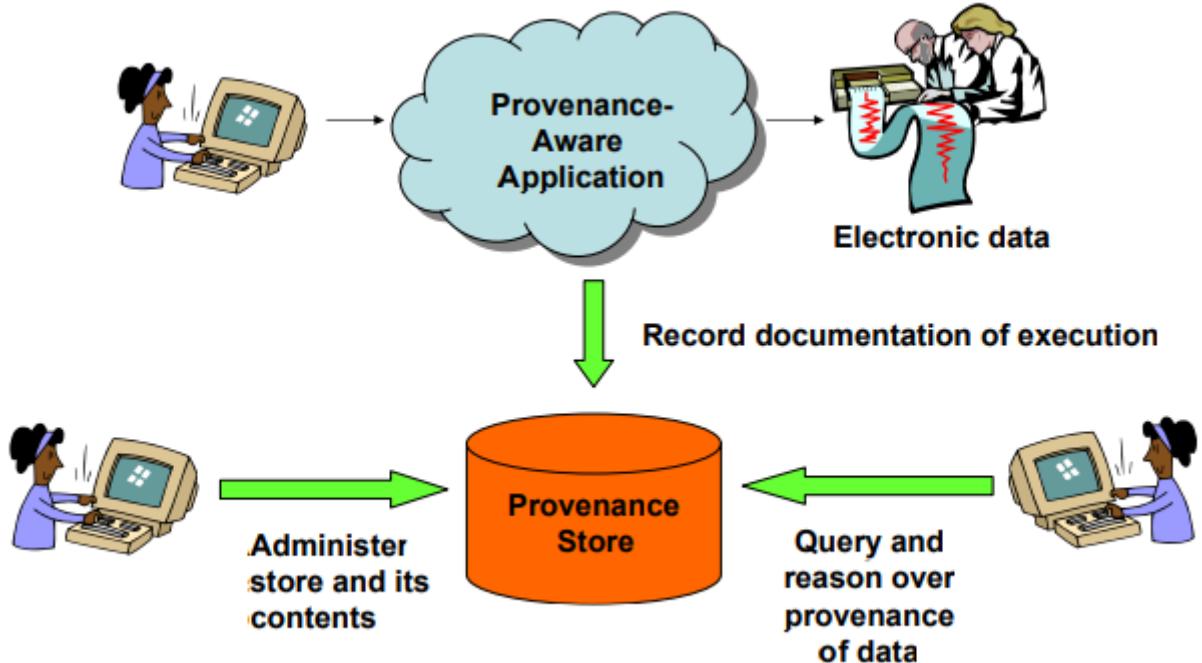


Figura 2.5: Ciclo de vida da proveniência. [66]

Uma vez registrada a documentação do processo, o resultado da proveniência dos dados pode ser recuperada consultando a base de proveniência e analisada para atender as necessidades do usuário da aplicação. Com o tempo, é preciso gerenciar e manter a base de proveniência e o conteúdo já obtido. Dado esse ponto, é possível dividir o ciclo de vida da proveniência de dados em quatro fases diferentes, sendo válido para todos os sistemas capazes de realizá-la [66]:

- Criação
- Registro
- Consulta
- Gerenciamento

É possível utilizar *workflows* para gerenciar a proveniência [35]. No domínio científico, um *workflow* é normalmente usado para executar tarefas complexas de processamento de dados. Um *workflow* pode ser pensado como um programa que é uma série de etapas de computação e etapas definidas por processos executados manualmente por uma ou mais pessoas. A proveniência do *workflow* refere-se ao registro de todo o histórico da saída final do *workflow* [84]. Nesse contexto, há duas formas distintas de proveniência [35]:

- **Prospectiva:** Trata-se da sequência de processos utilizados para a geração do dado, ou seja, a captura dos passos que devem ser seguidos para a geração de um dado produto.
- **Retrospectiva:** Trata-se das informações obtidas durante a execução dos processos de geração do dado. Compreende desde o tempo de duração de cada atividade executada até a origem dos dados de entrada. Além disso, não depende do tratamento da proveniência prospectiva para ser utilizado. Em outras palavras, é como se fosse um *log* detalhado da execução de uma tarefa.

Um importante componente da proveniência é a obtenção de informações sobre causalidade. Nesse componente é guardada a descrição do processo, ou a sequência das etapas, que, junto com dados de entrada e seus respectivos parâmetros, levam à criação de uma base de dados. Assim as dependências dos processos são usadas para documentar sua criação, bem como auxiliar na reprodução e validação desse processo. A causalidade pode ser inferida tanto para a forma prospectiva como para a forma retrospectiva de proveniência [35].

Outro componente-chave para a proveniência são as informações definidas pelo usuário, a documentação. Por serem dados, em geral, advindos dos processos de anotação, não são capturados automaticamente. Com isso, possuem diferentes níveis de granularidade e podem estar associados tanto para a forma prospectiva como para a forma retrospectiva de proveniência. Esse tipo de registro se torna muito importante, pois contém informações sobre decisões tomadas e observações feitas pelo usuário [35].

2.6 Arquitetura de Software

O consenso sobre a definição de arquitetura de software foi adotado 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 [25].

O nível de design da arquitetura de software em um projeto 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 [45]. 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.

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 [45]. 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 [25].

2.6.1 Arquitetura MAPE-K

Em 2001, Paul Horn introduziu o conceito de Computação Autônoma como alternativa a solução para a crescente complexidade dos sistemas da época, onde previa-se que os mesmos se tornariam muito grandes e complexos até mesmo para os profissionais mais qualificados configurarem e realizarem manutenção. Tal conceito qualifica sistemas de computação que podem se autogerenciar com relação aos objetivos de alto nível dados pelos administradores e é derivado da biologia, dado a grande variedade e hierarquia de sistemas autônomos presentes na natureza e na sociedade [55].

Em um ambiente autônomo e autogerenciado, os componentes de sistema podem incorporar como funcionalidade um *loop* de controle. Embora estes *loops* sejam divididos nos mesmos procedimentos, é possível categorizá-los em 4 categorias principais. Essas categorias são consideradas atributos dos componentes do sistema e são definidas como [12]:

- **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 [18], 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 2.6. O MAPE-K é um conceito que constitui um *loop* de controle, usado para monitorar e controlar um ou mais elementos gerenciados. Um elemento gerenciado (*Managed Element*) pode ser um hardware, como uma impressora, um software, como um banco de dados, outro Elemento Autônomo ou 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 efetor (*Effector*) ou mais executam as ações planejadas no recurso.
- **Conhecimento (Knowledge):** A base de conhecimento é central e acessível por todas as partes do *loop*. Separado a partir de dados coletados e analisados, ele contém conhecimento adicional, como modelos de arquitetura, modelos de metas, políticas e planos de mudança.

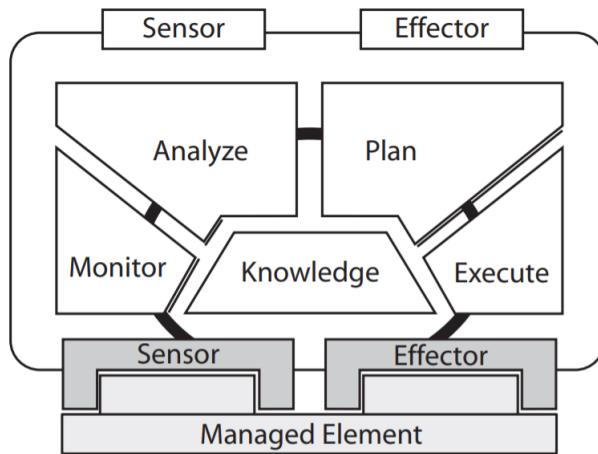


Figura 2.6: Diagrama de funcionamento da arquitetura MAPE-K [18].

2.6.2 Arquitetura *Pipe-and-Filter*

Em uma arquitetura *Pipe-and-Filter*, ilustrado na Figura 2.7, 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*) [45].

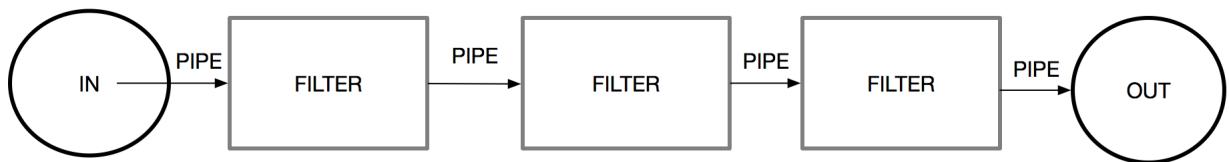


Figura 2.7: Diagrama de uma arquitetura *Pipe-and-Filter*.

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 *Pipe-and-Filter* possui como vantagens:

- Permitem 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.
- Suportam 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 *Pipe-and-Filter* 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 com um único filtro, 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.

2.7 Estado da Arte

Os principais trabalhos encontrados envolvendo métricas de *Fairness* e sua arquitetura vêm do time de engenharia do LinkedIn. Geyik et al. [46] falam sobre as métricas utilizadas e os algoritmos implementados sobre ranqueamento dos dados. Kenthapadi et al. [54] fala sobre como o conceito de *Fairness* pode ser implementado em uma aplicação de IA, e mostra a arquitetura sobre ela.

Quanto ao desenvolvimento do *Workflow* e arquiteturas, é possível mencionar a própria IBM AI Reference Architecture [5] como modelo para desenvolvimento de aplicações. Uma outra arquitetura que é comparável a arquitetura *Pipe-and-Filter* é a FBFlow do Facebook [38]. 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, onde é possível conectar o método de ML juntamente com o cálculo de métricas para serem automatizados e organizados de forma simultânea. Há diferenças entre DAGs e *pipes* e filtros, mas o objetivo de ambos acaba sendo o mesmo: Conectar diversos processos menores para formar um sistema coeso.

Quanto ao desenvolvimento da parte que utiliza o MAPE-K para garantir a autonomia, já há diversos artigos detalhando como o MAPE-K é importante para organizar o processo de decisão, como o "*An architectural blueprint for autonomic computing*" [12], que detalha todas as suas bases. Para o desenvolvimento da parte de análise, um trabalho que pode ser mencionado é o artigo "*The VADA Architecture for Cost-Effective Data Wrangling*" [60], que utiliza pesos para dar contexto aos dados medidos. Embora não seja o principal tema do artigo, indicar a importância dos dados analisados garantiu a autonomia necessária sem exigir manutenção na parte de análise.

Quanto a Proveniência de Dados, é possível citar o artigo "*Provenance and Scientific Workflows: Challenges and Opportunities*" [35], onde é discutido como há oportunidades de prover dados em *workflows*, como captura de etapas e informações de execução e geração do dado. No contexto deste trabalho, a proveniência é utilizada para alimentar a tomada de decisão realizada pela implementação da arquitetura MAPE-K e são utilizados metadados da própria execução do *workflow*, como algoritmos utilizados em cada etapa e conjunto de dados utilizado.

Referências	Descrição	Conceitos						
		Ciência/Eng.	Dados	ML	Fairness	Eng.	Software	Prov. dados
Geyik et al., 2019	Métricas utilizadas Rankeamento dos dados			✓	✓		✓	
Kenthapadi et al., 2019	Arquitetura/implementação de métricas de Fairness			✓	✓		✓	
IBM Analytics and AI Reference Architecture	Arquitetura de referência para Dados e ML	✓		✓			✓	
An architectural blueprint for autonomic computing	Arquitetura para sistemas autônomos					✓		✓
FBFlow - Facebook (Dunn, 2016)	Uso de DAGs para determinar <i>workflow</i> Execução/interpretação de processos de ML			✓	✓	✓		✓
Konstantinou et al., 2017	Arquitetura para entrelaçamento de dados Utilização de pesos para determinar contexto aos dados	✓					✓	✓
Davidson et al., 2008	Oportunidades de proveniência de dados em <i>workflows</i> Captura de passos e dados de execução						✓	

Capítulo 3

Metodologia

3.1 Detalhamento do processo

Dado as etapas da AI Reference Architecture, foram definidos os seguintes papéis onde um projeto de Machine Learning pode ter atuação e onde se encaixariam:

- **Especialista de domínio:** É a pessoa que detém de todo o conjunto de regras do qual a aplicação deve respeitar. Pode não ter conhecimento técnico, embora esse conhecimento possa ajudar na comunicação das regras com os demais papéis. 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. 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. 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 seu uso, 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.

Dado esses papéis e suas respectivas funções, foi desenhado um diagrama de atividades, presente na Figura 3.1, determinando como eles se encaixariam no processo. Como o foco está na parte de Machine Learning, o detalhamento maior ficará na parte responsável pelo Cientista de Dados.

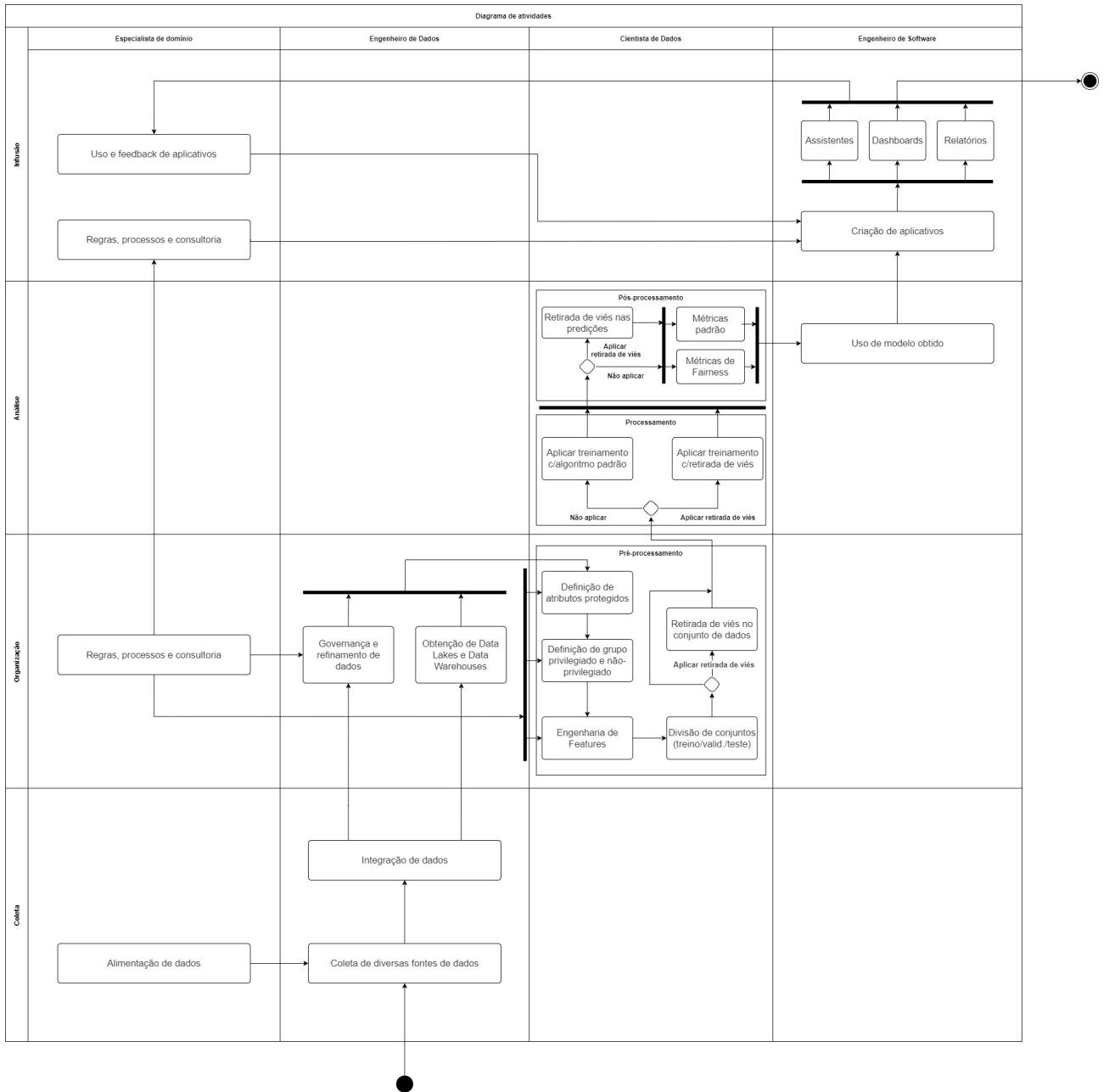


Figura 3.1: Diagrama de atividades com subdivisão de cada papel em uma aplicação de IA utilizando métricas de Fairness, com base na IBM AI Reference Architecture [5]

O desenvolvimento realizado neste trabalho pode ser dividido em 5 etapas maiores, que podem ser detalhadas e sub-divididas:

- **Obtenção dos conjuntos de dados:** Foi realizada uma pesquisa dos conjuntos de dados utilizados como experimento envolvendo algoritmos de redução de vieses e, após uma quantidade de conjuntos de dados, os mesmos foram encontrados e obtidos para serem analisados.
- **Transformação dos conjuntos de dados:** Como alguns dos conjuntos de dados obtidos diferentes valores qualitativos e codificados de acordo com o atributo,

primeiro foi realizado uma transformação dos dados para garantir uma melhor legibilidade, o que já equivaleria minimamente ao trabalho do Engenheiro de Dados presente neste trabalho de Mestrado. Posteriormente, novas transformações foram realizadas devido a necessidade de separar grupo privilegiado ao grupo não-privilegiado, mas neste trabalho já equivaleria a parte do Cientista de Dados.

- **Desenvolvimento do Workflow:** Como Machine Learning possui um processo muito bem definido, é possível subdividir este processo em etapas e subdividir cada uma dessas etapas em componentes isolados. O objetivo do desenvolvimento de um *Workflow* é sistematizar todo esse processo de modo que supostas atualizações sejam incrementais e que seja possível realizar uma execução de forma autônoma a partir da próxima etapa. Para simplificar o processo, foi utilizada a arquitetura *Pipe-and-Filter*, que é simples de ser entendida e pode ser encontrada em diferentes publicações.
- **Desenvolvimento dos processos de autonomia do Workflow:** Após o desenvolvimento do *Workflow*, foi desenvolvido um elemento autônomo para trabalhar em conjunto com o *Workflow* como elemento gerenciado. Através das métricas obtidas em execuções anteriores, é possível inferir sugestões de melhores combinações para execução e garantia de um melhor resultado de forma mais eficiente. Para garantir uma maior flexibilidade nos resultados, foram implementadas diversas estratégias para garantir que essa sugestão seja customizada pelo Cientista de Dados e/ou pelo Especialista de Domínio se isso for necessário para as necessidades do problema.
- **Interface Humano-Computador:** Para facilitar o entendimento dos arquivos utilizados para configuração e utilização das estratégias da arquitetura MAPE-K, foi criada uma interface onde é possível realizar a configuração pela mesma, podendo também realizar uma execução do *Workflow* isolada e uma execução com sugestões, acompanhando a análise realizada no gerenciador MAPE-K.

Para definição dos objetivos do *Workflow* e da sua autonomia, foi realizado um detalhamento na forma de *Assurance Cases*, onde o objetivo principal, que é a obtenção do modelo mais equilibrado entre métricas de avaliação, onde para melhor diferenciação de seu objetivo serão denominadas como métricas de Performance, e métricas de Fairness, é subdividido em diferentes estratégias, novamente subdividido em novos objetivos, e através de evidências é possível verificar o progresso do objetivo principal.

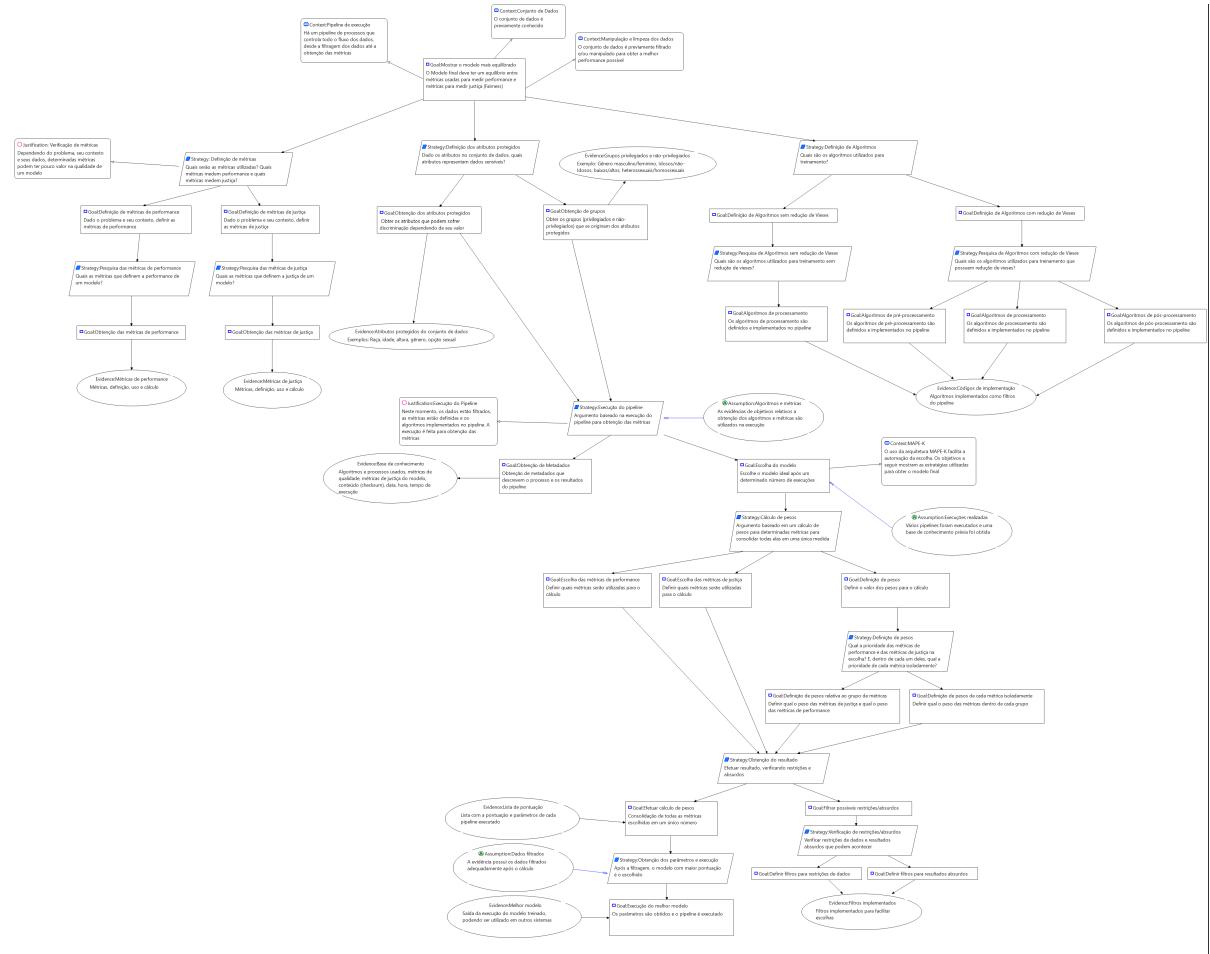


Figura 3.2: *Assurance Cases* feitos para detalhar os objetivos necessários para executar o Workflow mais adequado

3.2 Arquitetura do código

3.2.1 Transformação dos conjuntos de dados

Nesta etapa, os conjuntos de dados obtidos com atributos qualitativos codificados com uma documentação própria, como o *German Credit Dataset* [11], são transformados com nomenclaturas mais legíveis e colocados em arquivos CSV. O Trecho 3.1, colocado abaixo, ilustra um exemplo dessas operações.

```

1 def german_data_to_csv():
2     pd.set_option('display.max_columns', None)
3     df = pd.read_csv('datasets/german.data', sep=' ')
4     df.info()
5     print(df)
6
7     df['checking_account'] = df['checking_account'].map(
8         {'A11': '<0', 'A12': '0<=x<200', 'A13': '>=200', 'A14': 'None'})
9     .astype(str)
10    df['credit_history'] = df['credit_history'].map(
11        {'A30': 'no_credits_taken', 'A31': 'all_credits_paid_bank',
```

```

11     'A32': 'existing_credits_paid', 'A33': 'delay_in_past', 'A34':
12     'critical'}}).astype(str)
13     df['purpose'] = df['purpose'].map( 'radio/tv',
14         'A44': 'domestic_appliances', 'A45': 'repairs', 'A46': ,
15     education', 'A47':
16         {'A40': 'car_new', 'A41': 'car_used', 'A42': 'furniture/
17     equipment', 'A43': 'vacation', 'A48': 'retraining',
18     'A49': 'business', 'A410': 'others'}).astype(str)
19     df['savings_account'] = df['savings_account'].map(
20         {'A61': '<100', 'A62': '100<=x<500', 'A63': '500<=x<1000', 'A64':
21 : '>=1000', 'A65': 'unknown'}).astype(str)
22     df['present_employment_since'] = df['present_employment_since'].map(
23         {'A71': 'unemployed', 'A72': '<1', 'A73': '1<=x<4', 'A74': '4<=x
24 <7', 'A75': '>=7'}).astype(str)
25     df['personal_status_sex'] = df['personal_status_sex'].map(
26         {'A91': 'male_divorced/separated', 'A92': 'female_divorced/
27 separated/married', 'A93': 'male_single',
28     'A94': 'male_married/widowed', 'A95': 'female_single'}).astype(
29     str)
30     df['other_debtors_guarantors'] = df['other_debtors_guarantors'].map(
31         {'A101': 'None', 'A102': 'co-applicant', 'A103': 'guarantor'}).
32     astype(str)
33     df['property'] = df['property'].map(
34         {'A121': 'real_estate', 'A122': 'savings_insurance', 'A123': ,
35     car_other', 'A124': 'unknown'}).astype(str)
36     df['installment_plans'] = df['installment_plans'].map(
37         {'A141': 'bank', 'A142': 'stores', 'A143': 'None'}).astype(str)
38     df['housing'] = df['housing'].map(
39         {'A151': 'rent', 'A152': 'own', 'A153': 'for_free'}).astype(str)
40     df['job'] = df['job'].map(
41         {'A171': 'unemployed', 'A172': 'unskilled', 'A173': 'skilled', ,
42     A174': 'management'}).astype(str)
43     df['telephone'] = df['telephone'].map(
44         {'A191': 'none', 'A192': 'yes'}).astype(str)
45     df['foreign'] = df['foreign'].map(
46         {'A201': 'yes', 'A202': 'no'}).astype(str)
47     df['risk'] = df['risk'].map(
48         {1: 'good', 2: 'bad'}).astype(str)
49
50     print(df)
51     df.to_csv('datasets/german_credit_data.csv')

```

Código 3.1: Transformações do conjunto de dados *German Credit Dataset*

Nele, cada uma das colunas presentes no conjunto de dados que atribuem uma qualidade é renomeada de acordo com o equivalente presente em sua documentação. Esta etapa simularia uma transformação realizada por um Engenheiro de Dados para facilitar o trabalho de análise e implementação de um Cientista de Dados.

3.2.2 Workflow

Desenvolvimento de *Framework*

Para facilitar o desenvolvimento do *Workflow* e deixar seu código mais legível, foi vista durante seu desenvolvimento a necessidade de desenvolver um pequeno *Framework* baseado na arquitetura *Pipe-and-Filter* onde foram feitas adaptações devido a características e a limitações da linguagem *Python*.

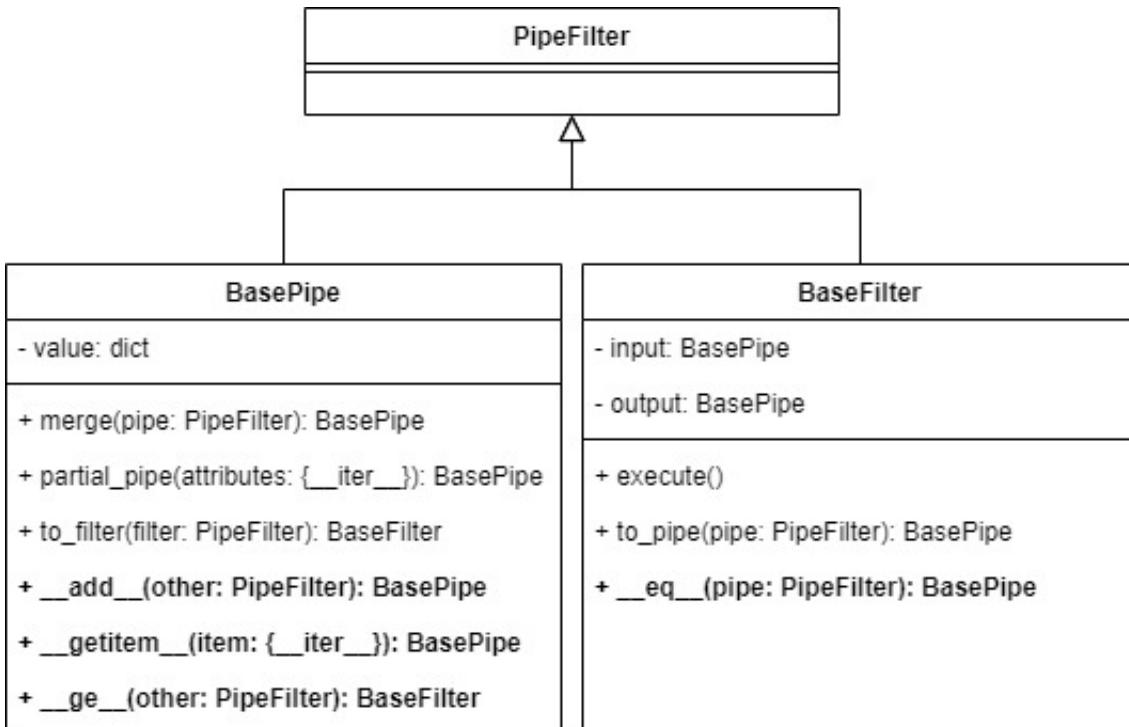


Figura 3.3: Diagrama de classes do *Framework* baseado na arquitetura *Pipe-and-Filter*

Conforme ilustrado na Figura 3.3, o *Framework* possui 3 classes: **PipeFilter**, **BasePipe** e **BaseFilter**. A classe **PipeFilter** foi criada para ser herdada pelas classes **BasePipe** e **BaseFilter** por duas limitações: Inexistência de interfaces no *Python* e evitar erro de recursão de heranças acusado no momento da compilação. A classe **BasePipe** possui apenas um atributo **value** que é um dicionário que simboliza os dados que trafegam por esse Pipe. Como é um dicionário, ele é totalmente livre de quantos atributos o Pipe trafega, sendo assim a sua validação é responsabilidade da aplicação. A classe **BaseFilter** possui dois atributos **input** e **output** que simbolizam os Pipes de entrada e de saída do Filtro, e o uso desses dois Pipes mais a execução do Filtro vai estruturar o *Workflow* como um todo.

O *Workflow* é realizado através de encadeamentos do método **to_filter** com o método **to_pipe**. O método **to_filter** prepara um Filtro para utilizar o Pipe, setando o atributo **input** presente na classe de Filtro passada como parâmetro, e o método **to_pipe** obtém o Pipe resultante do Filtro, chamando o método **execute** e obtendo o atributo **output** presente no Filtro para setar no atributo **value** presente na classe de Pipe passada como parâmetro. Isso forma encadeamentos de métodos que se assemelham muito com os de

uma *Fluent API* [43], presente no Apêndice A.2, criando uma DSL (*Domain-Specific Language*) cujo domínio é a estruturação de um *Workflow*. O Trecho 3.2 mostra um breve exemplo de como é possível encadeá-los, onde **Pipe1**, **Pipe2** e **Pipe3** são classes que herdam **BasePipe** e **Filter1** e **Filter2** são classes que herdam **BaseFilter**.

```

1 root_pipe = Pipe1()
2
3 pipe_with_filter_transformations = root_pipe.to_filter(Filter1())
4                                     .to_pipe(Pipe2())
5                                     .to_filter(Filter2())
6                                     .to_pipe(Pipe3())

```

Código 3.2: Exemplo de uso do *Framework* para encadeamento dos Workflows

Como a execução dos próprios filtros já é executada dentro do método **to_pipe**, basta apenas realizar a instância das classes e executar os métodos **to_filter** e **to_pipe** para a execução do *workflow*. Com isso, os algoritmos de Machine Learning ficam encapsulados em classes que herdam a classe **BaseFilter** e utilizam o método **execute** para executá-los. Desta forma, acontece a separação de interesses entre o componente que gerencia a execução do *Workflow* e os componentes que gerenciam cada passo do *Workflow* especificamente, garantindo um baixo acoplamento entre os componentes uma vez que é possível trocar de componentes que executam diferentes métodos sem afetar a execução do *Workflow*. Consequentemente, conforme novos métodos são descobertos, adicioná-los ao *Workflow* é extremamente simples e envolve poucas mudanças no código original. Ferramentas como o Apache Airflow [2] realizam uma abordagem semelhante através da classe **BaseOperator**, que pode ser herdada para implementar um operador customizado e executá-lo através de DAGs (*Direct Acyclic Graphs*)

Utilizando o recurso de métodos especiais presente no Python [8], é possível manipular alguns *tokens* como *proxies* para executar os métodos já presentes no *Framework*, simplificando a DSL e também simplificando o entendimento do *Workflow* caso o Cientista de Dados tenha ciência do recurso. Para o método **to_filter** foi escolhido o *token* `>=`, pela semelhança com um funil, que é geralmente assemelhado a filtros em sistemas atuais, e para o *token* o método `__ge__` é implementado como um *proxy* do mesmo. Para o método **to_pipe** foi escolhido o *token* `==`, pela semelhança com um cano (pipe, em inglês), e para este *token* o método `__eq__` é implementado como um *proxy*.

```

1 root_pipe = Pipe1()
2
3 pipe_with_filter_transformations = root_pipe >= Filter1() == Pipe2() >=
4                               Filter2() == Pipe3()

```

Código 3.3: Trecho 3.2 adaptado com métodos especiais da linguagem *Python*

Também foram implementados métodos para manipular os dicionários presentes nos Pipes conforme necessário. O método **merge(pipe)** junta os atributos presentes nos dicionários dos Pipes em um único Pipe, enquanto o método **partial_pipe(attributes)** pega apenas os atributos presentes nos dicionários dos Pipes que forem especificados em uma lista de atributos definida no parâmetro **attributes**, conforme ilustrado no Trecho 3.4.

Como métodos especiais, para o método **merge** foi escolhido o *token* `+`, pela semelhança com uma operação de adição, e para este *token* o método `__add__` é implementado como um *proxy*. E para o método **partial_pipe** foi escolhido os *tokens* `[]`, colocados após o dicionário, pela semelhança com a operação de procurar com um item no dicionário (neste caso, é possível procurar um ou mais itens), e para este *token* o método `__getitem__` é implementado como um *proxy*. O Trecho 3.5 mostra exemplos da modificação.

```

1 pipe1 = Pipe1()
2 pipe1.value = {'attribute1': 1, 'attribute2': 2}
3
4 pipe2 = Pipe2()
5 pipe2.value = {'attribute3': 3, 'attribute4': 4}
6
7
8 pipe3 = pipe1.merge(pipe2)
9 print(pipe3) # Output - {'attribute1': 1, 'attribute2': 2, 'attribute3':
    3, 'attribute4': 4}
10 pipe4 = pipe3.partial_pipe(['attribute1', 'attribute3'])
11 print(pipe4) # Output - {'attribute1': 1, 'attribute3': 3}
```

Código 3.4: Manipulações para Pipes presentes no *Framework*

```

1 pipe1 = Pipe1()
2 pipe1.value = {'attribute1': 1, 'attribute2': 2}
3
4 pipe2 = Pipe2()
5 pipe2.value = {'attribute3': 3, 'attribute4': 4}
6
7
8 pipe3 = pipe1 + pipe2
9 print(pipe3) # Output - {'attribute1': 1, 'attribute2': 2, 'attribute3':
    3, 'attribute4': 4}
10 pipe4 = pipe3['attribute1', 'attribute3']
11 print(pipe4) # Output - {'attribute1': 1, 'attribute3': 3}
```

Código 3.5: Manipulações para Pipes com métodos especiais

Após o desenvolvimento deste *Framework*, foi realizada uma refatoração com a sua utilização e o desenvolvimento do *Workflow* foi continuado.

Arquitetura e desenvolvimento do Workflow

O *Workflow* segue uma estrutura das fases de Pré-processamento, processamento e pós-processamento presentes em um processo para desenvolvimento de um modelo de Machine Learning, ilustrados na Figura 3.4. Os pontos de decisão entre redução de viés ou não presentes na figura são transparentes no código devido a implementação do *Framework* já explicado na seção anterior e foram colocados para ilustrar melhor onde atuam os algoritmos de redução dos vieses.

Além de tratar os dados, treinar o modelo e realizar testes sobre o mesmo, o *Workflow* possui etapas para obter os metadados durante o processo e salvá-los junto com as métricas para obter a proveniência dos dados necessária para integrá-lo a um elemento autônomo.

Para a execução dos algoritmos com redução de viés, 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 [10].

Devido a natureza deste trabalho de Mestrado ser voltada à Engenharia de Software e a métricas de Fairness, detalhes do processo que também poderiam ser utilizados como estratégias mais sofisticadas para validação dos modelos, como *k-fold Cross Validation*, e estratégias para regularização foram desconsiderados para evitar complexidade.

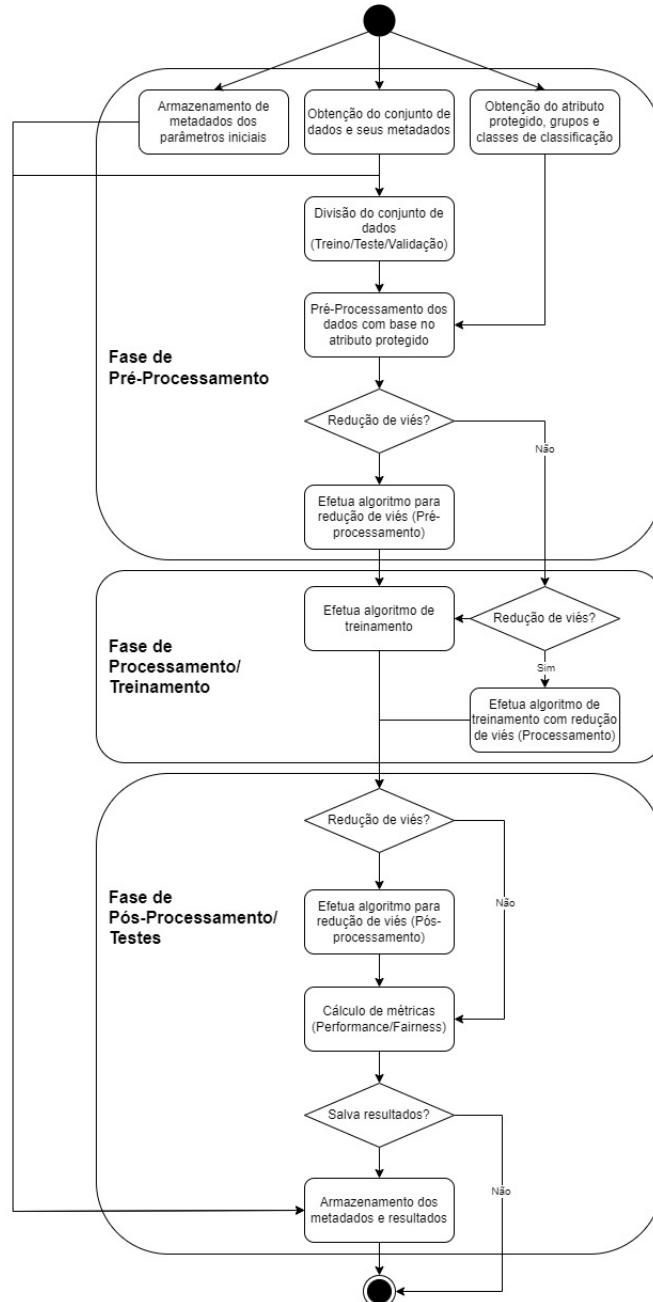


Figura 3.4: Fases e etapas do Workflow implementado

Cada etapa ilustrada na figura se comporta da seguinte maneira:

- **Fase de Pré-Processamento:**

- **Armazenamento de metadados dos parâmetros iniciais:** Os parâmetros que irão determinar quais serão os conjuntos de dados, atributos protegidos e algoritmos utilizados em cada fase do *Workflow* são armazenados em um Pipe e guardados para uma gravação em arquivo caso necessário.
- **Obtenção do conjunto de dados e seus metadados:** De acordo com o parâmetro passado para o *Workflow*, 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 *Workflow*, um Pipe relativo ao atributo protegido em questão é selecionado, e informações de grupos privilegiados e não-privilegiados e classes para classificação, necessárias para executar os algoritmos presentes no AIF360, 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 o AIF360.
- **Execução(ou não) de algoritmo para redução de viés:** De acordo com o parâmetro passado para o *Workflow*, um algoritmo para redução de viés na fase de Pré-Processamento é colocado ou não para ser executado no *Workflow*.

- **Fase de Processamento/Treinamento:**

- **Execução de algoritmo de treinamento com(ou sem) redução de viés:** De acordo com o parâmetro passado para o *Workflow*, um algoritmo de treinamento, podendo ter ou não ter redução de viés, é colocado ou não para ser executado no *Workflow*.

- **Fase de Pós-Processamento/Testes:**

- **Execução(ou não) de algoritmo para redução de viés:** De acordo com o parâmetro passado para o *Workflow*, um algoritmo para redução de viés na fase de Pós-Processamento é colocado ou não para ser executado no *Workflow*.
- **Cálculo de métricas (Performance/Fairness):** É realizado uma predição do algoritmo treinado com o conjunto de teste, e as previsões são comparadas com os valores verdadeiros para obtenção das métricas.
- **Armazenamento dos metadados e resultados:** Caso seja habilitado a gravação dos resultados, os metadados obtidos em etapas anteriores e o resultado das métricas são combinados e gravados em um arquivo JSON.

3.2.3 Componente MAPE-K

Os arquivos JSON gravados na última etapa a cada execução de *Workflow* vão ser utilizados como insumos para análises elemento autônomo, seguindo o modelo de arquitetura MAPE-K. Em um *Workflow* de Machine Learning, é possível guardar alguns metadados que podem servir de Base de Conhecimento inicial para a etapa de análise do componente a ser desenvolvido e que ajudariam na escolha do melhor modelo possível para execução:

- **Fase de Pré-Processamento:** Parâmetros utilizados para execução do *Workflow* (Conjuntos de dados, Atributos protegidos e Algoritmos utilizados) e "assinatura"/checksum do conjunto de dados utilizado.
- **Fase de Processamento:** Parâmetros utilizados para execução dos algoritmos de treinamento.
- **Fase de Pós-Processamento:** Métricas do modelo resultante (Metrics de Performance e métricas de Fairness).

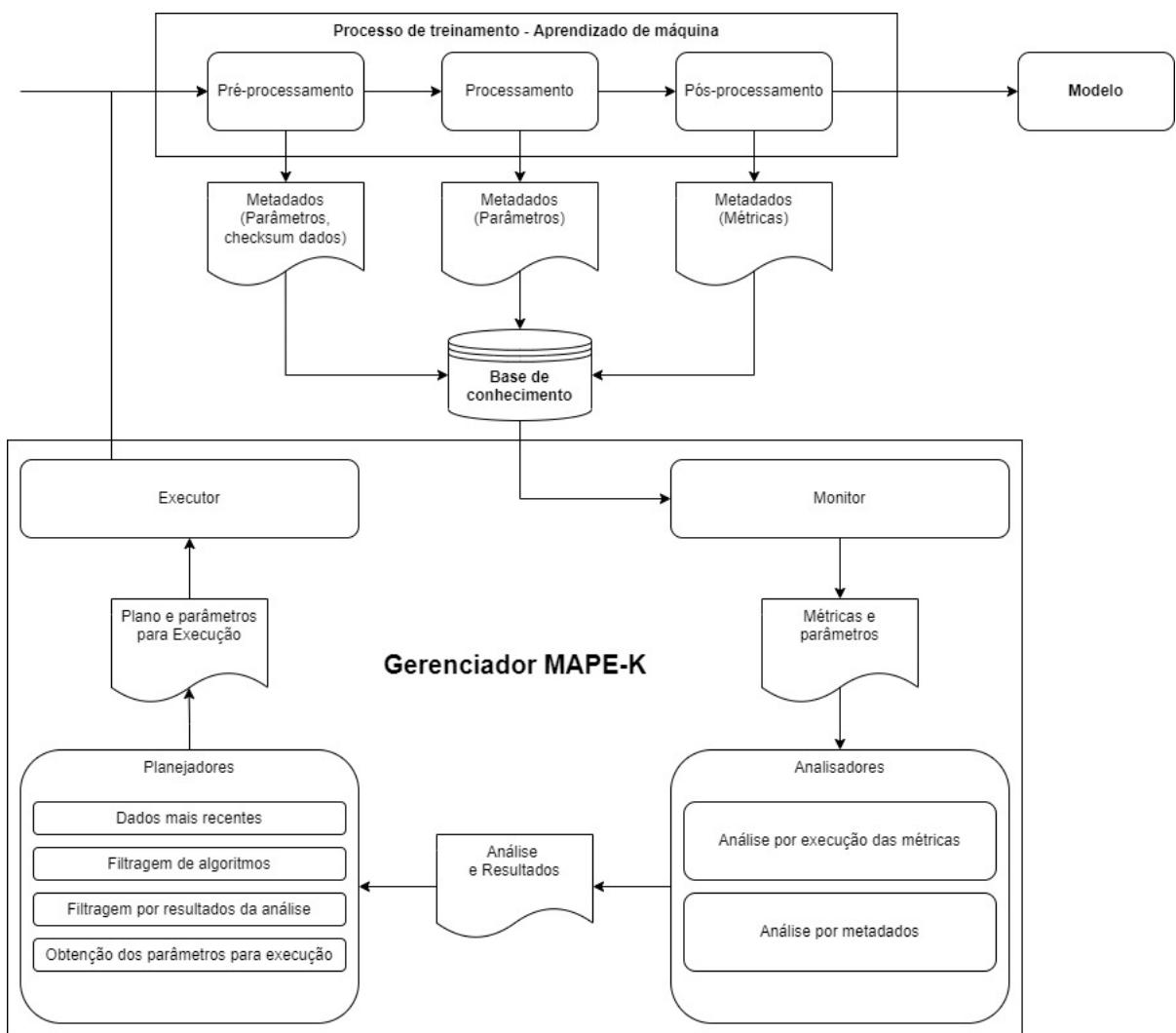


Figura 3.5: Adequação do workflow ao gerenciador MAPE-K

Após a obtenção de tal Base de Conhecimento, é possível verificar como as etapas de cada parte do MAPE-K serão desenvolvidas. Conforme ilustrado na Figura 3.5, as etapas de Análise e Planejamento foram subdivididas em diferentes estratégias para o MAPE-K ser configurável durante o processo de escolha, enquanto as etapas de monitoria e execução foram configuradas com uma única estratégia. As estratégias das etapas de Análise e Planejamento podem ser ativadas ou desativadas conforme o que é definido como *Feature Toggles* [72], presente no Apêndice A.3, por meio de arquivos JSON presentes no projeto conforme ilustrado no Trecho 3.6.

```

1 {
2     "ml_algorithm_validation": true,
3     "ml_data_checksum": false,
4     "ml_pipeline": false,
5     "ml_pipeline_threshold": true
6 }
```

Código 3.6: Exemplo de arquivo utilizando ([ver termo acima](#))

Monitoria

Para a etapa de monitoria, os dados são obtidos em cada arquivo presente na base de conhecimento e separados em dois conjuntos diferentes: Um contendo os metadados e informações relativas a execução do *Workflow* (checksum do conjunto de dados, estatísticas e parâmetros de execução do *Workflow*) e outro contendo as métricas obtidas pelo modelo resultante dessa execução, ambos possuindo um identificador do arquivo para estabelecer consistência entre os dados dos dois conjuntos. Os arquivos podem ser filtrados pelo conjunto de dados e pelo atributo protegido, ou não possuir filtro obtendo todos os dados possíveis.

Análise

Para a etapa de análise, é realizado um cálculo de pesos baseado em uma seleção livre das métricas. O motivo de existir esse cálculo é mensurar o contexto do problema de acordo com uma análise prévia do Cientista de Dados e do Especialista de Domínio, e consolidar todas as métricas para simplificar as estratégias de planejamento. É possível encontrar abordagens similares e com objetivos similares, porém em diferentes contextos, pesquisando em artigos acadêmicos [60].

```

1 {
2     "metrics_groups": [
3         {
4             "group_name": "standard",
5             "weight": 0.5,
6             "metrics": {
7                 "accuracy": {
8                     "weight": 0.5,
9                     "normalize": null
10                },
11                 "f1_score": {
12                     "weight": 0.5,
```

```

13     "normalize": null
14   }
15 }
16 ],
17 {
18   "group_name": "fairness",
19   "weight": 0.5,
20   "metrics": [
21     "statistical_parity_difference": {
22       "weight": 0.33,
23       "normalize": "diff"
24     },
25     "equal_opportunity_difference": {
26       "weight": 0.33,
27       "normalize": "diff"
28     },
29     "average_abs_odds_difference": {
30       "weight": 0.34,
31       "normalize": "diff"
32     }
33   }
34 ]
35 ]
36 }
```

Código 3.7: Exemplo de arquivo contendo informações sobre as métricas, grupos e seus respectivos pesos

As métricas são divididas em dois grupos (Métricas de performance e Metrics de Fairness), e dentro desse grupo pode-se colocar quantas métricas forem necessárias, desde que seja respeitado o contexto de cada grupo. A cada grupo é atribuído pesos diferentes, e a cada métrica desse grupo também é atribuído pesos diferentes, conforme ilustrado no Trecho 3.7. Primeiro, normaliza-se as métricas m_{F_i} para m'_{F_i} , referentes às métricas de Fairness, para todas ficarem em um intervalo de 0 a 1, conforme exibido na equação 3.1. Dessa forma, seus resultados ficam uniformes e é possível aplicar os pesos sem haver distorções no cálculo. No caso das métricas de Performance, todas possuem a mesma escala, por isso as métricas m_{P_i} não são normalizadas. Depois, multiplica-se cada uma por seus pesos correspondentes w_{P_i} e w_{F_i} , e realiza-se uma média ponderada dentro do grupo para atribuir uma pontuação S_P para o grupo das Métricas de Performance e S_F para o grupo das Metrics de Fairness, conforme exibido na equação 3.2. Para facilitar a visualização das pontuações, multiplica-se as pontuações por um fator $X = 1000$ para o intervalo da pontuação ser de 0 a 1000 e arredonda-se o número. Após tais pontuações serem obtidas, a pontuação geral S é calculada multiplicando-as por seus pesos correspondentes w_P e w_F e realizando a média ponderada, conforme exibido na equação 3.3.

$$m'_{F_i} = \begin{cases} 1 - |m_{F_i}| & \text{caso } m_{F_i} \text{ envolva diferença e } -1 < m_{F_i} < 1 \\ 0 & \text{caso } m_{F_i} \text{ envolva diferença, e } m_{F_i} \geq 1 \text{ ou } m_{F_i} \leq -1 \\ 1 - |\frac{1}{m_{F_i}} - 1| & \text{caso } m_{F_i} \text{ envolva razão e } m_{F_i} > 1 \\ 1 - |m_{F_i} - 1| & \text{caso } m_{F_i} \text{ envolva razão e } m_{F_i} \leq 1 \\ m_{F_i} & \text{caso contrário} \end{cases} \quad (3.1)$$

$$S_F = \left[X \times \frac{\sum_{i=1}^{n_F} w_{m'_{F_i}} \times m'_{F_i}}{\sum_{i=1}^n w_{m'_{F_i}}} \right] \quad (3.2)$$

$$S_P = \left[X \times \frac{\sum_{i=1}^{n_P} w_{m_{P_i}} \times m_{P_i}}{\sum_{i=1}^n w_{m_{P_i}}} \right]$$

$$S = \frac{w_F \times S_F + w_P \times S_P}{w_F + w_P} \quad (3.3)$$

Nesta etapa, foram desenvolvidas as seguintes estratégias, cujos motivos para existir e funcionamento estão explicados abaixo:

- **Análise por execução das métricas:** É atribuída a cada execução presente no conjunto uma pontuação conforme o cálculo explicado acima, e as pontuações são passadas adiante para a fase de planejamento.
- **Análise por metadados:** Para o desenvolvimento de melhores estratégias de planejamento, alguns metadados, como data de execução, são analisados e o conjunto resultante é enriquecido com esses metadados mais específicos.

Planejamento

Para a etapa de planejamento, foram desenvolvidas as seguintes estratégias, cujos motivos para existir e funcionamento estão explicados abaixo, para a escolha dos melhores modelos:

- **Dados mais recentes:** Os dados podem ser modificados de acordo com a execução e a qualidade das métricas são melhor definidas de acordo com a qualidade dos dados. Nesta estratégia, é realizado um filtro baseado na assinatura do dado com a execução mais recente, que só é possível de ser obtida se a análise dos metadados for executada.
- **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.
- **Fase por resultados da análise:** Pelo mesmo motivo da estratégia anterior, métricas não confiáveis significam uma distorção na pontuação final. Para esse caso, foi criado um limiar de pontuação mínimo e máximo para determinar pontuações que podem ser consideradas confiáveis para avaliação.

- **Obtenção dos parâmetros para execução:** Alguns dos modelos restantes são selecionados de acordo com as maiores pontuações e os parâmetros presentes nestes modelos selecionados são obtidos.

Após a execução das mesmas, caso estejam ativadas, os parâmetros selecionados são passados para a etapa de execução.

Execução

Para a etapa de execução, os parâmetros selecionados na fase de planejamento são configurados como parâmetros para a execução do *Workflow*, e esta execução pode ampliar a Base de Conhecimento ou não dependendo do valor da opção de gravação dos resultados.

3.2.4 Interface Humano-Computador

Durante o desenvolvimento do *Workflow* e de seu componente 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 das configurações e dos arquivos utilizados para a execução simples e autônoma do *Workflow*, além da própria realização destas execuções. A interface foi dividida em uma parte Frontend e outra parte Backend para ter mais flexibilidade, poder ter uma escolha de ferramentas mais adequada a cada parte e próxima ao padrão de aplicações atuais, conforme ilustrado na Figura 4.

O Backend foi desenvolvido em Python para reusar códigos já desenvolvidos em etapas anteriores, usando o framework Flask 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 *Workflow*.

O Frontend foi desenvolvido em JavaScript devido a facilidade e a robustez para construir interfaces com a linguagem e ao acervo grande de ferramentas para facilitar o desenvolvimento, usando as bibliotecas React e Redux e a biblioteca de componentes Material UI para economizar tempo com componentes já prontos, uma vez que o foco deste trabalho não exige componentes específicos para a interface. O uso da Material UI permite um *look-and-feel* similar a aplicativos desenvolvidos para o Sistema Operacional para dispositivos móveis Android, uma vez que é baseado nas *guidelines* do Material Design elaborados pelo Google.

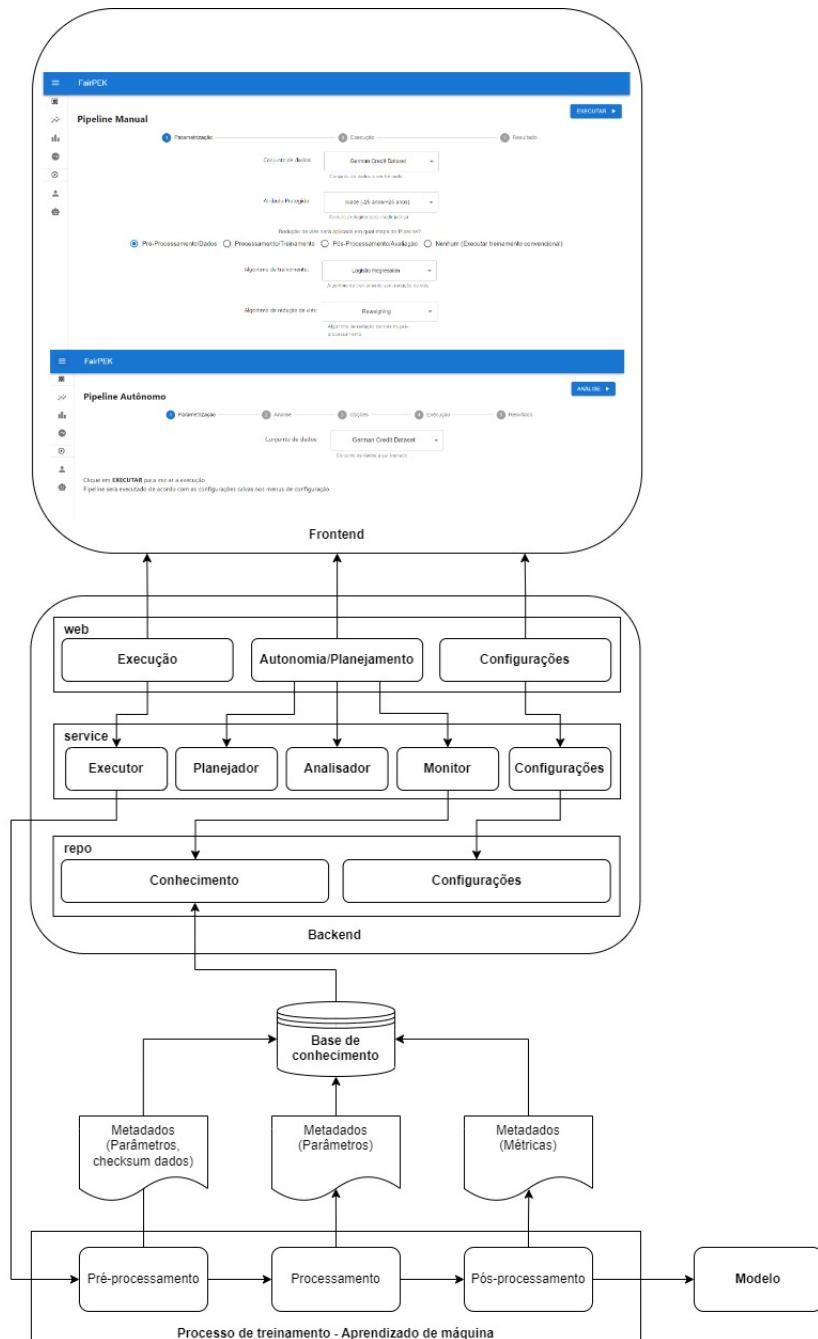


Figura 3.6: Aplicação inteira modificada e dividida em Backend e Frontend

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 3.7, 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.

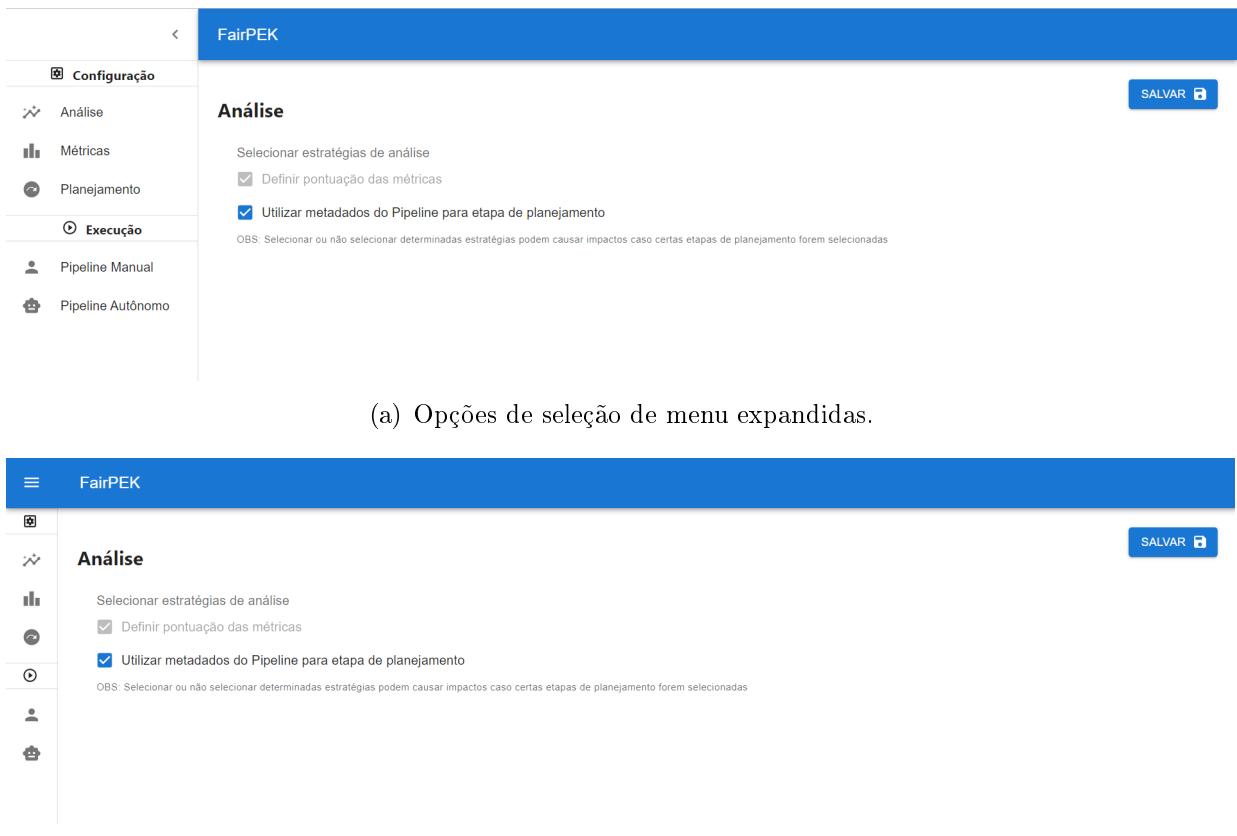


Figura 3.7: Comportamento das opções de menu.

No menu, as opções são divididas em Configuração e Execução. Em Configuração, há as opções Análise, Métricas e Planejamento relativas às configurações presentes no Componente MAPE-K. Em Execução, há as opções Pipeline Manual e Pipeline Autônomo relativas às maneiras de como realizar uma execução do *Workflow*.

Configurações para Análise

Ao clicar a opção do menu "Análise", é exibida a tela ilustrada na Figura 3.8. Ela possui apenas duas opções, relativas às estratégias desenvolvidas na parte de análise do componente MAPE-K. Ao clicar no botão "Salvar" localizado no canto superior direito, é chamada uma requisição que salva o arquivo com as opções selecionadas e é exibida uma indicação de sucesso no canto inferior esquerdo.

The figure consists of two screenshots of a software interface titled "FairPEK".

Screenshot (a): Configuration Options for Analysis Phase

- Header:** FairPEK
- Left Sidebar:** Icons for Home, Overview, Metrics, Pipeline, User, and Help.
- Section:** Análise
- Options:**
 - Selecionar estratégias de análise
 - Definir pontuação das métricas
 - Utilizar metadados do Pipeline para etapa de planejamento
- Note:** OBS: Selecionar ou não selecionar determinadas estratégias podem causar impactos caso certas etapas de planejamento forem selecionadas
- Buttons:** SALVAR (with a save icon)

Screenshot (b): Success Confirmation

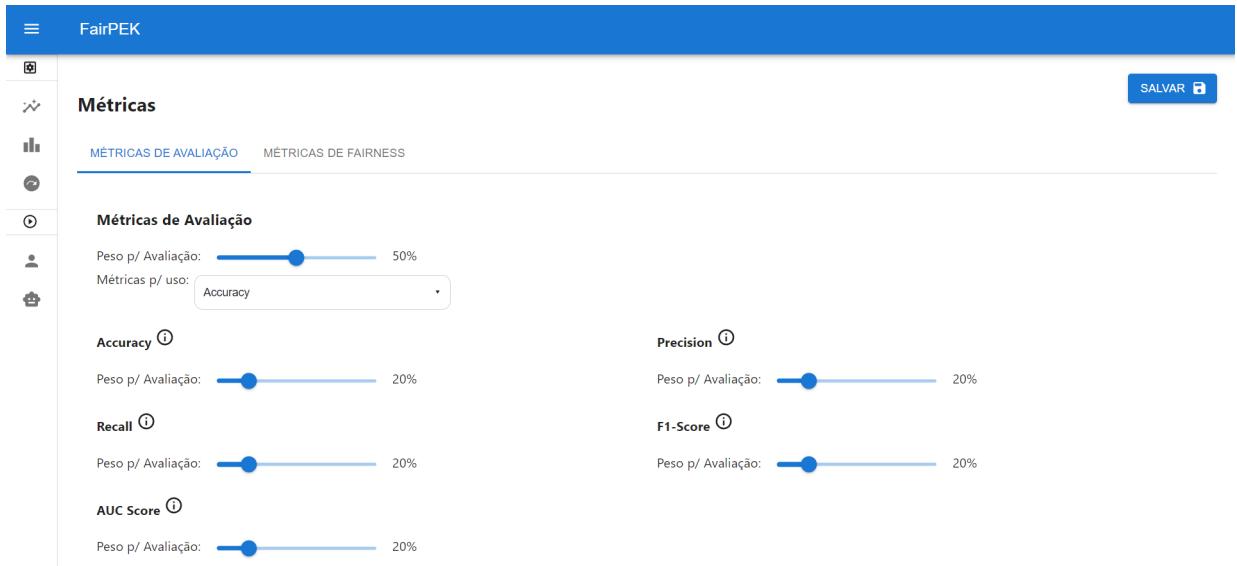
- Header:** FairPEK
- Left Sidebar:** Icons for Home, Overview, Metrics, Pipeline, User, and Help.
- Section:** Análise
- Options:**
 - Selecionar estratégias de análise
 - Definir pontuação das métricas
 - Utilizar metadados do Pipeline para etapa de planejamento
- Note:** OBS: Selecionar ou não selecionar determinadas estratégias podem causar impactos caso certas etapas de planejamento forem selecionadas
- Message:** Sucesso! Configurações salvas com sucesso!
- Buttons:** SALVAR (with a save icon)

Figura 3.8: Configuração da etapa de análise para o workflow autônomo.

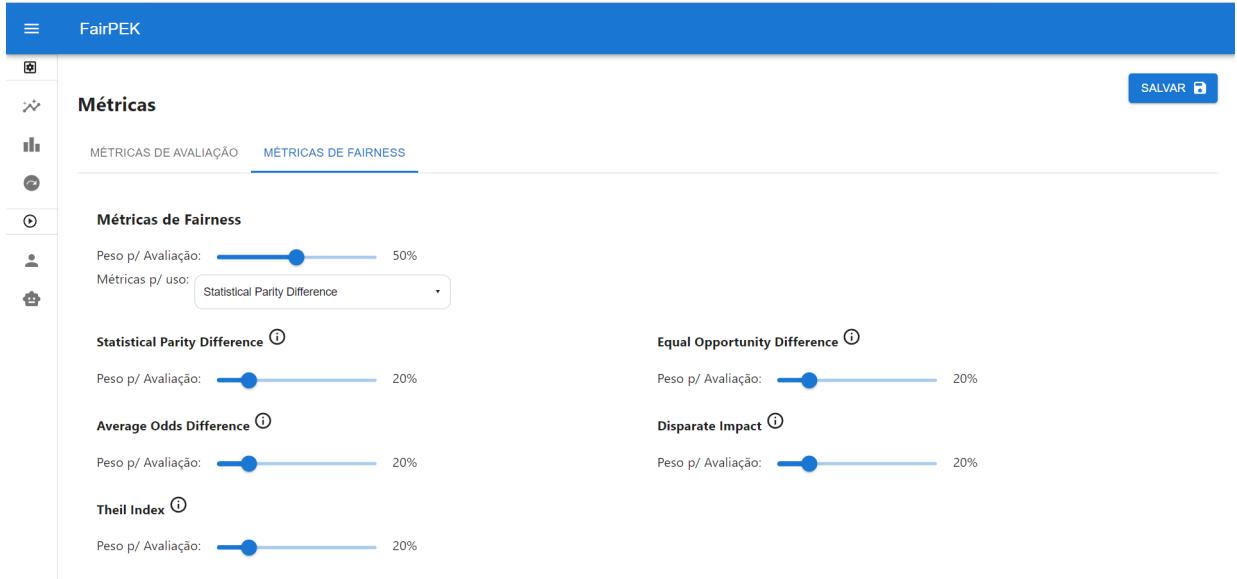
Uma das duas estratégias está desabilitada pois foi desenvolvida como a estratégia padrão usada pelo componente MAPE-K. Se a estratégia de utilizar metadados não for habilitada, certas opções marcadas na etapa de planejamento podem não funcionar.

Configurações das Métricas

Ao clicar a opção do menu "Métricas", é exibida a tela ilustrada na Figura 3.9. Ela é dividida em duas abas: Métricas de Avaliação e Métricas de Fairness, relativas aos grupos de métricas divididos no componente MAPE-K. 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.



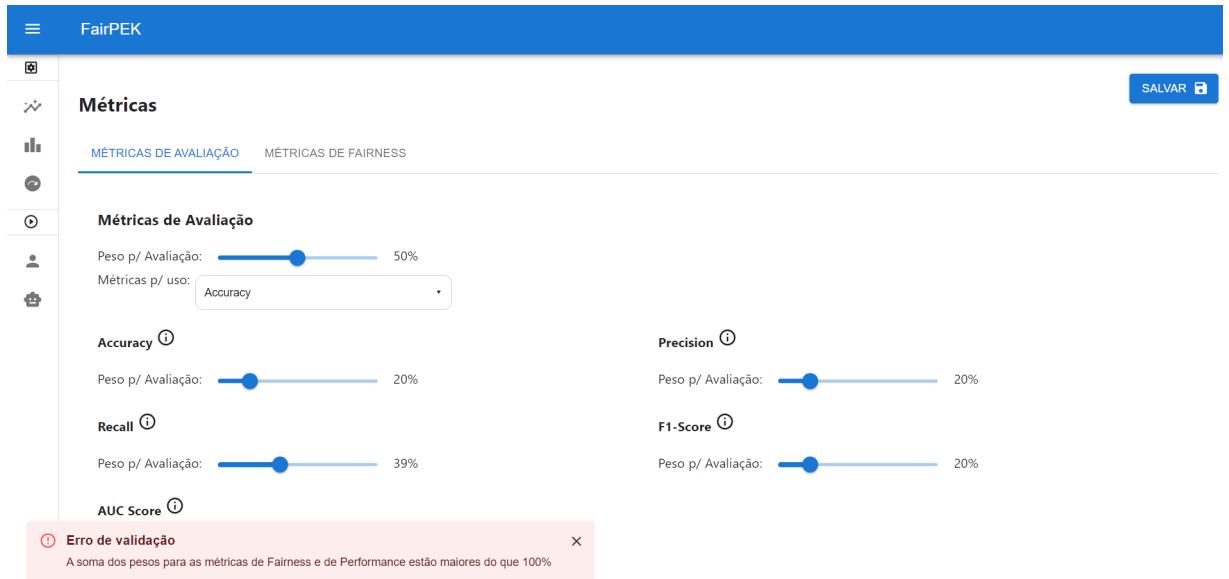
(a) Configuração para Métricas de Performance.



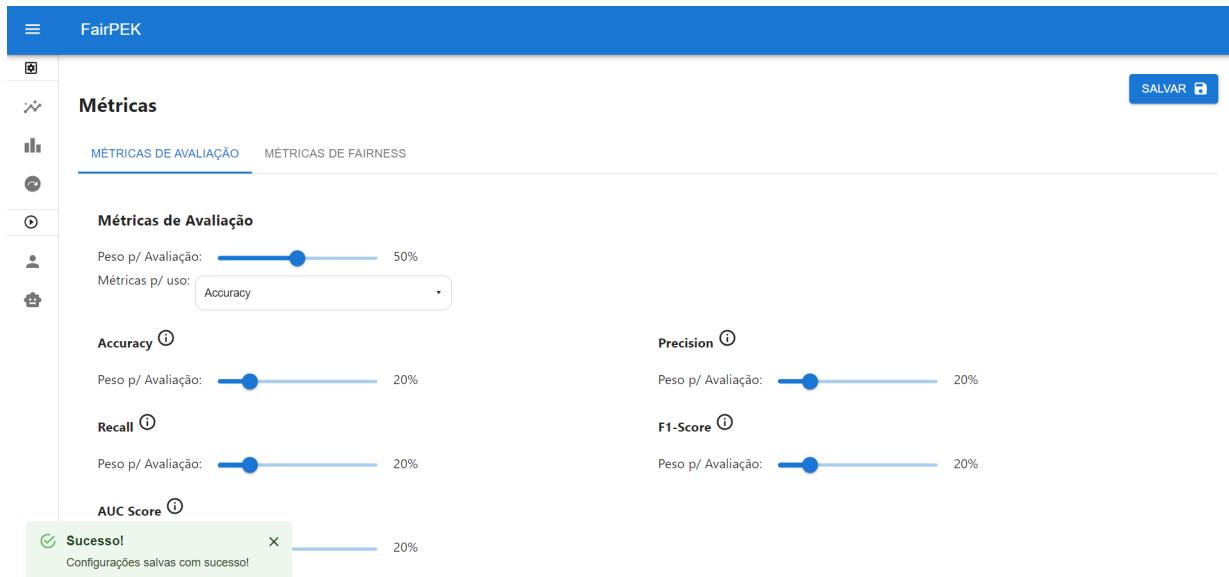
(b) Configuração para Métricas de Fairness.

Figura 3.9: Configuração das métricas para etapa de análise do workflow autônomo.

A alteração do campo de métricas para uso implicará na presença de mais ou menos métricas para alterar os pesos para o cálculo da pontuação para o grupo de métricas. Uma vez que eu posso ter mais de duas métricas nesse caso, implementar a soma de todas as métricas automaticamente para 100% seria uma tarefa com maior complexidade, e por isso foi substituída por uma validação, conforme ilustrado na Figura 3.10. 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 3.10: Cenários possíveis na configuração das métricas.

Configurações para Planejamento

Ao clicar a opção do menu "Planejamento", é exibida a tela ilustrada na Figura 3.11. Ela possui quatro opções, relativas às estratégias desenvolvidas na parte de análise do componente MAPE-K. 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.

Planejamento

Selecionar estratégias de planejamento

Escolher o modelo com a melhor pontuação

Escolher o modelo de acordo com os dados mais recentes encontrados em execuções prévias

Restringir conjunto de algoritmos escolhidos

Sem método
Logistic Regression
Sem método

Sem método
Logistic Regression
Equalized Odds

Sem método
Logistic Regression
Calibrated Equalized Odds

Restringir limiar de pontuação

500 ————— 881

OBS. Selecionar ou não selecionar determinadas estratégias podem causar impactos caso certas etapas de análise não forem selecionadas

(a) Opções para configurar etapa de planejamento.

Planejamento

Selecionar estratégias de planejamento

Escolher o modelo com a melhor pontuação

Escolher o modelo de acordo com os dados mais recentes encontrados em execuções prévias

Restringir conjunto de algoritmos escolhidos

Sem método
Logistic Regression
Sem método

Sem método
Logistic Regression
Equalized Odds

Sem método
Logistic Regression
Calibrated Equalized Odds

Restringir limiar de pontuação

Sucesso!
Configurações salvadas com sucesso!

inadas estratégias podem causar impactos caso certas etapas de análise não forem selecionadas

(b) Indicação de sucesso da operação.

Figura 3.11: Configuração da etapa de planejamento para o workflow autônomo.

Uma das quatro estratégias está desabilitada pois foi desenvolvida como a estratégia padrão usada pelo componente MAPE-K. Se a estratégia de restrição de algoritmos for selecionada, é exibido um submenu contendo as combinações de algoritmos que podem ser selecionadas 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.

Execução simples do Workflow

Ao clicar a opção do menu "Pipeline Manual", é exibida a tela ilustrada na Figura 3.12. 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 *Workflow* 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.

The screenshot shows the FairPEK Pipeline Manual interface. At the top, there are three tabs: 1. Parametrização, 2. Execução, and 3. Resultado. The 'Parametrização' tab is selected. Below the tabs, there are several configuration fields:

- Conjunto de dados:** German Credit Dataset (dropdown menu)
- Atributo Protegido:** Idade (-25 anos/+25 anos) (dropdown menu)
- Redução de viés será aplicada em qual etapa do Pipeline?** (radio buttons)
 - Pré-Processamento/Dados (selected)
 - Processamento/Treinamento
 - Pós-Processamento/Avaliação
 - Nenhum (Executar treinamento convencional)
- Algoritmo de treinamento:** Logistic Regression (dropdown menu)
- Algoritmo de redução de viés:** Reweighting (dropdown menu)

At the bottom right of the interface is a blue button labeled "EXECUTAR ▶".

(a) Opções para executar o workflow manualmente.

The screenshot shows the FairPEK Pipeline Manual interface during execution. The 'Parametrização' tab is selected. The status bar at the bottom indicates "Executando Pipeline...". A green notification bar at the bottom left says "Sucesso!" with a checkmark icon. The text below it reads "Execução será realizada em alguns segundos".

(b) Workflow em execução.

Figura 3.12: Execução simples e manual do Workflow.

Após a execução ser concluída, a indicação de etapa é atualizada para a etapa de resultados, conforme ilustração nas Figuras 3.13 e 3.14. Nela, os parâmetros gravados são organizados em 4 grupos: Execução, relativos aos parâmetros utilizados e estatísticas da execução, Métricas de Performance, relativas aos resultados das métricas de Performance,

Métricas de Fairness, relativas aos resultados das métricas de Fairness, e Pontuação, relativas ao cálculo realizado com as configurações utilizadas na parte de métricas.

The screenshot shows the FairPEK Pipeline Manual interface. The top navigation bar has tabs for 'Parametrização', 'Execução', and 'Resultado'. The 'Resultado' tab is active. Below the tabs, there are two main sections: 'Execução' and 'Parâmetros utilizados e estatísticas da execução'. The 'Parâmetros' section contains the following data:

Checksum do conjunto de dados	fcde189dd32a1631a474e1087a7f4835525194741309493e0475bce7cb3451d0a499c70007db6c4bf5e33cb4ecb2da9feb859b57cfa21255573103d92e6b497d
Conjunto de dados utilizado	Datasets.GERMAN_CREDIT
Atributo protegido	Preprocessors.AGE
Algoritmo de redução de viés no dado	UnbiasDataAlgorithms.NOTHING
Algoritmo de treinamento (redução de viés: Sim)	UnbiasInProcAlgorithms.META_FAIR_CLASSIFIER
Algoritmo de redução de viés no pós-processamento	UnbiasPostProcAlgorithms.NOTHING
Data de inicio da execução	02/05/2022 00:42:02.878556
Data de fim da execução	02/05/2022 00:42:04.598311
Tempo de Execução	1719 ms

Below this, there are two expandable sections: 'Métricas de Performance' and 'Métricas de Fairness'.

(a) Exibição dos parâmetros no resultado do workflow.

The screenshot shows the FairPEK Pipeline Manual interface. The top navigation bar has tabs for 'Parametrização', 'Execução', and 'Resultado'. The 'Resultado' tab is active. Below the tabs, there are three main sections: 'Execução', 'Parâmetros utilizados e estatísticas da execução', 'Métricas de Performance', 'Métricas de Fairness', and 'Pontuação'.

The 'Pontuação' section contains the following data:

Pontuação das métricas de performance	782
Pontuação das métricas de fairness	963
Pontuação geral	872

(b) Exibição das pontuações no resultado do workflow.

Figura 3.13: Informações do resultado do workflow.

The screenshot shows the FairPEK Pipeline Manual interface. At the top, there are three tabs: 'Parametrização' (Parametrization), 'Execução' (Execution), and 'Resultado' (Result). The 'Resultado' tab is selected and shows a summary of the execution. Below this, there are two expandable sections: 'Métricas de Performance' (Performance Metrics) and 'Métricas de Fairness' (Fairness Metrics).

Métricas de Performance:

Acurácia	0.75
Precisão	0.7660818713450293
Recall	0.9290780141843972
F1-Score	0.8397435897435898
AUC (Area Under the ROC Curve)	0.625555956244741

Métricas de Fairness:

Statistical Parity Difference	-0.125
Disparate Impact	0.8571428571428571
Average Odds Difference	0.14591291061879297
Equal Opportunity Difference	-0.07754010695187163
Theil Index	0.10133272912395277

(a) Exibição das métricas de performance no resultado do workflow.

The screenshot shows the FairPEK Pipeline Manual interface, similar to the previous one but with different metric data. The 'Resultado' tab is selected, showing the execution summary. Below it, the 'Métricas de Fairness' section is expanded.

Métricas de Fairness:

Statistical Parity Difference	-0.125
Disparate Impact	0.8571428571428571
Average Odds Difference	0.14591291061879297
Equal Opportunity Difference	-0.07754010695187163
Theil Index	0.10133272912395277

(b) Exibição das métricas de fairness no resultado do workflow.

Figura 3.14: Métricas do resultado do workflow.

Execução autônoma do Workflow

Ao clicar a opção do menu "Pipeline Autônomo", é exibida a tela ilustrada na Figura 3.15. 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 workflow de forma autônoma.

(b) Análise do componente MAPE-K em execução.

Figura 3.15: Execução autônoma do Workflow.

Após a etapa de análise ser concluída, a indicação de etapa é atualizada para a etapa de opções, conforme ilustração na Figura 3.16. Nela, os parâmetros sugeridos são organizados nos mesmos grupos presentes nas Figuras 3.13 e 3.14 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 componente MAPE-K.

Pipeline Autônomo

Escolha o pipeline a ser executado

Pipeline 1

Execução	Parâmetros utilizados e estatísticas da execução
Métricas de Performance	Métricas relacionadas a performance do modelo de execução mais recente
Métricas de Fairness	Métricas relacionadas a justiça do modelo de execução mais recente
Pontuação	Pontuação dada de acordo com os pesos e métricas utilizadas com base em todas as execuções

Pipeline 2

Execução	Parâmetros utilizados e estatísticas da execução
Métricas de Performance	Métricas relacionadas a performance do modelo de execução mais recente
Métricas de Fairness	Métricas relacionadas a justiça do modelo de execução mais recente
Pontuação	Pontuação dada de acordo com os pesos e métricas utilizadas com base em todas as execuções

(a) Opções para seleção de workflow para execução.

Pipeline 1

Execução	Parâmetros utilizados e estatísticas da execução
Checksum do conjunto de dados	fcde189dd32a1f631a474e1087a7f4835525194741309493e0475bce7cb3451d0a499c70007db6c4bf5e33cb4ecb2da9feb859b57cfa21255573103d92e6b497d
Conjunto de dados utilizado	Datasets.GERMAN_CREDIT
Atributo protegido	Preprocessors.AGE
Algoritmo de redução de viés no dado	UnbiasDataAlgorithms.LEARNING_FAIR REPRESENTATIONS
Algoritmo de treinamento (redução de viés: Não)	Algorithms.RANDOM_FOREST
Algoritmo de redução de viés no pós-processamento	UnbiasPostProcAlgorithms.NOTHING

Métricas de Performance	Métricas relacionadas a performance do modelo de execução mais recente
Métricas de Fairness	Métricas relacionadas a justiça do modelo de execução mais recente
Pontuação	Pontuação dada de acordo com os pesos e métricas utilizadas com base em todas as execuções

(b) Parâmetros a serem utilizados para execução.

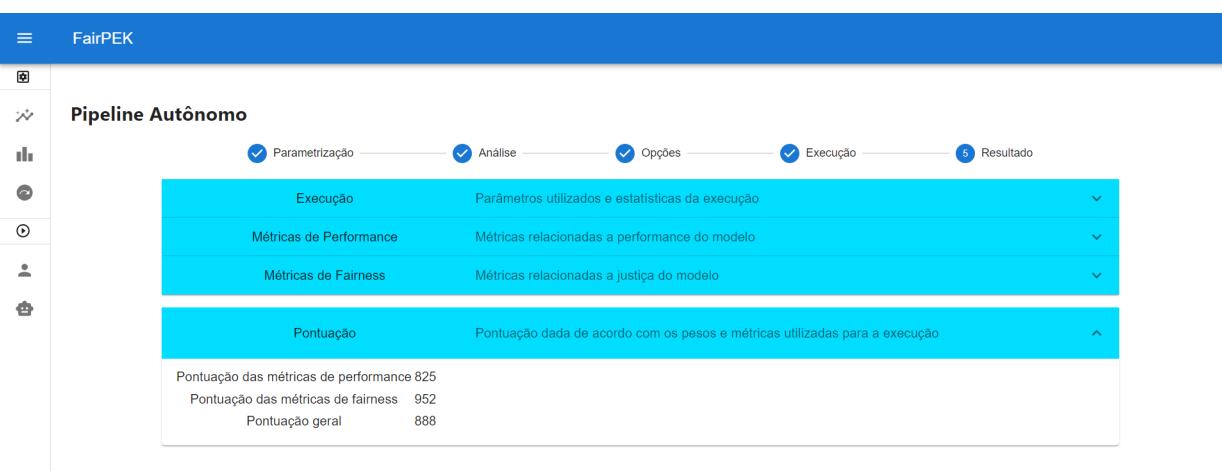
Figura 3.16: Seleção do workflow após análise.

Ao clicar novamente no botão "Executar" localizado no canto superior direito, é feita uma requisição para executar o *Workflow* 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 3.17. Nela, os parâmetros sugeridos são organizados nos mesmos grupos presentes na Figura 3.16, mas desta vez refletem as métricas e pontuação da execução realizada pelo *Workflow*.

,



(a) Workflow em execução.



(b) Resultados da execução realizada.

Figura 3.17: Execução do workflow após seleção.

Capítulo 4

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

Neste Estudo de Caso, o maior foco foi colocado em testar e verificar como o MAPE-K se comporta em um cenário com diversas execuções prévias de *Workflows*. Como as execuções geraram uma base de conhecimento, o componente MAPE-K pode executar uma análise através destes dados e determinar um plano indicando a gama de algoritmos que utilizará para obter melhores resultados. Foram notados resultados em outras escolhas realizadas para a construção desta aplicação, mas que serão melhor discutidos nos estudos seguintes.

4.1 Contexto e limitações

Foram realizados testes com o seguinte conjunto de dados e dado o seguinte objetivo:

- **Objetivo:** Obter classificação de crédito (boa ou ruim), através de uma série de *features*
- **Conjunto de dados:** German Credit Dataset, presente em [11].
- **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 posteriormente e de forma mais fácil no *Workflow*.
- **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 de um *workflow* de ML para obtenção de dados iniciais na base de conhecimento e discussão de um *workflow* de ML utilizando MAPE-K como facilitador da escolha de modelo.
- **Medição:** Serão medidas as pontuações de cada agrupamento de métricas (Performance e Fairness), e tais métricas também serão comparadas com o valor de cada uma para verificar seu impacto na pontuação.
- **Obtenção dos dados:** A execução do *Pipeline* utilizando o componente MAPE-K foi realizada com 3 pesagens diferentes na pontuação geral:
 - 50% para métricas de Performance e 50% para métricas de Fairness, para uma configuração equilibrada.
 - 75% para métricas de Performance e 25% para métricas de Fairness, para uma configuração que prioriza a performance em detrimento da justiça.
 - 25% para métricas de Performance e 75% para métricas de Fairness, para uma configuração que prioriza a justiça em detrimento da performance.

Em todas as execuções são utilizadas as métricas Acurácia, Precisão, *Recall*, *F1-Score* e AUC como métricas de Performance e as métricas *Statistical Parity Difference*, *Equal Opportunity Difference*, *Average Odds Difference*, *Disparate Impact* e *Theil Index* como métricas de Fairness, todas com pesagens iguais em seu respectivo agrupamento.

- **Pré-condição 1:** Houveram *workflows* que já foram pré-executados e já formaram uma base de conhecimento. Para o experimento, foi considerado ao menos 1 execução para cada combinação de conjunto de dados, atributo protegido e algoritmos utilizados.
- **Pré-condição 2:** A base de conhecimento será capaz de explicar a imparcialidade/justiça de um modelo, mas não será capaz de explicar a influência de cada *feature* aplicada no modelo.
- **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 [50] [41] [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, consequentemente, a situação de vida de uma pessoa.
- **Restrição 1:** O algoritmo de retirada de viés é feito em apenas uma parte das etapas (Pré-processamento/Processamento/Pós-processamento). Isto foi decidido pois não foram verificadas referências onde a aplicação desses algoritmos em duas ou em todas as três etapas impacta no desempenho.

- **Restrição 2:** Por não conseguirem rodar com sucesso, foram retirados os *Workflows* que rodaram os seguintes algoritmos: Pré-processamento Otimizado.
- **Restrição 3:** *Workflows* também foram retirados por apresentarem métricas com valores máximos, para evitar análises caso exista alguma falha não detectada na implementação. Como exemplo, os que possuíam Classificação baseada em Rejeição de Opções.
- **Restrição 4:** Para evitar *workflows* com conjuntos de algoritmos ruins e também pelo mesmo motivo da restrição anterior, o intervalo de pontuação para análise foi limitado de 500 a 950.

4.2 Resultados e Discussões

Os resultados baseados nas pré-condições e restrições já comentadas na seção anterior estão presentes abaixo nas Tabelas 4.1, 4.2 e 4.3:

Tabela 4.1: Melhores opções escolhidas pelo modelo MAPE-K

Todos os métodos - 50% Performance/50% Fairness

Atributo protegido	Pré-processamento	Workflow		Pontuação		
		Treinamento	Pós-processamento	Performance	Fairness	Geral
Idade	Nenhum	Régressão 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 4.2: Melhores opções escolhidas pelo modelo MAPE-K

Todos os métodos - 75% Performance/25% Fairness

Atributo protegido	Pré-processamento	Workflow		Pontuação		
		Treinamento	Pós-processamento	Performance	Fairness	Geral
Idade	Nenhum	Régressão 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 4.3: Melhores opções escolhidas pelo modelo MAPE-K

Todos os métodos - 25% Performance/75% Fairness

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

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

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

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

As pré-condições, restrições e pesagens nas pontuações usadas anteriormente foram mantidas e seus resultados estão presentes abaixo nas Tabelas 4.4, 4.5, 4.6, 4.7, 4.8, 4.9, 4.10, 4.11, 4.12, 4.13, 4.14 e 4.15:

Tabela 4.4: Melhores opções escolhidas pelo modelo MAPE-K
Apenas com redução de viés no dado - 50% Performance/50% Fairness

Atributo protegido	Workflow			Pontuação		
	Pré-processamento	Treinamento	Pós-processamento	Performance	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 4.5: Melhores opções escolhidas pelo modelo MAPE-K
Apenas com redução de viés no dado - 75% Performance/25% Fairness

Atributo protegido	Workflow			Pontuação		
	Pré-processamento	Treinamento	Pós-processamento	Performance	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 4.6: Melhores opções escolhidas pelo modelo MAPE-K
Apenas com redução de viés no dado - 25% Performance/75% Fairness

Atributo protegido	Workflow			Pontuação		
	Pré-processamento	Treinamento	Pós-processamento	Performance	Fairness	Geral
Idade	Disparate Impact Remover	Support Vector Machines	Nenhum	747	989	928
Nacionalidade	Disparate Impact Remover	Support Vector Machines	Nenhum	747	989	928
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

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

Tabela 4.7: Melhores opções escolhidas pelo modelo MAPE-K
Apenas com redução de viés no treinamento - 50% Performance/50% Fairness

Atributo protegido	Workflow			Pontuação		
	Pré-processamento	Treinamento	Pós-processamento	Performance	Fairness	Geral
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 4.8: Melhores opções escolhidas pelo modelo MAPE-K
Apenas com redução de viés no treinamento - 75% Performance/25% Fairness

Atributo protegido	Workflow			Pontuação		
	Pré-processamento	Treinamento	Pós-processamento	Performance	Fairness	Geral
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 4.9: Melhores opções escolhidas pelo modelo MAPE-K
Apenas com redução de viés no treinamento - 25% Performance/75% Fairness

Atributo protegido	Pré-processamento	Workflow		Pontuação		
		Treinamento	Pós-processamento	Performance	Fairness	Geral
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

Nos *workflows* utilizando apenas algoritmos com redução de viés no treinamento, percebe-se uma variedade maior nos algoritmos, até porque não há usos de redução de viés em um pré ou um pós-processamento, com destaque para o *Adversarial Debiasing* que foi bem avaliado pela pontuação alta em *Fairness*. Nestes *workflows*, as 2 observações percebidas nos *workflows* utilizando apenas algoritmos com redução de viés no dado são reforçadas por uma maior variedade de pontuações e pelo algoritmo *Exponentiated Gradient Reduction* com 2 exemplos diferentes na Tabela 4.8, onde o uso da Nacionalidade como atributo protegido possui pontuações de performance e *Fairness* piores que a Idade e concluindo que o processamento utilizado no atributo protegido pode afetar todas as métricas.

Tabela 4.10: Melhores opções escolhidas pelo modelo MAPE-K
Apenas com redução de viés no resultado - 50% Performance/50% Fairness

Atributo protegido	Pré-processamento	Workflow		Pontuação		
		Treinamento	Pós-processamento	Performance	Fairness	Geral
Idade	Nenhum	Régressão 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 4.11: Melhores opções escolhidas pelo modelo MAPE-K
Apenas com redução de viés no resultado - 75% Performance/25% Fairness

Atributo protegido	Pré-processamento	Workflow		Pontuação		
		Treinamento	Pós-processamento	Performance	Fairness	Geral
Idade	Nenhum	Régressão 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 4.12: Melhores opções escolhidas pelo modelo MAPE-K

Apenas com redução de viés no resultado - 25% Performance/75% Fairness

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

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

Tabela 4.13: Melhores opções escolhidas pelo modelo MAPE-K

Apenas sem redução de viés - 50% Performance/50% Fairness

Atributo protegido	Pré-processamento	Treinamento	Pós-processamento	Pontuação		
				Performance	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	Rregressão Logística	Nenhum	782	865	824
Nacionalidade	Nenhum	Gradient Boosting	Nenhum	817	784	800

Tabela 4.14: Melhores opções escolhidas pelo modelo MAPE-K

Apenas sem redução de viés - 75% Performance/25% Fairness

Atributo protegido	Pré-processamento	Treinamento	Pós-processamento	Pontuação		
				Performance	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 4.15: Melhores opções escolhidas pelo modelo MAPE-K
Apenas sem redução de viés - 25% Performance/75% Fairness

Atributo protegido	Pré-processamento	Workflow		Pontuação		
		Treinamento	Pós-processamento	Performance	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 os *workflows* sem algoritmos com redução de viés, é curioso notar que, ao comparar com *workflows* equivalentes mas com algoritmos usando redução de viés no dado, é possível notar que a hipótese principal se confirma em sua grande maioria: As pontuações em Performance são ligeiramente maiores e as pontuações em *Fairness* são ligeiramente menores. É possível notar uma exceção no *workflow* envolvendo o algoritmo *Random Forest*, mas nos outros casos a hipótese é verificada com sucesso. Ao comparar com *workflows* usando algoritmos com redução de viés no treinamento tal hipótese também se confirma, entretanto o uso de *Support Vector Machines* parece ser uma exceção a regra, implicando que o uso do algoritmo possibilita modelos mais justos para o conjunto de dados utilizado. Ao comparar com *workflows* usando algoritmos com redução de viés no resultado, a hipótese não se confirma devido a observação da grande melhora por parte das métricas de performance nos *workflows* usando algoritmos com redução de viés no resultado, mas ao verificar a pontuação em *Fairness* é possível notar uma ligeira melhora. Há a exceção de *workflows* envolvendo *Support Vector Machines* que são melhores nos *workflows* sem 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 4.16 e 4.17 exibidas abaixo.

Tabela 4.16: Métricas de performance das execuções de Workflows

Atributo protegido	Pré-processamento	Workflow	Treinamento	Pós-processamento	Métricas					
					Acurácia	Precisão	Recall	F1-Score	AUC	
Nacionalidade	Nenhum	Support Vector Machines	Nenhum	Mínimo Máximo Média	0,715 0,715 0,715	0,7143 0,7143 0,7143	0,9929 0,9929 0,9929	0,8309 0,8309 0,8309	0,5219 0,5219 0,5219	0,5219
Idade	Nenhum	Support Vector Machines	Nenhum	Mínimo Máximo Média	0,715 0,715 0,715	0,7143 0,7143 0,7143	0,9929 0,9929 0,9929	0,8309 0,8309 0,8309	0,5219 0,5219 0,5219	0,5219
Nacionalidade	Nenhum	Random Forest	Nenhum	Mínimo Máximo Média	0,76 0,79 0,774	0,7784 0,8037 0,7933	0,9007 0,9291 0,9192	0,8442 0,8618 0,8515	0,6474 0,6933 0,6731	0,6474
Nacionalidade	Nenhum	Régressão Logística	Nenhum	Mínimo Máximo Média	0,75 0,75 0,75	0,7661 0,7661 0,7661	0,9291 0,9291 0,9291	0,8307 0,8307 0,8307	0,6256 0,6256 0,6256	0,6256
Nacionalidade	Nenhum	Gradient Boosting	Nenhum	Mínimo Máximo Média	0,79 0,79 0,79	0,8194 0,8194 0,8194	0,9007 0,9007 0,9007	0,8581 0,8581 0,8581	0,7131 0,7131 0,7131	0,7131
Idade	Nenhum	Gradient Boosting	Nenhum	Mínimo Máximo Média	0,79 0,79 0,79	0,8235 0,8235 0,8235	0,8936 0,8936 0,8936	0,8571 0,8571 0,8571	0,718 0,718 0,718	0,718
Idade	Nenhum	Régressão Logística	Nenhum	Mínimo Máximo Média	0,75 0,75 0,75	0,7661 0,7661 0,7661	0,9291 0,9291 0,9291	0,8307 0,8307 0,8307	0,6256 0,6256 0,6256	0,6256
Idade	Reweighting	Gradient Boosting	Nenhum	Mínimo Máximo Média	0,775 0,775 0,775	0,8 0,8 0,8	0,9078 0,9078 0,9078	0,8505 0,8505 0,8505	0,6827 0,6827 0,6827	0,6827
Idade	Learning Fair Representations	Gradient Boosting	Nenhum	Mínimo Máximo Média	0,775 0,775 0,775	0,8 0,8 0,8	0,9078 0,9078 0,9078	0,8505 0,8505 0,8505	0,6827 0,6827 0,6827	0,6827
Idade	Disparate Impact Remover	Support Vector Machines	Nenhum	Mínimo Máximo Média	0,705 0,705 0,705	0,705 0,705 0,705	1 1 1	0,827 0,827 0,827	0,5 0,5 0,5	0,5
Nacionalidade	Disparate Impact Remover	Support Vector Machines	Nenhum	Mínimo Máximo Média	0,705 0,705 0,705	0,705 0,705 0,705	1 1 1	0,827 0,827 0,827	0,5 0,5 0,5	0,5
Nacionalidade	Learning Fair Representations	Support Vector Machines	Nenhum	Mínimo Máximo Média	0,715 0,715 0,715	0,7143 0,7143 0,7143	0,9929 0,9929 0,9929	0,8309 0,8309 0,8309	0,5219 0,5219 0,5219	0,5219
Nacionalidade	Learning Fair Representations	Gradient Boosting	Nenhum	Mínimo Máximo Média	0,785 0,785 0,785	0,8063 0,8063 0,8063	0,9149 0,9149 0,9149	0,8571 0,8571 0,8571	0,6947 0,6947 0,6947	0,6947
Nacionalidade	Reweighting	Gradient Boosting	Nenhum	Mínimo Máximo Média	0,785 0,785 0,785	0,8063 0,8063 0,8063	0,9149 0,9149 0,9149	0,8571 0,8571 0,8571	0,6947 0,6947 0,6947	0,6947
Nacionalidade	Learning Fair Representations	Random Forest	Nenhum	Mínimo Máximo Média	0,76 0,79 0,772	0,7831 0,8113 0,7949	0,8936 0,922 0,921	0,84 0,86 0,8494	0,6559 0,7032 0,6746	0,6559
Nacionalidade	Reweighting	Support Vector Machines	Nenhum	Mínimo Máximo Média	0,715 0,715 0,715	0,7143 0,7143 0,7143	0,9929 0,9929 0,9929	0,8309 0,8309 0,8309	0,5219 0,6219 0,5219	0,5219
Idade	Learning Fair Representations	Support Vector Machines	Nenhum	Mínimo Máximo Média	0,715 0,715 0,715	0,7143 0,7143 0,7143	0,9929 0,9929 0,9929	0,8309 0,8309 0,8309	0,5219 0,5219 0,5219	0,5219
Idade	Nenhum	Adversarial Debiasing	Nenhum	Mínimo Máximo Média	0,295 0,705 0,6317	0 0,7097 0,5888	0 1 0,8168	0 0,827 0,6842	0 0,5105 0,503	0,5
Nacionalidade	Nenhum	Grid Search Reduction	Nenhum	Mínimo Máximo Média	0,75 0,78 0,765	0,7791 0,7939 0,7857	0,9007 0,9362 0,9167	0,8355 0,8562 0,8461	0,6453 0,6764 0,6596	0,6453
Idade	Nenhum	Meta Fair Classifier	Nenhum	Mínimo Máximo Média	0,73 0,752 0,742	0,7278 0,9858 0,7466	0,9291 0,9858 0,9617	0,8374 0,8474 0,8402	0,5522 0,6256 0,5893	0,5522
Idade	Nenhum	Exponentiated Gradient Reduction	Nenhum	Mínimo Máximo Média	0,785 0,785 0,785	0,8063 0,8063 0,8063	0,9149 0,9149 0,9149	0,8571 0,8571 0,8571	0,6947 0,6947 0,6947	0,6947
Nacionalidade	Nenhum	Rich Subgroup Fairness	Nenhum	Mínimo Máximo Média	0,76 0,76 0,76	0,808 0,808 0,808	0,8653 0,8653 0,8653	0,8356 0,8356 0,8356	0,6869 0,6869 0,6869	0,6869
Nacionalidade	Nenhum	Exponentiated Gradient Reduction	Nenhum	Mínimo Máximo Média	0,775 0,785 0,7788	0,8038 0,8101 0,8057	0,9007 0,9078 0,9043	0,8495 0,8562 0,8521	0,6876 0,6997 0,6915	0,6876
Nacionalidade	Nenhum	Prejudice Remover	Nenhum	Mínimo Máximo Média	0,735 0,735 0,735	0,7683 0,7683 0,7683	0,8936 0,8936 0,8936	0,8262 0,8262 0,8262	0,6248 0,6248 0,6248	0,6248
Idade	Nenhum	Régressão Logística	Equalized Odds	Mínimo Máximo Média	0,965 0,965 0,965	0,9589 0,9589 0,9589	0,9929 0,9929 0,9929	0,9756 0,9756 0,9756	0,9456 0,9456 0,9456	0,9456
Nacionalidade	Nenhum	Random Forest	Calibrated Equalized Odds	Mínimo Máximo Média	0,82 0,99 0,8925	0,7966 0,9097 0,87	1 1 1	0,8868 0,9527 0,9299	0,6949 0,8814 0,8178	0,6949
Nacionalidade	Nenhum	Gradient Boosting	Calibrated Equalized Odds	Mínimo Máximo Média	0,84 0,875 0,855	0,8150 0,8194 0,8296	1 1 1	0,8981 0,9186 0,9068	0,7288 0,7881 0,7542	0,7288
Idade	Nenhum	Gradient Boosting	Equalized Odds	Mínimo Máximo Média	0,909 0,915 0,9083	0,9769 0,9919 0,9869	0,8723 0,9007 0,8818	0,9283 0,9373 0,9313	0,9249 0,9277 0,9268	0,9249
Idade	Nenhum	Random Forest	Equalized Odds	Mínimo Máximo Média	0,8 0,915 0,8733	0,9697 0,9903 0,9789	1 1 1	0,8361 0,9377 0,9011	0,8532 0,9200 0,897	0,8532

CAPÍTULO 4. ESTUDO DE CASO 1: CLASSIFICAÇÃO DE CRÉDITO (GERMAN CREDIT DATASET) 79

Tabela 4.17: Métricas de Fairness das execuções de Workflows

Atributo protegido	Pré-processamento	Workflow			Statistical Parity Difference	Equal Opportunity Difference	Average Odds Difference	Disparate Impact	Theil Index
		Treinamento	Pós-processamento	Métricas					
Nacionalidade	Nenhum	Support Vector Machines	Nenhum		Mínimo -0,0209 Máximo -0,0209 Média -0,0209	Mínimo -0,0075 Máximo -0,0075 Média -0,0075	Mínimo -0,290 Máximo -0,290 Média -0,290	Mínimo 0,791 Máximo 0,791 Média 0,791	0,0615 0,0615 0,0615
Idade	Nenhum	Support Vector Machines	Nenhum		Mínimo 0,0238 Máximo 0,0238 Média 0,0238	Mínimo 0,0084 Máximo 0,0084 Média 0,0084	Mínimo 0,0348 Máximo 0,0348 Média 0,0348	Mínimo 1,024 Máximo 1,024 Média 1,024	0,0615 0,0615 0,0615
Nacionalidade	Nenhum	Random Forest	Nenhum		Mínimo -0,09831 Máximo 0,03893 Média -0,0520	Mínimo 0,0273 Máximo 0,1899 Média 0,0733	Mínimo -0,2191 Máximo -0,1378 Média -0,1806	Mínimo 0,8894 Máximo 1,0301 Média 0,9428	0,0955 0,1173 0,1045
Nacionalidade	Nenhum	Regressão Logística	Nenhum		Mínimo -0,1518 Máximo -0,1518 Média -0,1518	Mínimo -0,0752 Máximo -0,0752 Média -0,0752	Mínimo -0,2014 Máximo -0,2014 Média -0,2014	Mínimo 0,8482 Máximo 0,8482 Média 0,8482	0,1013 0,1013 0,1013
Nacionalidade	Nenhum	Gradient Boosting	Nenhum		Mínimo 0,1134 Máximo 0,1134 Média 0,1134	Mínimo 0,2923 Máximo 0,2923 Média 0,2923	Mínimo -0,1211 Máximo -0,1211 Média -0,1211	Mínimo 1,1702 Máximo 1,1702 Média 1,1702	0,1137 0,1137 0,1137
Idade	Nenhum	Gradient Boosting	Nenhum		Mínimo -0,2111 Máximo -0,2111 Média -0,2111	Mínimo -0,1971 Máximo -0,1971 Média -0,1971	Mínimo 0,2337 Máximo 0,2337 Média 0,2337	Mínimo 0,7 Máximo 0,7 Média 0,7	0,1183 0,1183 0,1183
Idade	Nenhum	Regressão Logística	Nenhum		Mínimo -0,1518 Máximo -0,1518 Média -0,1518	Mínimo -0,0752 Máximo -0,0752 Média -0,0752	Mínimo -0,2014 Máximo -0,2014 Média -0,2014	Mínimo 0,8482 Máximo 0,8482 Média 0,8482	0,1013 0,1013 0,1013
Idade	Reweighting	Gradient Boosting	Nenhum		Mínimo -0,0595 Máximo 0,0595 Média 0,0595	Mínimo -0,0523 Máximo -0,0523 Média -0,0523	Mínimo -0,0517 Máximo -0,0517 Média -0,0517	Mínimo 0,9265 Máximo 0,9265 Média 0,9265	0,1118 0,1118 0,1118
Idade	Learning Fair Representations	Gradient Boosting	Nenhum		Mínimo -0,0595 Máximo 0,0595 Média 0,0595	Mínimo -0,0523 Máximo -0,0523 Média -0,0523	Mínimo -0,0517 Máximo -0,0517 Média -0,0517	Mínimo 0,9265 Máximo 0,9265 Média 0,9265	0,1118 0,1118 0,1118
Idade	Disparate Impact Remover	Support Vector Machines	Nenhum		Mínimo 0 Máximo 0 Média 0	Mínimo 0 Máximo 0 Média 0	Mínimo 0 Máximo 1 Média 1	Mínimo 0,573 Máximo 0,573 Média 0,573	0,573 0,573 0,573
Nacionalidade	Disparate Impact Remover	Support Vector Machines	Nenhum		Mínimo 0 Máximo 0 Média 0	Mínimo 0 Máximo 0 Média 0	Mínimo 0 Máximo 1 Média 1	Mínimo 0,573 Máximo 0,573 Média 0,573	0,573 0,573 0,573
Nacionalidade	Learning Fair Representations	Support Vector Machines	Nenhum		Mínimo -0,0209 Máximo 0,0209 Média -0,0209	Mínimo -0,0075 Máximo -0,0075 Média -0,0075	Mínimo -0,296 Máximo -0,296 Média -0,296	Mínimo 0,791 Máximo 0,791 Média 0,791	0,0615 0,0615 0,0615
Nacionalidade	Learning Fair Representations	Gradient Boosting	Nenhum		Mínimo -0,0931 Máximo 0,0931 Média 0,0931	Mínimo 0,0423 Máximo 0,0423 Média 0,0423	Mínimo -0,202 Máximo -0,202 Média -0,202	Mínimo 0,8953 Máximo 0,8953 Média 0,8953	0,1055 0,1055 0,1055
Nacionalidade	Reweighting	Gradient Boosting	Nenhum		Mínimo -0,0931 Máximo 0,0931 Média 0,0931	Mínimo 0,0423 Máximo 0,0423 Média 0,0423	Mínimo -0,202 Máximo -0,202 Média -0,202	Mínimo 0,8953 Máximo 0,8953 Média 0,8953	0,1055 0,1055 0,1055
Nacionalidade	Learning Fair Representations	Random Forest	Nenhum		Mínimo -0,0983 Máximo 0,0285 Média -0,0604	Mínimo 0,0197 Máximo 0,1673 Média 0,0658	Mínimo -0,2280 Máximo -0,1405 Média -0,1895	Mínimo 0,8891 Máximo 1,0367 Média 0,933	0,1025 0,1237 0,1095
Nacionalidade	Reweighting	Support Vector Machines	Nenhum		Mínimo -0,0209 Máximo 0,0209 Média -0,0209	Mínimo -0,0075 Máximo -0,0075 Média -0,0075	Mínimo -0,296 Máximo -0,296 Média -0,296	Mínimo 0,791 Máximo 0,791 Média 0,791	0,0615 0,0615 0,0615
Idade	Learning Fair Representations	Support Vector Machines	Nenhum		Mínimo 0,0238 Máximo 0,0238 Média 0,0238	Mínimo 0,0084 Máximo 0,0084 Média 0,0084	Mínimo 0,0348 Máximo 0,0348 Média 0,0348	Mínimo 1,024 Máximo 1,024 Média 1,024	0,0615 0,0615 0,0615
Idade	Nenhum	Adversarial Debiasing	Nenhum		Mínimo 0 Máximo 0,0104 Média 0,0932	Mínimo 0 Máximo 0,042 Média 0,0106	Mínimo -0,0086 Máximo 0,0017 Média 0,0012	Mínimo 1 Máximo 1,2208 Média N/A Média 0,2629	0,573 1,2208 0,2629
Nacionalidade	Nenhum	Grid Search Reduction	Nenhum		Mínimo -0,0826 Máximo -0,0512 Média -0,0669	Mínimo 0,0273 Máximo 0,0648 Média 0,0412	Mínimo -0,1933 Máximo -0,1639 Média -0,1827	Mínimo 0,9051 Máximo 0,9424 Média 0,9218	0,0932 0,1204 0,1076
Idade	Nenhum	Meta Fair Classifier	Nenhum		Mínimo -0,1548 Máximo -0,0208 Média -0,0935	Mínimo -0,0943 Máximo 0,0168 Média -0,0408	Mínimo -0,1840 Máximo 0,0406 Média -0,1186	Mínimo 0,8289 Máximo 0,9783 Média 0,8979	0,0652 0,1013 0,0802
Idade	Nenhum	Exponentiated Gradient Reduction	Nenhum		Mínimo -0,1339 Máximo -0,1339 Média -0,1339	Mínimo -0,1146 Máximo -0,1146 Média -0,1146	Mínimo -0,1328 Máximo -0,1328 Média -0,1328	Mínimo 0,887 Máximo 0,887 Média 0,887	0,1055 0,1055 0,1055
Nacionalidade	Nenhum	Rich Subgroup Fairness	Nenhum		Mínimo -0,1402 Máximo 0,1402 Média 0,1402	Mínimo -0,103 Máximo -0,103 Média -0,103	Mínimo -0,2638 Máximo -0,2638 Média -0,2638	Mínimo 0,8423 Máximo 0,8423 Média 0,8423	0,1427 0,1427 0,1427
Nacionalidade	Nenhum	Exponentiated Gradient Reduction	Nenhum		Mínimo -0,2109 Máximo -0,2147 Média -0,2186	Mínimo -0,1853 Máximo -0,0977 Média -0,1015	Mínimo -0,2989 Máximo -0,2903 Média -0,2943	Mínimo 0,7801 Máximo 0,7853 Média 0,7814	0,1101 0,1165 0,1135
Nacionalidade	Nenhum	Prejudice Remover	Nenhum		Mínimo 0,0442 Máximo 0,0442 Média 0,0442	Mínimo 0,1523 Máximo 0,1523 Média 0,1523	Mínimo -0,1049 Máximo 0,1049 Média -0,1049	Mínimo 1,057 Máximo 1,057 Média 1,057	0,1274 0,1274 0,1274
Idade	Nenhum	Regressão Logística	Equalized Odds		Mínimo 0,1726 Máximo 0,1726 Média 0,1726	Mínimo 0,0084 Máximo 0,0084 Média 0,0084	Mínimo 0,3042 Máximo 0,3042 Média 0,3042	Mínimo 1,2158 Máximo 1,2158 Média 1,2158	0,0199 0,0199 0,0199
Nacionalidade	Nenhum	Random Forest	Calibrated Equalized Odds		Mínimo -0,1193 Máximo -0,0041 Média -0,08	Mínimo 0 Máximo 0 Média 0	Mínimo 0,207 Máximo 0,3103 Média 0,1853	Mínimo 0,8656 Máximo 0,9594 Média 0,91	0,0203 0,04 0,0314
Nacionalidade	Nenhum	Gradient Boosting	Calibrated Equalized Odds		Mínimo -0,0617 Máximo -0,025 Média -0,0407	Mínimo 0 Máximo 0 Média 0	Mínimo 0,2155 Máximo 0,2759 Média 0,25	Mínimo 0,9306 Máximo 0,9719 Média 0,952	0,0362 0,0428 0,0401
Idade	Nenhum	Gradient Boosting	Equalized Odds		Mínimo 0,1176 Máximo 0,1563 Média 0,1305	Mínimo 0,1177 Máximo 0,1513 Média 0,1401	Mínimo 0,2156 Máximo 0,2088 Média 0,1334	Mínimo 1,1955 Máximo 1,25 Média 1,2137	0,0786 0,0964 0,0995
Idade	Nenhum	Random Forest	Equalized Odds		Mínimo 0,1682 Máximo 0,2126 Média 0,1974	Mínimo 0,1692 Máximo 0,3277 Média 0,1995	Mínimo 0,2139 Máximo 0,2506 Média 0,2280	Mínimo 1,2743 Máximo 1,5094 Média 1,3571	0,0751 0,2193 0,1279

Com estes dados, é possível reforçar com clareza observações notadas anteriormente: Boas métricas de Performance nos casos onde algoritmos com redução de viés no resultado foram utilizados, boas métricas de *Fairness* nos casos onde o algoritmo *Support Vector Machines* foi utilizado, mudanças nas métricas de *Fairness* dependendo do algoritmo de treinamento ou do atributo protegido utilizado independente de utilizar redução de viés no *workflow* ou não. Este reforço comprova que o cálculo utilizado para a etapa de Análise no componente MAPE-K foi eficaz em consolidar as métricas existentes sem afetar a análise dos resultados.

Entretanto, perceber qual *workflow* possui o melhor equilíbrio entre estes dois gru-

pos de métricas com contextos completamente diferentes ainda se torna difuso diante da grande quantidade de métricas e conjuntos de algoritmos utilizados. Além disso, a diferença entre as métricas é extremamente pequena e dificulta ainda mais a escolha. Nesse contexto, a consolidação das métricas em grupos simplifica a visualização de quais *workflows* são mais equilibrados, e o uso de pesos para cada métrica e para cada grupo pode calibrar qual o melhor equilíbrio desejado para determinada situação.

Deste modo, pode-se concluir também que, em um contexto de desenvolvimento, o processo simplifica a decisão do Cientista de Dados e reduz显著mente 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.

O uso da AI Reference Architecture para definição dos papéis permite 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.

O uso de *Assurance Cases* permitiu uma melhor visualização em no contexto da aplicação a ser implementada, com suas funcionalidades, objetivos e algoritmos a serem implementados e cumpridos. Desta forma, também é possível dividir melhor as tarefas para uma equipe implementá-la e permitir que todos os membros tenham uma visão de todos os detalhes a serem implementados e testados.

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 *workflow* 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 *workflow* muito bem definido para análise de dados e execução de ações, facilitando o entendimento de seu funcionamento e, consequentemente, de seu código.

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

Ao usar uma interface humano-computador para intermediar as interações entre o componente MAPE-K e as escolhas de um usuário, houve uma maneira completamente diferente de como enxergar as soluções que podem ser propostas. Inicialmente, teve-se a ideia de que a autonomia seria contínua, em uma espécie de *workflow* completamente "online", onde era possível obter resultados imediatos com o deploy de novos modelos através de um componente orquestrador das etapas do MAPE-K. Entretanto, foi notado que tal abordagem dificultava o entendimento de seu funcionamento, motivando a elaboração da interface. Após o desenvolvimento da mesma, foi possível notar e 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, implicando em resultados mais eficientes para o treinamento de novos usuários.

O uso da interface também levou a uma conclusão inesperada: Com alguns ajustes, é possível adaptar o sistema para processos de MLOps (ver Apêndice A.4), 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, consequentemente, melhores métricas no geral. Isso necessitaria de algumas adaptações, como um sistema para versionamento dos conjuntos de dados para armazenamento e economia de espaço, funcionalidades como opções para notificação em casos como piora das métricas e opções de visualização dos dados, e modificações no *workflow* para deixá-lo mais flexível, robusto e com suporte a técnicas bastante utilizadas em *Machine Learning* como *Data Augmentation* e *K-Fold Cross-Validation*.

Capítulo 5

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 MAPE-K em diferentes contextos, um maior foco foi colocado na manutenção do sistema, discutindo se as etapas e arquiteturas escolhidas são viáveis para evoluir e manter o *Workflow* sem grandes deteriorações nas ideias originais de seu desenvolvimento.

5.1 Contexto e limitações

Foram realizados testes com o seguinte conjunto de dados e dado o seguinte objetivo:

- **Objetivo:** Obter classificação de crédito (boa ou ruim), através de uma série de *features*.
- **Conjunto de dados:** Lendingclub Dataset, presente em [9].
- **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 [75], 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 *workflow* e reforço da viabilidade do MAPE-K 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 Workflow utilizando o componente MAPE-K 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, as pré-condições 1 e 2 do Estudo de Caso 1 foram mantidas, e a pré-condição 3 foi desconsiderada.
- **Restrições:** As 4 restrições determinadas no Estudo de Caso 1 foram mantidas, com uma diferença: Na Restrição 4, como o *Lendingclub Dataset* possui resultados mais eficientes do que o *German Credit Dataset*, o intervalo de pontuação para análise foi ampliado de 500 a 980, mas as motivações para esse valor permanecem as mesmas.

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 opções escolhidas pelo modelo MAPE-K
 Todos os métodos - 50% Performance/50% Fairness

Atributo protegido	Workflow			Pontuação		
	Pré-processamento	Treinamento	Pós-processamento	Performance	Fairness	Geral
Renda	Learning Fair Representations	Random Forest	Nenhum	991	968	979
Renda	Nenhum	Gradient Boosting	Equalized Odds	988	969	978
Renda	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 5.2: Melhores opções escolhidas pelo modelo MAPE-K
Todos os métodos - 75% Performance/25% Fairness

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

Tabela 5.3: Melhores opções escolhidas pelo modelo MAPE-K
Todos os métodos - 25% Performance/75% Fairness

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

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

Para processar o Lendingclub Dataset, foram necessárias modificações para realizar a evolução do sistema e adicionar este conjunto de dados como opção no *Workflow*. Estas foram contadas de acordo com seus *commits* realizados no repositório e exibidos na Tabela 5.4:

Tabela 5.4: Quantidade de modificações realizadas ao adicionar um novo conjunto de dados ao *Workflow*

Parte do Sistema	Quantidade de linhas alteradas	Total de linhas	Quantidade de arquivos alterados	Total de arquivos	Porcentagem de linhas alteradas	Porcentagem de arquivos alterados
Transformação do Conjunto de Dados	122	277	2	3	44,04%	66,67%
<i>Workflow</i> de IA	76	1982	5	38	3,84%	13,16%
Componente MAPE-K	0	457	0	10	0,00%	0,00%
Interface Humano-Computador (<i>Frontend</i>)	13	2905	2	14	0,45%	14,29%
Interface Humano-Computador (<i>Backend</i>)	4	432	1	7	0,93%	14,29%
TOTAL	215	6053	10	72	3,55%	13,89%

A primeira conclusão que é possível discutir é que a adição do conjunto de dados não exigiu modificações no Componente MAPE-K, mesmo os resultados sendo completamente diferentes do Estudo de Caso 1. Isto reforça a autonomia proposta no MAPE-K, possibilitando escolhas diferentes baseadas nos metadados presentes no *Workflow* sem realizar modificações. O Componente MAPE-K só exigirá modificações se os metadados salvos do *Workflow* forem modificados, ou se modificar alguma configuração intrínseca ao próprio MAPE-K, que não possui nenhuma relação com o *Workflow*.

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

Embora tenha a consequência de escrever algumas linhas a mais, o uso da arquitetura *Pipe-and-Filter* permite o encapsulamento dos algoritmos e a separação de interesses de forma simples, fazendo com que a parte de código existente para o *workflow* possa ter escolhas mais interessantes e elegantes para um bom *Design* do código. Como exemplo disso há o Trecho 5.1, onde há grande flexibilidade para configurar o método de pós-processamento e ele pode ser expandido conforme novas classes de filtro forem implementadas sem grande esforço.

```
1  def data_postprocess(self, test_pipe, prediction_pipe, fairness_pipe
2      , unbias_postproc_algorithm):
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
13     for option, filter in unbias_postproc_options:
14         if unbias_postproc_algorithm == option:
15             init_pipe = test_pipe + prediction_pipe + fairness_pipe[
16                 'unprivileged_group', 'privileged_group']
17             init_pipe >= filter == prediction_pipe
18             break
19
20
21     return prediction_pipe
```

Código 5.1: Método para escolha do algoritmo com redução de viés no pós-processamento

Dito isso, realizar a manutenção/evolução do sistema no *Workflow* e na Interface Humano-Computador é relativamente simples, desde que se saiba os arquivos onde as

modificações serão realizadas. Por isso, a criação de uma documentação, presente nos Anexos A e B, é extremamente importante para que um novo desenvolvedor entenda o todo do sistema e não adicione linhas em trechos desnecessários.

Capítulo 6

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 focará mais se as arquiteturas escolhidas são versáteis e simples o suficiente para que pessoas entendam o contexto do sistema e façam novas evoluções.

6.1 Contexto e limitações

Foram realizados testes com o seguinte conjunto de dados e dado o seguinte objetivo:

- **Objetivo:** Obter classificação de crédito (boa ou ruim), através de uma série de *features*.
- **Conjunto de dados:** Lendingclub Dataset, presente em [9].
- **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 [75], 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 *workflow*, 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.

6.2 Resultados e Discussões

Foi realizada uma sessão de desenvolvimento com um outro desenvolvedor, durando um tempo entre 1 e 2 horas, onde este adicionou um novo algoritmo de classificação dentro da estrutura do *workflow* com a documentação montada durante o Estudo de caso anterior. Durante a sessão, 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, 1 no Frontend e outro no MAPE-K, sendo detectados rapidamente, corrigidos e colocados na documentação.
- **Correções na documentação** - Foram colocados modificações em métodos que faltaram constar na documentação para obter certas parametrizações do **Workflow**, 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 Anexo 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. Dada a afirmação, é possível concluir que as decisões arquiteturais tomadas foram corretas e facilitaram o desenvolvimento de evoluções.

Outro item notado não apenas após o desenvolvimento, mas durante todo o desenvolvimento do *workflow*, foram notados 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 aprimorar tal *workflow* 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??, 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 6.1, 6.2 e 6.3:

Tabela 6.1: Melhores opções escolhidas pelo modelo MAPE-K
Todos os métodos - 50% Performance/50% Fairness

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

Tabela 6.2: Melhores opções escolhidas pelo modelo MAPE-K
Todos os métodos - 75% Performance/25% Fairness

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

Tabela 6.3: Melhores opções escolhidas pelo modelo MAPE-K
Todos os métodos - 25% Performance/75% Fairness

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

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

As evoluções realizadas no *Workflow* para adicionar o Naive Bayes foram contadas de acordo com seus *commits* realizados no repositório e exibidos na Tabela 5.4:

Tabela 6.4: Quantidade de modificações realizadas ao adicionar um novo algoritmo ao *Workflow*

Parte do Sistema	Quantidade de linhas alteradas	Total de linhas	Quantidade de arquivos alterados	Total de arquivos	Porcentagem de linhas alteradas	Porcentagem de arquivos alterados
Transformação do Conjunto de Dados	0	277	0	3	0,00%	0,00%
<i>Workflow</i> de IA	60	2042	5	39	2,94%	12,82%
Componente MAPE-K	1	458	1	10	0,22%	10,00%
Interface Humano-Computador (<i>Frontend</i>)	71	2948	3	14	2,41%	21,43%
Interface Humano-Computador (<i>Backend</i>)	1	433	1	7	0,23%	14,29%
TOTAL	132	6157	9	72	2,14%	12,50%

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

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

Capítulo 7

Conclusões

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

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

Embora olhar para a proveniência dos dados permita o armazenamento de dados importantes para estabelecer um elo entre o *workflow* e o componente MAPE-K, não foi possível garantir que ela substitua **Explainable AI** pois não foram encontradas evidências do porquê de determinados algoritmos funcionarem em determinados conjuntos de dados e não em outros com os metadados cadastrados na base de conhecimento. A qualidade dos dados também é um ponto que deverá ser observado conforme o número de dados for crescente: É possível ter resultados fora do comum (*outliers*), alguns atributos podem não permitir uma avaliação acurada, ou a quantidade de atributos pode não ser suficiente para avaliar. Antes da proveniência ser implementada, os próprios dados precisam ser

modelados conforme o problema exige. Após a implementação, é preciso realizar processos de limpeza dos dados conforme execuções no *workflow* vão sendo realizadas.

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

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 *workflow* 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 humano-computador simplifica o entendimento do processo utilizado para a obtenção do *workflow* 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.

Por fim, a AI Reference Architecture e os *Assurance Cases* são bons guias para evoluir a ideia de uma aplicação de IA de um contexto generalizado para um foco maior nos pequenos detalhes a serem realizados dentro da fase de desenvolvimento, e a evolução deste método para um foco mais profundo nos problemas de MLOps é um caminho interessante a ser seguido. Como trabalhos futuros, é possível focar em adaptações no sistema para tais problemas, como mecanismos para deploy dos modelos obtidos, um sistema para versionamento dos conjuntos de dados para armazenamento e economia de espaço, funcionalidades como opções para notificação em casos como piora das métricas e opções de visualização dos dados, e modificações no *workflow* para deixá-lo mais flexível, robusto e com suporte a técnicas bastante utilizadas em *Machine Learning* como *Data Augmentation* e *K-Fold Cross-Validation*. Também é possível explorar mais experimentos de como a mudança nas configurações presentes no componente MAPE-K afeta a escolha dos melhores *Workflows* e propor novas modificações após a realização dos mesmos.

Referências Bibliográficas

- [1] Ai fairness 360. URL <https://aif360.mybluemix.net>.
- [2] Statlog (german credit data) data set. URL <https://airflow.apache.org/docs/apache-airflow/stable/index.html>.
- [3] What's the difference between artificial intelligence, machine learning and deep learning? URL <https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/>.
- [4] Definition of modelops - gartner information technology glossary. URL <https://www.gartner.com/en/information-technology/glossary/modelops>.
- [5] IBM - Analytics and AI architecture. URL <https://www.ibm.com/cloud/architecture/architectures/aiAnalyticsArchitecture/reference-architecture/>.
- [6] Machine learning: o que é e qual sua importância? URL https://www.sas.com/pt_br/insights/analytics/machine-learning.html.
- [7] Aprendizado de máquina. URL https://pt.wikipedia.org/wiki/Aprendizado_de_máquina.
- [8] Python - special method names. URL <https://docs.python.org/3/reference/datamodel.html#special-method-names>.
- [9] Lendingclub dataset. URL <https://www.kaggle.com/datasets/wordsforthewise/lending-club>.
- [10] scikit-learn. URL <https://scikit-learn.org>.
- [11] Statlog (german credit data) data set. URL [https://archive.ics.uci.edu/ml/datasets/statlog+\(german+credit+data\)](https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data)).
- [12] An architectural blueprint for autonomic computing. Technical report, IBM, 2005. URL <https://www-03.ibm.com/autonomic/pdfs/AC%20Blueprint%20White%20Paper%20V7.pdf>.
- [13] *The elements of statistical learning: data mining, inference and prediction*. Springer, 2009. URL <https://hastie.su.domains/Papers/ESLII.pdf>.

- [14] Data engineering - introduction and epochs. Technical report, panopoly.io, 2017. URL <https://www-03.ibm.com/autonomic/pdfs/AC%20Blueprint%20White%20Paper%20V7.pdf>.
- [15] Machine learning: things are getting intense, 2018. URL <https://www2.deloitte.com/content/dam/Deloitte/global/Images/infographics/technologymediatelecommunications/gx-deloitte-tmt-2018-intense-machine-learning-report.pdf>.
- [16] Brasil se destaca com 42% das iniciativas de IA na América Latina, em 2020, 2021. URL <https://cio.com.br/tendencias/brasil-se-destaca-com-42-das-iniciativas-de-ia-na-america-latina-em-2020/>.
- [17] Proteção de Dados - LGPD, 2021. URL <https://www.gov.br/defesa/pt-br/acesso-a-informacao/lei-geral-de-protecao-de-dados-pessoais-lgpd>.
- [18] Nadeem Abbas, Jesper Andersson, and Welf Löwe. Autonomic software product lines. pages 324–331, 2010.
- [19] 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.
- [20] 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.
- [21] Alejandro Barredo Arrieta, Natalia Díaz Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-Lopez, Daniel Molina, Richard Benjamins, Raja Chatila, and Francisco Herrera. Explainable artificial intelligence (XAI): concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, 58:82–115, 2020.
- [22] Tom Begley, Tobias Schwedes, Christopher Frye, and Ilya Feige. Explainability for fair machine learning, 2021.
- [23] 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.
- [24] Sumon Biswas and Hridesh Rajan. Do the machine learning models on a crowd sourced platform exhibit bias? an empirical study on model fairness. page 642–653, 2020.
- [25] Jan Bosch. Software architecture: The next step. pages 194–199, 2004.
- [26] Rajendra Bose and James Frew. Lineage retrieval for scientific data processing: A survey. *ACM Comput. Surv.*, page 1–28, 2005.

- [27] L Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.
- [28] Peter Buneman, Sanjeev Khanna, and Wang Chiew Tan. Why and where: A characterization of data provenance. In *Proceedings of the 8th International Conference on Database Theory*, page 316–330, 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 Nateyan Ramamurthy, and Kush R Varshney. Optimized pre-processing for discrimination prevention. In *Advances in Neural Information Processing Systems*, 2017. URL <https://proceedings.neurips.cc/paper/2017/file/9a49a25d845a483fae4be7e341368e36-Paper.pdf>.
- [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] Susan B. Davidson and Juliana Freire. Provenance and scientific workflows: Challenges and opportunities. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, page 1345–1350, 2008.
- [36] Timothy Dozat. Incorporating nesterov momentum into adam. 2016.
- [37] 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.
- [38] Jeffrey Dunn. Introducing FB Learner Flow: Facebook’s AI backbone. URL <https://engineering.fb.com/2016/05/09/core-data/introducing-fblearner-flow-facebook-s-ai-backbone/>.
- [39] Luc (Editor, Beth Plale, Simon Miles, Carole Goble, Paolo Missier, Roger Barga, Yogesh Simmhan, Joe Futrelle, Robert McGrath, James Myers, Patrick Paulson, Shawn Bowers, Bertram Ludäscher, Natalia Kwasnikowska, Jan Van den Bussche,

- Tommy Ellqvist, Juliana Freire, and Paul Groth. The open provenance model (v1.01). 2009.
- [40] R. J. Erb. Introduction to backpropagation neural network computation. *Pharmaceutical Research*, 10:165–170, 1993.
- [41] Michael Feldman, Sorelle A. Friedler, John Moeller, Carlos Scheidegger, and Suresh Venkatasubramanian. Certifying and removing disparate impact. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 259–268, 2015. URL <https://doi.org/10.1145/2783258.2783311>.
- [42] Keith D. Foote. A brief history of data science, 2021. URL <https://www.dataversity.net/brief-history-data-science/>.
- [43] Martin Fowler. Fluentinterface, 2005. URL <https://martinfowler.com/bliki/FluentInterface.html>.
- [44] Jerome Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29, 2000.
- [45] David Garlan and Mary Shaw. An introduction to software architecture. In *Advances in Software Engineering and Knowledge Engineering*, 1993.
- [46] 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.
- [47] Alex Graves. Generating sequences with recurrent neural networks. 2013.
- [48] Moritz Hardt, Eric Price, Eric Price, and Nati Srebro. Equality of opportunity in supervised learning. In *Advances in Neural Information Processing Systems*, 2016. URL <https://proceedings.neurips.cc/paper/2016/file/9d2682367c3935defcb1f9e247a97c0d-Paper.pdf>.
- [49] Waldemar Hummer, Vinod Muthusamy, Thomas Rausch, Parijat Dube, Kaoutar El Maghraoui, Anupama Murthi, and Punleuk Oum. Modelops: Cloud-based lifecycle management for reliable and trusted ai. In *2019 IEEE International Conference on Cloud Engineering (IC2E)*, pages 113–120, 2019.
- [50] Faisal Kamiran and Toon Calders. Data pre-processing techniques for classification without discrimination. *Knowledge and Information Systems*, 2011. URL https://www.researchgate.net/publication/228975972_Data_Pre-Processing_Techniques_for_Classification_without_Discrimination.
- [51] 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.

- [52] Andrei Kapishnikov, Tolga Bolukbasi, Fernanda B. Viégas, and Michael Terry. XRAI: better attributions through regions. In *IEEE/CVF International Conference on Computer Vision*, pages 4947–4956, 2019.
- [53] 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.
- [54] Krishnaram Kenthapadi, Stuart Ambler, Ahsan Chudhary, Mark Dietz, Sahin Cem Geyik, Ian Koeppe, Varun Mithal, Guillaume Saint-Jacques, Amir Sepehri, Thanh Tran, and Sriram Vasudevan. Fairness, privacy, and transparency by design in ai/ml systems, 2019. URL <https://engineering.linkedin.com/blog/2019/fairness-privacy-transparency-by-design>.
- [55] Jeffrey Kephart and D.M. Chess. The vision of autonomic computing. pages 41 – 50, 2003.
- [56] Bahador Khalegi. The how of explainable ai: Explainable modelling, 2019. URL <https://towardsdatascience.com/the-how-of-explainable-ai-explainable-modelling-55c8c43d7bed>.
- [57] Bahador Khalegi. The how of explainable ai: Post-modelling explainability, 2019. URL <https://towardsdatascience.com/the-how-of-explainable-ai-post-modelling-explainability-8b4cbc7adf5f>.
- [58] Bahador Khalegi. The how of explainable ai: Pre-modelling explainability, 2019. URL <https://towardsdatascience.com/the-how-of-explainable-ai-pre-modelling-explainability-699150495fe4>.
- [59] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2014.
- [60] Nikolaos Konstantinou, Martin Koehler, Edward Abel, Cristina Civil, Bernd Neu-mayr, 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. URL <https://doi.org/10.1145/3035918.3058730>.
- [61] Rezvan Mahdavi-hezaveh, Jacob Dremann, and Laurie Williams. Software development with feature toggles: practices used by practitioners. *Empirical Software Engineering*, 2021.
- [62] Sasan Maleki, Long Tran-Thanh, Greg Hines, Talal Rahwan, and Alex Rogers. Bounding the estimation error of sampling-based shapley value approximation with/without stratifying. 2013.
- [63] Warren McCulloch and Walter Pitts. A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:127–147, 1943.

- [64] 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.
- [65] Luc Moreau, Bertram Ludäscher, Ilkay Altintas, Roger Barga, Shawn Bowers, Steven Callahan, George Chin, Ben Clifford, Shirley Cohen, Sarah Cohen-Boulakia, Susan Davidson, Ewa Deelman, Luciano Digiampietri, Ian Foster, Juliana Freire, James Frew, Joe Futrelle, Tara Gibson, Yolanda Gil, and Jun Zhao. The first provenance challenge. *Concurrency and Computation: Practice and Experience*, 2007.
- [66] Luc Moreau, Paul Groth, Simon Miles, Javier Vázquez-Salceda, John Ibbotson, Jian Sheng, Steve Munroe, Omer Rana, Andreas Schreiber, Victor Tan, and László Varga. The provenance of electronic data. *Communications of the ACM*, pages 52–58, 2008.
- [67] Carlos Mougan, José Manuel Álvarez, Gourab K. Patro, Salvatore Ruggieri, and Steffen Staab. Fairness implications of encoding protected categorical attributes. 2022.
- [68] Tomoki Nakamaru and Shigeru Chiba. Generating a generic fluent api in java. *The Art, Science, and Engineering of Programming*, 2020. URL <http://dx.doi.org/10.22152/programming-journal.org/2020/4/9>.
- [69] Geoff Pleiss, Manish Raghavan, Felix Wu, Jon Kleinberg, and Kilian Q Weinberger. On fairness and calibration. In *Advances in Neural Information Processing Systems*, 2017. URL <https://proceedings.neurips.cc/paper/2017/file/b8b9c74ac526ffffbeb2d39ab038d1cd7-Paper.pdf>.
- [70] Boris Polyak. Some methods of speeding up the convergence of iteration methods. *Ussr Computational Mathematics and Mathematical Physics*, 4:1–17, 1964.
- [71] Daphne Raban and Avishag Gordon. The evolution of data science and big data research: A bibliometric analysis. *Scientometrics*, 122:1563–1581, 2020.
- [72] Md Tajmilur Rahman, Louis-Philippe Querel, Peter C. Rigby, and Bram Adams. Feature toggles: Practitioner practices and a case study. In *Proceedings of the 13th International Conference on Mining Software Repositories*, page 201–211. Association for Computing Machinery, 2016.
- [73] C.v Ramamoorthy and Benjamin Wah. Knowledge and data engineering. *IEEE Transactions on Knowledge and Data Engineering*, 1:9 – 16, 1989.
- [74] 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.
- [75] Brian C. Ross. Mutual information between discrete and continuous data sets. *PLoS ONE*, 2014. URL <http://dx.plos.org/10.1371/journal.pone.0087357>.

- [76] Mark Schmidt, Nicolas Roux, and Francis Bach. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 162, 2013.
- [77] 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. URL <https://proceedings.neurips.cc/paper/2015/file/86df7dcfd896fcacf2674f757a2463eba-Paper.pdf>.
- [78] Colin Shearer. The crisp-dm model: The new blueprint for data mining. *Journal of Data Warehousing*, 2000.
- [79] Yashpal Singh and Alok Singh Chauhan. Neural networks in data mining. *Journal of Theoretical & Applied Information Technology*, 5, 2009.
- [80] Till Speicher, Hoda Heidari, Nina Grgic-Hlaca, Krishna P. Gummadi, Adish Singla, Adrian Weller, and Muhammad Bilal Zafar. A unified approach to quantifying algorithmic unfairness. 2018. URL <http://dx.doi.org/10.1145/3219819.3220046>.
- [81] Ingo Steinwart and Andreas Christmann. *Support Vector Machines*. Springer Science & Business Media, 2008. URL <http://www.lamda.nju.edu.cn/zhangt/seminar/RKHS/pdf/Support%20Vector%20Machines%20-%20Ingo%20Steinwart,%20Andreas%20Christmann.pdf>.
- [82] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3319–3328, 2017.
- [83] 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.
- [84] Wang-chiew Tan. Provenance in databases: Past, current, and future. *IEEE Data Eng. Bull.*, pages 3–12, 2007.
- [85] 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.
- [86] Ian H. Witten, Eibe Frank, and Mark A. Hall. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016. URL <https://www.wi.hs-wismar.de/~cleve/vorl/projects/dm/ss13/HierarClustern/Literatur/WittenFrank-DM-3rd.pdf>.
- [87] Allison Woodruff and Michael Stonebraker. Supporting fine-grained data lineage in a database visualization environment. Technical report, EECS Department, University of California, Berkeley, 1997.

- [88] Rich Zemel, Yu Wu, Kevin Swersky, Toni Pitassi, and Cynthia Dwork. Learning fair representations. In *Proceedings of the 30th International Conference on Machine Learning*, pages 325–333, 2013. URL <https://proceedings.mlr.press/v28/zemel13.html>.
- [89] Brian Zhang, Blake Lemoine, and Margaret Mitchell. Mitigating unwanted biases with adversarial learning. pages 335–340, 2018. URL https://www.researchgate.net/publication/330299272_Mitigating_Unwanted_Biases_with_Adversarial_Learning.

Apêndice A

Conceitos complementares

A.1 AI Explainability

AI Explainability é um conceito em IA que propõe a criação de um conjunto de técnicas de Aprendizado de Máquina (ou Machine Learning/ML) que produz modelos mais explicáveis, mantendo qualidade em suas métricas, e permite que os humanos entendam, confiem e gerenciem aplicações baseadas em IA. XAI também absorve conceitos das Ciências Sociais e considera a psicologia da expiação [21]. Um algoritmo de ML explicável precisa não apenas mostrar tomadas de decisão, mas também mostrar o processo que o levou ao tomar tal decisão, de modo que seja compreensível e transparente para humanos.

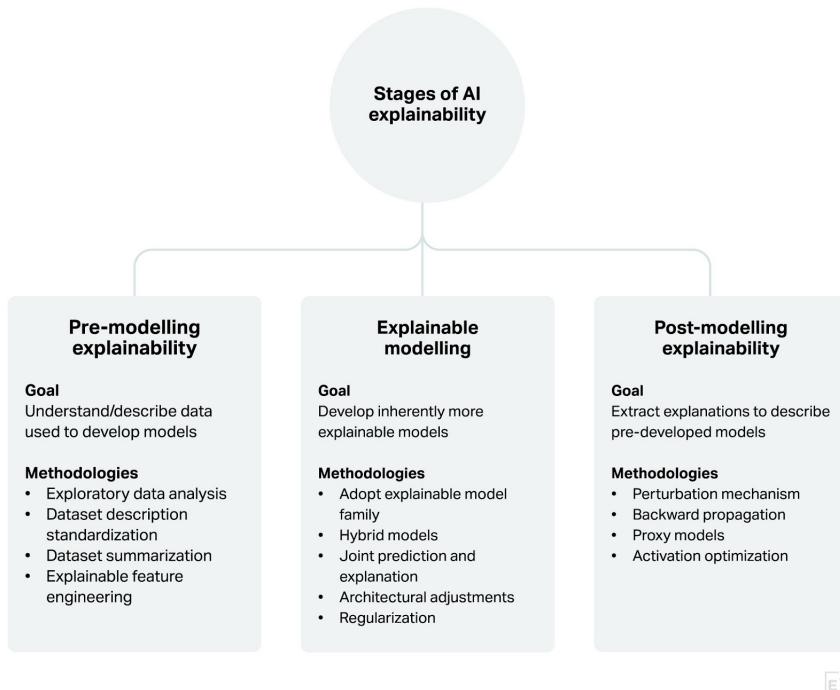


Figura A.1: Fases de um processo baseado em IA explicável.

Conforme ilustrado na Figura A.1, o um processo baseado em IA explicável pode ser classificado em 3 fases:

- **Pre-Modelling Explainability:** Esta fase se caracteriza por obter explicações no conjunto de dados usado para treino e validação, tendo como principal motivação o fato do comportamento de um modelo depender muito dos dados que o alimentam. Análises e visualizações dos dados, ou mesmo através de documentações do conjunto de dados se enquadram neste estágio. [58]
- **Explainable Modelling:** Esta fase se caracteriza por obter explicações por modelos considerados transparentes (ver adiante). Ao adotar modelos deste tipo, ele já pode ser considerado como explicável por um humano. Também é possível obter a explicação através de regularizadores, ou adotar abordagens onde explicação e resultado sejam obtidos simultaneamente, sejam via junção de dados ou sendo intrínseco a arquitetura do modelo. [56]
- **Post-Modelling Explainability:** Também chamada de *Post-hoc Explainability* e onde a maioria dos trabalhos disponíveis baseados em XAI tenta focar, esta fase se dedica a explicar modelos desenvolvidos anteriormente, ou modelos que não são considerados como transparentes. Pode-se explicar um modelo por métodos como árvores de decisão e estimativas como valores de Shapley e propagação inversa. [57]

Entretanto, estas 3 fases não necessariamente são executadas de maneira sequencial: Por ser realizada no conjunto de dados e não no modelo, a fase de *Pre-modelling Explainability* pode ser opcional, e a aplicação das fases de *Explainable Modelling* e *Post-Modelling Explainability* e seus métodos dependem exclusivamente se o modelo é classificado como transparente ou não, e tal critério é determinado pela adoção de uma (ou mais) das seguintes características [21]:

- **Simulabilidade:** Denota a capacidade de um modelo de ser simulado ou pensado estritamente por um humano. A complexidade assume um lugar dominante nesta classe: Sistemas com uma grande quantidade de regras, por mais simples que essas sejam, já não podem ser classificados como tal. O modelo que se adequa a essa característica precisa ser autocontido o suficiente para que um ser humano pense e raciocine sobre ele como um todo.
- **Decomponibilidade:** Também considerado como inteligibilidade, significa a capacidade de explicar cada uma das partes de um modelo. Se o modelo não for simples o suficiente, ele precisa ser divisível em várias pequenas partes e cada parte do modelo deve ser compreensível por um ser humano, sem a necessidade de ferramentas adicionais.
- **Transparência do algoritmo:** Trata-se da habilidade do usuário de entender o processo seguido pelo modelo para produzir qualquer saída dada a partir de seus dados de entrada. Colocando de outra forma, um modelo linear é considerado transparente porque sua superfície de erro pode ser entendida e fundamentada, permitindo ao usuário entender como o modelo irá agir em cada situação que pode enfrentar.

A.2 *Fluent API*

O conceito de *Fluent Interface* [43] (também conhecido como *Fluent API* [68]) é uma abstração de código que o organiza de modo a criar uma *Domain Specific Language* (DSL) interna, tendo como prioridade em seu *design* a legibilidade e a fluidez. Embora a construção de uma *Fluent API* consuma tempo, esse tempo pode ser recuperado com o tempo devido a maior facilidade no desenvolvimento e na manutenção dos componentes presentes no sistema.

Seu desenvolvimento lembra o *Design Pattern Builder*, onde seus métodos realizam ações específicas e retornam a referência do Objeto. Sua diferença está no objetivo em que foi desenvolvido: Enquanto no *Builder* o objetivo está na construção da instância de um Objeto, na *Fluent API* o objetivo está em desenvolver uma série de ferramentas e ações de modo a resolver um problema específico.

A melhor forma de descrever e entender as diferenças e objetivos da *Fluent API* é através de um exemplo: O Código A.1 representa uma implementação padrão de um sistema, com instanciações de novos Objetos e métodos determinando atribuições e adição do elemento em uma lista.

```

1  private void makeNormal(Customer customer) {
2      Order o1 = new Order();
3      customer.addOrder(o1);
4      OrderLine line1 = new OrderLine(6, Product.find("TAL"));
5      o1.addLine(line1);
6      OrderLine line2 = new OrderLine(5, Product.find("HPK"));
7      o1.addLine(line2);
8      OrderLine line3 = new OrderLine(3, Product.find("LGV"));
9      o1.addLine(line3);
10     line2.setSkippable(true);
11     o1.setRush(true);
12 }
```

Código A.1: Implementação padrão presente em um sistema [43]

O Código A.2 representa o mesmo sistema usando uma implementação com *Fluent API*. Além na diminuição no número de linhas, a implementação é mais legível devido ao menor número de elementos presentes no código: Todas as chamadas e instanciações do Código A.1 estão ímplicitas, e o encadeamento das funções possui uma progressão lógica, mais sucinta e de simples entendimento.

```

1  private void makeFluent(Customer customer) {
2      customer.newOrder()
3          .with(6, "TAL")
4          .with(5, "HPK").skippable()
5          .with(3, "LGV")
6          .priorityRush();
7 }
```

Código A.2: Implementação com o uso de *Fluent API* [43]

A.3 *Feature Toggles*

Feature toggles é um conceito antigo e conceitualmente simples: Basicamente, é uma variável usada em uma instrução condicional para proteger blocos de código, com o objetivo de habilitar ou desabilitar o código de uma feature nesses blocos para teste ou liberação [72]. Inicialmente elas foram pensadas para serem colocadas em tempo de compilação, excluindo a execução das features no binário de um aplicativo. Atualmente, é possível implementar *feature toggles* que permitem que as *features* sejam ativadas ou desativadas em tempo de execução, geralmente através de um arquivo ou através de uma variável introduzida no Sistema Operacional do *software* executado. O Código A.3 mostra um exemplo de *feature toggle*, onde a escolha dinâmica de um algoritmo de pesquisa depende do valor da *toggle* `useNewAlgorithm`. Se o valor dessa *toggle* for `true`, o novo algoritmo de pesquisa será usado, caso contrário, o método `Search` chamará o algoritmo de pesquisa antigo.

```

1 function Search() {
2     var useNewAlgorithm = false;
3     if(useNewAlgorithm){
4         return newSearchAlgorithm();
5     }else{
6         return oldSearchAlgorithm();
7     }
8 }
```

Código A.3: Implementação de uma *Feature Toggle* [61]

O uso de *feature toggles* é uma técnica frequentemente usada em contextos de integração contínua (CI) e entrega contínua (CD) que permite às equipes integrar e testar uma nova *feature* de forma incremental, mesmo quando ela não está pronta para ser lançada [61]. Os desenvolvedores também usam *feature toggles* para outros fins, como implantação gradual e experimentos. No entanto, a *feature toggle* pode se transformar em débito técnico. O uso de *feature toggles* adiciona mais pontos de decisão ao código, o que adiciona mais complexidade. Essa maior complexidade leva à necessidade de remover *toggles* quando a *feature* estiver concluída, ou olhar com mais atenção para os testes do sistema a ser implementado com a técnica.

Embora agilize testes ou coleta de dados importantes para as empresas, o uso de *feature toggles* sem seguir boas práticas pode ser prejudicial, levando a grandes prejuízos: Em 2012, os desenvolvedores do Knight Capital Group, uma empresa americana de serviços financeiros globais, atualizaram seu roteador algorítmico automatizado de alta velocidade que, inadvertidamente, reprovouitou um *feature toggle*, ativando a funcionalidade que não era utilizada há 8 anos. Em 2 minutos, os desenvolvedores perceberam que o código implantado se comportou incorretamente, mas levaram 45 minutos para interromper o sistema. Durante esse período, a Knight Capital perdeu quase 400 milhões de dólares, o que fez com que o grupo falisse [61].

Por ser uma técnica simples de ser implementada, as linguagens de programação fornecem o necessário para implementar *feature toggles* há muito tempo. No entanto, o primeiro uso desta técnica para suportar CI/CD foi no Flickr em 2009 [61]. Atualmente, empresas como Google, Facebook e Netflix utilizam esta técnica, podendo auxiliar na redução do

tempo de atualização de seus aplicativos para poucas semanas ou até diariamente [72]. Nos *softwares*, as *feature toggles* podem ser categorizadas em cinco tipos [61]:

- **Toggles de lançamento:** *Toggles* usadas para adicionar novas features em um contexto de *trunk-based development*. No *trunk-based development*, todos os desenvolvedores fazem o *commit* das alterações para uma *branch* compartilhada. Usando *toggles* de lançamento no desenvolvimento baseado em tronco suporta CI/CD para features parcialmente concluídas.
- **Toggles de experimento:** *Toggles* usadas para realizar experimentação no *software*, para avaliar novas alterações de recursos e observar sua influência no comportamento do usuário.
- **Toggles de operação:** *Toggles* usadas para controlar o aspecto operacional do comportamento do sistema. Quando uma nova feature é implantada, os operadores do sistema podem desabilitar o recurso rapidamente se ela apresentar alguma inconformidade.
- **Toggles de permissão:** *Toggles* usadas para fornecer a funcionalidade apropriada para um usuário, por exemplo *features* especiais para usuários *premium* ou pagos. *Toggles* de permissão também são chamados de *toggles* de negócios de longo prazo.
- **Toggles de desenvolvimento:** *Toggles* usados para habilitar ou desabilitar certos recursos para testar e depurar código.

Toggles de permissão, *toggles* de operação e *toggles* de desenvolvimento são *toggles* de longa duração com base em sua finalidade de uso no código. As *toggles* de lançamento e *toggles* de experimento são *toggles* de curta duração [61].

A.4 ModelOps

O ModelOps (ou operacionalização de modelos de IA) está focado principalmente na governança e gerenciamento do ciclo de vida de uma ampla gama de modelos de inteligência artificial (IA) e de decisão operacionalizados, incluindo aprendizado de máquina, grafos de conhecimento, regras, otimização, modelos linguísticos e modelos baseados em agentes [4]. Os principais recursos incluem integração contínua/entrega contínua (CI/CD), ambientes de desenvolvimento, sistema de controle de versão, armazenamento e reversão de modelos. O termo ModelOps é uma junção de modelos de IA e DevOps, sendo uma nova estrutura e plataforma para gerenciamento completo do ciclo de vida de artefatos de aplicativos de Inteligência Artificial [49] e cobrindo todos os termos relacionados a implantação de modelos baseados em IA, como MLOps e AIOps. Apesar do termo ModelOps ser criado como a generalização destes termos, a ideia de todos eles é utilizar os princípios de Integração Contínua (CI) e Entrega Contínua (Continuous Delivery, ou CD) presentes no DevOps para garantir que o modelo de IA seja sempre o melhor e mais próximo do cenário que aborda.

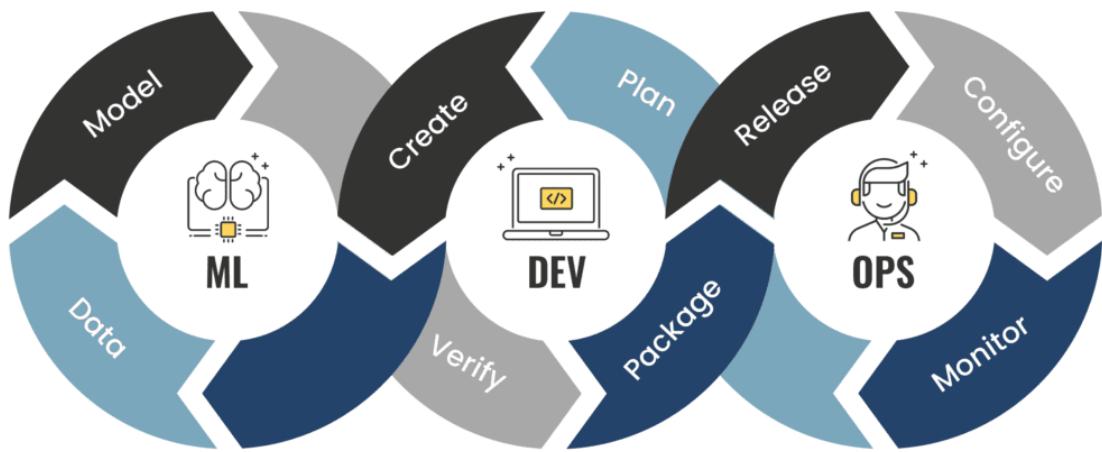


Figura A.2: Ciclo resumido do processo presente em MLOps

A operacionalização dos procedimentos em Software já é uma prática comum de ser adotada, mas em aplicações de IA sua adoção é mais recente, e possuem desafios diferentes de uma aplicação tradicional de Software [77]:

- **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 estas abstrações para sistemas de IA prescrevendo um comportamento pretendido específico. De fato, IA 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:** Débito de dependência é apontado como um dos principais contribuintes para a complexidade do código e débito técnico em configurações clássicas de Engenharia de Software. As dependências de dados em sistemas de IA possuem uma capacidade semelhante para aumentar o débito, mas podem ser mais difíceis de detectar. As dependências de código podem ser identificadas por meio de análise estática por compiladores e vinculadores. 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 IA é que eles geralmente acabam influenciando seu próprio comportamento se forem atualizados ao longo do tempo. Isso leva a uma forma de análise do débito, 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-Patterns nas aplicações:** É comum que sistemas de IA acabem com grandes débitos em padrões de design. Débitos 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, débitos de abstração, *code smells* (códigos que podem indicar problemas subjacentes em um componente ou sistema) devem ser evitados ou refatorados sempre que possível.

- **Débitos 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. 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 IA 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.

No caso de MLOps, ele pode ser visto como o processo de implantar os melhores modelos latentes de Machine Learning para o ambiente de produção, unindo os campos de Machine Learning, DevOps e Engenharia de Dados. Um projeto de MLOps, como exemplo mostrado na Figura A.3, deve dar suporte à automação, integração e monitoramento em todas as etapas da construção de um sistema de ML, incluindo treinamento, integração, teste, lançamento, implantação e gerenciamento de infraestrutura [85]. A ideia é projetar um pipeline automatizado usando o CI/CD, sendo a tarefa de CI de testar e validar dados, esquemas de dados e modelos, e a tarefa de CD um pipeline de ML que deve implantar automaticamente outro modelo de ML.

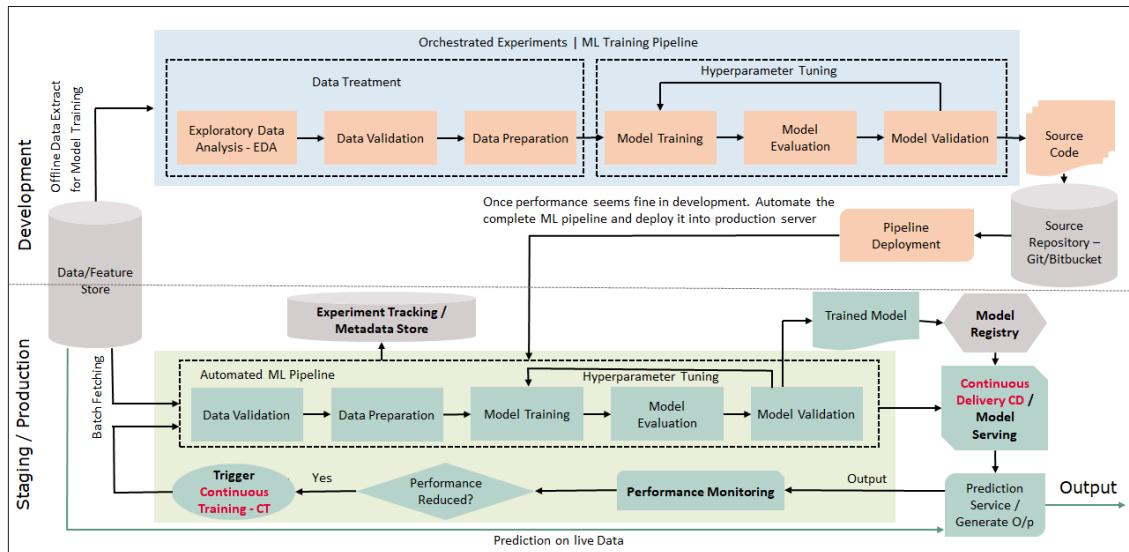


Figura A.3: Exemplo de Pipeline de MLOps com ambientes de Desenvolvimento e Produção

O ciclo de vida de Machine Learning tem metodologias diferentes para se adequar a diferentes cenários e tipos de dados. A abordagem mais utilizada por especialistas em mineração de dados é o Cross-Industry Standard Process for Data Mining (CRISP-DM) [78], 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: Os Cientistas de Dados, responsáveis pelo treinamento e teste do modelo, e os Engenheiros de Machine Learning, responsáveis pela produção e implantação. Tais papéis trabalhariam em conjunto em tarefas envolvendo as seguintes etapas:

- Análise do problema do Negócio
- Atributos e armazenamento do conjunto de dados
- Metodologia analítica de ML
- Componentes de um Pipeline de CI
- Componentes de um Pipeline de 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 IA ou através de frameworks de automação como o Amazon SageMaker, 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.

Anexo 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 4 etapas principais:

- **Engenharia de dados:** Etapa criada com o objetivo de simular processos de transformação e limpeza de dados.
- **Workflow de IA:** Etapa para execução de um Pipeline que simula o desenvolvimento de uma aplicação automatizada de IA, 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 (Componente MAPE-K):** 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 Componente MAPE-K.

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-archive> 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

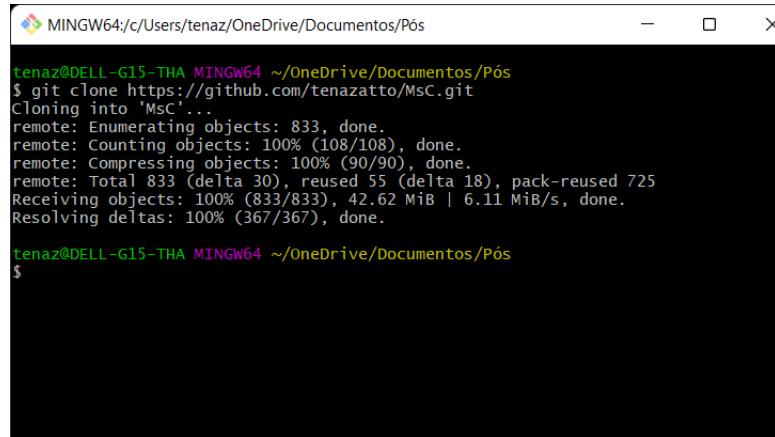
A partir desta parte, os exemplos serão realizados utilizando o Git Bash no Sistema Operacional Windows. Entretanto, no Linux e no Mac os passos são semelhantes por ambos também utilizarem esta linha de comando.

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**



The screenshot shows a terminal window titled 'MINGW64:/c/Users/tenaz/OneDrive/Documentos/Pós'. The command \$ git clone https://github.com/tenazatto/MsC.git was run, and the output shows the progress of cloning the repository. The process includes listing objects, counting objects, compressing objects, and receiving objects, all completed successfully at 100%.

```
MINGW64:/c/Users/tenaz/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.

MINGW64:/c/Users/tenaz/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
```

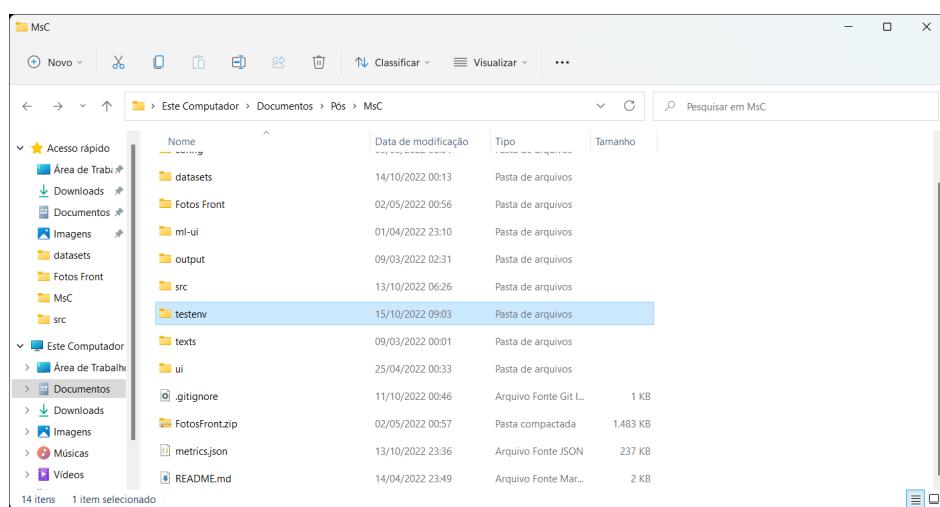
Para criar um novo ambiente virtual, é preciso digitar o comando

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

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

```
MINGW64:/c/Users/tenaz/OneDrive/Documentos/Pós/MsC
$ python3 -m venv ./testenv
$
```

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

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

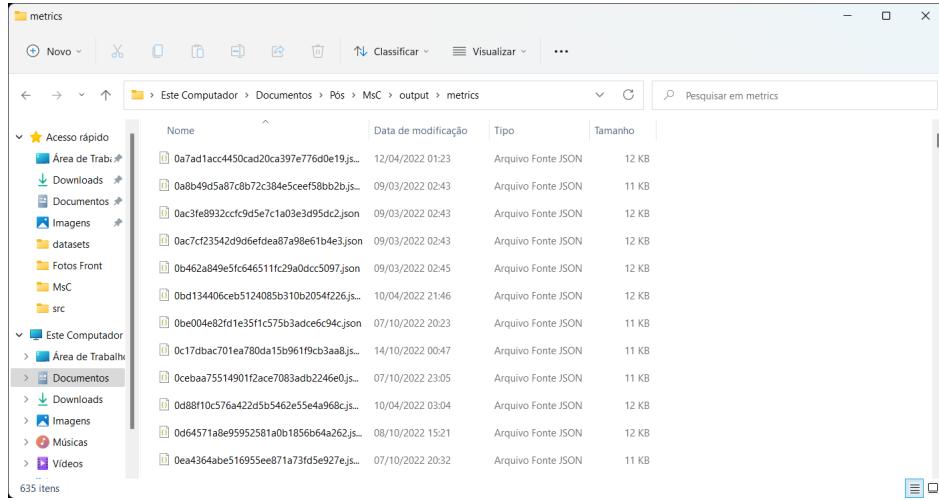
Dentro da pasta MsC e com o ambiente virtual criado e ativado, para rodar todos os Workflows possíveis basta digitar o seguinte comando

```
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.

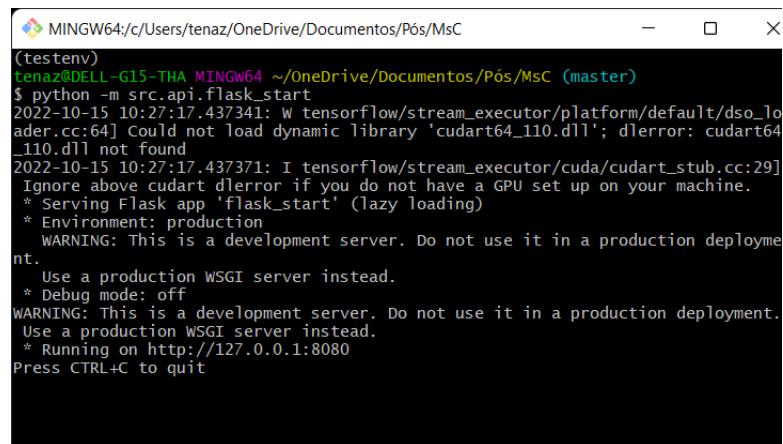
A.4.4 Interface

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



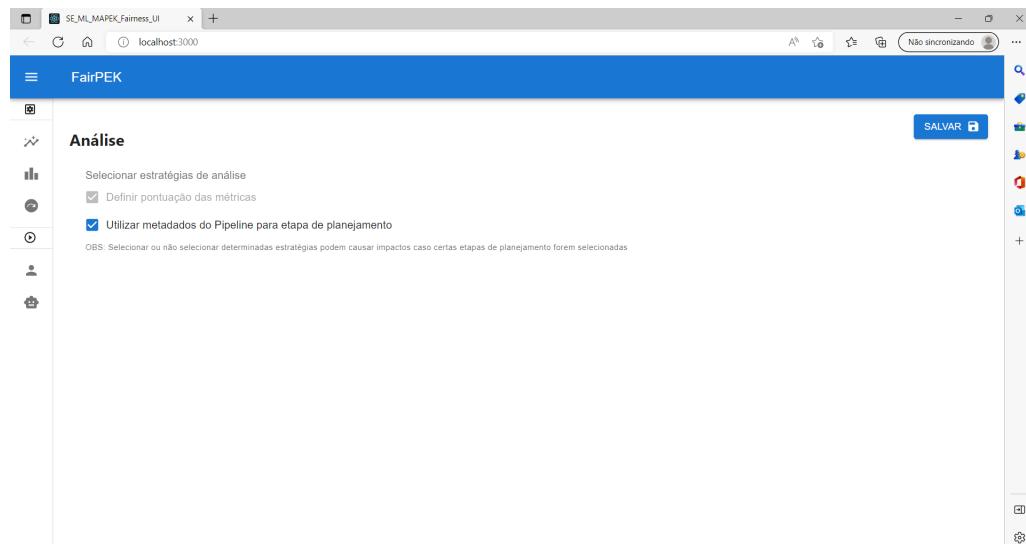
```
MINGW64:/c/Users/tenaz/OneDrive/Documentos/Pós/MsC
(testenv)
tenaz@DELL-G15-THA MINGW64 ~/OneDrive/Documentos/Pós/MsC (master)
$ python -m src.api.flask_start
2022-10-15 10:27:17.437341: W tensorflow/stream_executor/platform/default/dso_lo
ader.cc:64] Could not load dynamic library 'cudart64_110.dll'; dlsym: cudart64_
110.dll not found
2022-10-15 10:27:17.437371: I tensorflow/stream_executor/cuda/cudart_stub.cc:29]
Ignore above cudart dlsym 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 deployme
nt.
  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



Anexo 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 etapas principais:

- **Engenharia de dados:** Etapa criada com o objetivo de simular processos de transformação e limpeza de dados.
- **Workflow de IA:** Etapa para execução de um Pipeline que simula o desenvolvimento de uma aplicação automatizada de IA, 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 (Componente MAPE-K):** 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 algoritimo 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 Componente MAPE-K.

B.2 Arquitetura do Workflow e Framework

O Workflow usa uma arquitetura chamada de Pipe-and-Filter, 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 esiver 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

Pra 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`.

2. 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.
3. Importar o novo método no arquivo **data_engineering_start.py**.
4. 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])

```

5. Adicionar uma nova condição com a opção adicionada

Workflow

1. 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__()

```

2. Criar um novo arquivo.
3. 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_processors** com as novas opções colocadas no item anterior.

```

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

```

7. Na pasta **src/pipeline** e dentro do arquivo **pipeline.py**, encontrar o método **select_data_processor**, 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          ([Datasets.GERMAN_CREDIT, Preprocessors.FOREIGN], (
10             GermanDataset(), GermanForeignPreprocessor(),
11             GermanForeignFairnessPipe())),
12             ([Datasets.LENDINGCLUB, Preprocessors.INCOME], (
13                 LendingclubDataset(), LendingclubIncomePreprocessor(),
14                 LendingclubIncomeFairnessPipe())),
15             ]
16
17         for option, pipe_filter in options:
18             if choice == option:
19                 dataset_pipe, data_preprocessor_filter,
20                 fairness_pipe = pipe_filter
21                 preprocessed_data_pipe = dataset_pipe >=
22                 data_preprocessor_filter == self.new_pipe()
23                     break
24
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)

- 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
9 )))[1]

```

- 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.
9 items()))[1]

```

Interface (Frontend)

- 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>

```

- 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</
9     MenuItem>
10    <MenuItem value={'Datasets.GERMAN_CREDIT'}>German Credit Dataset
11      MenuItem>
12    <MenuItem value={'Datasets.LENDINGCLUB'}>Lendingclub Dataset</
13      MenuItem>
14  </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/
11         Feminino)</MenuItem>
12     </Select>
13   : dataset === 'Datasets.ADULT_INCOME' ?
14     <Select
15       sx={{fontSize: '14px'}}
16       value={protectedAtt}
17       onChange={handleProtectedAttChange}
18       displayEmpty
19       inputProps={{ 'aria-label': 'Without label' }}
20     >
21       <MenuItem value={'Preprocessors.AGE'}>Idade (-25 anos/+25
22         anos)</MenuItem>
23       <MenuItem value={'Preprocessors.FOREIGN'}>Nacionalidade (
24         Local/Estrangeiro)</MenuItem>
25     </Select>
26   :
27     <Select
28       sx={{fontSize: '14px'}}
29       value={protectedAtt}
30       onChange={handleProtectedAttChange}
31       displayEmpty
32       inputProps={{ 'aria-label': 'Without label' }}
33     >

```

```

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

```

4. 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

1. 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.

2. 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
3         unbias_data_algorithm == UnbiasDataAlgorithms.NOTHING and
4         unbias_postproc_algorithm == UnbiasPostProcAlgorithms.NOTHING)
5         or \

```

```

3         (...)

4             (algorithm == UnbiasInProcAlgorithms.

5             GRID_SEARCH_REDUCTION and unbias_data_algorithm ==
6             UnbiasDataAlgorithms.NOTHING and unbias_postproc_algorithm ==
7             UnbiasPostProcAlgorithms.NOTHING) or \
8                 (algorithm == Algorithms.NOVA_OPACAO and
9                  unbias_data_algorithm == UnbiasDataAlgorithms.NOTHING and
10                 unbias_postproc_algorithm == UnbiasPostProcAlgorithms.NOTHING)
11                 or \
12                     (algorithm == Algorithms.NOVA_OPACAO and
13                      unbias_data_algorithm == UnbiasDataAlgorithms.NOTHING and
14                      unbias_postproc_algorithm == UnbiasPostProcAlgorithms.NOTHING and
15                      EQUALIZED_ODDS) or \
16                         (algorithm == Algorithms.NOVA_OPACAO and
17                            unbias_data_algorithm == UnbiasDataAlgorithms.NOTHING and
18                            unbias_postproc_algorithm == UnbiasPostProcAlgorithms.NOTHING and
19                            CALIBRATED_EQUALIZED_ODDS) or \
20                               (algorithm == Algorithms.NOVA_OPACAO and
21                                 unbias_data_algorithm == UnbiasDataAlgorithms.NOTHING and
22                                 unbias_postproc_algorithm == UnbiasPostProcAlgorithms.NOTHING and
23                                 REJECT_OPTION_CLASSIFICATION) or \
24                                     (algorithm == Algorithms.NOVA_OPACAO and
25                                       unbias_data_algorithm == UnbiasDataAlgorithms.REWEIGHING and
26                                       unbias_postproc_algorithm == UnbiasPostProcAlgorithms.NOTHING)
27                                       or \
28                                         (algorithm == Algorithms.NOVA_OPACAO and
29                                           unbias_data_algorithm == UnbiasDataAlgorithms.
30                                           DISPARATE_IMPACT_REMOVER and unbias_postproc_algorithm ==
31                                           UnbiasPostProcAlgorithms.NOTHING) or \
32                                             (algorithm == Algorithms.NOVA_OPACAO and
33                                               unbias_data_algorithm == UnbiasDataAlgorithms.
34                                               OPTIMIZED_PREPROCESSING and unbias_postproc_algorithm ==
35                                               UnbiasPostProcAlgorithms.NOTHING) or \
36                                                 (algorithm == Algorithms.NOVA_OPACAO and
37                                                   unbias_data_algorithm == UnbiasDataAlgorithms.
38                                                   LEARNING_FAIR REPRESENTATIONS and unbias_postproc_algorithm ==
39                                                   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.GRADIENT_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
20     return next(filter(lambda a: a[1] == algorithm, indexes.
21 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         unbias_data_algorithm):
3             weighed_algorithm = unbias_data_algorithm ==
4             UnbiasDataAlgorithms.REWEIGHING or \
5                 unbias_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.GRADIENT_BOOST, GradientBoostFilter(weighed=
14                         weighed_algorithm)),
15                         (Algorithms.SUPPORT_VECTOR_MACHINES, SVMFilter(weighed=
16                             weighed_algorithm)),
17                             (UnbiasInProcAlgorithms.PREJUDICE_REMOVER,
18                               PrejudiceRemoverFilter()),
19                               (UnbiasInProcAlgorithms.ADVVERSARIAL_DEBIASING ,
20                                 AdversarialDebiasingFilter()),
21                                 (UnbiasInProcAlgorithms.
22                                     EXPONENTIATED_GRADIENT_REDUCTION,
23                                         ExponentiatedGradientReductionFilter(algorithm=Algorithms.
24                                             GRADIENT_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.

8. 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_OPACAO, UnbiasDataAlgorithms.NOTHING ,
3          UnbiasPostProcAlgorithms.NOTHING),
4              (Algorithms.NOVA_OPACAO, UnbiasDataAlgorithms.NOTHING ,
5                UnbiasPostProcAlgorithms.EQUALIZED_ODDS),
6                  (Algorithms.NOVA_OPACAO, UnbiasDataAlgorithms.NOTHING ,

```

```

5         UnbiasPostProcAlgorithms.CALIBRATED_EQUALIZED_ODDS),
6         (Algorithms.NOVA_OPACAO, UnbiasDataAlgorithms.NOTHING,
7          UnbiasPostProcAlgorithms.REJECT_OPTION_CLASSIFICATION),
8         (Algorithms.NOVA_OPACAO, UnbiasDataAlgorithms.REWEIGHING,
9          UnbiasPostProcAlgorithms.NOTHING),
10        (Algorithms.NOVA_OPACAO, UnbiasDataAlgorithms.
11          DISPARATE_IMPACT_REMOVER,
12            UnbiasPostProcAlgorithms.NOTHING),
13          # (Algorithms.NOVA_OPACAO, UnbiasDataAlgorithms.
14            OPTIMIZED_PREPROCESSING, UnbiasPostProcAlgorithms.NOTHING),
15          (Algorithms.NOVA_OPACAO, UnbiasDataAlgorithms.
16            LEARNING_FAIR REPRESENTATIONS,
17              UnbiasPostProcAlgorithms.NOTHING)
18        ]

```

MAPE-K

- 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.GRADIENT_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.ADVVERSARIAL_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
20     return next(filter(lambda a: a[0] == algorithm, indexes.
21 items()))[1]

```

Interface (Backend)

- 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.GRADIENT_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.ADVVERSARIAL_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
20    return next(filter(lambda a: a[0] == algorithm, indexes.
21 items()))[1]

```

Interface (Frontend)

- 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
9     Regression</MenuItem>
10    <MenuItem value={'Algorithms.RANDOM_FOREST'}>Random Forest</
11      MenuItem>
12    <MenuItem value={'Algorithms.GRADIENT_BOOST'}>Gradient Boost</
13      MenuItem>
14    <MenuItem value={'Algorithms.SUPPORT_VECTOR_MACHINES'}>Support
15      Vector Machines</MenuItem>
16  </Select>

```

- 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_OPACAO", "UnbiasDataAlgorithms.NOTHING",
3   "UnbiasPostProcAlgorithms.NOTHING"],
4   labels: ["Nome da nova opcao", "Sem metodo", "Sem metodo"],
5   selected: true
6 },

```

```
7     options: ["Algorithms.NOVA_OPÇÃO", "UnbiasDataAlgorithms.NOTHING",
8     "UnbiasPostProcAlgorithms.EQUALIZED_ODDS"],
9     labels: ["Nome da nova opção", "Sem método", "Equalized Odds"],
10    selected: true
11 },
12 {
13     options: ["Algorithms.NOVA_OPÇÃO", "UnbiasDataAlgorithms.NOTHING",
14     "UnbiasPostProcAlgorithms.CALIBRATED_EQUALIZED_ODDS"],
15     labels: ["Nome da nova opção", "Sem método", "Calibrated Equalized
16     Odds"],
17     selected: true
18 },
19 {
20     options: ["Algorithms.NOVA_OPÇÃO", "UnbiasDataAlgorithms.NOTHING",
21     "UnbiasPostProcAlgorithms.REJECT_OPTION_CLASSIFICATION"],
22     labels: ["Nome da nova opção", "Sem método", "Reject Option
23     Classification"],
24     selected: true
25 },
26 {
27     options: ["Algorithms.NOVA_OPÇÃO", "UnbiasDataAlgorithms.
28     REWEIGHING", "UnbiasPostProcAlgorithms.NOTHING"],
29     labels: ["Nome da nova opção", "Reweighting", "Sem método"],
30     selected: true
31 },
32 {
33     options: ["Algorithms.NOVA_OPÇÃO", "UnbiasDataAlgorithms.
34     DISPARATE_IMPACT_REMOVER", "UnbiasPostProcAlgorithms.NOTHING"],
35     labels: ["Nome da nova opção", "Disparate Impact Remover", "Sem
36     método"],
37     selected: true
38 },
39 {
40     options: ["Algorithms.NOVA_OPÇÃO", "UnbiasDataAlgorithms.
41     OPTIMIZED_PREPROCESSING", "UnbiasPostProcAlgorithms.NOTHING"],
42     labels: ["Nome da nova opção", "Optimized Preprocessing", "Sem
43     método"],
44     selected: true
45 },
46 {
47     options: ["Algorithms.NOVA_OPÇÃO", "UnbiasDataAlgorithms.
48     LEARNING_FAIR REPRESENTATIONS", "UnbiasPostProcAlgorithms.NOTHING"],
49     labels: ["Nome da nova opção", "Learning Fair Representations", "Sem
50     método"],
51     selected: true
52 }
```

Anexo 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.