1 Conceitos

- **Aprendizado de máquina (*Machine Learning*, ou ML):** Processo onde é extraído conhecimento de um conjunto de dados através de um algoritmo pré-estabelecido. Usando seus algoritmos padrão, pode ser classificado como supervisionado, onde se sabe suas entradas e saídas, ou não-supervisionado, onde sabe-se apenas os dados de entrada. Pode ter outras classificações quando redes neurais são utilizadas.

  No processo, 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. Ilustração na figura abaixo.

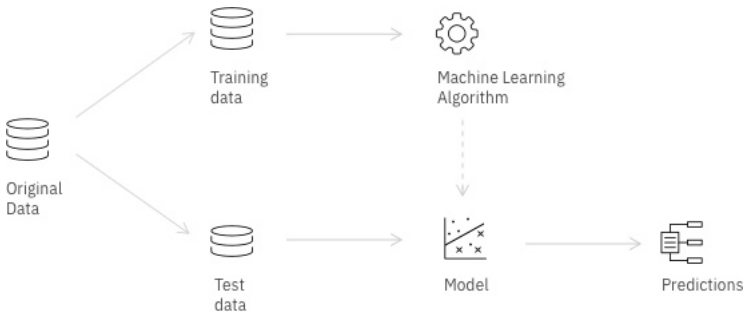

Figura 1: Processo padrão para aprendizado de máquina

- **Atributo protegido:** São características (*features*) que não podem ser usadas como base para decisões. Geralmente são considerados dados sensíveis, como raça, religião, nacionalidade, sexo, estado civil, idade e status socioeconômico.

- **Grupo privilegiado:** Agrupamento de dados relacionados a valores de atributos protegidos que sofrem discriminação e devem ser privilegiados na fase de treinamento para classificar o modelo final como justo.

- **Grupo não-privilegiado:** Agrupamento de dados relacionados a valores de atributos protegidos que não sofrem discriminação e devem ser desprivilegiados na fase de treinamento para classificar o modelo final como justo.

- **Verdadeiro positivo (TP):** Um caso em que os resultados previstos e reais estão ambos na classe positiva.

- **Falso positivo (FP):** Um caso previsto para estar na classe positiva quando o resultado real pertence à classe negativa.

- **Falso negativo (FN):** Um caso previsto para estar na classe negativa quando o resultado real pertence à classe positiva.

- **Verdadeiro negativo (TN):** Um caso em que os resultados previstos e reais estão ambos na classe negativa.

- **Regressão Logística**: Algoritmo utilizado em aprendizado de máquina para, em sua variação mais comum, realizar classificação binária de rótulos através de uma função sigmóide. Pode classificar mais de dois rótulos se utilizar a técnica *One-Vs-All* ou utilizar uma variação chamada Regressão Logística multinomial.
- **Regularização**: Conjunto de técnicas que tem como objetivo tirar ruídos presentes no processo de aprendizado que podem evitar o modelo final de ser generalizado, acarretando o que é chamado de *Overfitting*
- **AI Fairness 360**: Biblioteca para a linguagem Python criada pela IBM que cataloga diversas técnicas de aprendizado de máquina baseados em métricas de Fairness para serem utilizadas. Também encapsula conjuntos de dados com seus grupos protegidos e não-protegidos
- **Black box**: Forma de implementação de um modelo de ML onde o importante para determinação da qualidade de um modelo é sua entrada e sua saída apenas.

2 Métricas

2.1 Métricas primitivas

- **Precisão, ou valor preditivo positivo (PPV):** É a fração de casos positivos previstos corretamente para estar na classe positiva de todos os casos positivos previstos no modelo. Ou seja, $PPV = \frac{TP}{TP+FP}$
- **Taxa de descoberta falsa (FDR):** É a fração de casos negativos previstos incorretamente como estando na classe positiva de todos os casos positivos previstos no modelo. Ou seja, $FDR = \frac{FP}{TP+FP}$
- **Taxa de falsa omissão (FOR):** É a fração de casos positivos previstos incorretamente como estando na classe negativa de todos os casos negativos previstos no modelo. Ou seja, $FOR = \frac{FN}{TN+FN}$
- **Valor preditivo negativo (NPV):** É a fração de casos negativos previstos corretamente para estar na classe negativa de todos os casos negativos previstos no modelo. Ou seja, $NPV = \frac{TN}{TN+FN}$
- **Sensibilidade ou *recall*, ou taxa positiva verdadeira (TPR):** É a fração de casos positivos previstos corretamente para estar na classe positiva de todos os casos positivos reais. Ou seja, $TPR = \frac{TP}{TP+FN}$
- **Taxa de falsos positivos (FPR):** É a fração de casos negativos previstos incorretamente como estando na classe positiva de todos os casos negativos reais. Ou seja, $FPR = \frac{FP}{FP+TN}$
- **Taxa de falsos negativos (FNR):** É a fração de casos positivos previstos incorretamente como estando na classe negativa de todos os casos positivos reais. Ou seja, $FNR = \frac{FN}{TP+FN}$

- **Taxa negativa verdadeira (TNR):** É a fração de casos negativos previstos corretamente para estar na classe negativa de todos os casos negativos reais. Ou seja, $TNR = \frac{TP}{FP+TN}$

2.2 Métricas compostas

- **Diferença de paridade estatística, ou discriminação [17]:** Esta métrica é baseada na seguinte fórmula:

$$Pr(Y = 1|D = nao - privilegiado) - Pr(Y = 1|D = privilegiado)$$

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:** É a diferença entre a taxa positiva verdadeira do grupo não privilegiado e a taxa positiva verdadeira do grupo privilegiado:

$$TPR_{D=nao-privilegiado} - TPR_{D=privilegiado}$$

Um valor próximo de 0 é considerado justo, assim como na métrica anterior.

- **Diferença de probabilidade média:** 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:

$$\frac{1}{2}(|FPR_{D=nao-privilegiado} - FPR_{D=privilegiado}| + |TPR_{D=nao-privilegiado} - TPR_{D=privilegiado}|)$$

Também precisa ser próximo a 0 para ser considerado justo.

- **Impacto de disparidade:** Para esta métrica, é usada a seguinte fórmula:

$$\frac{Pr(Y = 1|D = nao - privilegiado)}{Pr(Y = 1|D = privilegiado)}$$

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.

- **Indíce de Theil:** Esta medida também é conhecida como índice de entropia generalizado, mas com α igual a 1 [16]. É calculado com a seguinte fórmula:

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

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

3 Algoritmos

3.1 Pré-processamento

- **Reposição (Reweighing) [10]:** 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) [8]:** 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) [17]:** Encontra uma representação latente que codifica os dados, mas ofusca informações dos atributos protegidos.
- **Pré-processamento otimizado [6]:** 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.

3.2 Processamento

- **Remoção de viés adversário (Adversarial debiasing) [18]:** 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) [8]:** 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) [7]:** 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) [12]:** 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) [4]:** 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) [4] [5]:** 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.

3.3 Pós-processamento

- **Igualdade de probabilidade calibrada (Calibrated Equalized odds) [15]:** 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) [9]:** 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) [11]:** Dá resultados favoráveis para grupos não privilegiados e resultados desfavoráveis para grupos privilegiados de acordo com uma faixa de confiança.

4 Problema

4.1 Motivação e Solução

- **Motivação 1:** Analisar um modelo justo não é uma tarefa simples, uma vez que as métricas utilizadas para determinar imparcialidade possuem baixa correlação com métricas utilizadas para determinar um modelo de qualidade. Ela pode aumentar ou diminuir essas métricas.(Qual o balanceamento adequado entre as métricas padrão e métricas de Fairness?)
- **Motivação 2:** Modelos são obtidos através de uma abordagem *Black box* e precisam de metadados para determinar alguma explicabilidade em seu processo.(O modelo é justo por qual razão? Por quê a imparcialidade não afeta sua qualidade?)
- **Motivação 3:** Embora os algoritmos para se obter modelos de ML acurados já estão estabelecidos, o mesmo não se pode dizer dos algoritmos para se obter modelos de ML justos. E, uma vez que eles podem atuar em diversas partes do processo, a complexidade de se medir o processo aumenta.(Como escolher a gama correta de algoritmos de forma a estabelecer um processo coeso e efetuar a medição para a escolha do melhor modelo?)
- **Solução 1:** O modelo MAPE-K pode ajudar em definir uma estratégia para balancear as métricas (Motivação 1) e escolher os modelos mais adequados (Motivação 3).
- **Solução 2:** Metadados são obtidos durante o processo(Motivação 2), e ajudam a alimentar o modelo MAPE-K como base de conhecimento (Motivações 1 e 3).
- **Solução 3:** Desenvolvimento de um *pipeline* de ML componentizado, para obter coesão nas abstrações de seu processo e facilitar sua expansão/manutenção através de reuso de código (Motivação 3).

4.2 Experimento e Restrições

- **Experimento:** Execução de um *pipeline* de ML para obtenção de dados iniciais na base de conhecimento e discussão de um *pipeline* de ML utilizando MAPE-K como facilitador da escolha de modelo.

- **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 algortimos em duas ou em todas as três etapas impacta no desempenho.

- **Restrição 2:** Por não conseguirem rodar com sucesso ou apresentarem métricas que destoam dos outros, o que implica em alguma falha não detectada na implementação, foram retirados os seguintes algoritmos: Pré-processamento otimizado e Classificação baseada em Rejeição de Opções

- **Restrição 3:** A base de conhecimento será capaz de explicar a imparcialidade de um modelo, mas não terá a explicabilidade da influência de cada *feature* aplicada no modelo.

4.3 Case Study

4.4 Problem Statement

Modelo Alg Pré-proc In-Proc Pós-Proc Metadados Métricas avaliação Fairness

Fazer Claims Ver Entradas e Saídas Rever Evidências Pensar em uma mentalidade de Requisitos ("Certificação")

"Rodar a aplicação"(Entradas, saídas, etapas) FAZER ESSA PORRA e O BAGULHO DO ALTIGRAN

4.5 GSN

4.6 Arquitetura

4.7 Exemplo de descrição

- **Objetivo:** Obter classificação de crédito (boa ou ruim), através de uma série de *features*

- **Conjunto de dados:** German Credit Dataset, presente em [3].

- **Atributo protegido:** Idade

- **Grupo privilegiado:** Idade maior ou igual a 25 anos

- **Grupo não-privilegiado:** Idade menor que 25 anos

5 Papeis de pessoas em ML

5.1 Diagrama de atividades

Ver Figura 3 e Figura 4.

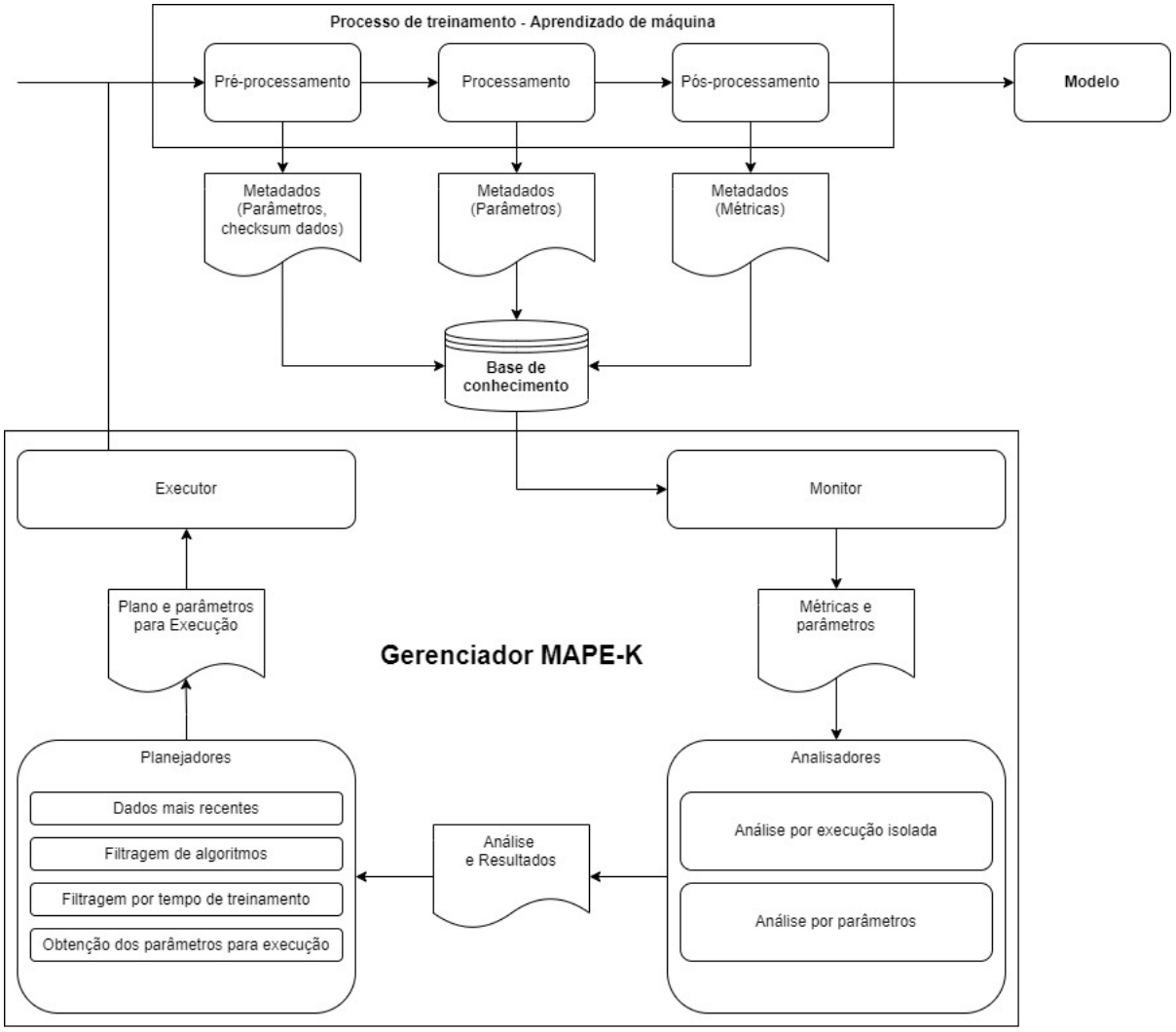

Figura 2: Arquitetura de um gerenciador MAPE-K "plugado"a um *pipeline* de ML

5.2 Cargos

PROCURAR PAPERS PROF. ALTIGRAN PARA REFERÊNCIA

- **Engenheiro de Dados:** Responsável pela coleta, estruturação, organização, consolidação/integração dos dados em *Data Lakes/Data Warehouses* e processos de governança.
- **Cientista de Dados:** Responsável pela análise de dados coletados e estruturados pelo Engenheiro de Dados e criação de processos de ML.
- **Engenheiro de Software:** Responsável pelo uso do modelo de ML gerado pelo cientista de dados e desenvolvimento de aplicações para uso geral.
- **Cargos Auxiliares:** É possível que outros cargos auxiliem o desenvolvimento destes, como *Product Owners* para entenderem os usos do cliente final e Advogados para determinarem boas práticas aos processos de governança.

5.3 Conteúdo

- **Ambiguidade entre os papeis de Ciência de Dados - Acadêmico:** `http://www.ijsrd.com/articles/IJSRDV8I90019.pdf`
- **Papel do Cientista de Dados - Acadêmico:** `https://dl.acm.org/doi/pdf/10.1145/2884781.2884783`
- **Perfis de Cientista de Dados - Acadêmico:** `http://www.ijsrd.com/articles/IJSRDV8I90019.pdf`
- **Diferença entre Cientista, Analista e Engenheiro de Dados - Miscelâneo:** `https://www.geeksforgeeks.org/what-are-the-roles-and-responsibilities-of-a-dat`

6 Arquitetura Pipes and Filters em ML

6.1 Pipes e Filtros

- **Classificação e sequência de Filtros:** Pré-Processamento > Processamento > Pós-Processamento
- **Entrada do Pipe para pré-processamento:** Conjunto de dados original
- **Filtros para pré-processamento:** Definições de atributos protegidos, Definições de grupos privilegiados e não-privilegiados, Engenharia de features, Divisão de conjuntos, técnicas de aumento/diminuição de dados (oversampling/undersampling) e algoritmos para retirada de vieses no dado.
- **Saída do Pipe para pré-processamento:** Conjunto de dados com tratamentos feitos na fase de pré-processamento e divididos para treinamento, validação (recomendável) e testes
- **Entrada do Pipe para processamento:** Conjunto de dados com tratamentos feitos na fase de pré-processamento e divididos para treinamento
- **Filtros para processamento:** Algoritmos para treinamento, independentemente de ter retirada de vieses ou não.
- **Saída do Pipe para processamento:** Modelos obtidos pós-treinamento
- **Entrada do pipe para pós-processamento:** Conjunto de dados com tratamentos feitos na fase de pré-processamento e divididos para testes, e modelos obtidos pós-treinamento
- **Filtros para pós-processamento:** Algoritmos para retirada de vieses nas predições, cálculo de métricas e escolha de modelo mais adequado.
- **Saída do pipe para pós-processamento:** Métricas utilizadas e modelo mais otimizado de acordo com elas.

6.2 Fluent API

É possível abstrair tal arquitetura através do conceito de Fluent Interface [1] (também conhecido como Fluent API [14]). Dessa forma, é possível encadear diversos processos que dependem uns dos outros em uma sintaxe mais intuitiva e que procura dar mais clareza ao fluxo dos dados.

Nele, os componentes de filtro representam as operações a modificar e/ou processar os dados, e os componentes de pipes representam os dados presentes no fluxo como um todo. Com ela, é possível com poucas linhas de código transformar um pipe através de um filtro passando componentes como entrada:

```
    initial_pipe.to_filter(Filter1())
                .to_pipe(Pipe())
                .to_filter(Filter2())
                .to_pipe(Pipe())
```

Trecho 1: Manipulações de pipes em filtros

É possível também criar novos pipes encadeando dados de outros pipes e obtendo parte dos mesmos:

```
    initial_pipe.merge(another_pipe)
                .partial_pipe('prop1', 'prop2')
```

Trecho 2: Conversões de pipes em outros pipes

Com o uso de métodos especiais presente no Python, é possível mudar alguns operadores para estes objetos com o objetivo de deixar a sintaxe ainda mais simples:

```
# Sintaxe equivalente ao encadeamento de pipes com filtros
    initial_pipe >= Filter1() == Pipe() >= Filter2() == Pipe()

# Sintaxes equivalentes ao encadeamento de pipes com outros
# e obtencao de pipes parciais
    (initial_pipe + another_pipe)['prop1', 'prop2']
    initial_pipe['prop1'] + another_pipe['prop2']
```

Trecho 3: Facilitando a sintaxe através de métodos especiais

É possível definir filtros não apenas para processar dados e uso de algoritmos de ML, mas também processos de visualização dos dados para simplificar a análise, como gráficos ou definir DSLs para obtenção dos dados.

6.3 Conteúdo

- **Pipes and Filters - Acadêmico:** `https://www.researchgate.net/publication/221034471_The_Pipes_and_Filters_Pattern_A_Functional_Parallelism_Architectural_Pattern_for_Parallel_Programming`
- **Pipes and Filters - Miscelâneo:** `https://www.linkedin.com/pulse/software-architect`
- **Pipes and Filters - Miscelâneo:** `https://docs.microsoft.com/en-us/azure/architecture/patterns/pipes-and-filters`
- **Arquitetando um Pipeline de ML:** `https://towardsdatascience.com/architecting-a-`

- **Estrutura de um Pipeline de ML (Pipes and Filters/Composite/DAGs):** `https://www.neuraxio.com/blogs/news/how-to-code-neat-machine-learning-pipeline`
- **Técnicas para aumentar/diminuir dados - Miscelâneo:** `https://dl.acm.org/doi/pdf/10.1145/2884781.2884783`

7 Ataques e segurança

O *Berryville Institute of Machine Learning* classifica 78 riscos de segurança que uma aplicação baseada em Machine Learning é capaz de sofrer [13]. Os 10 principais são:

- **Exemplos adversários:** São os ataques mais comuns a serem discutidos. A ideia é enganar uma aplicação de Machine Learning fornecendo dados maliciosos, muitas vezes envolvendo pequenas distorções que fazem com que o sistema faça uma previsão ou categorização falsa, por mais que os dados cubram boa parte das situações.
- **Envenenamento de dados:** Como um aplicação de Machine Learning aprende diretamente a partir dos dados que possui, os dados desempenham um papel desproporcional na segurança. Se um invasor puder manipular intencionalmente os dados usados de maneira coordenada, todo o modelo pode ser comprometido. Deve-se considerar que fração dos dados de treinamento um invasor pode controlar e até que ponto.
- **Manipulação do sistema online:** Uma aplicação de Machine Learning está "online" quando continua a aprender durante o uso operacional, modificando seu comportamento ao longo do tempo. Nesse caso, um invasor inteligente pode empurrar propositalmente o sistema que ainda está aprendendo na direção errada por meio da entrada do sistema e, lentamente, "retreinar" a aplicação para fazer a coisa errada. Esse tipo de ataque pode ser sutil e razoavelmente fácil de executar. Esse risco é complexo, exigindo considerar a proveniência dos dados, a escolha do algoritmo e as operações do sistema.
- **Ataque de *Transfer Learning*:** Em muitos casos no mundo real, aplicações de Machine Learning são construídas tirando proveito de um modelo padrão já treinado que é então ajustado para realizar uma tarefa mais específica. Um ataque de transferência de dados ocorre quando tal modelo padrão está comprometido, ou inadequado, tornando possível um comportamento imprevisto definido pelo invasor.
- **Confidencialidade dos dados:** Há um grande desafio em proteger dados sensíveis ou confidenciais que, por meio de treinamento, são integrados a um modelo. Ataques de extração sutis desses dados são um risco a ser considerado.
- **Confiabilidade dos dados:** Considerar a procedência e a integridade dos dados é essencial. Os dados precisam ser adequados e de qualidade, os sensores confiáveis, e a integridade dos dados deve ser preservada. Compreender a natureza das fontes de dados da aplicação (durante o treinamento e a execução) é de extrema importância. Os riscos de transmissão de dados são particularmente complexos quando se trata de fontes de dados públicas (que podem ser manipuladas ou envenenadas) e modelos online.

- **Reprodutibilidade:** Infelizmente, por causa de sua natureza impenetrável e do crescimento rápido do campo, os resultados das aplicações de Machine Learning são frequentemente sub-relatados, mal descritos e, de outra forma, impossíveis de reproduzir. Isso torna o futuro de sua manutenção incerto.

- ***Overfitting*:** Quando uma aplicação de Machine Learning "decora"seu conjunto de dados de treinamento, ele não generaliza para novos dados e é considerado *overfitting*. Tais modelos são particularmente fáceis de atacar. O *Overfitting* também é possível em conjunto com a manipulação do sistema online e pode acontecer enquanto um sistema está em execução.

- **Integridade da codificação dos dados:** Os dados são frequentemente codificados, filtrados, re-representados e processados de outra forma antes de serem usados (na maioria dos casos por um grupo de pessoas). Problemas de integridade de codificação podem distorcer um modelo de diferentes maneiras. Por exemplo, as codificações que incluem metadados podem permitir que um modelo "resolva"um problema de categorização enfatizando demais os metadados e ignorando o problema real de categorização, podendo acontecer algum tipo de ruído.

- **Integridade dos dados de saída:** Se um invasor puder se interpor entre uma aplicação de Machine Learning e seu uso no mundo real, um ataque direto na saída pode ser possível. A natureza impenetrável do modelo final (ou seja, não entender realmente como fazem o que fazem) pode tornar um ataque de integridade de saída muito mais fácil, pois uma anomalia pode ser mais difícil de detectar.

7.1 Possíveis soluções

- O AIF360 possui suporte ao ART (`https://github.com/Trusted-AI/adversarial-robustne` onde é possível simular ataques

- Uma outra alternativa é o `https://github.com/google-research/robustness_metrics`

7.2 Conteúdo

- **High Quality Machine Learning - Série de 3 artigos:**
  - `https://medium.com/codex/high-quality-machine-learning-part-1-d45259ee7613`
  - `https://medium.com/codex/high-quality-machine-learning-part-2-2643c144a2a7`
  - `https://medium.com/codex/high-quality-machine-learning-part-3-c660b885233c`

- **Adversarial ML Threat Matrix:** `https://github.com/mitre/advmlthreatmatrix`

- **Hands-On Guide To Adversarial Robustness Toolbox (ART):** `https://analyticsindi com/adversarial-robustness-toolbox-art/`

Referências

[1] Fluentinterface. URL https://martinfowler.com/bliki/FluentInterface.html.

[2] IBM - Analytics and AI architecture. URL https://www.ibm.com/cloud/architecture/architectures/aiAnalyticsArchitecture/reference-architecture/.

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

[4] 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.

[5] 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.

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

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] Gary McGraw, Harold Figueroa, Victor Shepardson, and Richie Bonett. An architectural risk analysis of machine learning systems: Toward more secure machine learning. URL `https://berryvilleiml.com/docs/ara.pdf`.

[14] 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`.

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

[16] 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`.

[17] 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`.

[18] 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`.

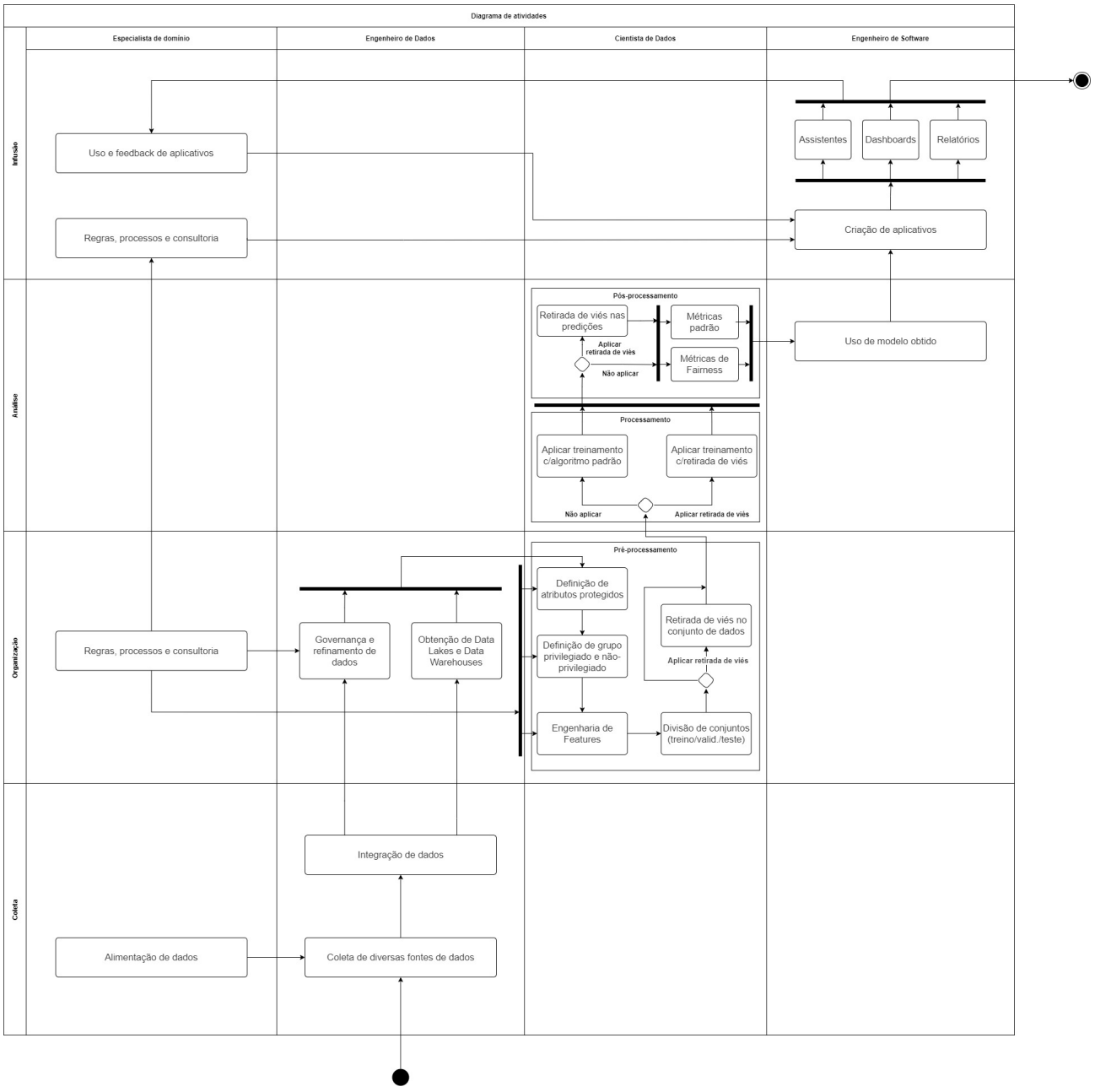

Figura 3: Diagrama de atividades com subdivisão de cada papel em uma aplicação de IA utilizando métricas de Fairness, com base na IBM AI Reference Architecture [2]

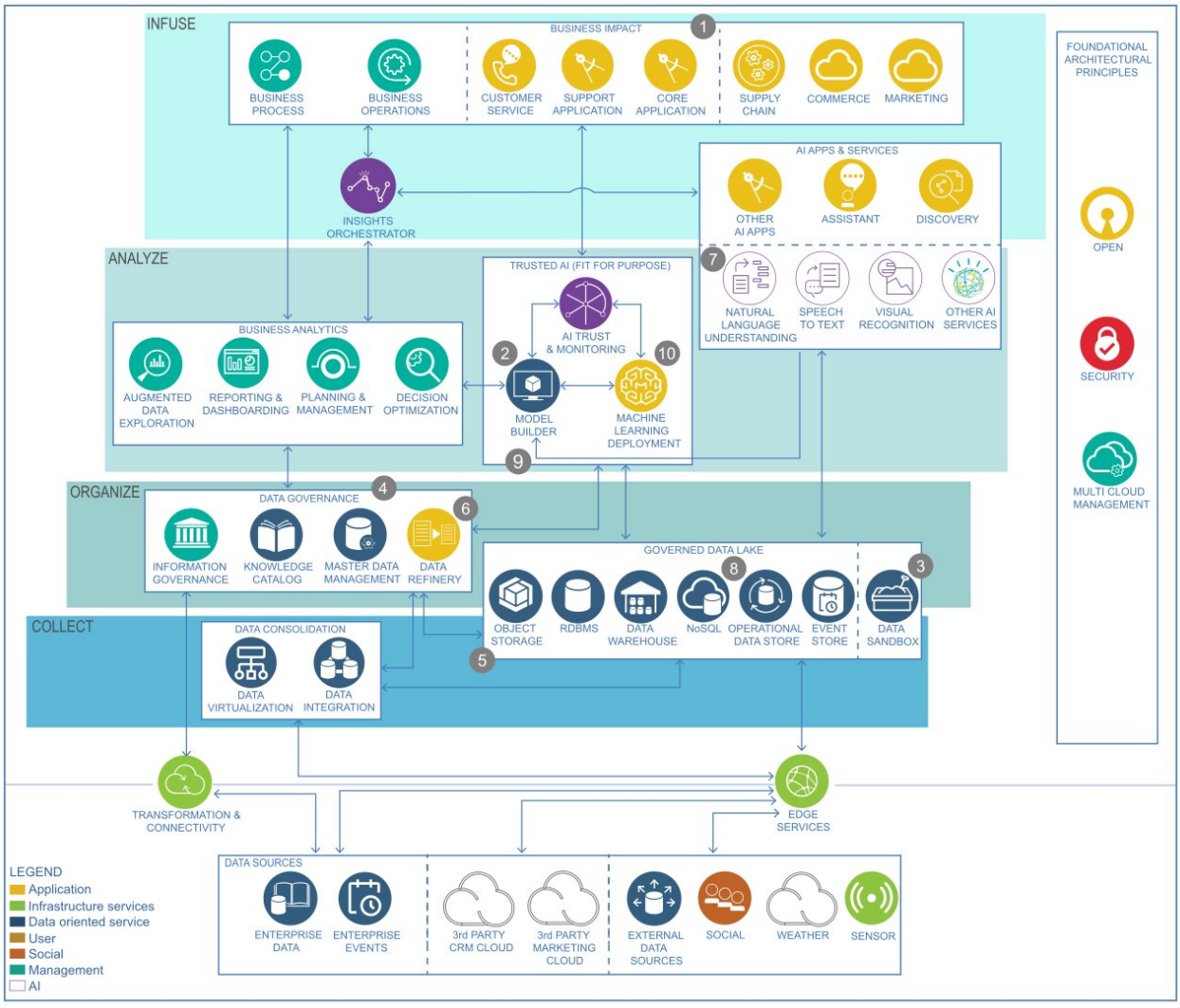

Figura 4: IBM Analytics and AI Reference Architecture.