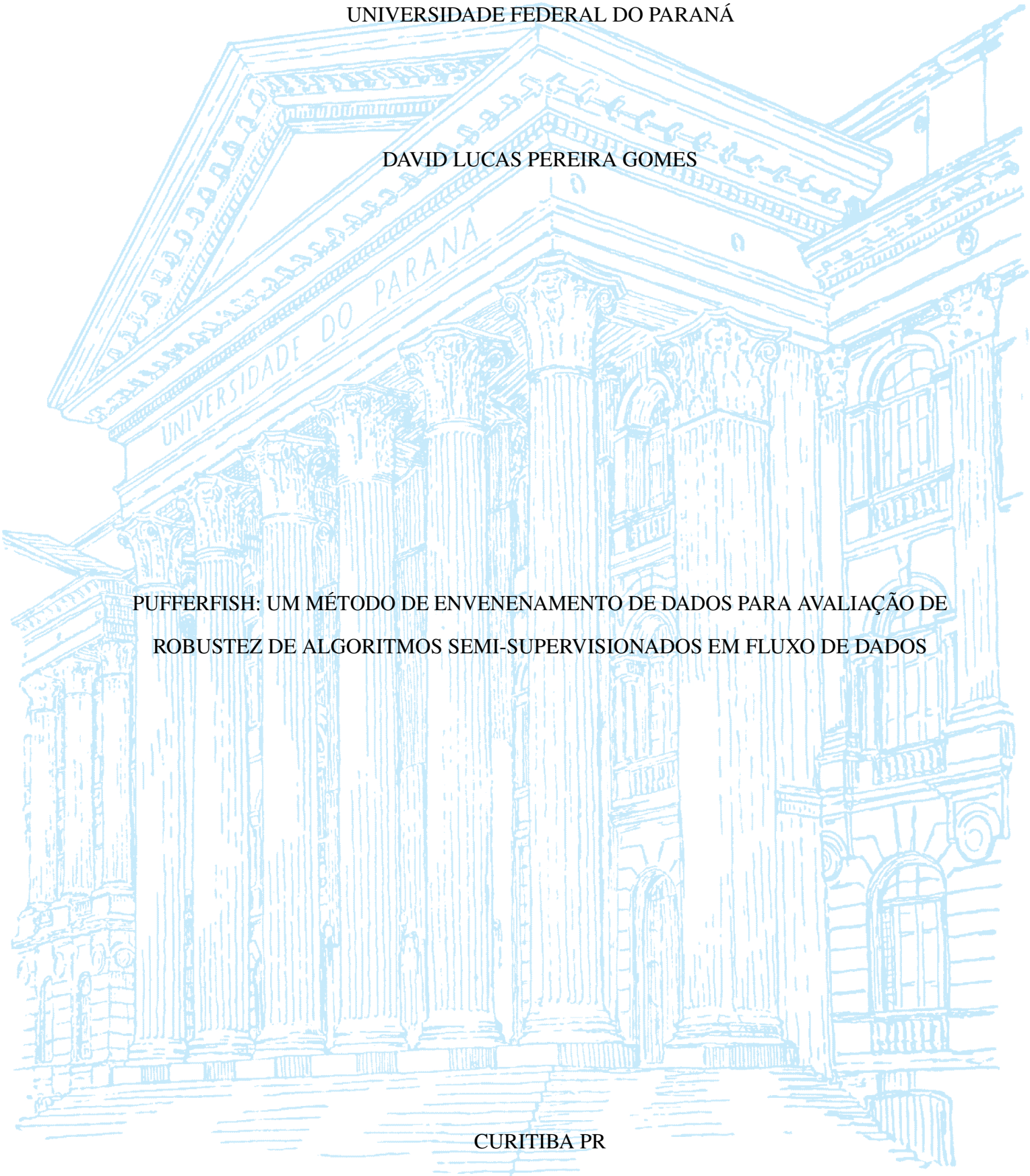
UNIVERSIDADE FEDERAL DO PARANÁ

DAVID LUCAS PEREIRA GOMES




# PUFFERFISH: UM MÉTODO DE ENVENENAMENTO DE DADOS PARA AVALIAÇÃO DE ROBUSTEZ DE ALGORITMOS SEMI-SUPERVISIONADOS EM FLUXO DE DADOS

CURITIBA PR

2025

DAVID LUCAS PEREIRA GOMES

# PUFFERFISH: UM MÉTODO DE ENVENENAMENTO DE DADOS PARA AVALIAÇÃO DE ROBUSTEZ DE ALGORITMOS SEMI-SUPERVISIONADOS EM FLUXO DE DADOS

Trabalho apresentado como requisito parcial à conclusão do Curso de Bacharelado em Ciência da Computação, Setor de Ciências Exatas, da Universidade Federal do Paraná.

Área de concentração: *Computação*.

Orientador: Paulo Ricardo Lisboa de Almeida.

Coorientador: André Ricardo Abed Grégio.

CURITIBA PR

2025

## AGRADECIMENTOS

Agradeço, primeiramente, aos meus orientadores, Prof. Dr. Paulo Ricardo Lisboa de Almeida e Prof. Dr. André Ricardo Abed Grégio, pela paciência e ensinamentos durante toda a realização deste trabalho. Também, agradeço a minha família e amigos pelo apoio incondicional e incentivo constante ao longo desta jornada acadêmica.

## RESUMO

Em cenários com grande volume de dados e geração contínua de amostras, algoritmos semi-supervisionados *online* são especialmente adequados, visto que utilizam tanto instâncias rotuladas quanto não rotuladas, além de, potencialmente, adequarem-se a mudanças de conceito. Entretanto, o uso de instâncias não rotuladas em treinamento traz possíveis vulnerabilidades a ataques por agentes mal intencionados, cujo objetivo visa manipular a fronteira de decisão de um modelo de forma que classifique erroneamente uma ou mais amostras. Desta forma, é necessária a avaliação de robustez de modelos semi-supervisionados *online* contra estes tipos de ataque, que ocorrem a partir de instâncias não rotuladas. Sendo assim, o presente trabalho propõe o método *Pufferfish*, cujo objetivo é avaliar a vulnerabilidade deste tipo de algoritmo. O método inclui: (i) um gerador sintético de ataques em fluxo de dados que desloca gradualmente instâncias não rotuladas, e (ii) um protocolo de avaliação por *holdout* periódico utilizando um segundo gerador de conjunto de dados. Experimentos foram feitos com cinco configurações de modelos, que utilizam algoritmos de *self-training*, *cluster-and-label* e *co-op training*. Avaliou-se alterações em suas fronteiras de decisão ao longo do treinamento, além de acurácia e taxa de falsos negativos nas predições no conjunto de avaliação, que quantifica a efetividade do ataque. Nesse contexto, identificou-se configurações robustas, vulneráveis e uma intermediária que recupera parcialmente o desempenho ao final do fluxo.



Palavras-chave: Aprendizado Semi-supervisionado. Fluxo de dados. Envenenamento de dados.

## ABSTRACT

In scenarios with a large amount of data and continuously arriving samples, online semi-supervised algorithms are well suited. They use both labeled and unlabeled instances and can adapt to changes in the data over time. However, using unlabeled instances during training can create vulnerabilities to attacks by malicious agents, whose goal is to push the model's decision boundary so it misclassifies one or more samples. Therefore, it is necessary to assess the robustness of online semi-supervised models against these attacks that act through unlabeled instances. This work presents the Pufferfish method, whose goal is to evaluate the vulnerability of such algorithms. The method includes: (i) a synthetic data-stream attack generator that gradually shifts unlabeled instances, and (ii) a periodic holdout evaluation protocol using a second data generator. We run experiments with five model configurations that use self-training, cluster-and-label, and co-op training. We analyze changes in their decision boundaries during training and report accuracy and the false negative rate on the evaluation set, which shows how effective the attack is. In this setting, we identify robust configurations, vulnerable ones, and an intermediate configuration that partially recovers its performance by the end of the stream.



Keywords: Semi-supervised Learning. Data Stream. Data Poisoning.

## LISTA DE FIGURAS

## LISTA DE TABELAS

## LISTA DE ACRÔNIMOS

| | |
|---|---|
| IoT | *Internet of Things* |
| MOA | *Massive Online Analysis* |
| MOA-SS | *Massive Online Analysis - Semi-supervised* |
| SSL | *Semi-supervised Learning* |
| OGD | *Orthogonal Gradient Descent* |
| MDP | *Markov Decision Process* |
| FNR | *False Negative Rate* |

## LISTA DE SÍMBOLOS

| | |
|---|---|
| $N$ | Número de instâncias no conjunto de dados |
| $\min(x_i)$ | Valor mínimo permitido para as características $i$ |
| $\max(x_i)$ | Valor máximo permitido para as características $i$ |
| $m$ | Margem de separação entre classes, distância mínima entre instâncias e a fronteira de decisão |
| $\alpha$ | Proporção inicial de instâncias rotuladas no conjunto de dados de envenenamento |
| $\gamma$ | Proporção de instâncias não rotuladas na fase de ataque do conjunto de dados com envenenamento |
| $\beta$ | Proporção de instâncias não rotuladas geradas no *cluster* de ataque |
| $\lambda$ | Fator de espalhamento das instâncias de ataque a partir da média |
| $\rho$ | Fator de controle de deslocamento máximo da média das instâncias de ataque |
| $s$ | Semente do gerador pseudo-aleatório |

**SUMÁRIO**

# 1 INTRODUÇÃO

Algoritmos de aprendizado de máquina *online* destacam-se por aprenderem uma instância de cada vez, em contraste com algoritmos tradicionais *offline*, que processam um conjunto de dados fixo (Hoi et al., 2021). Essa característica confere aos algoritmos *online* um papel fundamental em aplicações do mundo real que envolvem grande volume de dados, como transações bancárias e detecção de intrusão (Gomes et al., 2019). Nesse contexto, o alto custo de rotulação de instâncias torna-se um gargalo para algoritmos supervisionados (Fredriksson et al., 2020; Roh et al., 2021), demonstrando a importância de algoritmos semi-supervisionados, capazes de aproveitar tanto amostras rotuladas quanto não rotuladas.

Entretanto, o processamento de instâncias não rotuladas no treinamento introduz vulnerabilidades a envenenamento de dados. O objetivo desse ataque consiste em manipular a fronteira de decisão do modelo para induzir a predições errôneas, que, em sistemas críticos, pode resultar em falhas de segurança severas e prejuízos financeiros. Desta forma, torna-se necessária a avaliação da robustez de algoritmos semi-supervisionados, especialmente contra ataques por instâncias não rotuladas, que têm se mostrado altamente efetivos em estudos recentes.

Embora existam métodos e *frameworks* para envenenamento de dados de modelos semi-supervisionados *offline* (Carlini, 2021; Liu et al., 2019, 2022) e para modelos *online* supervisionados (Wang e Chaudhuri, 2018; Zhang et al., 2020), não se encontra na literatura métodos focados especialmente em algoritmos semi-supervisionados *online*, constituindo a principal lacuna no estado da arte que este trabalho visa preencher.

Desta forma, este trabalho propõe um método para análise de robustez de modelos de algoritmos *online* semi-supervisionados contra envenenamento de dados, focando em ataques por instâncias não rotuladas. Cria-se um conjunto de dados envenenado para treinamento dos modelos, assim como um conjunto de dados de grade para medir o impacto do ataque na fronteira de decisão dos modelos, e outro conjunto de dados para avaliação com o objetivo de mensurar a evolução de suas métricas durante o treinamento. Assim, o método pode encontrar suscetibilidade a envenenamento de dados antes que algum atacante a encontre e explore, evitando custos.

Os resultados obtidos mostram uma configuração de *self-training* (BDA) robusta ao ataque, o algoritmo *co-op training* com desempenho intermediário, e as configurações de *self-training* IPA e IDW, além do modelo *cluster-and-label*, demonstrando alta vulnerabilidade e suscetibilidade ao envenenamento de dados. Sendo assim, as contribuições deste trabalho são compostas por:

- Um método de análise de robustez de algoritmos semi-supervisionados *online* contra envenenamento de dados por instâncias não rotuladas.
- Avaliação da evolução da fronteira de decisão, acurácia e taxa de falsos negativos de modelos da literatura ao longo do envenenamento.

O restante deste documento está organizado da seguinte forma: o Capítulo 2 apresenta os conceitos necessários para compreender o método proposto neste trabalho. O Capítulo 3 explora a literatura e estado da arte sobre o presente tema. O Capítulo 4 detalha a geração dos conjuntos de dados utilizados pelo método proposto, assim como seu uso em algoritmos *online* semi-supervisionados. O Capítulo 5 apresenta o protocolo experimental, detalhando os modelos utilizados nos experimentos e seu processo de treinamento e avaliação. O Capítulo 6 apresenta os resultados dos experimentos, demonstrando visualmente a evolução das fronteira de decisão

dos modelos, suas acurácias e taxas de falsos negativos. Finalmente, o Capítulo 7 conclui este trabalho, trazendo os resultados e ponderações sobre os modelos avaliados.

# 2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo introduz os conceitos necessários para compreender a proposta deste trabalho. Inicialmente, serão introduzidos os conceitos de fluxo de dados, aprendizado semi-supervisionado *online* e envenenamento de dados. Também, será explicado como estes assuntos conectam-se na literatura.

## 2.1 FLUXO DE DADOS

O paradigma tradicional de aprendizado de máquina é conhecido como aprendizado em *batch* (ou *offline*), que necessita que todo o conjunto de dados esteja disponível antes do treinamento do modelo (Hoi et al., 2021). Nesse cenário, o modelo é treinado uma única vez com o conjunto completo de dados e, posteriormente, utilizado para fazer previsões em novas amostras. Em contraste, fluxos de dados são utilizados em aprendizado de máquina *online*, nos quais dados chegam em sequência e um modelo é atualizado a cada nova amostra rotulada (Hoi et al., 2021). Considera-se, também, que os dados chegam em altas taxas e que o fluxo é infinito, desta forma, cada instância é vista apenas uma vez pelo algoritmo. Assim, o aprendizado em fluxos de dados apresenta vantagens para cenários em que os dados são gerados continuamente, como em sistemas de monitoramento, transações bancárias, redes sociais e dispositivos IoT (Gama, 2010), apresentando algoritmos escaláveis para lidar com grandes volumes de dados.

Outro conceito importante no treinamento e avaliação dos modelos *online* é a estratégia *test-then-train* (também conhecida como avaliação *prequential*), na qual novas instâncias são usadas primeiramente para teste do modelo e, posteriormente, para seu treino, caso seus rótulos estejam disponíveis (Gama et al., 2009). Este método é mais utilizado em relação a outros tipos de avaliação em fluxos de dados, como a avaliação em *holdout* periódico, no qual um algoritmo é testado em intervalos fixos em um determinado conjunto de instâncias (Haug et al., 2022). Entre as avaliações periódicas, ocorre apenas o treinamento do modelo, e, como pode haver mudança de conceito, o conjunto de avaliação *holdout* deve ser atualizado conforme o progresso do fluxo.

Um algoritmo supervisionado *online* comum e utilizado como modelo base em alguns algoritmos semi-supervisionados é a árvore de Hoeffding (*Hoeffding Trees*). Este algoritmo consiste em um modelo de árvores de decisão para fluxos de dados, no qual o limite de Hoeffding é usado para decidir quando criar novas divisões (Domingos e Hulten, 2000; Kumar et al., 2015). Visto que foi criado considerando fluxos, apresenta tempo constante para processamento de um exemplo, além de não necessitar de armazenamento de instâncias, tornando-se adequado a fluxos potencialmente infinitos sob restrições de tempo e memória.

## 2.2 APRENDIZADO SEMI-SUPERVISIONADO ONLINE

Tradicionalmente, na área de classificação, o aprendizado de máquina é categorizado entre supervisionado e não supervisionado. No primeiro, durante o treinamento, um modelo recebe tanto os vetores de características das amostras quanto seus rótulos. Em contraste, no aprendizado não supervisionado, o modelo tem acesso apenas aos vetores de características, recorrendo, geralmente, a algoritmos de agrupamento e métricas de distância entre as amostras (Bishop, 2006). Visto que o processo de rotulação de amostras é caro e demorado, mas também benéfico para a aprendizagem, algoritmos de aprendizado semi-supervisionado vêm ganhando atenção (Gomes et al., 2022).

Nesta categoria, um algoritmo recebe tanto instâncias rotuladas quanto não rotuladas (van Engelen e Hoos, 2020), assim, pode-se tanto utilizar algoritmos de agrupamento (não supervisionados) e, por exemplo, os rótulos disponíveis para quantificar a precisão de um determinado grupo, quanto utilizar modelos supervisionados e suas predições de instâncias não rotuladas como rótulos verdadeiros para seu próprio treinamento. Este último caso é conhecido como *self-training*, no qual um modelo supervisionado é utilizado e, em seu treinamento, incorpora-se as instâncias rotuladas e, no caso das não rotuladas, utiliza-se suas próprias predições como rótulos para seu treinamento, denominadas pseudo-rótulos (Zhu, 2005; Amini et al., 2025; Lee, 2013).

Tratando-se de algoritmos semi-supervisionados no contexto de aprendizado online, em Le Nguyen et al. (2019) foi proposto o MOA-SS, uma extensão do *framework* MOA (*Massive Online Analysis*), uma biblioteca Java de aprendizagem de máquina com algoritmos de aprendizado *online* e avaliação em fluxos de dados (Bifet et al., 2010). No MOA-SS, foram propostos dois algoritmos de aprendizagem *online*, um baseado em agrupamento, denominado *Cluster-and-label*, e outro em *self-training*.

O algoritmo *Cluster-and-label* utiliza algum modelo de agrupamento *online* como base e uma estratégia de votação para seleção de rótulos de instâncias não rotuladas, que mantém uma contagem de instâncias por rótulo para cada grupo. Os autores, em seus experimentos, utilizaram como base o algoritmo *CluStream*, que funciona mantendo um número $m$ de *micro-clusters*, os quais armazenam estatísticas que representam os pontos de um grupo em vez de armazenar as instâncias em si (Aggarwal et al., 2003). No treinamento do *Cluster-and-label*, para cada nova instância é verificado seu grupo mais próximo e, se a amostra é rotulada, ela é usada para treinar o algoritmo de agrupamento e para incrementar a contagem de instâncias de sua classe neste grupo. Caso o ponto não possua rótulo, seu pseudo-rótulo é determinado pela classe mais frequente em seu grupo mais próximo, que por sua vez é usado para treinar o agrupador. Assim, a inferência do *Cluster-and-label* para um determinado ponto é dada pela classe com mais peso no *micro-cluster* mais próximo.

O outro método proposto no MOA-SS, *self-training*, utiliza um modelo supervisionado *online* (nos experimentos dos autores, árvore de Hoeffding) e pseudo-rotulagem, e pode variar em três aspectos: algoritmo de treinamento, pontuação de confiança e limiar de confiança. No treinamento, há a opção incremental, no qual uma instância é processada de cada vez e, caso seja não rotulada, seu pseudo-rótulo é utilizado caso a pontuação de confiança utilizada seja maior ou igual ao limiar de confiança escolhido. Por outro lado, há o treinamento incremental em *batch*, no qual novas amostras são acumuladas em um conjunto com um tamanho máximo e as instâncias rotuladas são utilizadas para atualizar o modelo normalmente. Quando o conjunto está cheio, os pseudo-rótulos das amostras não rotuladas são computados e as com maiores pontuações de confiança são utilizadas no treinamento. Assim, o conjunto é esvaziado e este processo reinicia.

Também, são disponibilizadas duas opções para a pontuação de confiança. A primeira consiste em utilizar a confiança de predição dada pelo modelo base. Por outro lado, a segunda opção se baseia no uso da distância de uma amostra não rotulada em relação às instâncias rotuladas da classe predita. Assim, sendo necessário o armazenamento destes pontos rotulados, a pontuação de confiança baseada em distância só pode ser utilizada junto com o treinamento incremental em *batch*. Por fim, há dois tipos de limiares de confiança: fixo (escolhido previamente) ou adaptativo. O limiar adaptativo sofre alterações de acordo com mudanças nas pontuações de confiança. Precisamente, a cada intervalo de tempo de tamanho pré-definido o limiar é atualizado para um valor próximo à média das pontuações de confiança anteriores. Desta forma, seis combinações de algoritmos de *self-training* são possíveis no *MOA-SS*:

- BDF: treinamento incremental em *batch* com pontuação de confiança baseada em distância e limiar de confiança fixo;
- BDA: similar ao BDF, mas utilizando limiar de confiança adaptativo;
- BPF: treinamento incremental em *batch* com pontuação de confiança baseada em confiança de predição e limiar de confiança fixo;
- BPA: similar ao BPF, mas utilizando limiar de confiança adaptativo;
- IPF: treinamento incremental com pontuação de confiança baseada em confiança de predição e limiar de confiança fixo;
- IPA: similar ao IPF, mas utilizando limiar de confiança adaptativo.

Além destas opções, os autores propuseram duas alternativas baseadas em peso em que não há limiar de confiança. Isto é, todas as instâncias não rotuladas são utilizadas para o treinamento do modelo, usando pesos diferentes. Assim, são disponibilizadas as configurações IDW, que utiliza treinamento incremental e confiança de predição do modelo base como peso para as amostras não rotuladas, e IEW, que também opta pelo treinamento incremental, porém é dado o mesmo peso para todas instâncias.

O algoritmo *Co-training* é um método semi-supervisionado *offline* em que dois classificadores são treinados com visões distintas de um conjunto de dados, assim, as predições com maior confiança de um modelo são usadas como pseudo-rótulos no conjunto do outro (Blum e Mitchell, 1998). Baseado neste algoritmo e no *self-training* proposto no *MOA-SS*, (Monteiro et al., 2021) propuseram o *Co-op training*, um algoritmo semi-supervisionado *online* que utiliza dois modelos supervisionados distintos, um classificador base e um sub-classificador (nos experimentos dos autores, árvore de Hoeffding e *Naive Bayes* respectivamente), e um limiar de confiança fixo. Em seu algoritmo de treinamento em fluxo de dados, ambos classificadores são treinados com as amostras rotuladas. Dado um ponto sem rótulo, os modelos são usados para inferência e, caso as predições coincidam e a confiança de predição do modelo base seja maior que o limiar de confiança previamente determinado, o pseudo-rótulo é usado apenas para atualização do classificador base.

## 2.3 ENVENENAMENTO DE DADOS

Envenenamento de dados trata-se de um tipo de ataque adversário cujo objetivo é a contaminação do treinamento de um modelo, levando-o a cometer erros de classificações de amostras de uma ou várias classes (Fan et al., 2022). Um exemplo comum de ataque em aprendizado supervisionado é a troca de rótulos de pontos em um conjunto de dados, de forma aleatória ou com objetivo de influenciar uma classe específica (Ramirez et al., 2022).

Entretanto, em aprendizado semi-supervisionado, há ataques a partir da inserção de instâncias maliciosas no conjunto de dados do modelo, principalmente de amostras não rotuladas. Visto que o ataque de envenenamento de dados visa comprometer o processo de treinamento do modelo, espera-se que a fronteira de decisão do modelo seja alterada de forma a favorecer o atacante. Assim, também, presume-se que a qualidade do modelo (e.g., acurácia) seja degradada gradualmente, principalmente em relação à classe alvo do ataque (Cinà et al., 2023).

A principal diferença entre ataques de envenenamento em aprendizado *offline* e *online* está no fato de que, no último, a ordem das instâncias importa e pode influenciar o sucesso do ataque (Wang e Chaudhuri, 2018). Em aprendizado *offline*, o atacante pode inserir todas as

instâncias maliciosas de uma só vez, enquanto em aprendizado *online*, as instâncias maliciosas são inseridas ao longo do tempo, exigindo que o atacante considere a sequência de dados para maximizar o impacto do ataque.

## 2.4 CONCLUSÃO

Neste capítulo, foram apresentados os conceitos fundamentais para a compreensão deste trabalho. Inicialmente, foi introduzido o conceito de fluxo de dados, destacando suas características e diferenças em relação ao aprendizado em *batch*. Em seguida, foi abordado o aprendizado semi-supervisionado *online*, detalhando os algoritmos propostos no *MOA-SS* e *Co-op training*. Por fim, foi discutido o envenenamento de dados, elucidando seu funcionamento e impacto nos modelos de aprendizado de máquina. Esses conceitos formam a base para a proposta apresentada nos capítulos subsequentes.

# 3 ESTADO DA ARTE

Neste Capítulo, será apresentado o estado da arte relacionado ao envenenamento de modelos de aprendizado de máquina, com foco em algoritmos semi-supervisionados e aprendizado *online*. Vários métodos e *frameworks* foram propostos na literatura visando o envenenamento de modelos, desde envenenamento utilizando instâncias não rotuladas em algoritmos semi-supervisionados quanto modelagem de ataques em fluxos de dados como problemas de otimização e controle ótimo. Inicialmente, a Seção 3.1 abordará trabalhos relacionados ao envenenamento de modelos semi-supervisionados. Em seguida, a Seção 3.2 apresentará trabalhos que propõem ataques de envenenamento em fluxos de dados.

## 3.1 ENVENENAMENTO DE MODELOS DE ALGORITMOS SEMI-SUPERVISIONADOS

Há alguns trabalhos que abordam o envenenamento de modelos semi-supervisionados, porém, focam em algoritmos tradicionais de aprendizado em *batch*, não abordando o cenário de fluxos de dados. Em (Carlini, 2021), o autor propôs um ataque de envenenamento para modelos semi-supervisionados *offline* com pseudo-rótulos, injetando um caminho de instâncias não rotuladas entre um exemplo da classe alvo ao exemplo visado, de forma à classificá-lo incorretamente. Este ataque, denominado *Interpolation Consistency Poisoning*, foi testado contra os modelos *MixMatch* (Berthelot et al., 2019), *UDA* (Xie et al., 2020) e *FixMatch* (Sohn et al., 2020), e demonstrou eficácia com apenas $0,1\%$-$0,5\%$ de inserções, concluindo que modelos mais acurados mostram-se mais vulneráveis a este ataque.

Outro exemplo é o trabalho de (Liu et al., 2019), no qual foi feito um estudo sistemático de ataques de envenenamento de dados contra modelos semi-supervisionados baseados em grafos (*GSSL*), nos quais as instâncias são representadas como nós em um grafo, e as relações entre elas são representadas como arestas ponderadas cujo peso indica a similaridade entre os nós (Song et al., 2023). Os autores propuseram um método de ataque formulado como um problema de otimização, cujo objetivo é a maximização de erros de predições do modelo. Tem como base a alteração de rótulos de algumas instâncias (*label poisoning*), bem como o envenenamento de características de instâncias não rotuladas (*feature poisoning*), de forma opcional.

De outra forma, em (Liu et al., 2022), os autores exploram envenenamento em aprendizado semi-supervisionado federado (*FSSL*), categoria de SSL na qual múltiplos dispositivos colaboram para treinar um modelo global sem compartilhar seus dados locais (Song et al., 2024). Neste caso, um servidor é treinado com dados rotulados e dispositivos clientes possuem exemplos não rotulados, que são pseudo-rotulados pelo modelo global. O método proposto, que utiliza instâncias não rotuladas, alcança uma boa qualidade de ataque com apenas $0,1\%$ de amostras maliciosas, e, ainda, os autores apresentam uma defesa por seleção de clientes baseada em otimização utilizando *minimax*.

## 3.2 ENVENENAMENTO DE MODELOS DE APRENDIZADO *ONLINE*

No contexto de fluxos de dados, o envenenamento de modelos tem sido explorado em poucos trabalhos, em geral focados em classificação *online* supervisionada. Em Wang e Chaudhuri (2018), estuda-se o envenenamento de classificadores lineares treinados utilizando o algoritmo de descida de gradiente *online* (OGD), formulando o ataque como um problema de otimização em que o atacante pode modificar no máximo K amostras do fluxo de treinamento. Os

autores propõem três estratégias: *Incremental Attack* (abordagem gulosa que altera os pontos com maior gradiente), *Interval Attack* (modifica sequência consecutiva ótima) e *Teach-and-Reinforce Attack* (corrompe início do fluxo e reforça ao longo do *stream*). Os experimentos mostram, dentre outras coisas, que o ataque, por explorar a ordem temporal das instâncias, é mais eficaz que ataques que ignoram a natureza sequencial dos dados.

De outra forma, em (Zhang et al., 2020), modela-se o envenenamento de dados *online* como um problema de controle ótimo em MDP (*Markov Decision Process*), no qual um atacante *white-box* observa e manipula apenas a amostra atual para afetar o processo de aprendizado *online*. Foram utilizados dois algoritmos, um baseado em planejamento com modelo e outro em aprendizado por reforço profundo, ambos demonstrando capacidade de ataque próximo ao de um atacante que possui conhecimento completo do fluxo de dados.

Não foram encontrados outros trabalhos que propõem ataques de envenenamento em fluxos de dados supervisionados, tampouco em fluxos de dados semi-supervisionados, demonstrando ser uma lacuna na literatura que este trabalho busca preencher. Assim, no próximo Capítulo, será proposto um método para avaliação da robustez de modelos semi-supervisionados contra ataques de envenenamento de dados em fluxos de dados, utilizando instâncias não rotuladas para o ataque.

## 3.3 CONCLUSÃO

Neste capítulo, foi apresentado o estado da arte relacionado ao envenenamento de modelos de aprendizado de máquina, com foco em algoritmos semi-supervisionados e aprendizado *online*. Foram discutidos trabalhos que abordam o envenenamento de modelos semi-supervisionados em cenários *offline*, destacando métodos como *Interpolation Consistency Poisoning* e ataques formulados como problemas de otimização. Em seguida, foram apresentados estudos sobre envenenamento em fluxos de dados supervisionados, incluindo estratégias baseadas em descida de gradiente *online* e modelagem como problemas de controle ótimo em *MDP*. Foi identificado que há uma lacuna na literatura em relação ao envenenamento de modelos semi-supervisionados em fluxos de dados, a qual este trabalho busca preencher. O próximo capítulo apresentará a proposta deste trabalho: um método para avaliação da robustez de modelos semi-supervisionados contra ataques de envenenamento de dados em fluxos de dados.

# 4 PROPOSTA

Este capítulo apresentará a proposta deste trabalho: um método para avaliação da robustez de modelos semi-supervisionados contra ataques de envenenamento de dados em fluxos de dados. Inicialmente, a Seção 4.1 descreverá a geração do conjunto de dados de envenenamento, baseado no ataque a partir de instâncias não rotuladas. Finalmente, a Seção 4.2 abordará a geração do conjunto de dados de avaliação, utilizado para medir o impacto do envenenamento no modelo ao decorrer do treinamento.

## 4.1 CONJUNTO DE DADOS DE ENVENENAMENTO

Para simular o envenenamento de dados em fluxos de dados e avaliar a robustez dos modelos usados, é proposta a criação de um conjunto de dados sintético que representa o ataque a partir de instâncias não rotuladas, utilizado para treinar o modelo de forma incremental. O conjunto simula um ataque temporal, no qual instâncias maliciosas não rotuladas são introduzidas gradualmente ao longo do tempo, com o objetivo de manipular a fronteira de decisão do modelo. Os dados são parametrizados para refletir diferentes cenários de ataque, incluindo número de instâncias, proporção de instâncias maliciosas e padrões de distribuição das classes.

Os parâmetros disponíveis estão descritos na Tabela 4.1, juntamente com suas respectivas descrições e valores padrão, os quais foram utilizados para os experimentos deste trabalho. Os dados são binários, ou seja, possuem duas classes, a positiva e a negativa. Também, adota-se dimensionalidade dos vetores de características como dois, o que facilita a visualização do conjunto de dados e a fronteira de decisão de modelos treinados em duas dimensões.

Tabela 4.1: Parâmetros da Geração do Conjunto de Dados de Envenenamento

| **Parâmetro** | **Descrição** | **Valor Padrão** |
|---|---|---|
| $N$ | Número de instâncias totais | 1.000 |
| $\min(x_i)$ | Valor mínimo das características | 0 |
| $\max(x_i)$ | Valor máximo das características | 50 |
| $m$ | Margem de distância entre instâncias e fronteira de decisão | 30 |
| $\alpha$ | Proporção inicial rotulada | 0,25 |
| $\gamma$ | Proporção não rotulada na fase de ataque | 0,9 |
| $\beta$ | Proporção de instâncias não rotuladas geradas no *cluster* de ataque | 0,8 |
| $\lambda$ | Fator de espalhamento das instâncias de ataque a partir da média | 0,15 |
| $\rho$ | Fator de controle de deslocamento máximo da média das instâncias de ataque | 0,8 |
| $s$ | Semente do gerador pseudo-aleatório | 42 |

Assim, o objetivo do ataque é modificar a fronteira de decisão do modelo, de forma que instâncias de uma classe sejam classificadas como pertencentes à outra. Por exemplo, em um cenário de classificação de transações bancárias maliciosas, um possível atacante poderia ter como finalidade a manipulação da fronteira de decisão do classificador de forma que instâncias maliciosas fossem erroneamente classificadas como benignas. Para isso, o meio de ataque seria a inserção de instâncias não rotuladas que, iniciadas com características similares às de amostras

maliciosas, gradualmente possuiriam características parecidas com exemplos benignos. Desta forma, caso o classificador fosse vulnerável a ataques por instâncias não rotuladas, o atacante obteria sucesso em seu propósito.

As instâncias são geradas aleatoriamente, seguindo uma distribuição uniforme dentro dos limites especificados ($\min(x_i)$ e $\max(x_i)$). A margem de separação de classe ($m$) define a distância mínima de qualquer ponto à fronteira de decisão, garantindo que as instâncias de diferentes classes estejam suficientemente separadas no espaço de características. Esta margem é necessária para que o ataque tenha um impacto significativo na fronteira de decisão do modelo, facilitando, também, a visualização dos resultados.

Para definir a classe das instâncias rotuladas, é utilizada uma fronteira de decisão linear, dada pela função $x_1 - x_2 = 0$. Ou seja, dado um ponto $(x_1, x_2)$, seu rótulo é definido como:

$$y(\mathbf{x}) = \textsc{ComputarRotulo}(\mathbf{x}) = \begin{cases} 0, & \text{se } x_1 < x_2 \\ 1, & \text{se } x_1 \geq x_2 \end{cases}$$

Assim, foi adotada a convenção do *cluster* superior, utilizado no ataque, ser representado pela classe negativa ($y = 0$), e o *cluster* inferior, pela classe positiva ($y = 1$). A geração do conjunto se dá em duas fases:

1. Geração das instâncias rotuladas iniciais;
2. Geração de instâncias com envenenamento.

**Fase 1**: o fluxo de dados inicia com a geração de um conjunto inicial de instâncias rotuladas, as quais serão utilizadas para o treinamento inicial do modelo. A proporção dessas instâncias é definida pelo parâmetro $\alpha$ e seus rótulos são atribuídos conforme a função de fronteira de decisão mencionada anteriormente. Essas instâncias são geradas aleatoriamente, respeitando a margem de separação de classe $m$ e compondo $N * \alpha$ de $N$ amostras totais.

**Fase 2**: na etapa de geração das amostras com envenenamento, cada instância tem uma probabilidade $\gamma$ de ser não rotulada. Dentre as instâncias não rotuladas, uma proporção de $\beta$ é gerada no *cluster* da classe negativa (superior), que é utilizada no ataque. O restante das instâncias não rotuladas é gerada no *cluster* da classe positiva (inferior), que não faz parte do envenenamento e, por isso, respeita a restrição de margem.

Para simular o ataque, as instâncias maliciosas não rotuladas geradas no *cluster* da classe negativa são deslocadas gradualmente em direção à fronteira de decisão e ao *cluster* inferior, da classe positiva. Estas amostras são criadas aleatoriamente seguindo uma distribuição normal isotrópica em duas dimensões, cuja média é calculada de forma dinâmica, por amostra, de acordo com o progresso do fluxo, e desvio-padrão computado a partir do fator de espalhamento $\lambda$. Assim, a única restrição adicional é de que o ponto gerado deve respeitar os limites das características $\min(x_i)$ e $\max(x_i)$.

O Algoritmo 1 descreve integralmente a geração do conjunto de envenenamento, e utiliza as seguintes funções:

- CriarPontoComMargem: recebe $min(x_i)$, $max(x_i)$ e $m$, retorna um ponto $p$ aleatório, seguindo uma distribuição uniforme, dentro dos limites das características e fora da margem, respeitando a condição $|p_1 - p_2| > m$;
- CriarPontoEnvenenado: recebe $min(x_i)$, $max(x_i)$, $\mu(t)$ e $\sigma$, retorna um ponto $p$ aleatório, seguindo a distribuição normal $\mathcal{N}(\mu(t), \sigma^2 I_2)$, dentro dos limites das características;

- CriarPontoInferior: recebe $min(x_i)$, $max(x_i)$ e $m$, retorna um ponto $p$ aleatório, seguindo uma distribuição uniforme, dentro do *cluster* inferior, respeitando a condição $p_2 \leq p_1 - m$.

Como dito anteriormente, a média da distribuição normal de geração de pontos envenenado é calculada por amostra. Precisamente, o centroide que define a média da distribuição normal é iniciado como o centroide do *cluster* superior, calculado na linha 10. Após isso, é feita uma interpolação com base no centroide de ambos os *clusters*, de forma que a média da distribuição avance em direção ao *cluster* inferior conforme o progresso do fluxo, chegando próximo ao final, computado na linha 22. Já o desvio-padrão é calculado uma única vez, também a partir dos centroides, demonstrado na linha 11.

A visualização deste processo é apresentada na Figura 4.1, onde é possível observar a progressão do envenenamento ao longo do tempo, com as instâncias maliciosas se aproximando da fronteira de decisão e, depois, do *cluster* da classe positiva.

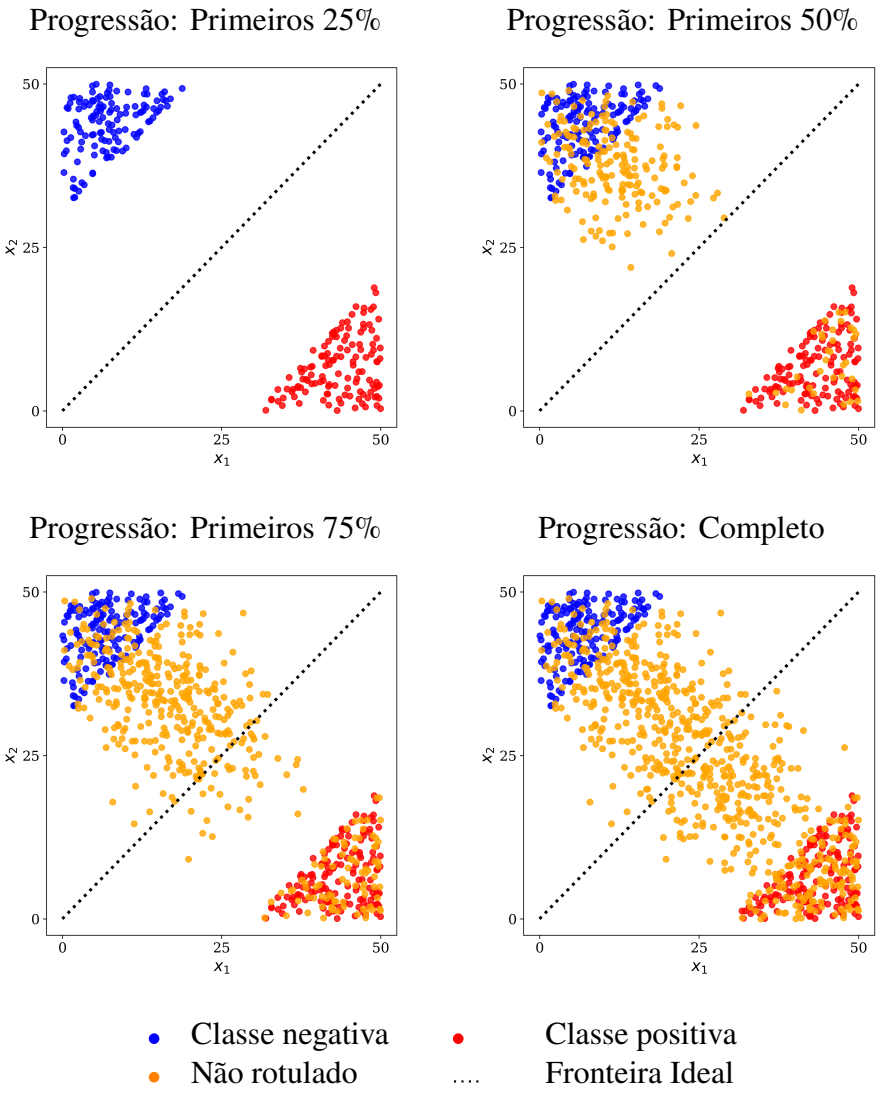



Figura 4.1: Conjunto de dados de envenenamento proposto ao longo do tempo.

**Algoritmo 1** GerarConjuntoEnvenenamento($N, \min(x_i), \max(x_i), m, \alpha, \gamma, \beta, \lambda, s$)

**Entrada:** Número de instâncias $N$; limites $[\min(x_i), \max(x_i)] \subset \mathbb{R}^2$; margem $m > 0$; proporção inicial rotulada $\alpha \in (0, 1)$; proporção não rotulada $\gamma \in [0, 1]$; proporção de instâncias não rotuladas no *cluster* superior $\beta \in [0, 1]$; fator de espalhamento $\lambda > 0$; fator de controle de deslocamento máximo $\rho \in (0, 1]$; semente $s$.
**Saída:** Conjunto $\mathcal{D}$ com $N$ amostras (rotuladas e não rotuladas).

Inicialize o gerador pseudoaleatório com semente $s$.
$n_{\text{init}} \leftarrow \lfloor N \cdot \alpha \rfloor$; $T \leftarrow N - n_{\text{init}}$; $\mathcal{D} \leftarrow \emptyset$.
// Fase inicial: amostragem uniforme com margem.
**for** $i = 1$ **to** $n_{\text{init}}$ **do**
  $\mathbf{x} \leftarrow$ CriarPontoComMargem($\min(x_i), \max(x_i), m$).
  $y \leftarrow$ ComputarRotulo($\mathbf{x}$).
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{x}, y)\}$.
**end for**
// Centroides a partir das amostras rotuladas iniciais.
$\mu_S \leftarrow$ média$\{\mathbf{x} \in \mathcal{D} : y = 0\}$; $\mu_I \leftarrow$ média$\{\mathbf{x} \in \mathcal{D} : y = 1\}$.
$\mathbf{d} \leftarrow \mu_I - \mu_S$; $\sigma \leftarrow \lambda \cdot \|\mathbf{d}\|_2 / \sqrt{2}$. // Dimensionalidade = 2
// Gera as amostras restantes do fluxo.
**for** $t = 0$ **to** $T - 1$ **do**
  **if** RAND() $> \gamma$ **then**
    // Ponto rotulado.
    $\mathbf{x} \leftarrow$ CriarPontoComMargem($\min(x_i), \max(x_i), m$).
    $y \leftarrow$ ComputarRotulo($\mathbf{x}$).
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{x}, y)\}$.
  **else**
    // Ponto não rotulado.
    **if** RAND() $< \beta$ **then**
      $\mu(t) \leftarrow \mu_S + \rho * \frac{t}{T-1} * \mathbf{d}$.
      $\mathbf{x} \leftarrow$ CriarPontoEnvenenado($\min(x_i), \max(x_i), \mu(t), \sigma$).
    **else**
      $\mathbf{x} \leftarrow$ CriarPontoInferior($\min(x_i), \max(x_i), m$).
    **end if**
    $y \leftarrow null$.
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{x}, y)\}$.
  **end if**
**end for**
**return** $\mathcal{D}$

## 4.2 CONJUNTO DE DADOS DE AVALIAÇÃO

Para medir o impacto do envenenamento no modelo ao decorrer do treinamento, é proposta a criação de um conjunto de dados de avaliação, que será utilizado para avaliar o desempenho do modelo em diferentes estágios do treinamento. Este conjunto também é binário e com dimensionalidade igual a dois, seguindo as configurações do conjunto de envenenamento. Contudo, diferentemente do conjunto de dados de envenenamento, todas as instâncias são rotuladas e geradas sem a restrição da margem de separação entre as instâncias e a fronteira de decisão. Isso garante que o conjunto de avaliação represente integralmente o espaço de características e permita uma avaliação precisa do desempenho do modelo. Desta forma, os parâmetros necessários para a geração do conjunto de dados de avaliação são compostos por $N$, $min(x_i)$, $max(x_i)$ e $s$, listados na Tabela 4.1.

Assim, o modelo é avaliado periodicamente, a cada 10% do fluxo de treinamento, utilizando todos os pontos deste conjunto de dados de avaliação, provendo uma métrica clara do impacto do envenenamento no desempenho do modelo. Essa abordagem permite identificar pontos críticos onde o envenenamento começa a afetar significativamente a fronteira de decisão do modelo, utilizando métricas como a acurácia. O Algoritmo 2 descreve a criação do conjunto de avaliação, dados os parâmetros necessários, utilizando a função CriarPontoUniforme, que retorna um ponto aleatório, seguindo uma distribuição uniforme, dentro dos limites das características.

**Algoritmo 2** GerarConjuntoAvaliacao$(N, \min(x_i), \max(x_i), s)$

**Entrada:** Número de instâncias $N$; limites $[\min(x_i), \max(x_i)] \subset \mathbb{R}^2$; semente $s$.
**Saída:** Conjunto $\mathcal{D}$ com $N$ amostras rotuladas.

1: Inicialize o gerador pseudoaleatório com semente $s$.
2: $\mathcal{D} \leftarrow \emptyset$.
3: **for** $i = 1$ **to** $N$ **do**
4: $\quad \mathbf{x} \leftarrow$ CriarPontoUniforme$(\min(x_i), \max(x_i))$.
5: $\quad y \leftarrow$ ComputarRotulo$(\mathbf{x})$.
6: $\quad \mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{x}, y)\}$.
7: **end for**
8: **return** $\mathcal{D}$

Por fim, a Figura 4.2 apresenta o conjunto de avaliação gerado a partir dos valores padrão dos parâmetros.

## 4.3 CONCLUSÃO

Neste capítulo, foi apresentada a proposta de um método para avaliação da robustez de modelos semi-supervisionados contra ataques de envenenamento de dados em fluxos de dados. A geração do conjunto de dados de envenenamento foi detalhada, incluindo os parâmetros utilizados e o processo de simulação do ataque. Além disso, foi descrita a criação do conjunto de dados de avaliação, que permite medir o impacto do envenenamento no desempenho do modelo ao longo do treinamento.

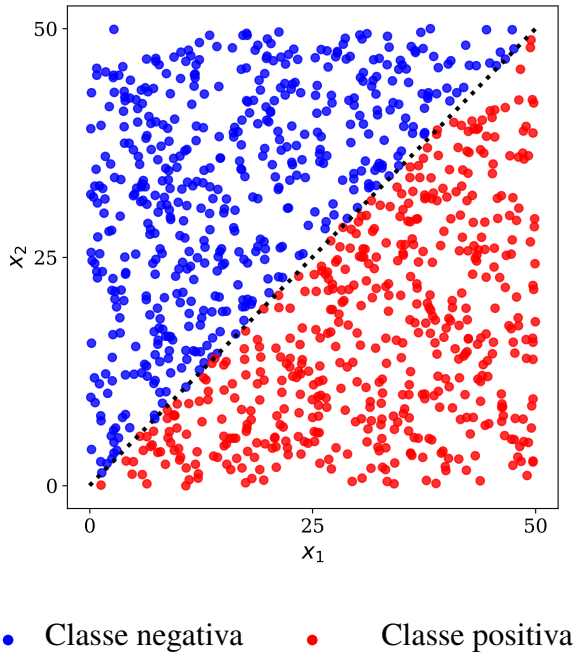



Figura 4.2: Conjunto de dados de avaliação criado a partir dos valores padrão dos parâmetros.

# 5 PROTOCOLO EXPERIMENTAL

Este capítulo descreverá o protocolo experimental utilizado para avaliar a robustez de modelos semi-supervisionados contra ataques de envenenamento de dados em fluxos de dados. Inicialmente, a Seção 5.1 detalhará os modelos avaliados nos experimentos. Em seguida, a Seção 5.2 apresentará o protocolo experimental usado, incluindo a forma de treinamento e avaliação. Por fim, a Seção 5.3 abordará os detalhes de implementação do método desenvolvido para este trabalho.

## 5.1 MODELOS SEMI-SUPERVISIONADOS

Para avaliar o impacto do envenenamento de dados utilizando o conjunto de dados proposto no Capítulo 4, foram selecionados três paradigmas de aprendizado semi-supervisionado: o *self-training* (Le Nguyen et al., 2019), *cluster-and-label* (Le Nguyen et al., 2019) e o *co-op training* (Monteiro et al., 2021). Cada abordagem possui uma estratégia específica sobre o uso dos dados não rotulados durante o treinamento do modelo.

*Self-training* é um método iterativo onde um modelo inicial é treinado com um conjunto pequeno de dados rotulados. Em seguida, o modelo faz previsões sobre os dados não rotulados e as instâncias com maior confiança são utilizadas para treinamento com seus rótulos previstos (pseudo-rótulos). Para estes modelos, foi utilizada a implementação do *MOA-SS* (Le Nguyen et al., 2019), que possui modelos com variações de estilo de treinamento, pontuação de confiança e limiar de confiança, utilizado para selecionar os pseudo-rótulos a serem utilizados para treinamento.

Nestes experimentos, foram escolhidas três variações dos modelos de *self-training* disponíveis no *MOA-SS*: (i) *Self-training BDA*, o qual utiliza treinamento incremental em *batch*, pontuação de confiança baseada na distância da instância em relação às amostras rotuladas da classe predita e, por fim, limiar de confiança adaptativo, que evolui de acordo com as mudanças na média das pontuações de confiança, (ii) *Self-training IPA*, que, apesar de também possuir limiar de confiança adaptativo, utiliza treinamento incremental e pontuação de confiança baseada na probabilidade predita pelo modelo e (iii) *Self-training IDW*, o qual utiliza treinamento incremental e todas as instâncias não rotuladas para treinamento, ponderadas pela confiança de predição do modelo base.

O algoritmo de *cluster-and-label* utiliza um modelo *online* não supervisionado de *clustering* e, para cada instância não rotulada, utiliza como pseudo-rótulo o rótulo mais frequente dentre as instâncias rotuladas presentes no grupo mais próximo. Neste caso, o modelo de *clustering* utilizado foi o *CluStream*, assim como os autores em (Le Nguyen et al., 2019).

Por fim, o *co-op training* utiliza dois modelos supervisionados distintos que são treinados simultaneamente: o modelo base e o sub-modelo. Ambos os modelos são treinados com as amostras rotuladas e, para cada instância não rotulada, caso as predições dos dois modelos coincidam e a confiança do modelo base seja maior que um limiar pré-definido, a instância, junto com seu rótulo previsto, é utilizada para treinar o modelo base. Assim como nos experimentos dos autores, foi utilizado o *Hoeffding Tree* como modelo base, *Naive Bayes* como sub-modelo e limiar de confiança fixo igual a 0,9.

## 5.2 PROTOCOLO EXPERIMENTAL

No treinamento dos modelos citados com o conjunto de dados de envenenamento proposto, foi utilizada a abordagem *online test-then-train*, que simula um cenário realista de aprendizado *online*. Isto é, a cada instância no fluxo, o modelo produz uma predição e, em seguida, é atualizado com a mesma. As predições sobre as amostras do conjunto de dados com envenenamento, do treinamento, são utilizadas externamente apenas para visualização, ou seja, não são usadas para calcular métricas. Isto se dá pois as métricas são computadas apenas sobre as predições no conjunto de avaliação, demonstrado no capítulo anterior.

Precisamente, a avaliação é por *holdout* periódico. Após cada 10% do fluxo processado (i.e., em 10%, 20%, 30% etc.), o estado atual do modelo é usado para classificar toda instância no conjunto de dados de avaliação. Em seguida, computa-se acurácia e *FNR* (*False Negative Rate*). Visto que o conjunto de amostras de avaliação é balanceado, a acurácia torna-se uma métrica adequada para medir a qualidade preditiva dos modelos. Por outro lado, a taxa de falsos negativos representa quantas instâncias positivas foram preditas, erroneamente, como negativas pelo classificador, ou seja, a taxa de amostras do *cluster* inferior que foram classificadas como do *cluster* superior. Sendo assim, torna-se uma métrica apropriada para medir a efetividade do ataque, visto que espera-se o aumento da *FNR* conforme o progresso do ataque em modelos vulneráveis.

Para visualizar o impacto do ataque na fronteira de decisão do modelo ao longo do fluxo, foi gerado um conjunto de dados de grade uniforme, cobrindo todo o espaço de características (utilizando $\min(x_i)$ e $\max(x_i)$) e possuindo resolução de $200 \times 200$. O modelo é usado para prever os pontos da grade após cada 25% do fluxo de treinamento (i.e., em 25%, 50%, 75% e 100%), previsões que são usadas para visualizar a fronteira de decisão aprendida.

Finalmente, os experimentos foram conduzidos em uma máquina com processador *Intel Core i7-12700K* e 32GB de memória DDR4.

## 5.3 DETALHES DE IMPLEMENTAÇÃO

A base de código do *Pufferfish* foi desenvolvida utilizando a linguagem de programação *Python*. Para o uso das implementações dos modelos semi-supervisionados do *MOA-SS*, foi utilizado o módulo *JPype*, que permite a integração entre código *Java* e *Python*, possibilitando a utilização das classes do *MOA-SS* diretamente no código em *Python*.

Para os modelos de *self-training* e *cluster-and-label*, foram utilizadas as implementações disponíveis no *MOA-SS* sem modificações. Já para o modelo de *co-op training*, foi implementada uma versão em *Python* baseada na versão dos autores (Monteiro et al., 2021), utilizando a implementação do *Hoeffding Tree* e do *Naive Bayes* disponíveis na biblioteca *MOA* (Bifet et al., 2010).

## 5.4 CONCLUSÃO

Neste capítulo, foi apresentado o protocolo experimental para avaliar a robustez de modelos semi-supervisionados contra ataques de envenenamento de dados em fluxos de dados. Foram descritos três paradigmas utilizados: *self-training*, *cluster-and-label* e *co-op training*. O protocolo adotou a abordagem *test-then-train* para treinamento, *holdout* periódico para avaliações e uso de conjunto de grade para a visualização das fronteiras de decisão. O método *Pufferfish* foi implementado em *Python* com integração a bibliotecas *Java* via *JPype*. O próximo capítulo apresentará os resultados dos experimentos.

# 6 EXPERIMENTOS E RESULTADOS

Este capítulo apresentará os resultados dos experimentos realizados para avaliar o impacto do envenenamento de dados em algoritmos semi-supervisionados para fluxos de dados. Inicialmente, na Seção 6.1 serão mostradas as mudanças na fronteira de decisão dos modelos ao longo do fluxo de dados envenenado. Em seguida, a Seção 6.2 apresentará os resultados quantitativos obtidos pelos modelos no conjunto de dados de avaliação.

## 6.1 IMPACTO DO ENVENENAMENTO NA FRONTEIRA DE DECISÃO

Visto que o conjunto de dados utilizado é bidimensional, é possível visualizar a fronteira de decisão dos modelos ao longo de seu treinamento, o que permite observar diretamente o impacto do ataque na aprendizagem dos modelos. Como dito anteriormente, para cada modelo avaliado, são apresentadas quatro etapas do treinamento: após os primeiros 25%, 50%, 75% e 100% do fluxo de dados. Nestes estágios, os algoritmos foram utilizados para predição de amostras de um conjunto de dados de grade, cujas instâncias são distribuídas uniformemente pelo espaço de característica. Isto permite inferir a fronteira de decisão aprendida pelo modelo, como detalhado no capítulo anterior.

### 6.1.1 *Self-training* BDA

A Figura 6.1 apresenta as predições do modelo *self-training* BDA ao longo do treinamento com o fluxo de dados envenenado. Observa-se que, mesmo com o avanço do envenenamento, i.e. instâncias não rotuladas em direção ao *cluster* inferior, o modelo mantém uma fronteira de decisão próxima à real, sem grandes variações, demonstrando robustez contra o ataque.

Isto deve-se, grandemente, ao uso da pontuação de confiança baseado na distância entre a instância não rotulada e as amostras rotuladas da classe predita presentes no *batch* de instâncias. Como as instâncias não rotuladas ficam, gradualmente, mais distantes das amostras rotuladas da classe negativa, suas pontuações de confiança diminuem, o que reduz a probabilidade de serem selecionadas para treinamento do modelo. Desta forma, o modelo consegue evitar o impacto do envenenamento na fronteira de decisão.

### 6.1.2 *Self-training* IPA

A Figura 6.2 apresenta as predições do modelo *self-training* IPA ao longo do treinamento com o fluxo de dados envenenado. Diferentemente do modelo BDA, o IPA demonstra uma vulnerabilidade significativa ao ataque de envenenamento. Observa-se que, conforme o fluxo de dados envenenado avança, a fronteira de decisão do modelo se desloca progressivamente, afastando-se da fronteira real e se aproximando do *cluster* da classe positiva.

Este comportamento pode ser explicado pela estratégia de seleção de pseudo-rótulos utilizada pelo IPA, que se baseia na pontuação de confiança fornecida pelo classificador base. Como as instâncias não rotuladas envenenadas são movidas gradualmente, e são aprendidas como amostras rotuladas, as pontuações de confiança permanecem elevadas, levando o modelo a incorporar também as próximas amostras maliciosas em seu treinamento. Consequentemente, resulta em uma fronteira de decisão distorcida.

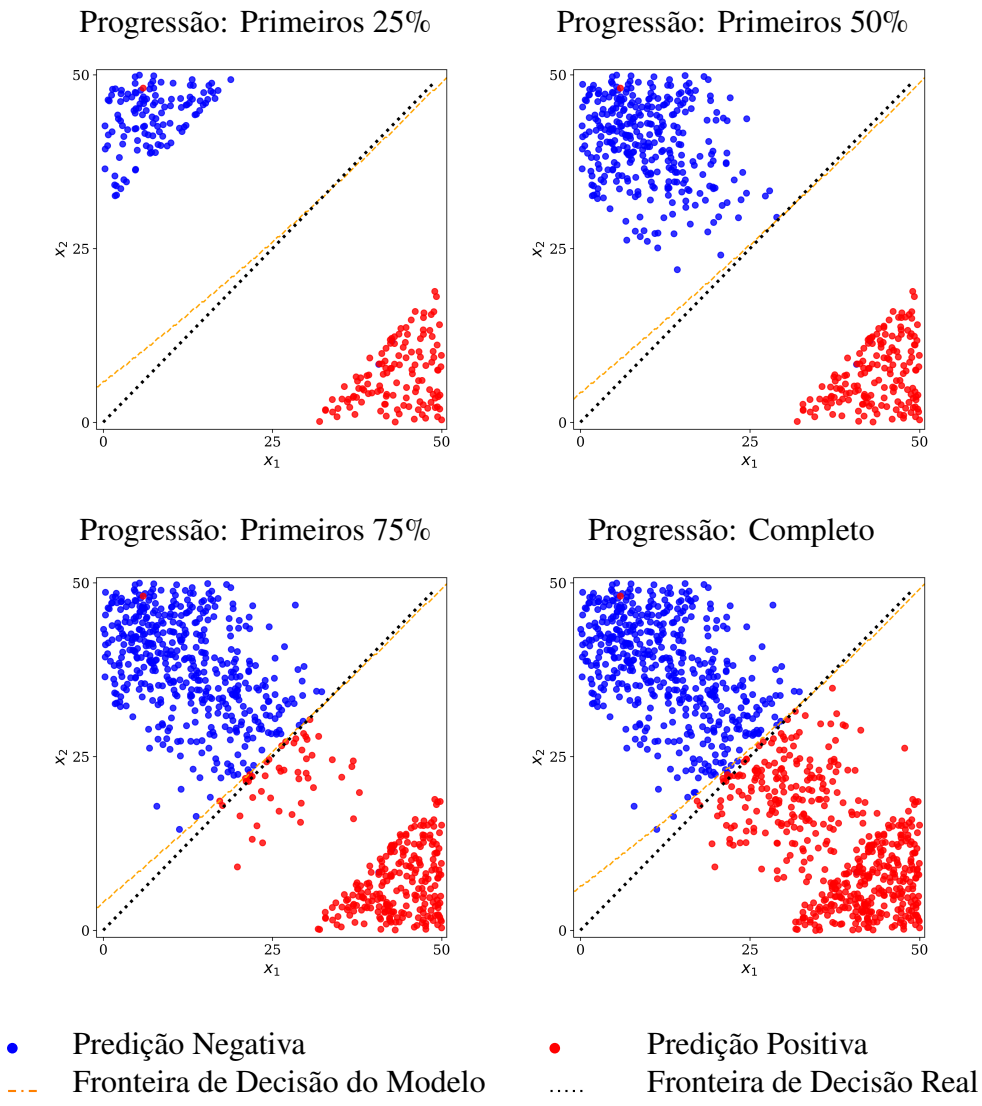



Figura 6.1: Predições do modelo *self-training* BDA durante o treinamento.

### 6.1.3 *Self-training* IDW

A Figura 6.3 apresenta as predições do modelo *self-training* IDW ao longo do treinamento com o fluxo de dados envenenado. Assim como o modelo IPA, o IDW apresenta uma grande vulnerabilidade ao ataque, demonstrando alteração na fronteira de decisão de forma similar ao IPA. Como o IPA, o IDW também utiliza treinamento incremental e, apesar de não usar um limiar de confiança, usa a confiança de predição do modelo base como peso para as amostras aprendidas.

### 6.1.4 *Cluster-and-label*

A Figura 6.4 apresenta as predições do modelo *cluster-and-label* ao longo do treinamento com o fluxo de dados envenenado. De forma similar aos modelos IPA e IDW, o *cluster-and-label* também demonstra vulnerabilidade ao ataque de envenenamento, com o deslocamento de sua fronteira de decisão de forma mais brusca.

Como comentado anteriormente, o algoritmo *cluster-and-label* utiliza todas as instâncias não rotuladas para treinamento do modelo de *clustering* base, cujos pseudo-rótulos são dados pela classe mais frequente no *cluster* selecionado. Assim, visto que não há um critério de seleção

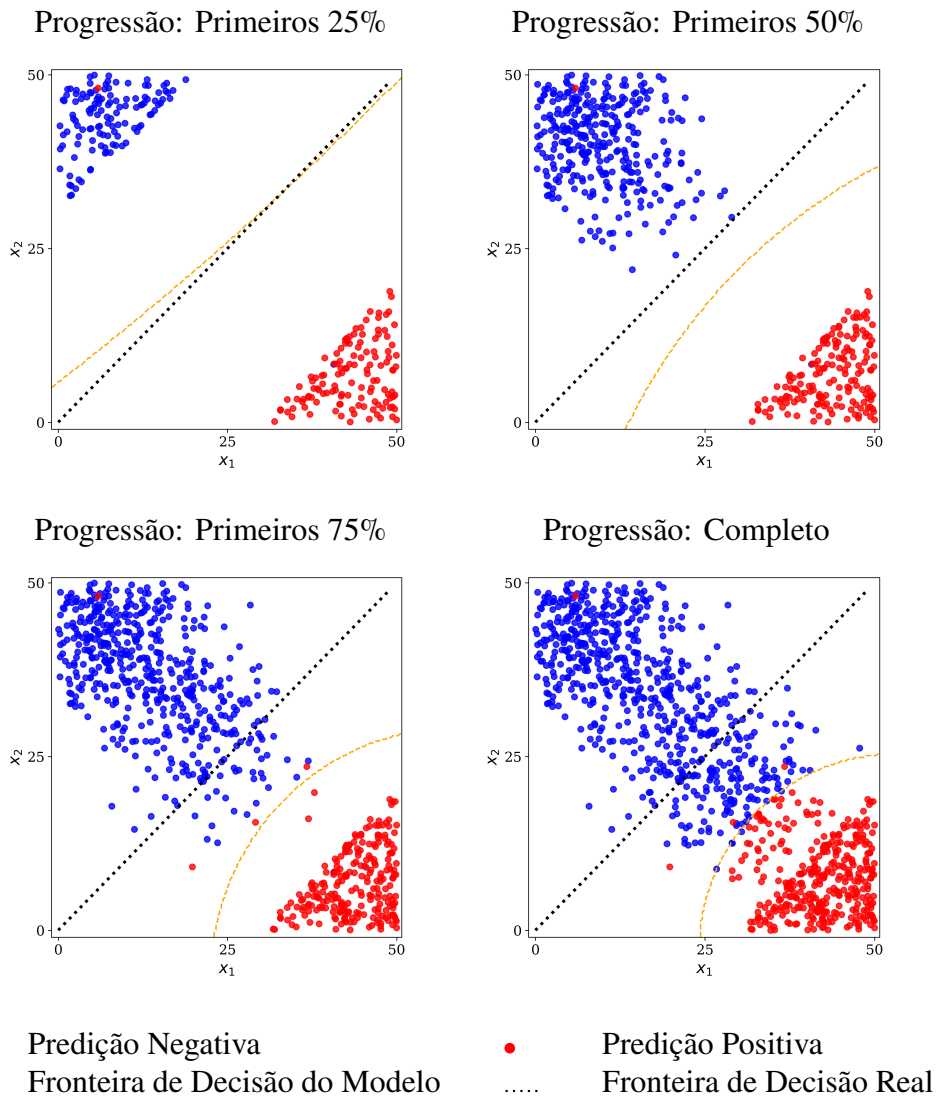



Figura 6.2: Predições do modelo *self-training* IPA durante o treinamento.

das instâncias não rotuladas, todas as instâncias envenenadas são incorporadas ao modelo, o que leva a uma distorção significativa na fronteira de decisão aprendida, mais extrema que a observada no modelo IPA, que utiliza um critério de seleção baseado em confiança.

### 6.1.5 *Co-op training*

Por fim, a Figura 6.5 apresenta as predições do modelo *co-op training* ao longo do treinamento com o fluxo de dados envenenado. O modelo apresenta um comportamento intermediário entre o BDA robusto e os modelos IPA, IDW e *cluster-and-label* vulneráveis. Observa-se que inicialmente a fronteira de decisão é afetada pelo envenenamento, deslocando-se da posição real, porém o modelo demonstra uma leve capacidade de recuperação nas fases finais do treinamento.

Este comportamento pode ser atribuído à estratégia cooperativa entre classificadores utilizada pelo *co-op training*. Além de possuir um limiar de confiança para seleção dos pseudo-rótulos, o modelo base só incorpora instâncias não rotuladas quando o sub-modelo concorda com a predição. Essa abordagem reduz a probabilidade de incorporar instâncias envenenadas, visto que o sub-modelo só é treinado com as instâncias rotuladas, que não são afetadas pelo ataque.

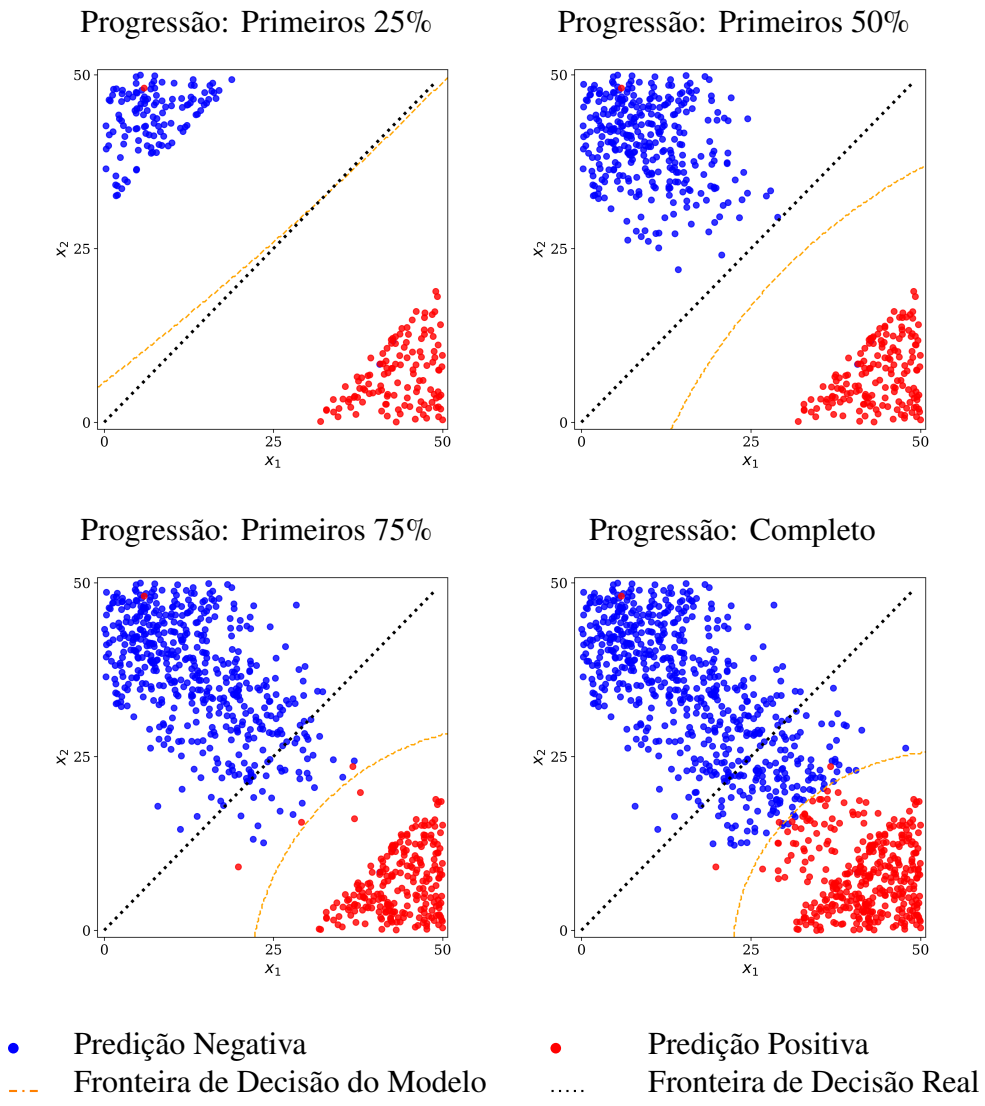



Figura 6.3: Predições do modelo *self-training* IDW durante o treinamento.

Assim, o algoritmo apresenta uma vulnerabilidade moderada ao envenenamento, conseguindo mitigar parcialmente seus efeitos.

## 6.2 DESEMPENHO NO CONJUNTO DE DADOS DE AVALIAÇÃO

Para medir quantitativamente o impacto do envenenamento de dados nos algoritmos semi-supervisionados avaliados, foram coletadas métricas de desempenho ao longo do fluxo de dados envenenado. A Tabela 6.1 apresenta as acurácias dos modelos após o processamento de cada 10% do fluxo de dados envenenado, desde 30% até 100%, calculadas utilizando o conjunto de dados de avaliação estático.

Visto que este conjunto é balanceado em relação ao número de amostras por classe, a acurácia é uma métrica adequada para quantificar o desempenho dos algoritmos semi-supervisionados. Observa-se que o modelo *self-training* BDA mantém uma acurácia elevada e estável ao longo do fluxo de dados envenenado, corroborando a observação visual da fronteira de decisão. Em contrapartida, os modelos IPA, IDW e *cluster-and-label* apresentam uma queda significativa na acurácia conforme o envenenamento avança, refletindo sua vulnerabilidade ao

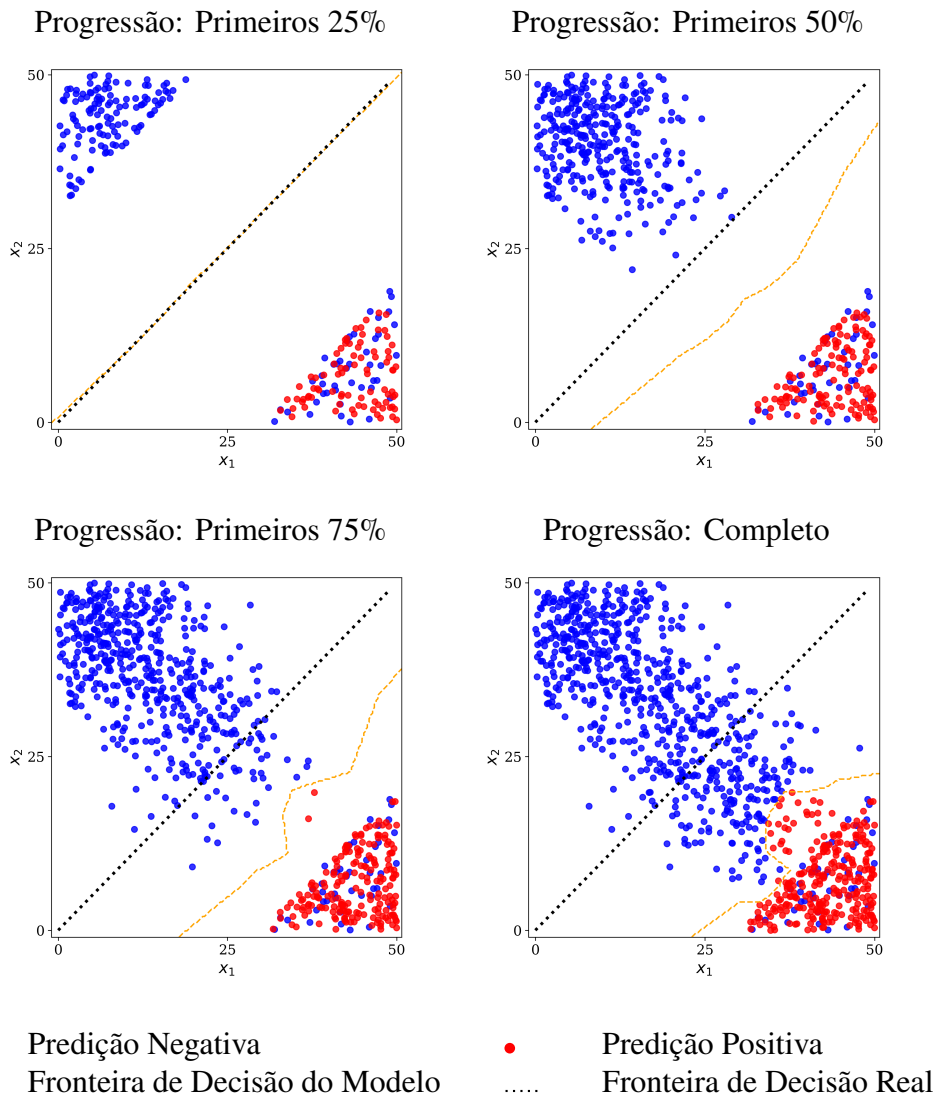



Figura 6.4: Predições do modelo *cluster-and-label* durante o treinamento.

ataque. O modelo *co-op training*, por sua vez, demonstra um desempenho intermediário, com uma queda inicial na acurácia seguida por uma leve recuperação nas fases finais do treinamento.

Também, a Figura 6.6 ilustra graficamente a evolução da acurácia dos modelos ao longo do fluxo de dados envenenado. A visualização reforça as observações anteriores, destacando a robustez do modelo BDA e a vulnerabilidade dos modelos IPA, IDW e *cluster-and-label*, bem como o desempenho intermediário do *co-op training*.

Outra métrica computada foi a taxa de falsos negativos (*FNR*), que representa quantas instâncias positivas foram erroneamente preditas como negativas (classe utilizada pelo ataque). Sendo assim, espera-se que um modelo robusto contra o envenenamento apresente um *FNR* baixo, enquanto um classificador vulnerável demonstre um aumento em seu *FNR* conforme o ataque avança. A Tabela 6.2 apresenta a taxa de falsos negativos dos classificadores testados ao longo do fluxo de dados, calculado a partir das predições no conjunto de dados de avaliação a cada 10% do fluxo de dados de treinamento.

De forma similar, a Figura 6.7 demonstra visualmente a evolução das taxas de falsos negativos dos modelos ao longo de seu treinamento. Percebe-se, novamente, que o classificador BDA demonstra a maior robustez, visto que seu *FNR* permanece abaixo de 1% durante todo o fluxo. Também, os modelos de *self-training* IPA e IDW apresentam robustez similares, cenário

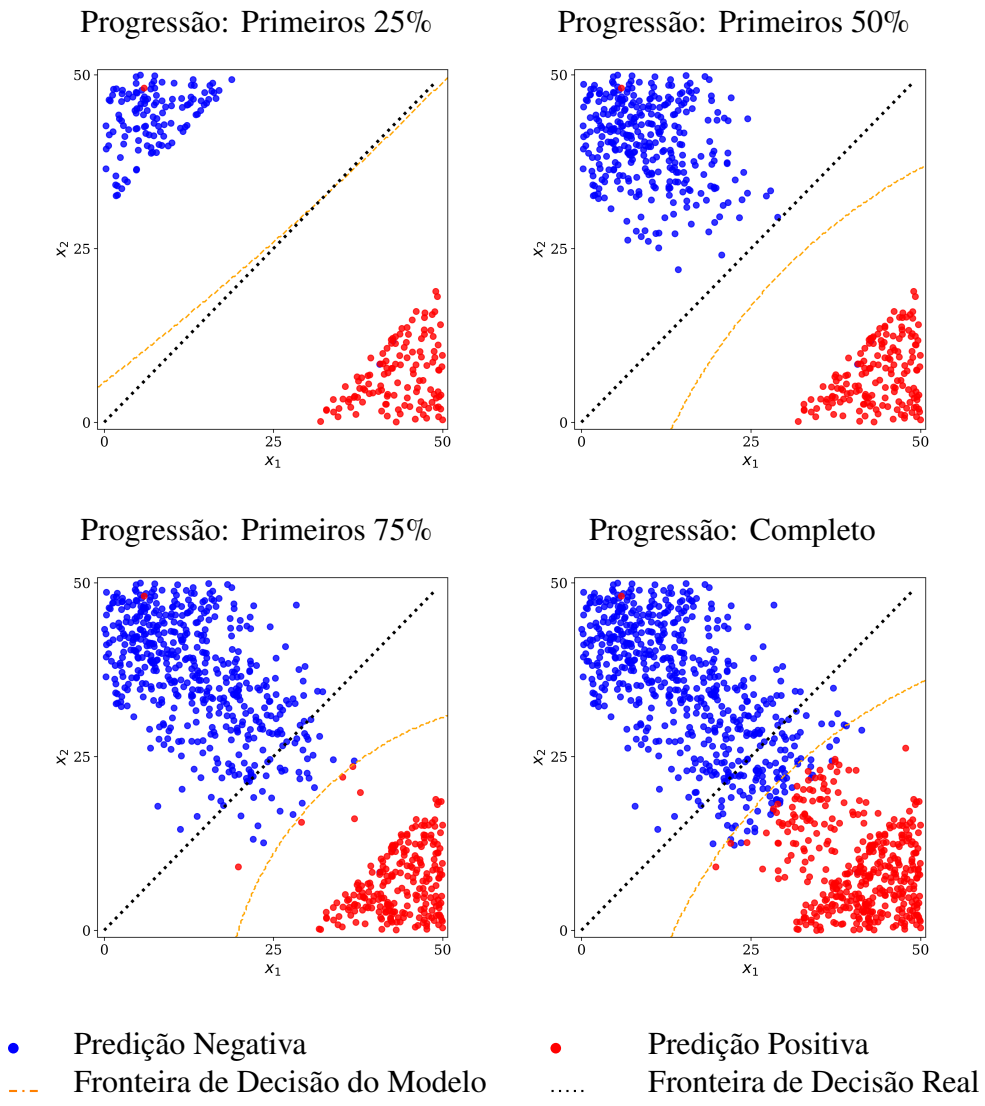



Figura 6.5: Predições do modelo *co-op training* durante o treinamento.

em que sua taxa de falsos negativos cresce conforme o avanço do treinamento, estabilizando em torno de 60%.

Por outro lado, o *cluster-and-label*, que apresentou a maior queda em *acurácia*, também demonstrou o maior aumento em seu *FNR*, que atingiu o pico no final do fluxo com uma proporção de 73%. Por fim, o *co-op training*, também demonstrou uma piora e recuperação em seu *FNR*, que atingiu o valor máximo de 54% e uma posterior queda para 38%.

## 6.3 CONCLUSÃO

Neste capítulo, foram apresentados os resultados dos experimentos realizados para avaliar o impacto do envenenamento de dados em algoritmos semi-supervisionados *online*. A análise visual das fronteiras de decisão dos modelos, junto com as acurácias e taxas de falsos negativos dos algoritmos no conjunto de dados de avaliação, revelaram diferentes níveis de vulnerabilidade ao ataque, com o modelo BDA demonstrando robustez e o *co-op training* apresentando um desempenho intermediário, enquanto os modelos IPA, IDW e *cluster-and-label* foram significativamente afetados.

Tabela 6.1: Acurácia dos modelos semi-supervisionados ao longo do fluxo de dados envenenado.

| **Modelo** | 30% | 40% | 50% | 60% | 70% | 80% | 90% | 100% |
|---|---|---|---|---|---|---|---|---|
| BDA | 95% | 97% | 97% | 97% | 97% | 96% | 96% | 95% |
| IPA | 97% | 89% | 80% | 74% | 71% | 67% | 67% | 67% |
| IDW | 97% | 89% | 80% | 75% | 72% | 68% | 68% | 69% |
| CL | 91% | 84% | 77% | 69% | 65% | 66% | 65% | 63% |
| Co-op | 97% | 89% | 80% | 76% | 73% | 72% | 75% | 81% |

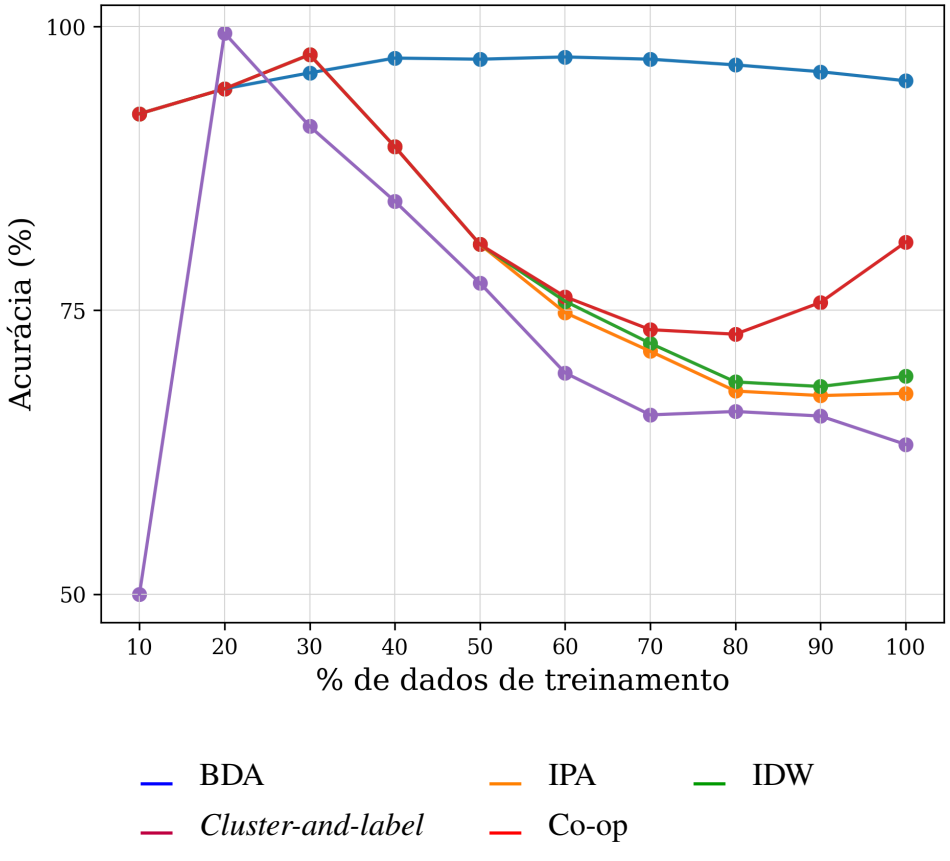



Figura 6.6: Evolução da acurácia dos modelos ao longo do fluxo de dados envenenado.

Tabela 6.2: Taxa de falsos negativos dos modelos semi-supervisionados ao longo do fluxo de dados envenenado.

| **Modelo** | 30% | 40% | 50% | 60% | 70% | 80% | 90% | 100% |
|---|---|---|---|---|---|---|---|---|
| BDA | 0% | 0% | 0% | 0% | 0% | 0% | 0% | 0% |
| IPA | 2% | 21% | 38% | 50% | 57% | 64% | 65% | 64% |
| IDW | 2% | 21% | 38% | 48% | 55% | 62% | 63% | 61% |
| CL | 17% | 30% | 45% | 61% | 68% | 67% | 68% | 73% |
| Co-op | 2% | 21% | 38% | 47% | 53% | 54% | 48% | 38% |

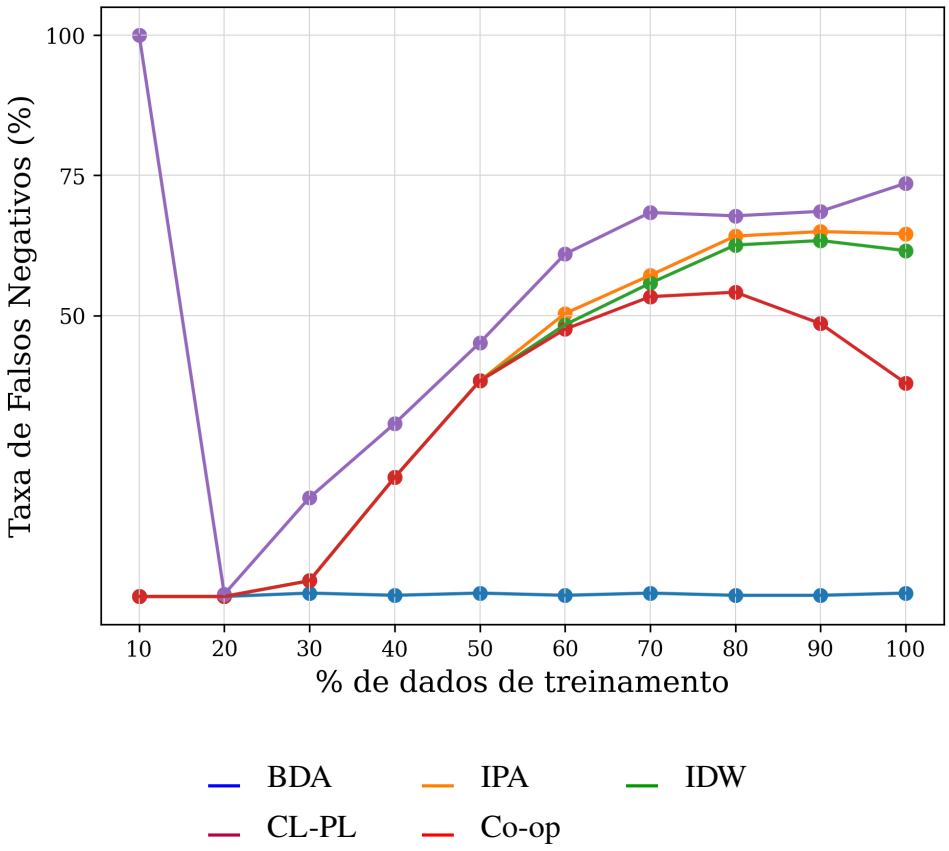



Figura 6.7: Evolução da taxa de falsos negativos (*FNR*) dos modelos ao longo do fluxo de dados envenenado.

# 7 CONCLUSÃO

Neste trabalho, foi proposto um método de envenenamento de dados para avaliação de robustez de modelos de aprendizado semi-supervisionado em fluxos de dados. Um conjunto de dados sintético de envenenamento é criado, e o ataque é dado por instâncias não rotuladas que, inicialmente geradas no *cluster* superior, gradualmente são deslocadas em direção à ao *cluster* inferior. Assim, cinco configurações de algoritmos semi-supervisionados foram testadas, treinados no conjunto de dados de envenenamento e avaliados em dois conjuntos distintos em determinadas etapas do fluxo: um conjunto de grade, para possibilitar a visualização da fronteira de decisão dos modelos, e um conjunto de avaliação, com o propósito de medir a acurácia e taxa de falsos negativos dos modelos ao longo do treinamento.

Desta forma, mostrou-se que o algoritmo *self-training* BDA foi o mais robusto ao envenenamento, não demonstrando impacto significativo ao ataque, dado, principalmente, ao seu tipo de pontuação de confiança baseado na distância entre uma instância não rotulada e os exemplos rotulados de sua classe predita presentes no *batch*, que aumenta conforme o ataque avança. Também, o algoritmo *co-op training* demonstrou um impacto intermediário, conseguindo recuperar sua performance parcialmente ao final do treinamento, efeito dado pelo uso das predições do sub-modelo para a seleção dos pseudo-rótulos, visto que este é treinado apenas com os exemplos rotulados.

Por fim, concluiu-se que os algoritmos *self-training* IPA, IDW e *cluster-and-label* foram os mais suscetíveis ao ataque, demonstrado tanto pela evolução de suas fronteiras de decisão, quanto por suas acurácias e taxas de falsos negativos no conjunto de avaliação ao longo do treinamento. No primeiro caso, o uso da confiança de predição dado pelo modelo base foi o principal fator da suscetibilidade, visto que os pseudo-rótulos das instâncias envenenadas causam um efeito de realimentação de confiança para os pseudo-rótulos posteriores. Similarmente, no IDW ocorre o mesmo efeito, dado que todas as amostras não rotuladas são usadas para treinamento, ponderadas pela confiança de predição do modelo base. Finalmente, a suscetibilidade do ataque no *cluster-and-label* é dada pelo critério de determinação de pseudo-rótulo para as instâncias não rotuladas, que é computado pelo rótulo mais frequente no *cluster* mais próximo, causando, também, um efeito de realimentação do envenenamento.

Assim, este trabalho demonstrou ser possível avaliar o impacto de um ataque por instâncias não rotuladas, demonstrando que esse tipo de algoritmo não deve ser utilizado em ambientes em que adversários maliciosos podem estar presentes, como, por exemplo, em sistemas de monitoramento de intrusão. Como limite do trabalho, entende-se que os conjuntos gerados são simples, e não refletem conjuntos do mundo real. Como trabalho futuro, pode-se explorar outros parâmetros, como diferentes proporções de amostras não rotuladas e número de características, investigar ataques no contexto de mudanças de conceito e avaliar ataques em conjuntos reais.

# REFERÊNCIAS


Aggarwal, C. C., Philip, S. Y., Han, J. e Wang, J. (2003). A framework for clustering evolving data streams. Em *Proceedings 2003 VLDB conference*, páginas 81–92. Elsevier.

Amini, M.-R., Feofanov, V., Pauletto, L., Hadjadj, L., Devijver, E. e Maximov, Y. (2025). Self-training: A survey. *Neurocomputing*, 616:128904.

Berthelot, D., Carlini, N., Goodfellow, I., Papernot, N., Oliver, A. e Raffel, C. A. (2019). MixMatch: A Holistic Approach to Semi-Supervised Learning. Em *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Bifet, A., Holmes, G., Kirkby, R. e Pfahringer, B. (2010). Moa: Massive online analysis. *J. Mach. Learn. Res.*, 11:1601–1604.

Bishop, C. M. (2006). Machine learning. *Mach. Learn*, 128(10.1117):1–2819119.

Blum, A. e Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. Em *Proceedings of the eleventh annual conference on Computational learning theory*, páginas 92–100.

Carlini, N. (2021). Poisoning the unlabeled dataset of Semi-Supervised learning. Em *30th USENIX Security Symposium (USENIX Security 21)*, páginas 1577–1592. USENIX Association.

Cinà, A. E., Grosse, K., Demontis, A., Vascon, S., Zellinger, W., Moser, B. A., Oprea, A., Biggio, B., Pelillo, M. e Roli, F. (2023). Wild patterns reloaded: A survey of machine learning security against training data poisoning. *ACM Comput. Surv.*, 55(13s).

Domingos, P. e Hulten, G. (2000). Mining high-speed data streams. Em *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, páginas 71–80.

Fan, J., Yan, Q., Li, M., Qu, G. e Xiao, Y. (2022). A survey on data poisoning attacks and defenses. Em *2022 7th IEEE International Conference on Data Science in Cyberspace (DSC)*, páginas 48–55.

Fredriksson, T., Mattos, D. I., Bosch, J. e Olsson, H. H. (2020). Data Labeling: An Empirical Investigation into Industrial Challenges and Mitigation Strategies. Em Morisio, M., Torchiano, M. e Jedlitschka, A., editores, *Product-Focused Software Process Improvement*, páginas 202–216, Cham. Springer International Publishing.

Gama, J. (2010). Knowledge Discovery from Data Streams (Chapman & Hall/CRC Data Mining and Knowledge Discovery Series).

Gama, J., Sebastião, R. e Rodrigues, P. P. (2009). Issues in evaluation of stream learning algorithms. Em *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '09, páginas 329–338, New York, NY, USA. Association for Computing Machinery.

Gomes, H. M., Grzenda, M., Mello, R., Read, J., Le Nguyen, M. H. e Bifet, A. (2022). A survey on semi-supervised learning for delayed partially labelled data streams. *ACM Computing Surveys*, 55(4):1–42.

Gomes, H. M., Read, J., Bifet, A., Barddal, J. P. e Gama, J. a. (2019). Machine learning for streaming data: state of the art, challenges, and opportunities. *SIGKDD Explor. Newsl.*, 21(2):6–22.

Haug, J., Tramountani, E. e Kasneci, G. (2022). Standardized Evaluation of Machine Learning Methods for Evolving Data Streams. arXiv:2204.13625 [cs].

Hoi, S. C., Sahoo, D., Lu, J. e Zhao, P. (2021). Online learning: A comprehensive survey. *Neurocomputing*, 459:249–289.

Kumar, A., Kaur, P. e Sharma, P. (2015). A survey on hoeffding tree stream data classification algorithms. *CPUH-Res. J*, 1(2):28–32.

Le Nguyen, M. H., Gomes, H. M. e Bifet, A. (2019). Semi-supervised learning over streaming data using moa. Em *2019 IEEE International Conference on Big Data (Big Data)*, páginas 553–562.

Lee, D.-H. (2013). Pseudo-Label : The Simple and Efficient Semi-Supervised Learning Method for Deep Neural Networks. *ICML 2013 Workshop : Challenges in Representation Learning (WREPL)*.

Liu, X., Si, S., Zhu, X., Li, Y. e Hsieh, C.-J. (2019). A unified framework for data poisoning attack to graph-based semi-supervised learning.

Liu, Y., Yuan, X., Zhao, R., Wang, C., Niyato, D. e Zheng, Y. (2022). Poisoning semi-supervised federated learning via unlabeled data: Attacks and defenses.

Monteiro, P. M., Soares, E. e de Barros, R. S. M. (2021). Co-op training: a semi-supervised learning method for data streams. Em *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, páginas 933–938.

Ramirez, M. A., Kim, S.-K., Hamadi, H. A., Damiani, E., Byon, Y.-J., Kim, T.-Y., Cho, C.-S. e Yeun, C. Y. (2022). Poisoning attacks and defenses on artificial intelligence: A survey. *arXiv preprint arXiv:2202.10276*.

Roh, Y., Heo, G. e Whang, S. E. (2021). A Survey on Data Collection for Machine Learning: A Big Data - AI Integration Perspective. *IEEE Transactions on Knowledge and Data Engineering*, 33(4):1328–1347.

Sohn, K., Berthelot, D., Carlini, N., Zhang, Z., Zhang, H., Raffel, C. A., Cubuk, E. D., Kurakin, A. e Li, C.-L. (2020). FixMatch: Simplifying Semi-Supervised Learning with Consistency and Confidence. Em *Advances in Neural Information Processing Systems*, volume 33, páginas 596–608. Curran Associates, Inc.

Song, Z., Yang, X., Xu, Z. e King, I. (2023). Graph-Based Semi-Supervised Learning: A Comprehensive Review. *IEEE Transactions on Neural Networks and Learning Systems*, 34(11):8174–8194.

Song, Z., Yang, X., Zhang, Y., Fu, X., Xu, Z. e King, I. (2024). A Systematic Survey on Federated Semi-supervised Learning. volume 9, páginas 8244–8252. ISSN: 1045-0823.

van Engelen, J. E. e Hoos, H. H. (2020). A survey on semi-supervised learning. *Machine Learning*, 109(2):373–440.

Wang, Y. e Chaudhuri, K. (2018). Data poisoning attacks against online learning. *CoRR*, abs/1808.08994.

Xie, Q., Dai, Z., Hovy, E., Luong, T. e Le, Q. (2020). Unsupervised Data Augmentation for Consistency Training. Em *Advances in Neural Information Processing Systems*, volume 33, páginas 6256–6268. Curran Associates, Inc.

Zhang, X., Zhu, X. e Lessard, L. (2020). Online data poisoning attacks. Em Bayen, A. M., Jadbabaie, A., Pappas, G., Parrilo, P. A., Recht, B., Tomlin, C. e Zeilinger, M., editores, *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, volume 120 de *Proceedings of Machine Learning Research*, páginas 201–210. PMLR.

Zhu, X. J. (2005). Semi-Supervised Learning Literature Survey. Technical Report, University of Wisconsin-Madison Department of Computer Sciences. Accepted: 2012-03-15T17:19:12Z.