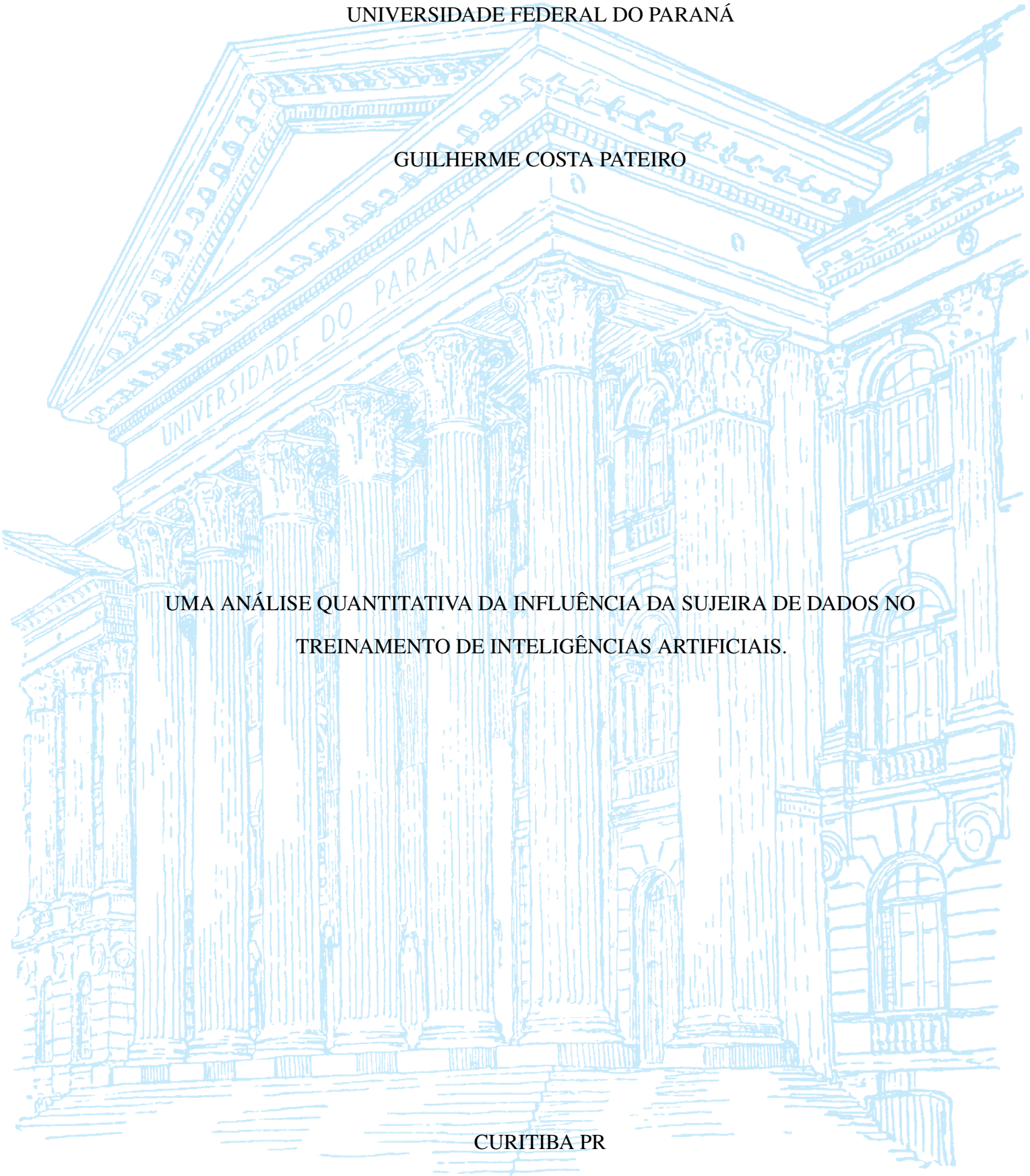
UNIVERSIDADE FEDERAL DO PARANÁ

GUILHERME COSTA PATEIRO



UMA ANÁLISE QUANTITATIVA DA INFLUÊNCIA DA SUJEIRA DE DADOS NO TREINAMENTO DE INTELIGÊNCIAS ARTIFICIAIS.

CURITIBA PR

2023

GUILHERME COSTA PATEIRO

UMA ANÁLISE QUANTITATIVA DA INFLUÊNCIA DA SUJEIRA DE DADOS NO TREINAMENTO DE INTELIGÊNCIAS ARTIFICIAIS.

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: *Ciência da Computação*.

Orientador: Guilherme Alex Derenievicz.

CURITIBA PR

2023

RESUMO

O treinamento de aprendizagem de máquina emerge como um campo crucial no avanço da computação. A sujeira de dados refere-se a qualquer irregularidade, imprecisão ou ruído nos conjuntos de dados utilizados para treinar modelos de inteligência artificial, e sua presença pode ter implicações significativas na eficácia e na generalização desses modelos. Este trabalho propõe uma investigação aprofundada, empregando métodos quantitativos para avaliar como a presença de sujeira de dados afeta o desempenho de algoritmos de aprendizado de máquina. No decorrer deste trabalho, problemas simulados foram empregados, juntamente com conjuntos de dados igualmente simulados, a fim de estabelecer um ambiente controlado para a análise da influência da sujeira no treinamento de aprendizagem de máquina, com foco em redes perceptron múlti camada. Esta abordagem permitiu uma manipulação precisa dos fatores envolvidos, possibilitando a criação de cenários específicos e permitindo isolar os diferentes tipos de viés, evitando a coexistência de diferentes tipos de sujeiras nos conjuntos de dados simulados. Reconhecendo a necessidade de ir além da avaliação tradicional, busca-se a implementação de uma heurística para classificar os erros resultantes desse fenômeno. Essa heurística, baseada na quantidade de erros e na confiança do modelo, visa proporcionar uma compreensão mais aprofundada dos padrões de falha, destacando preocupações cruciais relacionadas à generalização, também demostrando casos onde certas porcentagens de erro não foram capazes de causar erros significativos, enquanto outros casos causaram erros que podem inviabilizar a generalização do modelo. O objetivo é contribuir para o entendimento do papel da sujeira de dados no treinamento de aprendizagem de máquina, fornecendo informações uteis para pesquisadores, desenvolvedores e profissionais envolvidos na criação e implementação de sistemas de aprendizado de maquina robustos.

ABSTRACT

The training of machine learning emerges as a crucial field in the advancement of computing. Data dirt refers to any irregularity, inaccuracy, or noise in the datasets used to train artificial intelligence models, and its presence can have significant implications for the effectiveness and generalization of these models. This work proposes an in-depth investigation, employing quantitative methods to assess how the presence of data dirt affects the performance of machine learning algorithms. Throughout this work, simulated problems were employed, along with equally simulated datasets, in order to establish a controlled environment for the analysis of the influence of dirt on machine learning training, focusing on multi-layer perceptron networks. This approach allowed for a precise manipulation of the factors involved, enabling the creation of specific scenarios and isolating different types of biases, avoiding the coexistence of different types of dirt in the simulated datasets. Recognizing the need to go beyond traditional evaluation, an heuristic is sought for the classification of errors resulting from this phenomenon. This heuristic, based on the quantity of errors and model confidence, aims to provide a deeper understanding of failure patterns, highlighting crucial concerns related to generalization, also demonstrating cases where certain error percentages were unable to cause significant errors, while other cases caused errors that could make impossible the model's generalization. The goal is to contribute to the understanding of the role of data dirt in machine learning training, providing useful information for researchers, developers, and professionals involved in the creation and implementation of robust machine learning systems.

LISTA DE FIGURAS

2.1 Exemplo de árvore de decisão. Imagem por T-kita, em domínio público. 10
2.2 Exemplo de problemas que são e não são linearmente separáveis,. 12
2.3 Exemplo de um neurônio, com entradas X, pesos W, função de ativação phi, e saída O. imagem por Martin Thoma sob licença Creative Commons CC0 1.0 Universal Public Domain Dedication 13
2.4 Exemplo de um MLP, em vermelho os neurônios de entrada, em verde oliva, a camada intermediária, e em verde escuro a camada de saída. imagem por sky-99 sobre licença Creative Commons Attribution-Share Alike 3.0 Unported 14

4.1 Representação de 1 dado 21

5.1 Média de pesos, por bit quando executado com 1 neurônio na camada intermediaria.. 25
5.2 Gráfico com taxas de acerto quando se varia o tamanho da máscara, mascara inserida por substituição. 26
5.3 Gráfico com taxas de acerto quando se varia o tamanho da máscara, máscara inserida por concatenação. 27
5.4 Estrutura de um dado padrão seguindo modelo de concatenação, e um dado esparso.28
5.5 Gráfico com taxas de acerto, com máscara de tamanho 5. 28
5.6 Gráfico com taxas de acerto, com máscara de tamanho 5 esparsa e bit útil no meio. 29
5.7 Tabela com taxas de acerto médias para cada tamanho de arquitetura.. 29
5.8 Média de pesos, por bit, quando executado com 1 neurônio na camada intermediaria, com 2 bits para classificar.. 30
5.9 Gráfico com taxas de acerto, com máscara de tamanho 5 e 2 bits para classificar.. 31
5.10 Comparação de taxas de erros entre o problema do par/impar e o dos múltiplos de 4, com relação ao tamanho da arquitetura. 32
5.11 Taxas de erros quando quantidade de viés é variável. 33
5.12 Taxas de erros com problemas de 109 bits e máscara de tamanho variável. 34
5.13 Gráfico com taxas de acerto com máscara de tamanho 80 e variando o tamanho dos dados úteis. 34
5.14 Gráfico com taxas de acerto com problemas de 109 bits e diferentes tamanhos de arquitetura. 35
5.15 Gráfico com taxas de acerto, com valor de erro fixo e 2 classes. 36
5.16 Gráfico com taxas de acerto, com valor de erro fixo e 4 classes. 37
5.17 Gráfico com taxas de acerto, com quantidade de erro variável, erro presente em todos os bits, 2 classes. 38

5.18 Gráfico com taxas de acerto, com quantidade de erro variável, erro presente em todos os bits neutros, 2 classes. 39
5.19 Gráfico com taxas de acerto, erro somente no bit útil, 2 classes. 40
5.20 Gráfico com taxas de acerto, erros no classificador, duas classes. 41
5.21 Gráfico com taxas de acerto, erros no classificador, quatro classes. 41
5.22 Gráfico com taxas de acerto, erros no classificador em ambas as classes, 2 classes. 42

SUMÁRIO

1 INTRODUÇÃO 8
2 FUNDAMENTAÇÃO TEÓRICA 9
2.1 TIPOS DE PROBLEMAS ABORDADOS 9
2.1.1 Classificação 9
2.1.2 Regressão 9
2.2 DEFINIÇÃO DOS ALGORITMOS 10
2.2.1 Árvores de Decisão 10
2.2.2 Lista de Decisão 11
2.2.3 Rede Perceptron 11
2.2.4 Multi Layer Perceptron 14
2.3 TIPOS DE SUJEIRA 16
2.3.1 Dados Errados 16
2.3.2 Dados Enviesados 16
2.3.3 Dados Incompletos 17
2.3.4 Outliers 17
3 REVISÃO DA LITERATURA 18
4 METODOLOGIA 20
4.1 DADOS NEUTROS, ÚTEIS E O TREINAMENTO PERFEITO 20
4.2 GERAÇÃO DE DADOS E MÁSCARAS 21
4.3 ARQUITETURA ESCOLHIDA E PARÂMETROS DE TREINO 22
5 RESULTADOS EXPERIMENTAIS 24
5.1 DADOS ENVIESADOS 24
5.1.1 Influência de Cada Bit e Dados Neutros 24
5.1.2 Tamanho da Máscara 25
5.1.3 Dados Úteis e Viés Esparsos 27
5.1.4 Classificadores Com Mais De 1 Bit de Dados Úteis 30
5.1.5 Presença Parcial de Viés 32
5.1.6 Escalabilidade 33
5.2 DADOS ERRADOS 35
5.2.1 Amplitude de Erro 36
5.2.2 Erros em Toda a Entrada 37
5.2.3 Erros Somente nos Dados Neutros 38
5.2.4 Erros Somente nos Dados Úteis 39
5.2.5 Erro no Classificador 40

5.2.6 Erro no Classificador de Todas as Classes 42
6 CONCLUSÃO 44
6.1 TRABALHOS FUTUROS 45
REFERÊNCIAS 46

1 INTRODUÇÃO

A presença de dados sujos, também conhecidos como dados não autorizados ou *rogue data* em inglês, é uma questão recorrente e de grande relevância na área da computação, especialmente no campo da análise de dados e inteligência artificial. Pesquisas realizadas nos Estados Unidos da América (EUA) revelaram que empresas enfrentam prejuízos superiores a 600 bilhões de dólares anualmente devido a dados sujos (Chu, 2004). Esse problema tem sido objeto de estudo abrangente, dada sua significativa influência no ambiente corporativo, especialmente com o advento do big data e o crescimento atual da aprendizagem de máquina e do deep learning.

Dados sujos são informações contaminadas por imprecisões, irregularidades, ruído ou vieses. Por exemplo, em um conjunto de dados de pesquisa, valores mal registrados ou outliers podem introduzir sujeira nos resultados. Da mesma forma, dados tendenciosos, informações incorretas ou incompletas constituem exemplos de dados sujos.

O presente estudo aborda uma lacuna identificada na literatura, na qual muitos artigos se concentram na avaliação de algoritmos de limpeza de dados, com pouca ênfase na própria natureza da sujeira de dados (como observado em (Qi et al., 2021) e (Taghvaei et al., 2017)); outros artigos abordam a tentativa de identificar ou classificar dados como sujos; ou adotam uma abordagem simplificada ao descrever os efeitos negativos da sujeira de dados, como afirmar que sujeira de dados gera imprecisão no treinamento, ou afirmar que ouliers existem em um conjunto de dados, sem fornecer uma quantificação precisa desses efeitos (como apresentado em (Ressan e Hassan, 2022) e (Ghiassi et al., 2019)).

O objetivo deste estudo é preencher essa lacuna por meio de uma análise aprofundada da sujeira de dados, utilizando uma metodologia experimental que compara casos onde um tipo de sujeira se faz presente tando no conjunto de treino quando no de teste.

Para isso, delineamos um conjunto de problemas de classificação caracterizados por lógicas distintas, além de estabelecer um conjunto de treinamento específico para esses desafios. A fim de abordar a presença de dados sujos, desenvolvemos variações para esses conjuntos de treinamento, isolando dados por tipo de sujeira. Consequentemente, geramos conjuntos de testes abrangendo diferentes categorias de sujeira identificadas.

Em seguida, procedemos ao treinamento de uma rede neural do tipo perceptron de múltiplas camadas utilizando esses conjuntos de dados ao longo de um número predefinido de épocas. Buscamos, assim, analisar os resultados do treinamento quando a rede é confrontada com testes enviesados, considerando um cenário adverso de dados de teste.

Por fim, busca-se determinar uma heurística para ser utilizada; definir a quantidade de sujeira de dados necessária para que um sistema comece a apresentar inconsistências, considerando diversos tipos de sujeira de dados, e buscar estabelecer um valor numérico ou uma porcentagem aceitável que não afete significativamente os resultados.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo discorrerá sobre diversos algoritmos empregados no contexto do aprendizado de máquina, concentrando-se particularmente naqueles que serão aplicados no âmbito deste trabalho. Adicionalmente, serão abordados de maneira mais aprofundada os conceitos relativos à sujeira de dados, elucidando seus distintos tipos e características.

2.1 TIPOS DE PROBLEMAS ABORDADOS

Uma tarefa de aprendizado de máquina refere-se a capacitar um modelo de computador a aprender com os dados disponíveis e fazer previsões com base nesse aprendizado para resolver um problema específico ou alcançar uma meta.

Quando o número de saídas de uma tarefa for um conjunto finito de valores, o problema de aprendizagem será chamado de classificação e as saídas possíveis do problema serão chamadas de rótulos. Caso o problema se trate de encontrar uma expectativa de valor numérico, o problema será de regressão. (Russell e Norvig, 2010)

Por exemplo, baseado em dados climáticos de uma região, prever a temperatura de um dia é um problema de regressão, pois, a temperatura é uma escala numérica continua, enquanto prever se o dia estará ensolarado, nublado ou chuvoso é um problema de classificação, pois, de trata de um conjunto finito de valores.(Russell e Norvig, 2010)

2.1.1 Classificação

Um problema de classificação refere-se a uma tarefa de aprendizado de máquina em que o objetivo é atribuir uma ou mais categorias ou rótulos a um conjunto de dados. Essas categorias ou rótulos são geralmente predefinidos e representam diferentes classes ou grupos aos quais os dados podem pertencer. O objetivo é construir um modelo que seja capaz de aprender a mapear corretamente as entradas (características) para as saídas (categorias) correspondentes.

Se houverem apenas duas saídas (categorias) para o problema, o problema de classificação será chamado de classificação booleana ou binária. (Russell e Norvig, 2010)

Em um problema de classificação supervisionado, os dados de treinamento são compostos por exemplos rotulados, onde cada exemplo consiste em uma combinação de características (ou atributos) e o rótulo associado a ele. O modelo de aprendizado de máquina é treinado com esses dados rotulados para aprender padrões e relações entre as características e os rótulos correspondentes. Em seguida, o modelo pode ser usado para classificar novos exemplos não rotulados, atribuindo-lhes uma ou mais classes previstas com base em suas características.

2.1.2 Regressão

Um problema de regressão refere-se a uma tarefa de aprendizado de máquina em que o objetivo é prever um valor numérico com base em um conjunto de variáveis de entrada. (Alpaydin, 2010)

Em um problema de regressão, os dados de treinamento consistem em pares de entrada e saída, onde a entrada é composta por um conjunto de características (ou atributos) e a saída é o valor numérico que se deseja prever. O modelo de aprendizado de máquina é treinado com esses dados para aprender a relação entre as características de entrada e os valores de saída correspondentes.

2.2 DEFINIÇÃO DOS ALGORITMOS

Ao longo deste trabalho, os seguintes algoritmos foram estudados com objetivo de descobrir bons candidatos a treinamento e posterior análise de dados de treinamento com sujeira de dados.

2.2.1 Árvores de Decisão

Uma árvore de decisão é uma representação funcional que recebe como entrada um vetor contendo valores atribuídos aos atributos e retorna uma "decisão". A árvore de decisão alcança essa decisão por meio da execução de uma sequência de testes. Cada nó interno da árvore corresponde a um teste realizado em relação aos valores dos atributos. (Russell e Norvig, 2010)

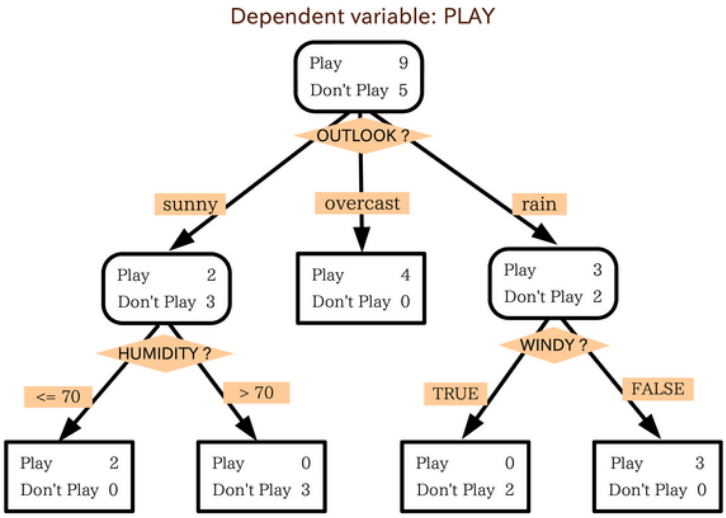


Figura 2.1: Exemplo de árvore de decisão. Imagem por T-kita, em domínio público.

Durante a etapa de construção da árvore de decisão, é necessário determinar qual característica é a mais adequada para dividir os dados em cada nó, a fim de maximizar a pureza das classes resultantes. No entanto, uma das dificuldades associadas ao uso de árvores de decisão é o seu custo computacional elevado durante a etapa de construção. O tamanho de uma árvore pode ser de ordem $O(2^{2^n})$, onde n é o número de atributos, se todas as possíveis opções de árvore (com redundância) forem geradas. Porém, é importante ressaltar que a execução de uma árvore de decisão é realizada em tempo linear O(n), o que torna esse método altamente eficiente durante a fase de execução. (Russell e Norvig, 2010)

Com o intuito de contornar a problemática relacionada ao alto custo computacional para gerar a árvore de decisão ótima, uma abordagem gulosa é empregada para encontrar uma árvore suficientemente boa, embora não necessariamente a melhor (Russell e Norvig, 2010). Esse algoritmo baseia-se em identificar, no conjunto de treinamento, subconjuntos nos quais todos os elementos apresentam o mesmo resultado para uma determinada escolha. Por exemplo, considerando os dias da semana, todas as quartas-feiras apresentam o mesmo resultado. Nesse caso, a decisão é formulada para englobar esses elementos do conjunto, eliminando todas as quartas-feiras do conjunto de testes, uma vez que elas já possuem uma decisão de classificação pré-determinada. Esse processo é repetido com o conjunto de entrada remanescente. No caso em que um valor não seja unânime, a decisão é baseada na maioria dos valores.

No entanto, um desafio significativo durante o treinamento é que a árvore resultante pode não estar adequadamente preparada para lidar com casos que não se assemelham aos casos

de treinamento. Essa situação reflete a insuficiência do conjunto de treinamento, e quanto maior o tamanho desse conjunto, maior será o tempo necessário para construir a árvore. No entanto, um conjunto de treinamento mais amplo aumenta a confiabilidade dos resultados. (Russell e Norvig, 2010)

Outro problema que pode surgir é a superadaptação, um fenômeno comum a todos os algoritmos de aprendizado de máquina, no qual o algoritmo toma decisões que não se alinham com a generalização do problema, mas sim com respostas específicas para o conjunto de treinamento utilizado. (Russell e Norvig, 2010)

2.2.2 Lista de Decisão

Uma lista de decisão consiste em um conjunto de testes, nos quais cada teste é composto pela combinação de literais. Se um teste for bem-sucedido, a lista retorna um valor; caso contrário, a lista avança para o próximo teste. (Russell e Norvig, 2010)

As listas de decisão compartilham semelhanças com árvores de decisão, pois, quando os testes não possuem um limite de tamanho, é possível generalizar qualquer função booleana por meio de uma lista de decisão. No entanto, se um teste tiver um limite de K literais, o algoritmo é capaz de generalizar um número reduzido de exemplos, resultando em uma linguagem conhecida como K-DL. (Russell e Norvig, 2010)

O algoritmo para a geração eficiente de uma lista de decisão baseia-se em uma abordagem gulosa, na qual um teste que corresponde exatamente a um subconjunto do conjunto de treinamento é repetidamente identificado e adicionado à lista de decisões, ao mesmo tempo em que os exemplos correspondentes são removidos. A fim de garantir que a lista de decisão seja do tipo PAC (provavelmente aproximadamente correta), é necessário que o conjunto de treinamento para uma função K-DL seja polinomial em relação ao número N de atributos (Russell e Norvig, 2010), ou seja para que a lista de decisão apresente um comportamento aproximadamente correto, existe um conjunto mínimo de treinamento que precisa ser inserido no treinamento.

2.2.3 Rede Perceptron

Um perceptron é um modelo básico de neurônio artificial. Uma rede perceptron é uma estrutura que engloba um conjunto de perceptrons, essa rede é capaz de generalizar modelos lineares. Os modelos lineares são um conjunto de métodos de aprendizado de máquina usados para estabelecer a relação entre variáveis. Esses modelos assumem que essa relação é linear, o que significa que as mudanças em uma variável estão diretamente relacionadas às mudanças em outra variável, multiplicadas por um coeficiente constante. Um modelo linear é a regressão linear, que procura encontrar a melhor linha (ou hiperplano em espaços de maior dimensão) que se ajusta aos dados observados. (Russell e Norvig, 2010)

O conceito de modelos linearmente separáveis denota conjuntos de dados que possuem a propriedade de serem divisíveis por uma linha (ou hiperplano) de modo que todas as instâncias pertencentes a uma classe estejam localizadas de um lado da linha, enquanto todas as instâncias da outra classe estejam situadas no lado oposto(Russell e Norvig, 2010). Um exemplo de problema linearmente separável é operação AND, enquanto a operação XOR não.

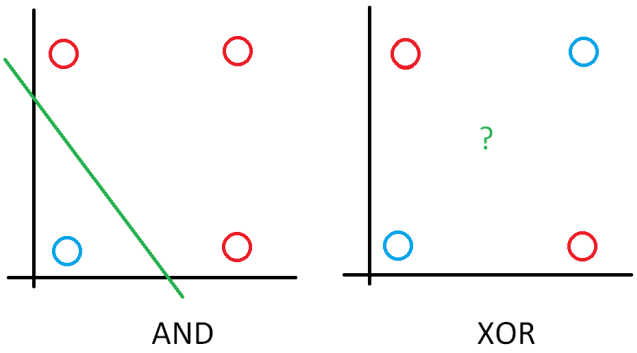


Figura 2.2: Exemplo de problemas que são e não são linearmente separáveis,

No espaço bidimensional, problemas de regressão linear apresentam como solução uma reta. A equação da reta pode ser dada em fator de (x, y) na forma de $y = w_1x + w_0$, sendo w_0 e w_1 coeficientes, também chamados pesos. Assim, o treinamento de um sistema linearmente separável é fazer uma regressão linear, com objetivo de minimizar a perda empírica. É comum, utilizar a função de perda quadrática, que envolve a soma dos quadrados das diferenças entre os valores previstos e reais em todos os exemplos de treinamento. (Russell e Norvig, 2010)

O treinamento do problema linearmente separável é dado iterando o método do gradiente descendente, até encontrar a convergência. Para esse trenamento é definida um valor α chamado de **taxa de aprendizagem**. (Russell e Norvig, 2010)

Assim a equação que deve ser resolvida a cada interação se da por:

$$w_0 = w_0 + \alpha(y - h_w(x)) \tag{2.1}$$

e

$$w_1 = w_1 + \alpha x(y - h_w(x)) \tag{2.2}$$

sendo $h_w(x)$ o resultado atual da equação linear, ou seja $h_w(x) = w_1x + w_0$. Como se pode perceber, quanto maior a diferença entre o resultado obtido e o resultado esperado y, maior será a mudança nos pesos. (Russell e Norvig, 2010)

Tanto o número de pesos quanto o numero de variáveis x de entrada podem ser generalizada para qualquer tamanho fazendo com que $h_w(x)$ se torne o produto escalar entre um vetor de valores x e um vetor de pesos w. E o treinamento de um peso se da por:

$$w_i = w_i + \alpha \sum_j x_{i,j}(y_j - h_w(x_j)) \tag{2.3}$$

A regressão univariada (com somente w_0 e w_1) não são afetadas por superadaptação, porém as multivariadas (com um valor maior de pesos) pode sofrer de superadaptação. Para lidar com esse problema, é comum o uso de regularização , como por exemplo o L_1 que minimiza a soma dos valores ou L_2 que minimiza a soma dos quadrados. (Russell e Norvig, 2010)

Até o momento, foi discutido como funciona um problema de regressão linearmente separável. No entanto, é possível realizar uma classificação ao modificar a lógica por trás da linha gerada. Essa linha é agora denominada **fronteira de decisão** , ela divide o plano onde está em dois sub-planos onde cada sub-plano tem um conjunto com a mesma característica da classificação. Os dados que podem ser divididos dessa maneira são chamados de linearmente separáveis. (Russell e Norvig, 2010)

Para um problema de classificação binária, os valores que a função $h_w(x)$ deve assumir mudaram para um conjunto de apenas dois valores: 0 ou 1. Para isso $h_w(x)$ agora representa a função limiar. Funções limiares, tem forma: $h_w(x) = Limiar(w \cdot x)$, onde $Limiar(n) = 1$, se $n \geq 0$, ou 0 caso contrario. (Russell e Norvig, 2010)

O Problema pode ser treinado pela equação 2.3, no caso de apenas 2 classes, y assumirá valores de 0 ou 1. Como $h_w(x)$ e y podem assumir apenas valores 0 ou 1, $(y_i - h_w(x))$ pode assumir somente 3 valores, -1, 0 ou 1. Essa regra de aprendizado é chamada de "regra de aprendizado perceptron". (Russell e Norvig, 2010)

Um aspecto importante do treinamento do perceptron é que, especialmente para dados que não são linearmente separáveis, é benéfico reduzir a taxa de aprendizado à medida que as iterações avançam. Isso ajuda a diminuir a taxa de erro e o ruído durante o treinamento. (Russell e Norvig, 2010)

A função de ativação não precisa necessariamente ser o produto escalar entre os pesos (w) e as entradas (x). Outras funções de ativação podem ser definidas, e algumas podem ser mais adequadas que o produto escalar. Um exemplo de função de ativação alternativa é a função logística, também conhecida como sigmoide. Essa função tem a forma:

$$logistica(w \cdot x) = \frac{1}{1 + e^{-w \cdot x}} \tag{2.4}$$

A função logística tem uma forma que se aproxima mais da curva normal, e o limiar de decisão é o valor intermediário da função (geralmente 0,5). (Russell e Norvig, 2010)

Por fim, basta abordar a parte central da rede perceptron. Inicialmente, é necessário definir o conceito de neurônio. Um neurônio é uma unidade fundamental que recebe um conjunto de entradas, representadas pelo vetor x. Internamente, o neurônio possui pesos w associados a essas entradas, e ele gera uma saída por meio de uma função de ativação, calculando a soma ponderada entre x e w. Os neurônios podem ser treinados utilizando a regra de aprendizado perceptron. (Russell e Norvig, 2010)

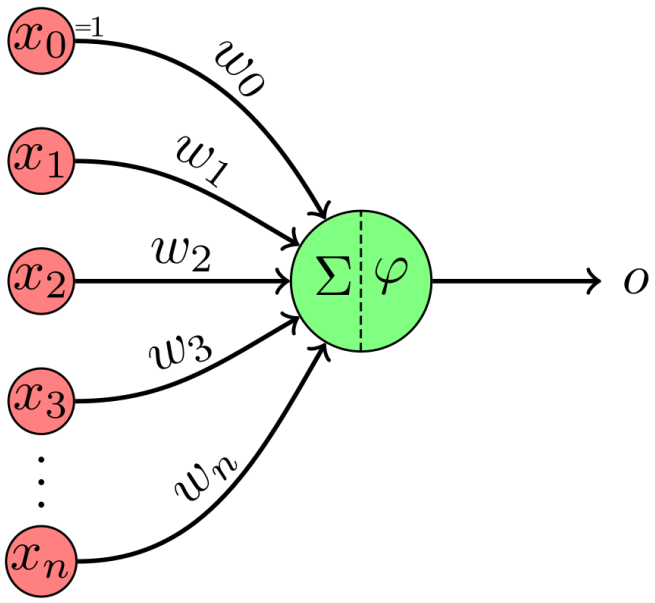


Figura 2.3: Exemplo de um neurônio, com entradas X, pesos W, função de ativação phi, e saída O. imagem por Martin Thoma sob licença Creative Commons CC0 1.0 Universal Public Domain Dedication

Redes neurais, de forma simplificada, são conjuntos de neurônios interconectados. Geralmente, esses neurônios estão organizados em camadas, em que os neurônios de uma camada recebem entradas provenientes das unidades das camadas anteriores. Se a rede possui apenas duas camadas, uma de entrada e uma de saída, ela é denominada **rede de camada única**. Por outro lado, se existem uma ou mais camadas intermediárias entre a entrada e a saída, a rede é chamada de **rede multicamada**. No contexto de redes multicamadas, todos os neurônios que não pertencem às camadas de entrada ou saída são referidos como "unidades ocultas". (Russell e Norvig, 2010)

No caso em que as conexões da rede ocorrem somente em uma direção, ou seja, um neurônio nunca está conectado a uma camada subsequente, a rede é denominada **rede feedforward** ou **rede com alimentação para frente**, formando um grafo direcionado acíclico. A presença de pelo menos uma conexão que não segue essa regra transforma o sistema em uma "rede recorrente". (Russell e Norvig, 2010)

Portanto, uma rede perceptron é uma rede neural de camada única com alimentação para frente. Nessa estrutura, todos os neurônios de entrada estão conectados a todos os neurônios de saída. (Russell e Norvig, 2010)

2.2.4 Multi Layer Perceptron

Um multi layer perceptron (MLP) é uma rede neural multicamada com alimentação para frente, onde todos os neurônios de uma camada recebem como entrada, todos os neurônios da camada anterior. (Russell e Norvig, 2010)

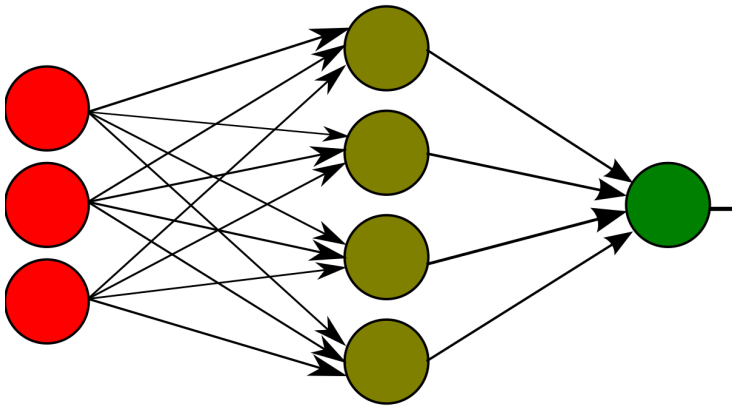


Figura 2.4: Exemplo de um MLP, em vermelho os neurônios de entrada, em verde oliva, a camada intermediária, e em verde escuro a camada de saída. imagem por sky-99 sobre licença Creative Commons Attribution-Share Alike 3.0 Unported

Ao contrário do perceptron, o MLP tem a capacidade de atuar como um aproximador universal para qualquer problema contínuo dentro de um intervalo fechado. A demonstração matemática desse fato estabelece que qualquer caso ou região de um plano pode ser delimitado por dois hiperplanos. O primeiro hiperplano é posicionado na primeira camada da rede, também conhecida como camada de entrada. O segundo hiperplano é localizado na segunda camada, denominada camada oculta. Por fim, a terceira camada, conhecida como camada de saída, realiza uma operação de conjunção (AND) com as duas regiões delimitadas pelos hiperplanos da camada anterior. (Alpaydin, 2010)

O treinamento de um MLP consiste na atualização dos pesos de todos os neurônios presentes na rede. Esse processo de atualização é realizado por meio de um algoritmo conhecido como "backpropagation". O algoritmo de backpropagation soluciona o problema de calcular a diferença entre o valor alvo y_i e a saída $h_w(x)$através da aplicação da regra da cadeia. É importante ressaltar que, nas camadas ocultas da rede, não há um valor y correspondente que possa ser diretamente comparado com a saída gerada pelo neurônio. (Alpaydin, 2010)

A essência do algoritmo de backpropagation consiste em transferir uma fração do erro de um neurônio para todos os neurônios da camada anterior. Essa propagação do erro é realizada de forma proporcional ao peso das conexões entre os neurônios. Dessa forma, os erros são distribuídos de maneira ponderada, levando em consideração a importância de cada neurônio na contribuição para o erro total. Com a contribuição do erro definida os pesos podem ser atualizados por exemplo pela técnica do gradiente descendente, permitindo a minimização gradual do erro ao longo das iterações. (Russell e Norvig, 2010)

As MLPs podem ser submetidas a métodos de treinamento visando aprimorar a eficiência do processo e reduzir o ruído resultante do treinamento. Duas técnicas relevantes nesse contexto são o **momentum** e a **taxa de aprendizagem adaptativa**, ambas as técnicas podem ser usadas concomitantemente ao backpropagation, alterando levemente como os pesos são atualizados.

A técnica do **momentum** consiste em utilizar o resultado gerado pela mesma entrada na iteração anterior como parte da atualização dos pesos durante a iteração atual. Por exemplo, se um peso foi aumentado na iteração anterior, na iteração subsequente, caso esse peso volte a aumentar, seu incremento será maior; da mesma forma, caso diminua, sua redução será menor. Essa abordagem leva em consideração a influência da atualização anterior no processo de ajuste dos pesos, proporcionando uma maior coerência e estabilidade nas atualizações. (Alpaydin, 2010)

Já a técnica da **taxa de aprendizagem adaptativa** consiste em modificar a taxa de aprendizagem à medida que a rede neural alcança taxas de acerto mais elevadas. Conforme mencionado anteriormente, a redução da taxa de aprendizagem tem o intuito de mitigar o ruído gerado durante o treinamento. Essa adaptação permite ajustar a velocidade de aprendizado da rede de acordo com seu desempenho, tornando-a mais sensível às necessidades específicas do problema em questão. (Alpaydin, 2010)

Quando uma rede neural assume uma dimensão consideravelmente extensa, é possível observar uma prevalência significativa de pesos nulos, um fenômeno que impacta negativamente a capacidade de generalização e pode conduzir a um estado de superadaptação. Com o intuito de determinar o tamanho ideal da rede que melhor se adéqua ao problema em questão, duas abordagens se destacam: a construtiva e a destrutiva. (Alpaydin, 2010)

A abordagem construtiva fundamenta-se na implementação de uma rede neural inicialmente pequena, que é então treinada até que o progresso estagne. Em seguida, um novo neurônio é adicionado à rede para lidar com uma porção específica do conjunto de treinamento que ainda não apresenta um nível satisfatório de generalização. Esse processo é repetido iterativamente, com a inclusão de novos neurônios à medida que se busca aprimorar a capacidade de generalização da rede. (Alpaydin, 2010)

Por outro lado, a técnica destrutiva adota uma abordagem inversa. Inicialmente, uma rede neural de grandes dimensões é criada e, posteriormente, são removidos os neurônios que apresentam um número significativo de pesos nulos. Essa estratégia visa forçar a rede a se adaptar aos possíveis pesos não nulos que cada neurônio possuía, demandando uma readaptação do modelo aos padrões de conectividade mais relevantes. (Alpaydin, 2010)

Para o método construtivo, existem 2 abordagem de criação de neurônios, a criação dinâmica, que cria neurônios em camadas existentes, e a criação em cascata que cria um novo

neurônio que recebe ligação de todos os neurônios existentes na rede, nesse caso cada neurônio está em uma nova camada com somente 1 neurônio (Alpaydin, 2010)

Alguns problemas enfrentados ao utilizar MLPs podem estar relacionados à dependência temporal dos dados. Para lidar com esses problemas, existem abordagens possíveis. Uma delas é a introdução de um atraso (delay) e uma sequência de entrada que desliza ao longo do tempo. Essa abordagem envolve o descarte dos dados mais antigos e a incorporação dos dados mais recentes, semelhante a uma fila. (Alpaydin, 2010)

Outra opção é utilizar uma arquitetura de rede recorrente, na qual as conexões retroalimentadas na rede funcionam como uma memória de curto prazo dos eventos passados. Essa configuração permite que a rede aprenda e mantenha informações sobre o que ocorreu anteriormente, tornando-a capaz de lidar com a dependência temporal dos dados. (Alpaydin, 2010)

2.3 TIPOS DE SUJEIRA

Sujeira de dados é definida como qualquer dado que está incompleto, inválido ou impreciso; em resumo qualquer dado que está de alguma forma errado (Chu, 2004). Esses dados podem levar a análises incorretas e decisões equivocadas. Se os dados utilizados para treinar modelos de aprendizado de máquina estiverem sujos, isso pode resultar em modelos imprecisos ou enviesados.

2.3.1 Dados Errados

Dados errados ou incorretos são caracterizados por uma incoerência lógica entre a saída real desejada e a saída apresentada pelo próprio dado durante o processo de treinamento. Para ilustrar esse conceito, considere um problema em que se deseja determinar se um número é par ou ímpar. Nesse contexto, um dado incorreto seria, por exemplo, o número 5, que foi erroneamente associado à saída esperada "par"no conjunto de treinamento.

Além disso, outro tipo de erro de base de dados que merece destaque é a presença de informações inválidas provenientes de memória não alocada corretamente ou de entradas aleatórias, resultando em saídas igualmente aleatórias.

2.3.2 Dados Enviesados

Dados enviesados são uma distorção presente no conjunto de dados, resultante de uma inadequação entre o subconjunto de treinamento utilizado e a amostragem completa da população em questão. No contexto de um problema que visa determinar se um número é par ou ímpar, um conjunto de dados enviesado poderia ser exemplificado por uma amostra em que todos os números pares também são múltiplos de 6, ou ainda por um conjunto de dados que contém apenas valores pares abaixo de um determinado limiar.

Durante o processo de treinamento, um sistema que utiliza dados enviesados pode chegar a conclusões que são coerentes apenas com o conjunto de treinamento em questão, mas que não são generalizáveis para outras situações. Esse viés introduzido nos dados pode levar a uma falta de representatividade da diversidade presente na população total, comprometendo, assim, a capacidade de generalização do sistema.

2.3.3 Dados Incompletos

Dados incompletos ocorrem quando parte ou todo o dado de um conjunto está faltando, por exemplo, se a entrada de um problema for uma linha de uma tabela de um banco de dados, a ausência de uma coluna, ou do registo completo de uma linha. Muitos algoritmos acusariam erro ou pararíam por causa de uma entrada nula.

2.3.4 Outliers

Outliers podem ser vistos como sendo um tipo de viés de dado, mas diferente da proporção de saídas não condizer com a realidade do conjunto, o peso da saída tem uma grande influência no momento da generalização. Por exemplo, se uma rede é treinada com valores de entrada pequenos (digamos 1 a 1000) e depois com valores muito maiores (na casa dos milhões ou bilhões), pode ocorrer dos valores muito grandes terem um peso para a rede muito maior que os valores pequenos, desbalanceando o sistema.

3 REVISÃO DA LITERATURA

O artigo mais próximo ao tema proposto foi (Qi et al., 2021). Em seu projeto eles testam a eficacia de vários algoritmos de limpeza de databases, mesmo com o foco na limpeza de dados, eles apresentam comparações entre os dados limpos e os dados sujos sem a limpeza feita, inclusive com diferentes tipos de erro, porém os autores não falam de outras características mais especificas de algoritmos de aprendizado de maquina como tempo de treinamento e tamanho dos dados, que podem ser causa de dados enviesados, além de não possuir um exemplos relacionados com MLPs, que são o foco desse trabalho.

(Qi e Wang, 2021) que tem parte dos autores do primeiro artigo, fizeram um experimento parecido com esse primeiro citado, agora para problemas de regressão. Os temas abordados e o foco da pesquisa são os mesmos do anterior, abordando uma análise das diferentes soluções.

A regularização de dados é dada com uma tarefa que visa evitar problemas como a superadaptação, porém (Taghvaei et al., 2017) apresenta uma pesquisa que indica que a regularização de um problema pode afetar os pontos criticos de uma função, o que impediria o algoritmo dos mínimos quadrados de achar com certeza o mínimo global, diminuindo a robustez do algoritmo. No entanto, é importante notar que, enquanto esse estudo aponta para possíveis implicações da regularização nos pontos críticos, ele pode não abordar de maneira abrangente os desafios decorrentes dos próprios dados utilizados no processo. Erros inerentes aos dados, não são abordados de forma específica nesta análise.

Novas abordagens de correção de dados são abordadas como no caso de (Cerda et al., 2018), a abordagem proposta busca encontrar similaridades entre as categorias e atribuir valores numéricos com base nessas similaridades. Isso permite lidar de forma mais eficiente com categorias sujas ou com alta cardinalidade. Embora ofereça uma solução, ele não explora detalhadamente a natureza e a extensão dos problemas relacionados aos dados defeituosos ou de alta cardinalidade. Assim, enquanto a solução proposta é valida, seria enriquecedor se houvesse uma discussão mais abrangente sobre a complexidade dos desafios inerentes aos dados e como essa solução específica se encaixa no contexto mais amplo da qualidade dos dados.

O artigo de (Ressan e Hassan, 2022) aborda a limpeza de dados como um processo essencial antes da utilização desses dados em conjuntos de treinamento.Sua abordagem é mais abrangente do que o trabalho de (Ghiassi et al., 2019), que se concentra na limpeza de dados em bancos de dados específicos, como mídias provenientes de bancos de imagens e redes sociais. Ambos os estudos têm como foco o pré-processamento e a limpeza de bancos de dados antes do treinamento.

Já o artigo de (L'Heureux et al., 2017) discute os desafios enfrentados no treinamento de aprendizado de máquina com big data, apresentando perspectivas e obstáculos, como a heterogeneidade, a linearidade, a sujeira de dados e a incerteza dos dados. A sujeira de dados é abordada como um ponto de incerteza e um desafio a ser enfrentado, e o artigo oferece uma análise mais aprofundada sobre o assunto, o que é compreensível considerando o contexto de utilização de big data.

Um assunto que nesse trabalho será pouco abordado, mas ainda relevante, é como identificar um dados sujos, aqui os dados errados e enviesados serão pré definidos e já marcados no momento de sua geração. Encontrar um viés em meio aos dado é o objetivo do artigo (Köhler et al., 2010), que propõe um algoritmo que busca similaridade de dados para identificar dados sujos, nesse artigo a idéia seria identificar uma pequena amostra de dados considerados sujo e

passar um algoritmo que compararia esses dados com o resto da base de dados classificando probabilisticamente um dado como sujo.

No âmbito da análise de inconsistências nos dados em aprendizado de maquina, este estudo se distingue de diversos artigos revisados em vários aspectos. Enquanto (Qi et al., 2021) e (Qi e Wang, 2021) abordam a eficácia de algoritmos de limpeza de dados, este trabalho se dedica à avaliação e compreensão das inconsistências nos dados, não tratando de algoritmos que procuram soluções. No entanto, o presente trabalho brevemente considera conjuntos de dados com erros conhecidos. (Qi et al., 2021) e esse trabalho apresentam uma intersecção ao concentrar o foco do estudo em problema de classificação.

Ao contrário de (Taghvaei et al., 2017), que destaca as implicações da regularização nos pontos críticos das funções, este estudo não discute algoritmos específicos de solução; a regularização é vista como uma estratégia para aprimorar a base de dados, podendo ser interpretada como um algoritmo de limpeza. Contrariamente, essa técnica pode gerar vieses. Analogamente, este trabalho aborda casos nos quais a geração de inconsistências nos dados pode resultar em vieses ocultos, levando a perdas de controle na qualidade dos dados.

Comparado a (Cerda et al., 2018) e (Ressan e Hassan, 2022), que propõem algoritmos de limpeza em diferentes bases de dados, com foco nas soluções para o problema em vez das inconsistências em si, (L'Heureux et al., 2017), que explora desafios no treinamento de aprendizado de máquina com big data, não apresenta algoritmos de limpeza. Em vez disso, é um artigo mais amplo que apresenta perspectivas e noções que podem ser desenvolvidas posteriormente em outros artigos. De certa forma, este trabalho realiza testes de escalabilidade com seus problemas, tentando generalizar o tamanho do problema para possível aplicação em big data, destacando a importância do tamanho dos dados. A tentativa de demonstrar escalabilidade contribui para a compreensão do uso de dados em larga escala, relacionando-se, assim, com as questões apresentadas em (L'Heureux et al., 2017). Por outro lado, o artigo de (Köhler et al., 2010) concentra-se em encontrar erros por meio de algoritmos de busca, diferindo deste trabalho, que possui dados de erros gerados manualmente em ambiente controlado. No entanto, ambos os trabalhos buscam classificar os tipos de inconsistências que existem, cada um em suas proporções específicas. No entanto, a ausência de algoritmos de solução direta e a ênfase na criação de métricas diferenciam significativamente este trabalho de todos os outros abordados, destacando sua contribuição única para a compreensão de sujeira de dados.

4 METODOLOGIA

Para a realização dos testes, optou-se por abordar um problema matemático de simplicidade destacada, suscetível à geração e validação tanto por agentes humanos quanto por computadores: a classificação de números como pares ou ímpares. Para o entendimento humano, frequentemente ancorado no sistema decimal, essa classificação pode ser efetuada observando se o número finda em 0,2,4,6,8 para pares ou 1,3,5,7,9 para ímpares. No contexto das máquinas, que operam com representação binária, um número é considerado par se seu último dígito é 0 e ímpar se termina em 1. A título exemplificativo, o número 18, identificado como par, apresenta sua representação binária como 10010, finalizando em 0, corroborando sua caracterização como par.

É relevante ressaltar que esse problema apresenta uma particularidade notável em ambas as representações (decimal ou binária): apenas o último dígito do número demanda consideração, ou seja é um **dado útil**, sendo possível desconsiderar os demais sem prejuízo à classificação. Essa particularidade é explorada para ilustrar a noção de **dados neutros**, isto é, dados que teoricamente não devem exercer influência relevante sobre os pesos da MLP utilizada para a classificação.

Quando um dado é empregado no treinamento de uma MLP supervisionada, uma tupla composta por um dado de entrada e uma resposta predefinida, denominada classificador, é inserida no sistema. Com a presença de duas classes, os valores aceitáveis são 0 e 1; no caso de quatro classes, os valores variam de 0 a 3.

4.1 DADOS NEUTROS, ÚTEIS E O TREINAMENTO PERFEITO

Conforme estabelecido, os dados neutros são caracterizados como informações que carecem de relevância substancial para a resolução do problema em questão. Predominantemente, tais dados são empregados para ampliar o conjunto de entradas no processo de treinamento.Esses dados são incorporados ao conjunto de treinamento para fornecer variedade e generalização ao modelo Entretanto, surge uma questão crítica: como assegurar que esses dados não exerçam uma influência significativa no treinamento? Em um sistema ideal, a abordagem apropriada seria atribuir um peso de 0 a todos os dados, exceto ao último dígito do número, indicando que esse peso pode ser desconsiderado. Uma abordagem mais pragmática consiste em garantir que a soma dos pesos associados aos dados neutros seja inferior ao menor peso relevante:

$$\sum_{i=1}^{k} w_{n_i} \leq \{w_i \mid w_i \text{é peso útil}\} \tag{4.1}$$

onde k representa o número de dados neutros e w_{n_i} denota o peso associado ao i-ésimo dado neutro.Essa formulação visa estabelecer uma condição que limita a contribuição dos dados neutros, assegurando que a soma de seus pesos não ultrapasse o menor peso significativo.

No entanto, é importante salientar que uma MLP carece da capacidade intrínseca de classificar de maneira direta a utilidade de suas entradas. O que a MLP realiza, na verdade, é a aproximação dos pesos que se mostram aparentemente irrelevantes para o valor de 0. Em uma execução ideal, o resultado seria congruente com a expressão apresentada na Equação 4.1.

Uma estratégia para designar um dado como neutro consiste em garantir que o referido dado apareça em todas as classes com proporções equitativas. Tomando como exemplo um classificador que opera com representações binárias, uma abordagem seria assegurar que todas as outras entradas possuam uma distribuição uniforme de 50% de zeros e 50% de uns em cada

posição. Além disso, para cada classe (par ou ímpar), essa proporção de 50/50 seria mantida. Outra alternativa seria uniformizar todas as entradas em uma posição específica.

Se o treinamento de uma MLP fosse conduzido utilizando a totalidade dos dados disponíveis, a MLP apresentaria a capacidade intrínseca de mapear todo o espaço de treinamento. Dado que todos os conjuntos de teste seriam, por definição, subconjuntos do conjunto de treinamento, o resultado seria um treinamento perfeito, culminando em um modelo que se ajusta perfeitamente aos dados de treinamento. No entanto, a problemática subjacente reside na incapacidade de avaliar a generalização do modelo em casos de treinamento perfeito, uma vez que este encobre uma tendência à sobreajustagem ao conjunto de teste. Contudo, dada a geração abrangente de todas as entradas possíveis e a correção de todos os casos, torna-se desafiador discernir se as repostas geradas são específicas do conjunto de treinamento ou genuinamente generalizações do problema a ser resolvido.

4.2 GERAÇÃO DE DADOS E MÁSCARAS

Os dados de entrada a serem empregados obedecem a um padrão consistente, compreendendo a representação binária de todos os números no intervalo de 0 a X, onde X é um número expresso como uma potência de 2. Adicionalmente, a cada número foi incorporada uma máscara ou um valor aleatório. Por fim, o último dígito, assumindo valores de 0 ou 1, destaca-se como o único componente numericamente útil para a resolução do problema em questão.

A represtanção binária de um número entre 0 e X	Máscara ou número aleatório (em binário)	0 ou 1

Figura 4.1: Representação de 1 dado

A máscara, enquanto dado de valor fixo, desempenha a função de simular um viés nos dados, contribuindo para a introdução de assimetrias na distribuição dos dados. A presença da máscara desequilibra a igualdade pretendida na proporção de dados, quebrando a paridade entre 0 e 1 em determinadas posições da representação. As máscaras exibem uma variabilidade em seu tamanho, oscilando entre 1 e 14 bits , sendo que o valor mais recorrente é 5.

Para os testes e experimentos subsequentes, dois métodos distintos foram empregados na geração de dados: a concatenação e a substituição. Na maioria dos casos, a concatenação foi adotada como método preferencial.

No método de concatenação, as três partes dos dados são geradas individualmente e posteriormente unidas. O tamanho do dado resultante é a soma das dimensões das partes. Este método apresenta a garantia de não gerar todas as entradas possíveis, devido à incorporação da máscara, evitando assim o cenário de aprendizado perfeito.

Já o método de substituição de dados fundamenta-se na seleção da parte correspondente a um número de 0 a X e na subsequente injeção da máscara, com a remoção de alguns números da entrada para a sobreposição da máscara. Este método, por sua natureza, é mais propenso a contradições, uma vez que pode resultar em dados repetidos, o que pode ser interpretado como um viés nos dados. Para mitigar essa possibilidade, foram estabelecidas regras visando minimizar a repetição de dados, reduzindo assim o potencial viés fora do controle dos experimentos.

Ao gerar dados para experimentos de viés de dados, o conjunto de treinamento uma classe teve uma sujeira aplicada em todos os seus elementos, enquanto no conjunto de teste esse mesmo viés aplicado a classe oposta. O objetivo dessa escolha é mostrar um caso ruim, onde o viés presente no treinamento se mostra presente no conjunto teste. Ao gerar dados para dados

errados, um conjunto de dados limpos é gerado e depois erros são inseridos nesses dados. O conjunto de teste é o mesmo conjunto de dados limpos sem os dados inseridos.

4.3 ARQUITETURA ESCOLHIDA E PARÂMETROS DE TREINO

A biblioteca utilizada para gerar a MLP, treina-la e avalia-la em todos os experimentos foi a tensoflow (TensorFlow, 2023) na linguagem de programação python. Os dados utilizados também foram gerados em python. Todas as partes que baseadas em aleatoriedade foram feitas utilizando a biblioteca random, em especial o método randint, que gera números inteiros pseudo-aleatórios entre 2 intervalos, em uma distribuição discreta uniforme (Python, 2023). Os gráficos gerados foram feitos em python utilizando a biblioteca "matplotlib.pyplot ".

Em todos os experimentos realizados, a arquitetura escolhida para a MLP segue um padrão com três camadas distintas:

1. **Camada de Entrada:** Possui um número de neurônios equivalente ao tamanho da entrada.

2. **Camada Intermediária:** Configurada com um número variável de neurônios, podendo oscilar entre 1 e o tamanho da entrada menos 1. A escolha dessa faixa decorre da consideração de que um número de neurônios superior ao tamanho da entrada pode induzir a uma superadaptação do sistema (Alpaydin, 2010). Nesta camada, foi empregada a função de ativação ReLU (*Rectified Linear Unit*), com comportamento definido como:

$$f(x) = \max(0, x) \tag{4.2}$$

   Em outras palavras, ela retorna x se x for positivo, e zero caso contrário. A escolha dessa função se deu principalmente porque ela permite que alguns neurônios tenham mais influência do que outros, ao contrário da função $Limiar(n)$, onde o valor máximo de um neurônio é sempre 1. Além disso, essa função ajuda a identificar neurônios que não estão ativos, já que seus pesos são zerados. Por último, vale mencionar que essa função é fácil de calcular computacionalmente, o que a torna prática para o treinamento. (Fukushima, 1975)

3. **Camada de Saída:** A última camada, responsável pela produção da saída da rede, é configurada com um número de neurônios equivalente à quantidade de classes passíveis de serem categorizadas. Por exemplo, ao abordar a classificação de um número como par ou ímpar, são designados 2 neurônios de saída, enquanto em cenários com 4 classes, torna-se necessário o emprego de 4 neurônios de saída. A função de ativação selecionada para esta camada foi a função 2.4. A escolha da função 2.4 fundamenta-se na sua capacidade de proporcionar uma transição mais suave nas respostas.

Não serão aplicadas técnicas de regularização nos experimentos. Regularização também é considerado uma técnica de evitar os problemas causados pelo treinamento, o que sai do escopo proposto.

Em todas as instâncias de avaliação, o processo de treinamento do modelo foi conduzido ao longo de 10 épocas, incorporando atualizações ponderadas pelo momentum. Durante cada uma dessas épocas, a totalidade do conjunto de treinamento foi empregada, aproveitando a distinção entre os conjuntos de treinamento e teste. Essa prática foi viabilizada pela divergência dos conjuntos de treino e teste, permitindo uma abordagem integral durante o treinamento.

Além disso, a validação do modelo foi realizada mediante a utilização de um conjunto distinto, sendo essencial para a verificação de indícios de superadaptação (*overfitting*) do modelo ao conjunto de teste. A empregabilidade de um conjunto de validação independente desempenhou um papel crucial na garantia da generalização eficaz do modelo para dados não observados durante o treinamento.

5 RESULTADOS EXPERIMENTAIS

Este capitulo apresenta os experimentos que foram conduzidos para avaliar diferentes tipos de sujeira de dados, fornecendo uma análise dos resultados obtidos ao longo dessas investigações. Na seção 5.1 são apresentados os resultados relacionados a dados enviesados, enquanto na seção 5.2 são apresentados os resultados considerando dados errados.

5.1 DADOS ENVIESADOS

5.1.1 Influência de Cada Bit e Dados Neutros

Este caso de teste foi concebido com o propósito de verificar a aparente redundância de parte das entradas no processo de aprendizado. Com esse intuito, foram gerados 16383 números de 19 bits. Os bits de 0 a 13 abrangem todos os números no intervalo de 0 a 8191, enquanto os bits 14 a 18 são compostos por uma máscara de tamanho 5 com o padrão “11010”. Esta máscara foi aplicada a todos os números pares. Nos números ímpares, os bits 14 a 18 foram substituídos por um número aleatório de 5 bits, garantindo que este último nunca coincidisse com a máscara utilizada nos números pares. O último bit, representando 0 ou 1, é o único efetivamente empregado no processo de classificação.

Após o período de treinamento, os mesmos dados utilizados durante essa fase foram submetidos a modificações, consistindo na alteração de todos os bits 0 para 1 e vice-versa em um único índice da entrada. Por exemplo, apenas os bits na posição 15 foram modificados nessas versões alteradas. Essas modificações pontuais visavam avaliar qualquer alteração nas taxas de acerto, uma vez que qualquer variação seria atribuível à influência desse único bit modificado.

O modelo adotado para esse experimento compreende único neurônio na camada intermediária. A justificativa dessa escolha foi a facilidade de se extrair e interpretar os pesos obtidos durante o pós-treinamento.

Após a execução do experimento, observa-se que o bit mais significativo identificado é o último, associado a uma máscara de valor “11010”, com os dois bits “0” presentes sendo os dois mais relevantes posteriormente. Esses dois bits, individualmente, têm a capacidade de alterar a resposta do sistema, ao contrário de todos os outros. Esse resultado indica que o modelo sofreu uma interferência real e converteu dois bits que deveriam ser desconsiderados em elementos relevantes para a classificação.

Bit	Média do peso	tipo
0	0.25880411	Dados neutros
1	0.09089345	
2	0.10006244	
3	0.09104682	
4	0.0826893	
5	0.10319777	
6	0.09391043	
7	0.10315883	
8	0.10197139	
9	0.10197139	
10	0.12389981	
11	0.10109332	
12	0.10050061	
13	0.10894105	
14	0.31919351	Mascara
15	0.29105754	
16	0.91514542	
17	0.31820572	
18	0.95830864	
19	2.72166558	Par\Impar

Figura 5.1: Média de pesos, por bit quando executado com 1 neurônio na camada intermediaria.

Ao analisar as médias dos pesos do modelo, presente da Figura 5.1, observa-se que todos os pesos da máscara são superiores aos pesos provenientes dos dados neutros. Além disso, é relevante notar que a soma das médias de todos os pesos da máscara se aproxima do peso associado ao bit Par/Ímpar.

Diante disso, conclui-se que a máscara está demonstrando habilidade para confundir o modelo, levando-o a acreditar que o problema a ser resolvido não se resume à classificação de números pares ou ímpares. Pelo contrário, sugere que os bits da máscara possuem valores superiores aos normalmente encontrados em outros dados, destacando-se especialmente os bits 16 e 18.

5.1.2 Tamanho da Máscara

Com a consideração de apenas 1 bit de dado relevante, é observável que uma máscara com um grande número de bits pode ocasionar um aumento progressivo na soma dos bits enviesados, mesmo que a influência individual destes seja inferior àquela do bit de classificação.

Com essa premissa, foram conduzidos dois testes para validar tal hipótese. O primeiro teste consistiu na geração de dados por meio da concatenação, mantendo o tamanho dos bits "neutros"constante, enquanto o tamanho da máscara variava de 1 a 20. Com 20 bits, 57.82% dos bits em cada entrada foram comprometidos por enviesamento (13 bits de dados inalterados, 20 bits de máscara e 1 bit de classificação).

O segundo teste foi realizado por meio do método de substituição, no qual um bit da máscara substituía um bit de dado neutro à medida que a máscara aumentava. Este teste, embora apresente desafios devido à repetição de dados com o crescimento da máscara, permite alcançar

uma porcentagem significativamente maior de dados contaminados por entrada. Com 8 bits de máscara, 57.14% dos dados encontram-se contaminados (5 bits de dados neutros, 8 bits de dados enviesados e 1 bit de classificação), e com 13 bits de máscara, 92.85% dos dados estão sujeitos a contaminação (13 bits de dados enviesados e 1 bit de classificação).

O modelo adotado para este teste compreendeu 13 neurônios na camada intermediária.

Cabe salientar que a máscara utilizada é composta pelo número "10101010101010101010", e, quando necessário, esta representação é ajustada ao tamanho adequado.

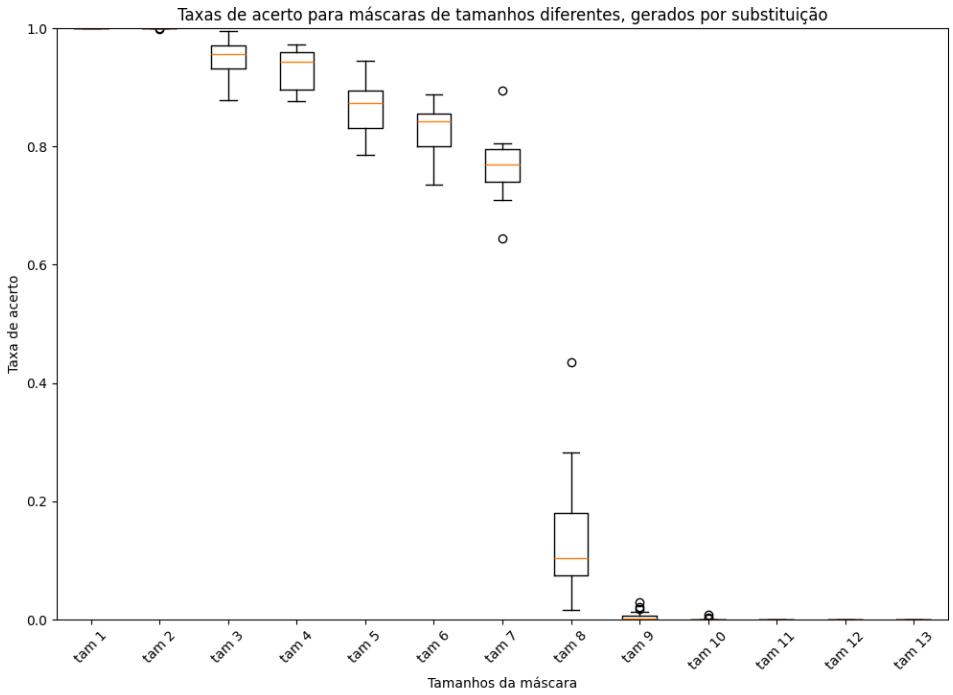


Figura 5.2: Gráfico com taxas de acerto quando se varia o tamanho da máscara, mascara inserida por substituição.

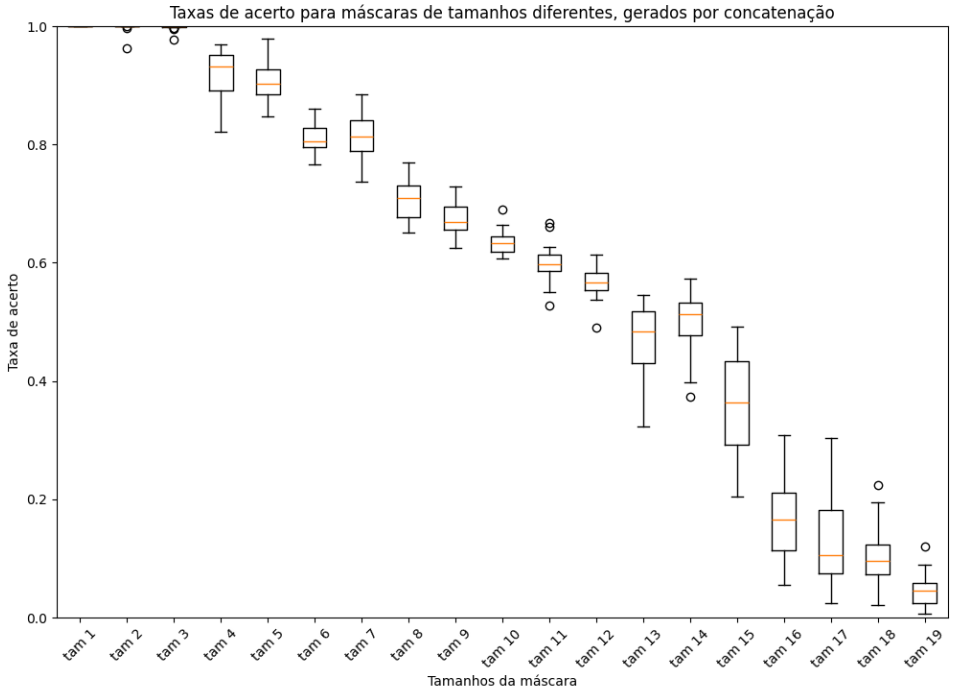


Figura 5.3: Gráfico com taxas de acerto quando se varia o tamanho da máscara, máscara inserida por concatenação.

O primeiro aspecto notável reside na significativa disparidade no comportamento dos gráficos em relação aos métodos de geração de dados.A conclusão mais direta deste experimento é que, quanto maior o tamanho da máscara, maiores serão as taxas de erro. Generalizando, quanto maior a porcentagem de dados corrompidos por entrada, maiores serão as taxas de erro.

O comportamento identificado na Figura 5.2 pode ser atribuído à remoção de dados neutros do sistema ao inserir dados enviesados. Adicionalmente, o fato de muitos dados repetidos serem inseridos à medida que a máscara aumenta pode resultar em números maiores de dados idênticos, criando uma camada adicional de erros e complicando a influência do viés nas taxas de erro.

5.1.3 Dados Úteis e Viés Esparsos

Uma generalização do problema de identificação de números pares ou ímpares pode ser alcançada por meio do problema de “copiar o bit localizado na posição X da representação binária”. Neste novo contexto, a decisão pode ser tomada com base apenas no conteúdo do bit específico na posição X, desconsiderando integralmente os demais bits da representação. É pertinente destacar que o desbalanceamento na quantidade de 0s e 1s em uma posição específica pode ocorrer em qualquer localidade.

No cenário proposto, o bit classificador foi posicionado no centro , e uma máscara de 5 bits foi concebida. Esta máscara foi posteriormente subdividida em 5 bits distintos, sendo distribuída em cinco posições, também no centro dos dados neutros.

DADOS PADRÃO																			
BIT	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
TIPO	N	N	N	N	N	N	N	N	N	N	N	N	N	V	V	V	V	V	0/1
DADOS ESPARSOS																			
BIT	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
TIPO	N	V	N	N	V	N	0/1	N	N	V	N	N	V	N	N	V	N	N	N

Figura 5.4: Estrutura de um dado padrão seguindo modelo de concatenação, e um dado esparso.

A Figura 5.4 exemplifica a estrutura de um dado, dividindo-o entre neutro(N), enviesado(V) e o bit de classificação(0/1).

Adicionalmente, um teste, mantendo o padrão dos dados, foi conduzido para estabelecer um grupo de controle.

O teste foi realizado com diferentes tamanhos de arquitetura, ou seja, variamos o número de neurônios na camada intermediária.

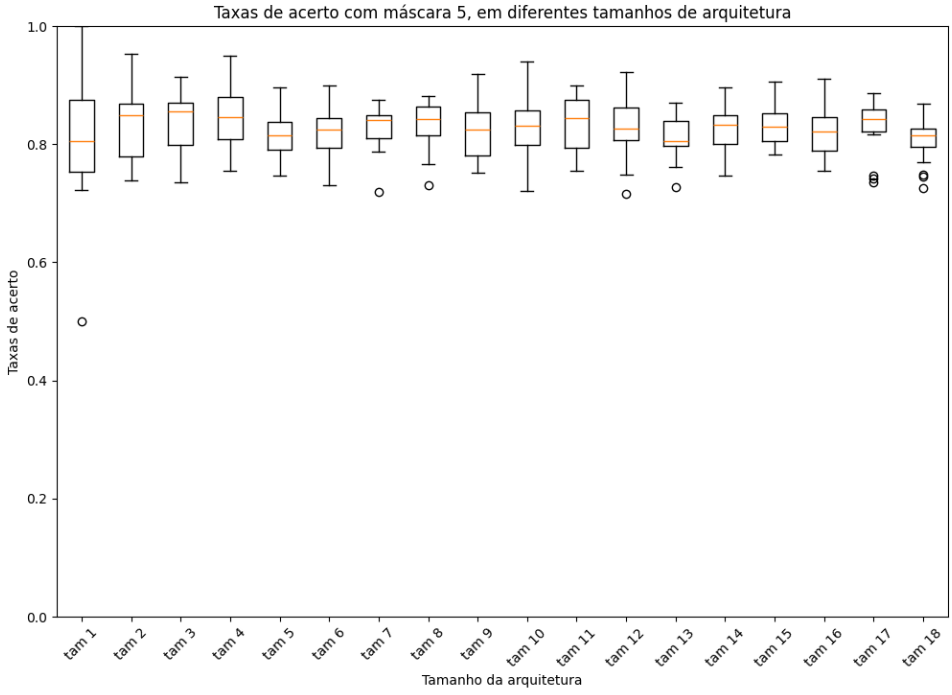


Figura 5.5: Gráfico com taxas de acerto, com máscara de tamanho 5.

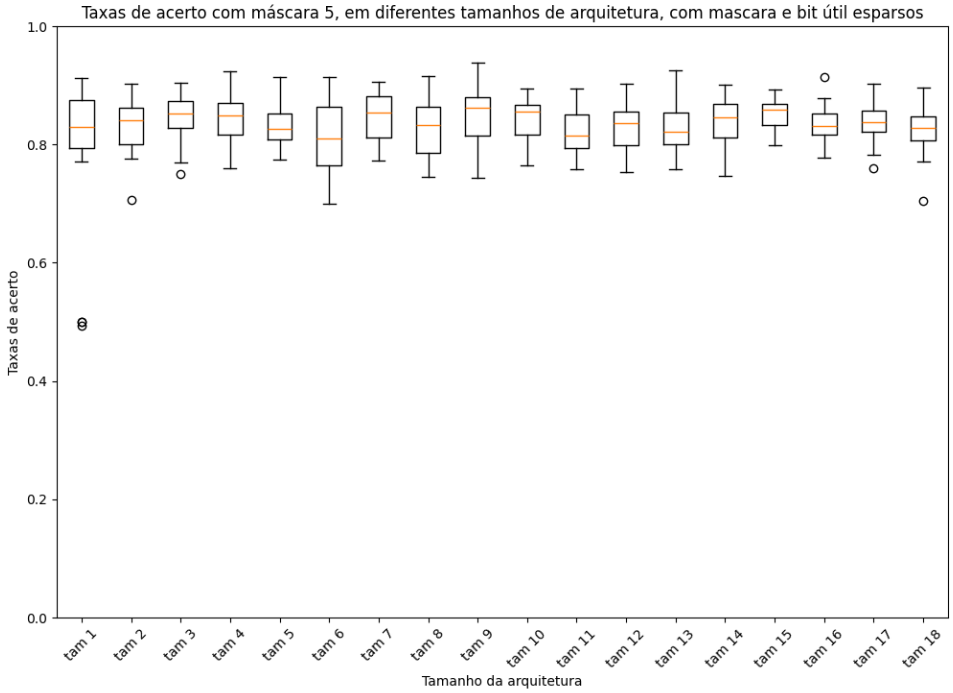


Figura 5.6: Gráfico com taxas de acerto, com máscara de tamanho 5 esparsa e bit útil no meio.

Tamanho	Esparsa	Padrão	Diferença
1	81,02%	79,40%	-1,63%
2	83,53%	83,51%	-0,02%
3	83,92%	84,43%	0,51%
4	84,43%	84,51%	0,08%
5	81,87%	83,05%	1,18%
6	82,00%	80,75%	-1,25%
7	82,87%	84,63%	1,77%
8	83,32%	82,66%	-0,66%
9	82,36%	85,14%	2,78%
10	82,85%	84,33%	1,47%
11	83,80%	82,29%	-1,51%
12	83,02%	83,08%	0,06%
13	81,29%	82,91%	1,61%
14	82,69%	83,85%	1,16%
15	83,20%	85,28%	2,07%
16	82,08%	83,55%	1,47%
17	83,17%	83,79%	0,61%
18	80,91%	82,20%	1,29%
Média	82,86%	83,53%	0,89%

Figura 5.7: Tabela com taxas de acerto médias para cada tamanho de arquitetura.

Ao analisar os gráficos representados pela Figuras 5.5 e 5.6 e a tabela de médias presente na Figura 5.7, constata-se que a diferença entre os dois experimentos foi inferior a 1% na taxa de acerto; nos piores casos o erro se mostrou inferior a 3%.

5.1.4 Classificadores Com Mais De 1 Bit de Dados Úteis

Até o momento, a proporção entre as duas classes permanecia equitativa, com metade dos números sendo pares e metade ímpares. Introduziu-se uma modificação para alterar esse padrão, criando assim um novo desafio que consiste em identificar números múltiplos de 4. Em notação binária, os números múltiplos de 4 são caracterizados por terminarem em "00". Apesar de ainda se ter apenas duas classes, a proporção de dados foi desequilibrada, com 75% representando números que não são múltiplos de 4 e os restantes 25% sendo números múltiplos de 4. O objetivo do teste é determinar se o viés em uma classe, mesmo diante desse desequilíbrio, exerce influência sobre a outra classe.

Durante o treinamento, a máscara foi aplicada exclusivamente aos números múltiplos de 4, ou seja, a 1/4 do conjunto de dados. Na fase de validação, o viés foi introduzido nos números que não são múltiplos de 4, simulando assim um cenário de pior caso.

Bit	Média do peso	tipo
0	0,3268965	Dados neutros
1	0,0538333	
2	0,0763069	
3	0,0556310	
4	0,0578116	
5	0,0533482	
6	0,0603139	
7	0,0806430	
8	0,0804541	
9	0,0543316	
10	0,0631202	
11	0,0626518	
12	0,0669995	
13	0,4643848	Mascara
14	0,4544609	
15	1,0350224	
16	0,4528839	
17	1,0571334	
18	2,0980586	múltiplo de 4
19	2,1037214	

Figura 5.8: Média de pesos, por bit, quando executado com 1 neurônio na camada intermediaria, com 2 bits para classificar.

Ao observar a Figura 5.8, nota-se que seu comportamento se assemelha consideravelmente à Figura 5.1. Os bits relevantes apresentam um peso consideravelmente maior do que os dados neutros, e a máscara ainda possui valores superiores aos dados neutros. Uma diferença é que em 5.8 a soma dos pesos dos bits de máscara é superior que um peso de dado útil, mas inferior a soma dos pesos dos dados úteis.

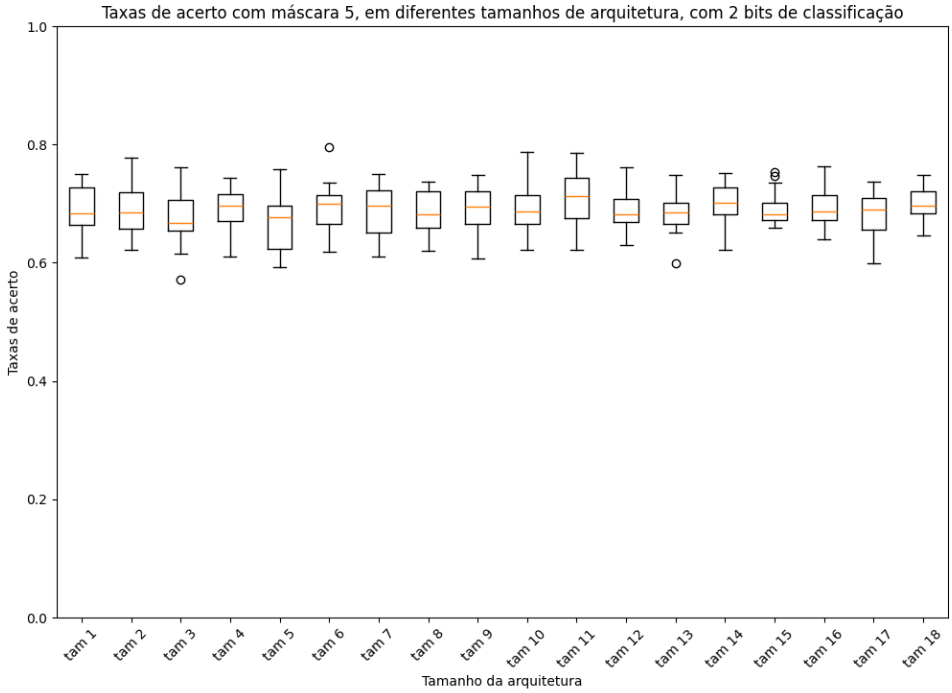


Figura 5.9: Gráfico com taxas de acerto, com máscara de tamanho 5 e 2 bits para classificar.

Ao comparar a Figura 5.9 com a Figura 5.5, observa-se que as taxas de acerto estão mais baixas. Isso pode ocorrer devido ao fato de que, mesmo com 2 bits influenciando as decisões, os pesos dos bits de viés estão mais elevados.

A maneira como os dados foram gerados também desempenha um papel significativo. Durante o treinamento, o viés estava presente apenas no conjunto dos números múltiplos de 4; no entanto, durante o teste, ele estava presente em todos os números que não são múltiplos de 4. Em outras palavras, houve um desequilíbrio de 3 vezes mais viés no conjunto de treinamento.

Taxa de erros			
Tamanho	Par impar	multiplos de 4	Crescimento
1	20,61%	31,13%	51%
2	16,49%	31,01%	88%
3	15,57%	32,13%	106%
4	15,49%	31,01%	100%
5	16,95%	32,84%	94%
6	19,25%	30,70%	59%
7	15,37%	31,19%	103%
8	17,34%	31,25%	80%
9	14,86%	30,66%	106%
10	15,67%	30,65%	96%
11	17,71%	29,19%	65%
12	16,92%	31,24%	85%
13	17,09%	31,65%	85%
14	16,15%	30,15%	87%
15	14,72%	30,75%	109%
16	16,45%	30,67%	86%
17	16,21%	31,67%	95%
18	17,80%	29,85%	68%
Média	16,47%	31,01%	88%

Figura 5.10: Comparação de taxas de erros entre o problema do par/impar e o dos múltiplos de 4, com relação ao tamanho da arquitetura.

Tanto a alteração no balanceamento dos dados, quanto a diferença no uso do viés foram capazes de gerar um aumento na taxa de erro, com uma média de 88%, como apresentado na figura 5.10.

5.1.5 Presença Parcial de Viés

Assim como no problema anterior, onde usamos uma máscara de tamanho 5 em dados organizados, agora estamos estudando um novo cenário. Nele, introduzimos viés apenas em algumas entradas para simular uma situação mais realista, parecida com o que ocorre na prática.

A geração dessa probabilidade de viés foi realizada mediante o emprego de um gerador de números pseudo-aleatórios, esse mesmo gerador é empregado na geração de números aleatórios quando é gerado números aleatórios no espaço ocupado pela máscara, esse gerador tem uma distribuição discreta uniforme.

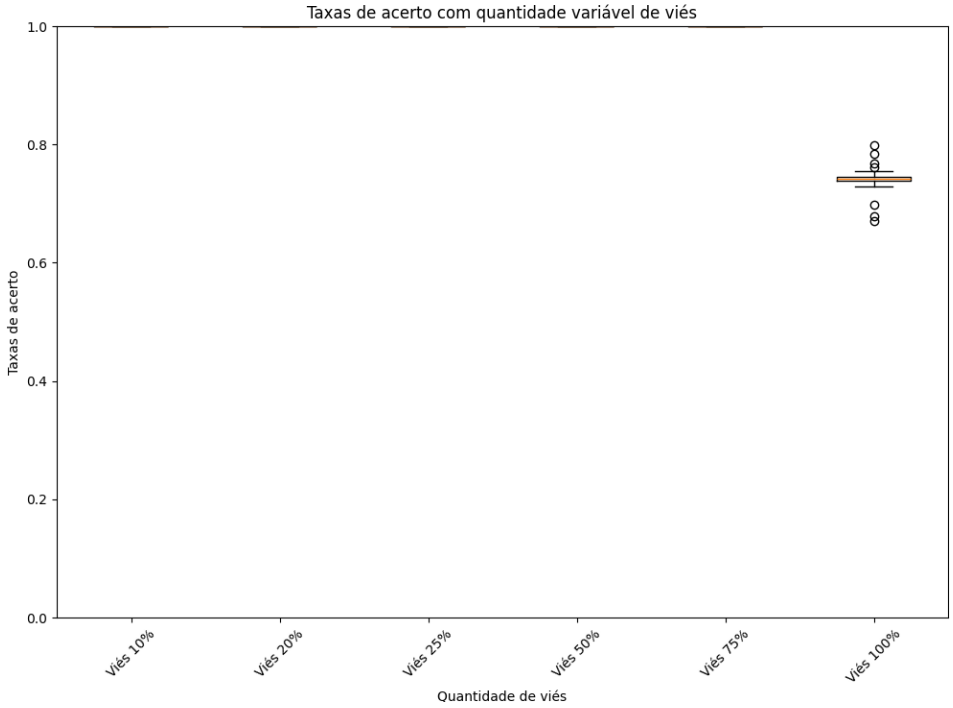


Figura 5.11: Taxas de erros quando quantidade de viés é variável.

Observando a Figura 5.11,o caso com 100% de viés foi o único que apresentou uma queda nas taxas de acerto, alinhando-se com os valores observados em outros gráficos que utilizam máscaras de tamanho 5, como o apresentado na Figura 5.5. A aparente ausência de erros em outros tamanhos deve-se ao fato de que o modelo consegue generalizar melhor quando a máscara não está sempre presente, permitindo que o sistema aprenda a inutilidade dos dados da máscara e os ignore.

5.1.6 Escalabilidade

A escalabilidade de um problema refere-se à capacidade intrínseca deste de manter inalteradas suas propriedades diante do aumento de tamanho ou complexidade. Em termos mais precisos, um método é considerado escalável quando consegue processar entradas mais extensas sem modificar seu comportamento original.

Para avaliar a escalabilidade, foi necessário criar uma entrada substancialmente maior, mantendo, contudo, uma relação tangível com entradas de maior complexidade. Nesse contexto, foi gerado um conjunto de dados com 94 a 109 bits de entrada. Este novo conjunto é constituído por 13 bits de dados representando todos os números de 0 a 8191, 80 bits de máscara ou dados gerados aleatoriamente, e de 1 a 16 bits de dados de classificação.

O experimento foi parametrizado com o ajuste do tamanho da máscara em múltiplos de 16, o dimensionamento dos dados úteis variando de 1 a 16, e o tamanho da camada intermediária da rede variando entre 1 e 150 neurônios. A escolha de limitar o tamanho da máscara e da camada intermediária é estratégica, uma vez que tamanhos excessivamente grandes podem resultar em casos de superadaptação. Mesmo diante da geração de entradas de até 109 bits, o tamanho máximo de 150 foi selecionado para explorar eventuais comportamentos associados à superadaptação.

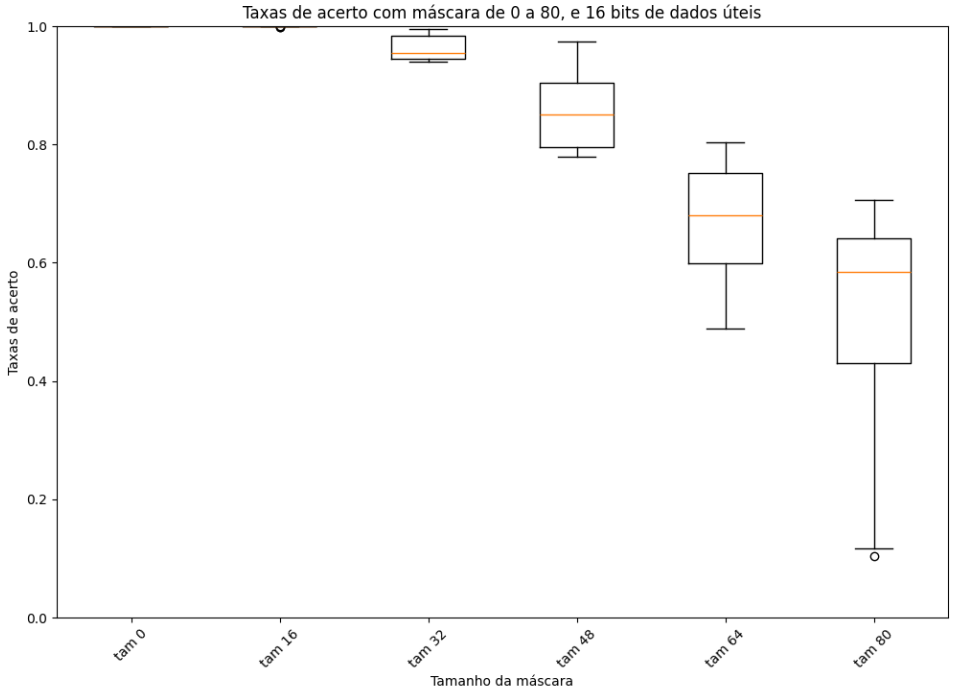


Figura 5.12: Taxas de erros com problemas de 109 bits e máscara de tamanho variável.

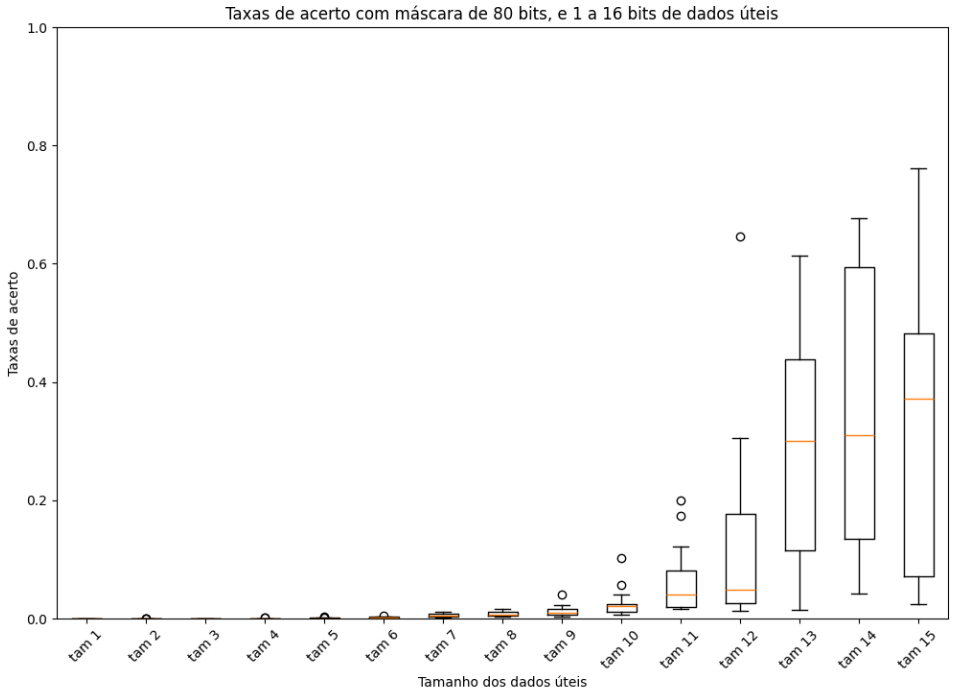


Figura 5.13: Gráfico com taxas de acerto com máscara de tamanho 80 e variando o tamanho dos dados úteis.

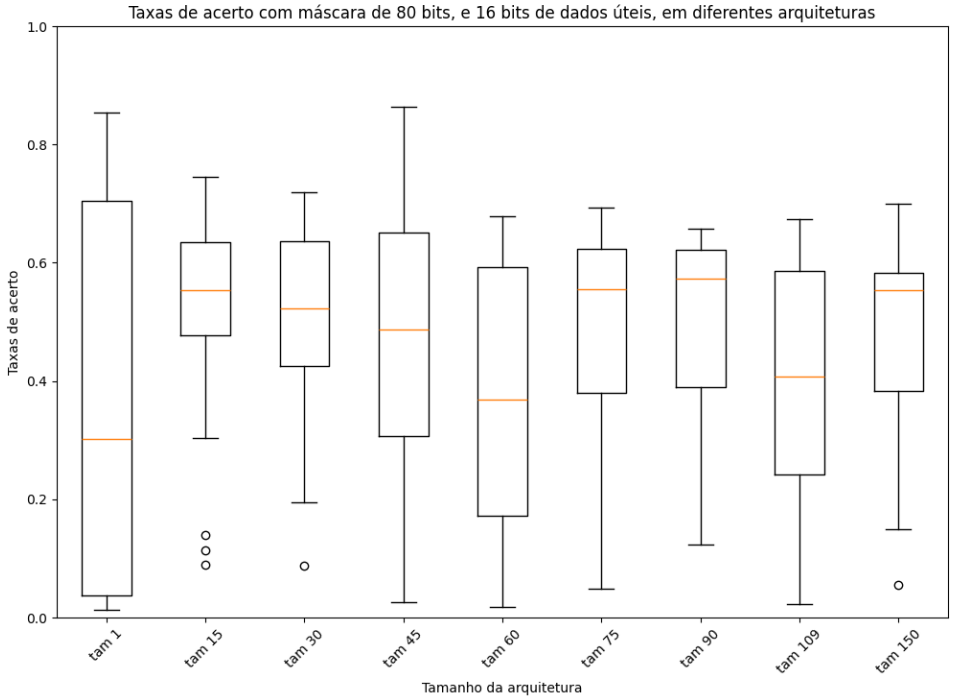


Figura 5.14: Gráfico com taxas de acerto com problemas de 109 bits e diferentes tamanhos de arquitetura.

O tamanho das entradas influencia diretamente nas taxas de acerto, conforme evidenciado na Figura 5.13. Quanto maior o tamanho dos dados úteis, melhores as taxas de acerto. O comportamento observado, considerando o topo dos desvios padrão, assemelha-se a uma exponencial, com um crescimento mais lento das médias e um grande salto de desempenho nos últimos tamanhos.

Quando a arquitetura do sistema aumenta, não é possível notar um padrão consistente de melhoria ou piora. Essa falta de padrão também é observada nos casos menores, como na Figura 5.5. Como uma relação não pode ser identificada em ambos os casos, essa característica parece ser escalável.

O tamanho da máscara tem uma forte influência no desempenho do sistema. Em proporção, um caso com máscara de tamanho 5 e um tamanho total de 19 tem 26.31% de dados sujos, o que se assemelha a um caso com 29 bits de viés em 109 bits no total. No tamanho mais próximo, de 32 bits, presente na Figura 5.12, as taxas de acerto são consideravelmente maiores. As taxas de acerto ficam mais semelhantes no tamanho 48, ambas próximas a 80%. O comportamento dos dados, se faz presente, indicando que o crescimento dos vieses tem um comportamento escalável.

5.2 DADOS ERRADOS

Dados incorretos, conforme previamente definidos, denotam falhas na lógica e na consistência das entradas, suscetíveis a comprometer a eficácia do aprendizado do problema em questão.

Dado que as entradas geradas seguem um padrão binário, com apenas duas classes para classificação, a simulação de dados incorretos ocorre mediante a introdução de números que diferem de 0 e 1. Devido aos parâmetros de uma MLP aceitarem valores reais, o sistema permite

a inclusão dessa entrada, mesmo quando esta carece de pertinência para a resolução do problema proposto.

Analogamente, podem ocorrer falhas no classificador, resultando na alocação inadequada de uma entrada em sua classe correspondente. Dado que os valores aceitos são 0 e 1 (ou de 0 a 3 em situações de quatro classes), a simulação de erro se efetua ao atribuir valores superiores aos admitidos.

Nos experimentos que incorporam dados incorretos, alguns foram conduzidos com duas e quatro classes para classificação. No caso das quatro classes, o problema selecionado foi o módulo 4, ou seja, o resto da divisão por 4, distribuindo igualmente os dados em 4 conjuntos distintos.

Ao gerar um dado, há uma probabilidade associada à possibilidade de ele conter erros, sendo que a localização do erro é determinada de maneira aleatória. Esses erros podem ser inseridos nos bits classificadores, conforme requisitado. A probabilidade de erro varia de 0 a 1500%, onde valores superiores a 100% indicam que todas as entradas terão pelo menos 1 erro. Por exemplo, um erro de 250% resultará em 2 bits incorretos em todas as entradas, com uma probabilidade adicional de 50% para um terceiro bit errado.

5.2.1 Amplitude de Erro

Quando um dado é corrompido, resultando em um número superior ao aceitável, torna-se imperativo examinar se esses valores elevados não estão exercendo uma influência desproporcional sobre os pesos, conduzindo a uma margem de erro ampliada. Na situação em que um bit da entrada é selecionado para ser incorreto, ele é substituído por um número constante, variando entre 3 e 9 (ou 5 e 9).

A probabilidade de erro é estabelecida em 300%, indicando que todos os dados terão 3 bits substituídos pelo valor incorreto.

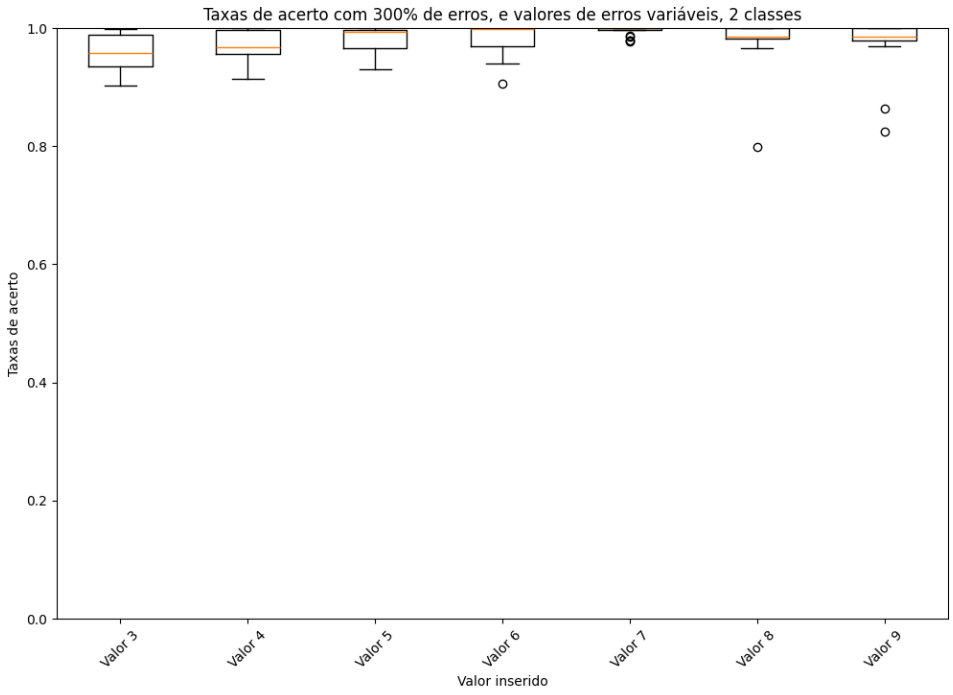


Figura 5.15: Gráfico com taxas de acerto, com valor de erro fixo e 2 classes.

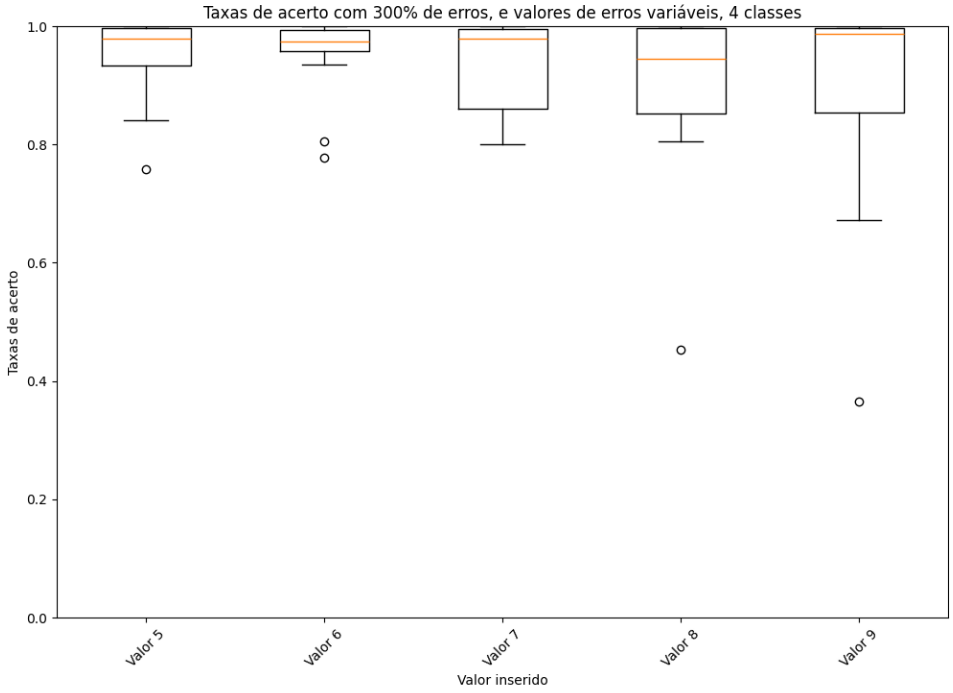


Figura 5.16: Gráfico com taxas de acerto, com valor de erro fixo e 4 classes.

Ao analisar as figuras 5.15 e 5.16, o primeiro ponto a ser destacado nos estatísticas geradas é a diferença de comportamento entre os dois gráficos. No gráfico com 4 classes, observa-se uma tendência de piora nos casos com valores mais elevados. Entre os valores 7 e 8, a média diminuiu, e um outlier muito baixo surgiu. Entre os valores 8 e 9, a média cresceu, ficando muito próxima de 99%; no entanto, o desvio padrão indica que os valores abaixo da média são muito esparsos, indicando uma piora no pior caso.

Em contradição com o caso com 4 classes, o caso com 2 classes manteve médias piores com valores mais baixos, melhorando as médias em valores intermediários e apresentando uma leve diminuição nos valores mais altos, mas ainda maiores que os presentes nos valores mais baixos.

Em valores mais altos de ambos os problemas, ocorre a criação de outliers, indicando uma piora nos piores casos.

5.2.2 Erros em Toda a Entrada

Não é sempre que os erros identificados apresentarão um padrão predefinido de serem um valor fixo. Portanto, incorporou-se um gerador de números aleatórios para assegurar a não utilização de valores válidos.

Neste contexto, qualquer dado, inclusive os dados úteis da entrada, pode ser corrompido. Foram gerados valores de erro variando de 10% a 1400%, sendo aplicados em ambas as classes.

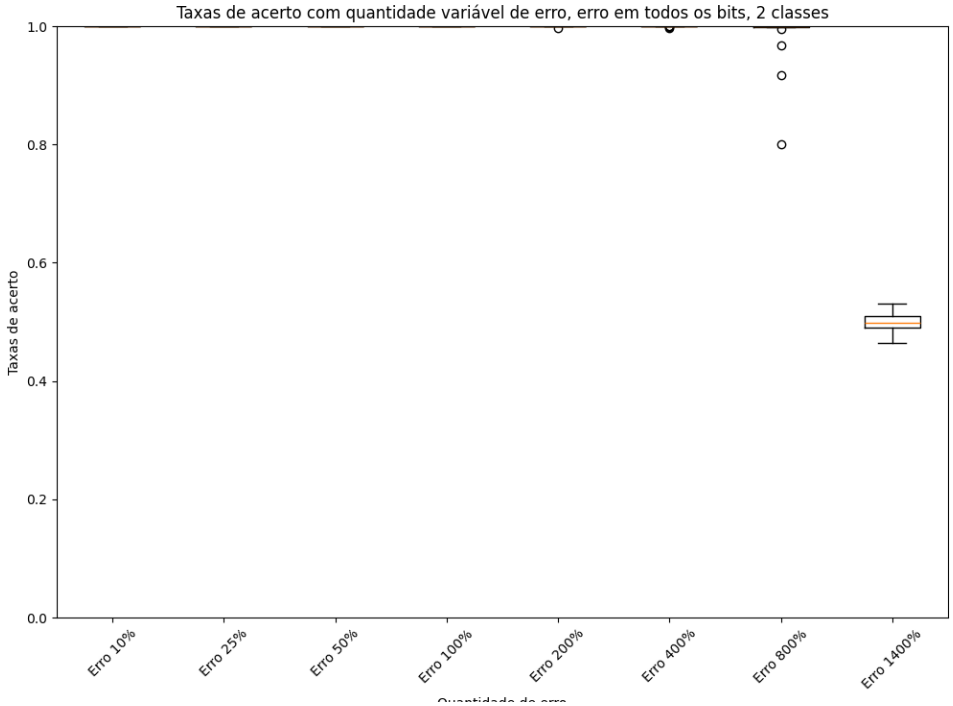


Figura 5.17: Gráfico com taxas de acerto, com quantidade de erro variável, erro presente em todos os bits, 2 classes.

Na Figura 5.17 observa-se que os erros presentes não foram relevantes até o tamanho de 800% ou 8 erros por entrada. Isso se deve ao fato de que a localização do erro pode ocorrer em qualquer posição. Com 8 dados errados, sorteio aleatório, sem repetição e 15 números no total, a probabilidade de um bit específico ser escolhido é de 53.33%. Essa é a probabilidade do bit útil ser modificado. Então, em ambas as classes, admitindo uma distribuição perfeita dos números aleatórios, a divisão dos dados ocorre em 46.33% dados corretos e 53.33% de dados errados. Esses valores ainda geram uma generalização do problema, como mostrado pelas taxas de acerto muito altas, porém com vários casos com porcentagens inferiores.

Como os dados estavam divididos entre o dado real da classe e todos os números inválidos possíveis, o resultado foi uma diminuição na confiança e na generalização do modelo.

Com erros de 1400%, todos os dados foram substituídos por dados errados. Como o sistema não pôde encontrar um padrão, ele acabou por responder todas as entradas como sendo de uma classe, com pequenas variações para mais ou para menos.

5.2.3 Erros Somente nos Dados Neutros

No cenário anterior, os dados podiam ser incorretamente inseridos em qualquer posição, inclusive nos dados úteis fundamentais para a resolução do problema. Nesta instância, essa condição foi reformulada; agora, é obrigatório que os dados úteis de cada entrada sejam preservados, sendo somente os dados neutros suscetíveis a incorreções. Novamente, um valor inválido para um bit é selecionado quando é necessário introduzir um erro em uma entrada.

Os valores associados aos erros variam entre 10% e 1400%. No caso de 1400%, todos os bits, com exceção daqueles empregados pelo classificador, encontram-se incorretos. Dada a aleatoriedade na escolha dos números, é crucial destacar a possibilidade de ocorrência de entradas idênticas, suscitando uma ressalva em relação a porcentagens excessivamente elevadas.

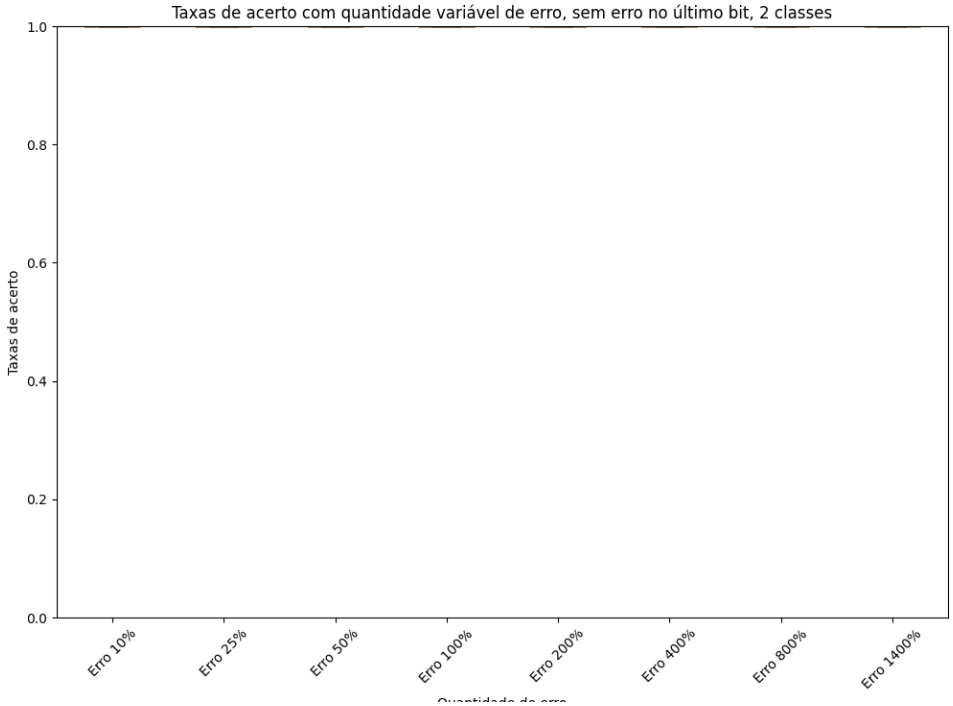


Figura 5.18: Gráfico com taxas de acerto, com quantidade de erro variável, erro presente em todos os bits neutros, 2 classes.

O gráfico apresentado pela Figura 5.18, de difícil observação, apresenta todos os casos com 100% de taxa de acerto, mesmo com todos os bits neutros sendo trocados, ele preservou sua taxa de acerto. Isso se deve à teórica proporção dos dados errados, pois ambas as classes têm seus dados errados, e a proporção dos números é igual. Não existe viés presente, e todos os dados errados continuam sendo tratados como neutros. Mesmo não existindo garantia de ausência de dados repetidos no conjunto de teste, os desbalanceamentos e possíveis vieses gerados não são suficientes para desestabilizar o sistema.

5.2.4 Erros Somente nos Dados Úteis

Na geração do caso em que todos os dados foram corrompidos e no caso em que somente os dados neutros foram corrompidos, neste experimento, exclusivamente os dados úteis serão incorretos.

O propósito deste experimento é elucidar a influência dos erros diretamente na parte mais crucial do problema. Vale ressaltar que os dados incorretos consistem em números aleatórios não interpretáveis pelo sistema, e são teoricamente gerados de maneira homogênea, ou seja, eles tendem a converter os dados úteis em dados neutros.

Como apenas um bit será incorreto, os valores de erro variam entre 10% e 100%.

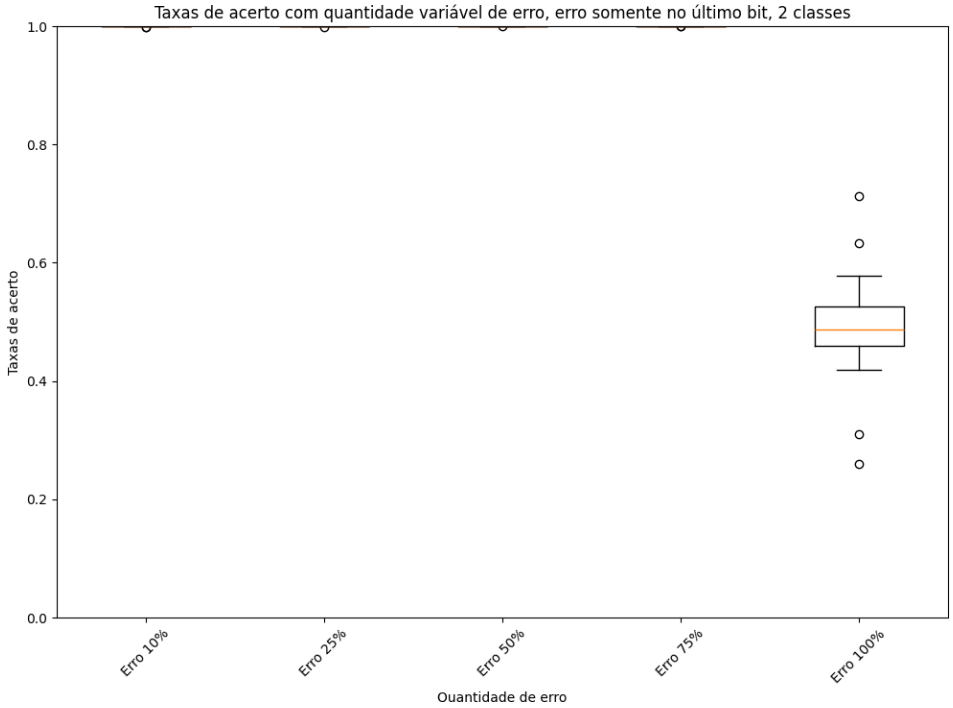


Figura 5.19: Gráfico com taxas de acerto, erro somente no bit útil, 2 classes.

Assim como visto na seção 5.2.2, o comportamento do gráfico se dá pela quantidade cada vez menor de dados úteis que possuem alguma lógica para uma possível generalização do problema. Com 100% de dados errados, todos os números têm a mesma proporção; novamente, o sistema tentará generalizar uma classe, mas não conseguirá.

Nesse caso, sem a interferência dos outros bits, o desvio padrão e os outliers foram maiores, mostrando que os melhores e piores casos de treino são mais esparsos, indicando mais instabilidade no treinamento.

5.2.5 Erro no Classificador

Conforme mencionado, os dados de treinamento consistem em tuplas que contêm a representação binária de um número e uma resposta predefinida. É possível que ocorra um erro na resposta predefinida, por exemplo, devido a problemas no classificador.

Neste cenário, é simulado um erro em apenas uma classe. Diferentemente dos casos em que um erro se traduz em um dado inválido, aqui, o erro manifesta-se como a classificação equivocada de uma classe para outra. No caso de duas classes, ela é erroneamente atribuída à outra classe.

É importante destacar que, ao contrário dos erros anteriores que tendiam a neutralizar o dado, aqui os dados estão ativamente contribuindo para confundir o classificador, uma vez que as respostas incorretas são válidas para o problema em questão.

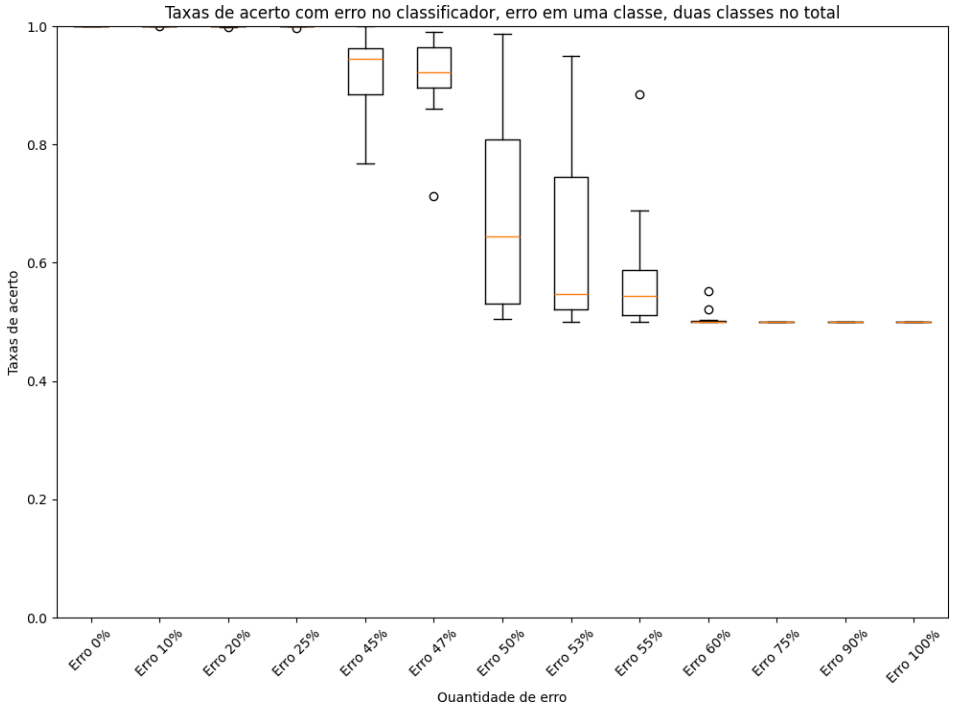


Figura 5.20: Gráfico com taxas de acerto, erros no classificador, duas classes.

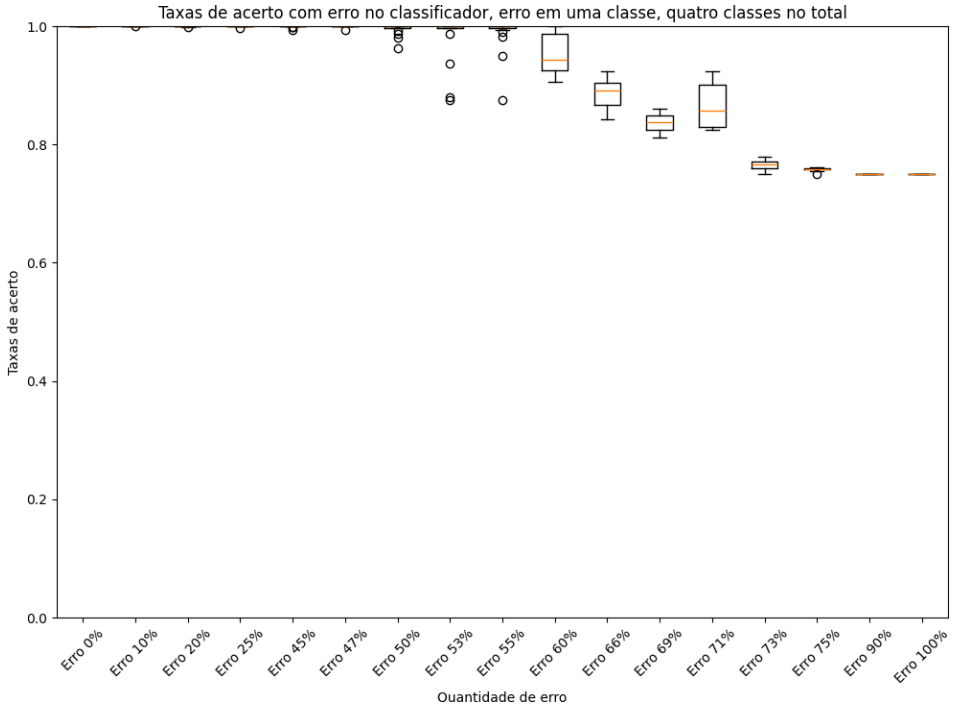


Figura 5.21: Gráfico com taxas de acerto, erros no classificador, quatro classes.

Os equívocos derivados da inversão do classificador ocasionam a reclassificação da classe afetada para a outra classe. Quando as taxas de erro ultrapassam 50%, o sistema tende a optar por classificar todas as instâncias como pertencentes a uma única classe. Essa decisão se

baseia na busca pela maximização da assertividade, assegurando uma taxa de acerto de 75% para todos os dados.

Ao analisar o gráfico na Figura 5.20, nota-se que as taxas de erro começam a declinar a partir de uma taxa de acerto de 45%. Com 60%, a generalização equivocada mencionada anteriormente parece estar plenamente estabelecida. Os desvios padrões revelam que alguns casos ainda foram capazes de manter a generalização.

A análise do caso com 4 classes, apresentada no gráfico da Figura 5.21, revela um padrão de decréscimo nas taxas de acerto à medida que os valores se aproximam de 75%, correspondendo à proximidade das taxas de erro em relação à quantidade de dados não corrompidos. Simultaneamente, outliers começam a surgir em tamanhos próximos de 50%, indicando que, um pior caso já pode resultar em problemas significativos.

Nos dois cenários analisados, a taxa de erro torna-se significativa quando se aproxima da porcentagem de classes não corrompidas, isto é, $(n-1)/n$ classes.

5.2.6 Erro no Classificador de Todas as Classes

Diferente do caso anterior com uma classe sendo erroneamente classificada, é simulado um erro que afeta todas as classes, resultando em uma modificação global na classificação. Diferentemente dos casos em que um erro se traduz em um dado inválido, aqui, o erro manifesta-se como a classificação equivocada de todas as classes. Todas as respostas predefinidas são alteradas, promovendo uma troca global nas classificações.

No exemplo de 2 classes, ambas são intercambiadas de maneira incorreta.

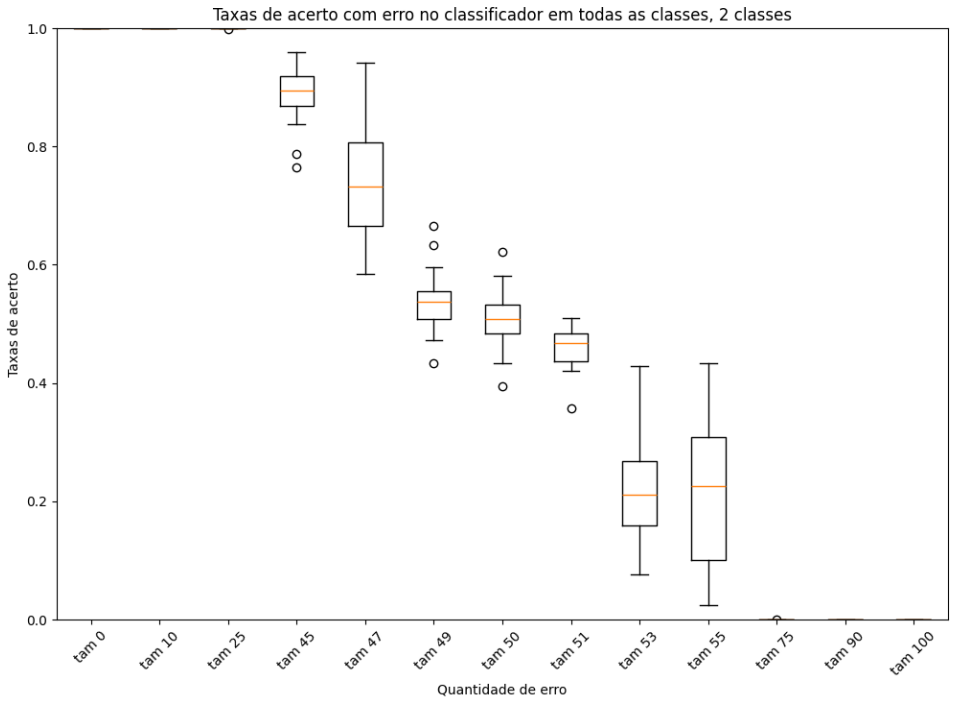


Figura 5.22: Gráfico com taxas de acerto, erros no classificador em ambas as classes, 2 classes.

Assim como no caso com somente 1 classe errada, quando a quantidade de erros se torna próxima ou maior que o de dados corretos, o sistema é incapaz de aprender a generalização

do problema, e sem a generalização, o treinamento não consegue classificar os dados sem erros de testes.

6 CONCLUSÃO

A generalização da análise pode ser realizada separadamente para casos de dados enviesados e dados errados.

Para dados enviesados, é crucial analisar a porcentagem de sujeira presente em cada classe, supondo o pior caso onde o conjunto de teste tem o mesmo viés. Uma comparação útil envolve um treinamento sem viés com o resultado obtido, multiplicando o resultado pela taxa de aprendizado obtida no treinamento controle, se houver.

A localização dos dados revelou-se uma variável que, quando preservada uniformemente para todas as entradas, não exerceu influência significativa no desempenho do sistema.Similarmente, a presença de viés, mesmo quando distribuído de forma esparsa, não induziu mudanças substanciais em comparação com os cenários nos quais as entradas seguiram o modelo de dados concatenados.

Além disso, quando lidamos com mais classes, é importante notar que o viés em uma única classe pode ter um impacto maior do que em situações com menos classes. Isso acontece porque o viés em uma classe acaba afetando todas as outras, e quanto mais classes temos, pior pode ser essa influência.

Ao analisar as taxas de erro, é recomendável pensar de maneira não linear sobre os resultados, considerando modelos analíticos como exponencial, parabólico ou logístico.

O viés de dados pode ser classificado em três grupos com base na quantidade de viés presente por dado:

1. Valores Inferiores a 15% de Viés:
   - Não apresentaram impacto considerável no sistema.
   - Taxas de erro geralmente menores que 5%.
2. Valores entre 15% a 40% de Viés:
   - Apresentaram até 40% de erro.
   - Ainda é possível manter alguma generalização, mesmo com as taxas de acerto baixas.
3. Valores acima de 40% de Viés:
   - Apresentar taxas de erro superiores a 50% e que crescem rapidamente conforme o viés aumenta. Mais de 50% de viés têm taxas de erro que ultrapassam 80%.
   - Não aparenta ter generalização.

Observa-se que o viés de dados tem um comportamento mais tolerante à medida que a entrada cresce, e os valores mencionados anteriormente podem aumentar conforme o tamanho da entrada também aumenta.

Para dados incorretos, a estabilidade do sistema tende a diminuir à medida que a quantidade de erros aumenta. Contudo, observa-se uma relativa estabilidade em cenários com taxas de erro elevadas, como 75%. Este comportamento está sujeito a um equilíbrio específico de erros, que pode não ocorrer em contextos normais.

Esse desafio está intrinsecamente relacionado à definição de um conjunto mínimo de entradas capazes de generalizar de forma eficaz um problema. Devido à ausência de uma

exploração aprofundada dessa métrica específica, quaisquer afirmações conclusivas sobre seu impacto seriam prematuras.

No caso de erros em que dados são substituídos por outros dados válidos, a dinâmica é diferente. Erros nos dados úteis abaixo de 40% parecem ter um impacto reduzido no treinamento de sistemas de inteligência artificial, graças à propensão à generalização do problema, que tende a favorecer o resultado com maior taxa de acerto. Entretanto, valores de erro superiores a 45% podem resultar em quedas notáveis na acurácia. Taxas iguais ou superiores a 50% tornam-se praticamente inviáveis, pois comprometem a capacidade de generalização do modelo.

Erros em dados neutros, quando distribuídos homogeneamente, podem não afetar significativamente o treinamento, isso se deve ao fato que a substituição de um dado neutro balanceado por outro dado balanceado não aparenta exercer nenhuma influencia na generalização do sistema. No entanto, desequilíbrios nesses dados têm o potencial de introduzir viés no sistema, impactando negativamente seu desempenho.

6.1 TRABALHOS FUTUROS

É importante salientar que a metodologia, desenvolvida e aplicada nesse trabalho ocorreu em ambiente controlado, com dados fabricados e sujeira de dados bem definidas, a fim de validar o uso dessa metodologia é necessário o uso dela em ambiente reais, com bases de dados não fabricadas, e aplicações que envolvam complexidades diferentes das utilizadas.

Com base nas considerações acerca do balanceamento de dados e nos conceitos de dados neutros, enviesados e úteis, surge a indagação: seria possível desenvolver uma heurística capaz de criar "dados antígeno"? Em outras palavras, seria concebível formar um conjunto de dados contaminados que, quando combinado aos dados sujos preexistentes, reduzisse o viés, resultando em taxas de acerto mais elevadas e em um aprendizado menos enviesado?

Ciente das taxas de viés que provocam impacto irrelevante na generalização, torna-se plausível explorar métodos para "neutralizar"os erros ao introduzir mais dados comprovadamente válidos. Este enfoque busca mitigar o efeito do viés, promovendo uma abordagem mais equilibrada no treinamento de modelos de aprendizado de máquina.

Uma forma de gerar dados válidos é a aplicação de **hints**, hints são informações adicionais que podem ser fornecidas para orientar o sistema durante o treinamento . Essas "hints"podem ser utilizadas para fornecer informações sobre o espaço de busca, preferências ou restrições específicas do problema, ou qualquer outra informação útil para melhorar o desempenho do sistema. Podem ser usados para expandir o conjunto de treinamento criando "exemplos virtuais". Um exemplo é rotacionar imagens, uma vez que essas imagens possuem as mesmas características da imagens original, como as 2 imagens são consideradas o mesmo exemplo, um erro em qualquer versão das imagens deve ser tratado (Alpaydin, 2010).

Todos os casos analisados até o momento abordaram problemas simples, como o de classificação entre par ou ímpar. Seria enriquecedor estender essa análise para problemas mais complexos, especialmente aqueles que envolvem um maior número de bits de dados úteis. O experimento mencionado contemplou, no máximo, 16 bits úteis.

A análise realizada focou-se exclusivamente em problemas de classificação. Há espaço para estender essa mesma análise a problemas de regressão, proporcionando uma compreensão mais abrangente do impacto desses diferentes tipos de erros em diversos contextos de aprendizado de máquina.

REFERÊNCIAS

Alpaydin, E. (2010). *Introduction to Machine Learning*. The MIT Press, 2nd edition.

Cerda, P., Varoquaux, G. e Kégl, B. (2018). Similarity encoding for learning with dirty categorical variables.

Chu, M. (2004). *Blissful Data: Wisdom and Strategies for Providing Meaningful, Useful, and Accessible Data for All Employees*. AMACOM.

Fukushima, K. (1975). Cognitron: A self-organizing multilayered neural network. *Biological Cybernetics*, 20:121–136.

Ghiassi, A., Younesian, T., Zhao, Z., Birke, R., Schiavoni, V. e Chen, L. Y. (2019). Robust (deep) learning framework against dirty labels and beyond. Em *2019 First IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*, páginas 236–244.

Köhler, H., Zhou, X., Sadiq, S., Shu, Y. e Taylor, K. (2010). Sampling dirty data for matching attributes. Em *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, SIGMOD '10, página 63–74, New York, NY, USA. Association for Computing Machinery.

L'Heureux, A., Grolinger, K., Elyamany, H. F. e Capretz, M. A. M. (2017). Machine learning with big data: Challenges and approaches. *IEEE Access*, 5:7776–7797.

Python (2023). random — generate pseudo-random numbers.

Qi, Z. e Wang, H. (2021). Dirty-data impacts on regression models: An experimental evaluation. Em Jensen, C. S., Lim, E.-P., Yang, D.-N., Lee, W.-C., Tseng, V. S., Kalogeraki, V., Huang, J.-W. e Shen, C.-Y., editores, *Database Systems for Advanced Applications*, páginas 88–95, Cham. Springer International Publishing.

Qi, Z., Wang, H., Li, J. e Gao, H. (2021). Impacts of dirty data: and experimental evaluation.

Ressan, M. e Hassan, R. (2022). Improving machine learning performance by eliminating the influence of unclean data. *Engineering and Technology Journal*, 40(4):546–539.

Russell, S. e Norvig, P. (2010). *Artificial Intelligence: A Modern Approach*. Prentice Hall, 3 edition.

Taghvaei, A., Kim, J. W. e Mehta, P. G. (2017). How regularization affects the critical points in linear networks.

TensorFlow (2023). Criar modelos de machine learning no nível de produção com o tensorflow.