Charles Giovane de Salles

Sistema de Aprendizagem de Máquina Distribuído utilizando o VCube

Cascavel - PR

2024

Charles Giovane de Salles

Sistema de Aprendizagem de Máquina Distribuído utilizando o VCube

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre pelo Programa de Pós-Graduação em Ciência da Computação (PPGComp) da Universidade Estadual do Oeste do Paraná – Unioeste, campus de Cascavel.

Universidade Estadual do Oeste do Paraná – Unioeste – Cascavel

Centro de Ciências Exatas e Tecnológicas – CCET

Programa de Pós-Graduação em Ciência da Computação – PPGComp

Orientador: Dr. André Luiz Brun

Coorientador: Dr. Luiz Antonio Rodrigues

Cascavel - PR

2024

Charles Giovane de Salles
Sistema de Aprendizagem de Máquina Distribuído utilizando o VCube/ Charles Giovane de Salles. – Cascavel - PR, 2024. – 80p. : il. (algumas color.) ; 30 cm.

Orientador: Dr. André Luiz Brun
Coorientador: Dr. Luiz Antonio Rodrigues

Dissertação – Universidade Estadual do Oeste do Paraná – Unioeste – Cascavel
Centro de Ciências Exatas e Tecnológicas – CCET
Programa de Pós-Graduação em Ciência da Computação – PPGComp, 2024.

1. Classificação. 2. Combinação de classificadores. 3. Sistemas distribuídos. I. Brun, André Luiz. II. Rodrigues, Luiz Antonio. III. Universidade Estadual do Oeste do Paraná. IV. Faculdade de Ciência da Computação. V. Sistema de Aprendizagem de Máquina Distribuído utilizando o VCube

Charles Giovane de Salles

Sistema de Aprendizagem de Máquina Distribuído utilizando o VCube

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre pelo Programa de Pós-Graduação em Ciência da Computação (PPGComp) da Universidade Estadual do Oeste do Paraná – Unioeste, campus de Cascavel.

Trabalho aprovado. Cascavel - PR, 23 de abril de 2024:

Dr. André Luiz Brun
Orientador(a)

Dr. Luiz Antonio Rodrigues
Unioeste

Dr. Ronan Assumpção Silva
IFPR, UFPR

Cascavel - PR
2024

Este trabalho é dedicado a minha família que sempre acreditou em mim e me apoiou em todos os momentos.

Agradecimentos

Agradeço a Deus por todas as bençãos recebidas em minha vida. Agradeço a minha mãe Marlene, minha esposa Tais, meus filhos Arthur, Antônia e Kemilly pelo incentivo incondicional que me deram. Agradeço também aos meus orientadores pelo apoio e por sempre estarem dispostos a me ajudar na construção deste trabalho. Agradeço a todos os professores do Programa de Mestrado que me ajudaram a chegar até aqui.

"Ciência da Computação tem tanto a ver com o computador como a Astronomia com o telescópio, a Biologia com o microscópio, ou a Química com os tubos de ensaio. A Ciência não estuda ferramentas, mas o que fazemos e o que descobrimos com elas."

(Edsger Wybe Dijkstra)

Resumo

SALLES, Charles Giovane de. **Sistema de Aprendizagem de Máquina Distribuído utilizando o VCube**. Orientador: Dr. André Luiz Brun. 2024. 80f. Dissertação (Mestrado em Ciência da Computação) – Universidade Estadual do Oeste do Paraná, Cascavel – Paraná, 2024.

Com o crescimento do volume de informações geradas, seu armazenamento deixou de ser realizado de forma local, criando os cenários de dados distribuídos. Assim, quando é necessário realizar um processo de classificação, que é o processo de predizer a categoria de novas entradas com base em dados de treinamento, seria necessária a consolidação das informações em um ponto central da rede para efetuar o aprendizado. No entanto, em algumas situações a movimentação de dados pela rede não é viável, seja pela sobrecarga do enlace ou por sujeitar as informações a ataques. Como forma de contornar tais dificuldades neste trabalho, é proposto um método de classificação distribuída empregando-se uma estratégia do tipo peer-to-peer em conjunto com VCube que é um algoritmo de diagnóstico distribuído que organiza os nós da rede em uma topologia virtual de hipercubo permitindo a detecção eficiente de falhas nos nós da rede. Na solução proposta os modelos são treinados localmente e então compartilhados, evitando a necessidade de envio e exposição das informações. Durante os experimentos foram utilizados oito nós na rede no qual cada um realizava o treinamento local com o algoritmo Perceptron Multicamadas. Diferentes cenários de distribuição de dados na rede foram testados, desde variando-se a quantidade de instâncias quanto a distribuição das classes. Além disso, simulamos casos em que algum dos nós da rede poderia não estar disponível. Os resultados mostraram que o treinamento local é mais rápido em comparação ao treinamento centrado em um único nó. Os desempenhos em termos de acurácia foram maiores quando cada nó recebia os modelos treinados nos outros nós, ou seja, o sistema distribuído obteve acurácia maior em comparação à solução individual. Os resultados evidenciaram a aplicabilidade do VCube como topologia de compartilhamento dos modelos treinados. Nos casos em que um dos nós estava indisponível, a estratégia permitiu que o sistema de aprendizado distribuído pudesse funcionar adequadamente, obtendo desempenho superior aos modelos gerados em cada nó individualmente.

Palavras-chave: Classificação; Sistemas de Múltiplos Classificadores; Sistemas de Aprendizado Federado.

Abstract

SALLES, Charles Giovane de. **Distributed Machine Learning System using VCube**. Orientador: Dr. André Luiz Brun. 2024. 80f. Dissertação (Mestrado em Ciência da Computação) – Universidade Estadual do Oeste do Paraná, Cascavel – Paraná, 2024.

As the amount of data generated increases, it is no longer stored locally, resulting in distributed data scenarios. Therefore, if it is necessary to perform a classification process, i.e. the process of predicting the category of new entries based on training data, it would be necessary to consolidate the information at a central point in the network to perform the learning. However, in some situations, it is not practical to move the data across the network because the connections are congested or the information is exposed to attacks. To overcome such difficulties, a distributed classification method using a peer-to-peer strategy in conjunction with VCube is proposed in this paper. VCube is a distributed diagnosis algorithm that organizes the network nodes in a virtual topology of a hypercube, enabling efficient detection of failures in the network nodes. In the proposed solution, the models are trained locally and then shared so that no information needs to be sent and displayed. During the experiments, eight nodes were used in the network, each of which performed local training using the multilayer perceptron algorithm. Different scenarios of data distribution in the network were tested, varying the number of instances and the distribution of classes. We also simulated cases where one of the network nodes was unavailable. The results show that local training is faster than training that focuses on a single node. The performance in terms of accuracy was greater when each node received models trained on other nodes, i.e. the distributed system achieved higher accuracy than the individual solution. The results emphasize the applicability of VCube as a topology for sharing trained models. In cases where one of the nodes was unavailable, the strategy allowed the distributed learning system to function properly and achieve better performance than the models generated on each individual node.

Keywords: Classification; Multiple Classifier Systems; Federated Learning.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

API	Application Programming Interface
ANN	Artificial Neural Networks
AWS	Amazon Web Services
BRF	Radial-Basis Function
CSV	Comma-separated values
DT	Decision Trees
GB	Gigabyte
HDD	Hard Disk Drive
IDE	Integrated Development Environment
IR	Imbalance Ratio
IoT	Internet of Things
KB	Kilobyte
MB	Megabyte
MLP	Multilayer Perceptron
RNN	Redes Neurais Artificiais
SDI	Sistemas de Detecção de Intrusão
SMC	Sistemas de Múltiplos Classificadores
SVM	Support Vector Machines
TB	Terabyte
UCI	University of California, Irvine

Lista de símbolos

ξ	Letra grega minúscula csi
β	Letra grega maiúscula beta
Δ	Letra grega maiúscula delta
σ	Letra grega minúscula sigma
log	Logaritmo
$\prod$	Produtório
$\sum$	Somatório

Sumário

1 Introdução

Uma grande quantidade de dados é gerada diariamente, tanto por programas de computadores quanto aplicativos móveis, principalmente pela popularização de dispositivos como celulares, tablets e outros. Segundo Jiao et al. (2023) tecnologias inteligentes como Internet das Coisas (*Internet of Things* - IoT) se desenvolveram rapidamente com a evolução dos computadores, tornando todos os aspectos da vida das pessoas gradualmente mais inteligentes. Esse conceito está se difundindo nas mais diversas áreas como economia, defesa nacional, indústria militar e energética, e assim, gerando cada vez mais informação.

Neste sentido, Zaharia et al. (2016) afirmam que o crescimento no volume de dados tanto na pesquisa quanto na indústria, apresenta grandes oportunidades, mas também grandes desafios computacionais. Os autores relatam que o tamanho dos dados ultrapassou os recursos de uma única máquina e que os usuários precisam de novos sistemas para dimensionar os dados para outros nós.

Muitas dessas informações não se encontram agrupadas em um local único, ou seja, eles formam uma rede de dados distribuída no qual uma parte da informação pode estar perto geograficamente ou então estar em um servidor em outro país a milhares de quilômetros. Para Popovic (2017), uma rede de dados distribuída geralmente permite acesso a mais informações do que em um local único e, ao combinar dados de vários pontos de tal forma que cada um mantenha a propriedade de seus próprios dados, pode fornecer benefícios principalmente em uma tomada de decisão.

Nas áreas da ciência, finanças e medicina, os analistas lidam frequentemente com a tarefa de classificar itens com base em dados históricos ou mensuráveis. Porém, uma grande dificuldade enfrentada pelos analistas é que os dados a serem classificados muitas vezes podem ser complexos (MATISA; MAMAT, 2011), pois em algum caso estes dados não se apresentam de forma estruturada.

Segundo Brun (2017), um classificador tem como função principal atribuir a um objeto um determinado rótulo. Esse processo ocorre ao analisar o conjunto de características de um elemento e assim definir o grupo a que ele pertence. De acordo com autor, um classificador pode não ter um bom desempenho em cenários mais complexos. Nesse sentido, Kittler et al. (1998) sugere que diferentes classificadores podem oferecer informações complementares sobre um padrão a ser classificado podendo assim melhorar o desempenho do classificador em questão.

Ao associar vários classificadores se espera obter desempenhos melhores (GUNES et al., 2003). Com o objetivo de utilizar um sistema de múltiplos classificadores é necessário uma maneira de unir as opiniões dos classificadores utilizados, neste sentido, Kittler et al.

(1998) demostra algumas abordagens, onde é possível combinar diferentes classificadores, como as regras do produto, da soma, do mínimo, do máximo, da média e voto majoritário.

Neste ponto surge um questionamento: como proceder quando uma grande quantidade de dados for utilizada para classificação e reconhecimento de padrões. De acordo com Mu (2014), a migração e replicação de dados pela rede está sujeita a gargalos ao lidar com um volume de dados muito grande. Desta maneira, seria indesejável movimentar todos as informações por meio da rede até um ponto central para serem utilizados para treinamento ou classificação. Além disso, há casos em que uma organização não quer e nem pode disponibilizar as informações para outros nós. Nestes cenários, apenas o modelo de classificação pode ser compartilhado.

Sendo assim, nesta pesquisa foi elaborada uma estratégia para treinamento distribuído, no qual cada classificador utiliza os dados disponíveis localmente e após seu aprendizado o nó compartilha o seu modelo de classificação com os outros nós da rede, ao mesmo tempo que recebe modelos de classificação deles também.

A comunicação entre os nós da rede segue a topologia virtual VCube que funciona conforme as informações de diagnósticos obtidas por meio de um sistema de monitoramento de processos descritas em Duarte, Bona e Ruoso (2014b).

Essa topologia foi escolhida devido as suas propriedades logarítmicas que otimiza a detecção de falhas com baixa complexidade computacional. A estrutura do VCube consegue se adaptar dinamicamente, permitindo que nós falhem e se recuperem, enquanto a topologia se reorganiza para manter a eficiência do diagnóstico.

Na medida em que um modelo de classificação é enviado o processo de combinação de classificadores é feito de forma independente em cada nó, resultando em um modelo de classificação parcial, até que algoritmo VCube conclua sua execução e faça com que todos os nós da rede se comuniquem entre si e recebam todos os modelos dos seus vizinhos.

A utilização do VCube garante que a comunicação entre os nós se propague de forma logarítmica. Além disso, o algoritmo tem a capacidade de detectar nós em falhas dentro da rede garantindo que a comunicação dos outros nós não seja prejudicada, tornando o sistema de classificação tolerante a falhas.

1.1 Objetivos

O objetivo deste trabalho é propor uma abordagem de classificação distribuída, onde o sistema como um todo não possui todos os dados centralizados, apenas as informações presentes em seu nó. A estratégia proposta Utiliza o algoritmo VCube para comunicação entre todos os nós garantindo um sistema tolerante a falhas dinâmico com propriedades de execução logarítmicas.

1.1.1 Objetivos Específicos

• Implementação de um framework para simulação de uma rede geograficamente distribuída, em que cada nó contém parte do conjunto de dados;

• Protocolos experimentais com diferentes configurações de distribuição de dados;

• Avaliação dos resultados obtidos em termos de tempo de treinamento e custo de transmissão dos modelos pela rede;

• Avaliação dos resultados obtidos em termos de acurácia para determinar a viabilidade do cenário distribuído frente à abordagem centralizadora tradicional, assim como identificar quais configurações são mais promissoras em termos de taxa de acertos e de adaptabilidade em momentos de falhas de comunicação entre os nós da rede;

Espera-se que o treinamento distribuído proposto neste trabalho obtenha um bom desempenho, sendo similar à estratégia que agruparia todos os dados disponíveis em um ponto central e assim realizasse o treinamento.

1.2 Organização do Trabalho

Este trabalho está organizado da seguinte forma: O Capítulo 2 contém uma visão geral sobre algoritmos de Classificação e sistemas de múltiplos classificadores. O Capítulo 3 contém técnicas e formas de aprendizagem distribuída e como funciona o algoritmo de comunicação tolerante a falhas VCube. Já no Capítulo 4, é apresentada a metodologia que foi implementada. No Capítulo 5 é abordado o principais resultados dos experimentos realizados a partir da metodologia adotada. No Capitulo 6 é destacado as principais conclusões obtidas com base no conhecimento construído ao longo da produção deste trabalho. Por fim as referência utilizadas como embasamento teórico do trabalho.

2 Classificação

A expansão da internet e tecnologias de comunicação contribui para a geração *terabytes* de dados todos os dias, nas mais diversas áreas, como bancos, hospitais, laboratórios científicos entre outros. O desafio é fazer uso efetivo dessas informações. Por exemplo, um banco poderia aprender sobre o padrão de gastos dos seus clientes e assim detectar uma fraude nos seus cartões de créditos rapidamente, como demonstrado em Dornadula e Geetha (2019). Da mesma forma, carros inteligentes poderiam informar previamente seus proprietários sobre problemas mecânicos prestes a acontecer (MARSLAND, 2014). Além disso, computadores poderiam aprender, a partir de registros médicos, qual tratamento é mais efetivo para uma nova doença, casas poderiam aprender a partir da experiência a diminuir o uso de energia baseado no padrão de uso dos seus ocupantes ou um software de assistência pessoal aprender sobre os interesses dos seus usuários com o objetivo de destacar histórias relevantes das notícias matinais diárias (MITCHELL, 1997).

Nas áreas da ciência, finanças e medicina, os analistas lidam frequentemente com a tarefa de classificar itens com base em dados históricos ou mensuráveis. Porém, uma grande dificuldade enfrentada pelos analistas é que os dados a serem classificados muitas vezes podem ser bastante complexos (MATISA; MAMAT, 2011). Segundo Marsland (2014) o tamanho e a complexidade das informações geradas inferem que humanos são incapazes de extrair informações pertinentes delas, assim, uma mescla de ideias da neurociência, biologia, estatísticas, matemática e físicas contribuíram para os computadores aprendessem.

Um sucesso na compreensão de como fazer as máquinas aprenderem possibilitaria novas aplicações para o computador e novos níveis de competências e customização, assim como um entendimento detalhado de um algoritmo de processamento de informação para aprendizado de máquina pode levar a uma melhor compreensão das habilidades de aprendizagem humana (MITCHELL, 1997).

Tendo em vista as diversas áreas de aplicação, é necessário definir uma forma de verificar se o computador está ficando melhor em alguma tarefa estabelecida. Uma dessas formas é apresentar ao algoritmo de aprendizado a resposta correta para o problema. Assim esperamos que o algoritmo possa trabalhar para conseguir respostas corretas para outros problemas. Desta maneira, Marsland (2014) classifica os problemas de aprendizado em Supervisionado e não-supervisionado.

A primeira estratégia dá-se quando um conjunto de treino com as respostas corretas é fornecido e baseado nisso o algoritmo constrói um modelo para responder corretamente todas as entradas possíveis, essa forma também é chamada de aprendizado por exemplares.

Por sua vez, o aprendizado não-supervisionado ocorre quando as respostas corretas

não são informadas. Assim, o algoritmo tenta identificar similaridades entre as entradas, de forma que aquelas que têm algo em comum são categorizadas juntas.

Uma vez que as abordagens mais comuns de classificação lidam com aprendizagem supervisionada, esta receberá o foco nesta pesquisa.

Neste tipo de aprendizado existe um conjunto de dados que consiste em um grupo de características de entrada que está associada a um dado de saída, que é a saída que o algoritmo deve gerar. Normalmente intitulada classe. Isso é geralmente escrito como um conjunto de dados $(a_1, a_2, ..., a_i, t_i)$, onde $a_1, a_2, ..., a_i$ representam as entradas do problema e t_i a saída (MARSLAND, 2014).

2.1 Problemas de Classificação

Segundo Marsland (2014) os problemas de classificação consistem analisar um conjunto de entradas e decidir em qual das N classes disponíveis elas pertencem, baseado no treinamento dos exemplares de cada classe. Uma característica importante sobre o problema de classificação é que ele é discreto, ou seja, cada exemplar pertence a uma classe precisamente e que o conjunto de classe cobre todo o espaço de entradas possíveis.

Neste sentido, uma técnica utilizada na área da Inteligência Artificial mais precisamente no campo de Aprendizado de Máquina, é a utilização de classificadores, onde o objetivo é encontrar um padrão nas informações analisadas. Desta maneira apresentamos o classificador que, segundo Brun (2017), tem como função principal atribuir a um objeto um determinado rótulo. Esse processo ocorre ao analisar o conjunto de características de um elemento e, a partir delas, definir o grupo a que ele pertence. De acordo com autor, um classificador pode não ter um bom desempenho em cenários mais complexos.

Para ilustrar o processo de classificação vamos considerar a configuração de um classificador de moedas. Quando a moeda é colocada na máquina algumas informações sobre esta são capturadas. Tais informações podem se referir ao diâmetro, peso ou forma. Essas características formaram o vetor de entrada do classificador, chamado vetor de características. Neste caso um vetor com três elementos, onde cada um será representado por um valor, inclusive o formato (neste caso escolhendo um número para representar determinada forma, por exemplo, $1 = círculo, 2 = hexágono$ e assim por diante) (MARSLAND, 2014).

Existem outras características que poderiam ser levadas em consideração como a densidade, ou então, utilizar descritores visuais a partir de imagens capturadas das moedas. O fato é que a escolha das características que serão utilizadas também é uma tarefa difícil (ZHONG; DONG; OHSUGA, 2001).

Deve existir um meio termo na quantidade de informações que serão utilizadas, pois se temos muitas entradas, mais tempo de treinamento o algoritmo de classificação vai

levar (MARSLAND, 2014).

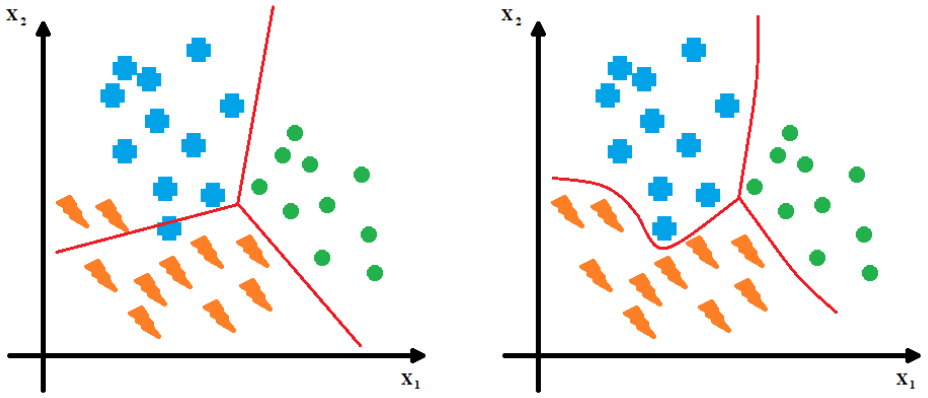
Outro ponto importante, como apontado por Marsland (2014), é que precisamos ter certeza que podemos separar de forma confiável as classes com base nas características escolhidas.

Tomando por exemplo o problema da classificação das moedas (MARSLAND, 2014), se quisermos separá-las apenas pela sua cor, seria uma tarefa muito difícil, pois alguns tipos de moedas só possuem como cor uma tonalidade de prata. Porém, ao unirmos a tonalidade e o diâmetro teríamos um bom conjunto de descritores. Neste sentido existem característica que podem não contribuir para a resolução do problema, por exemplo, se classificarmos as moedas de "Real" e considerarmos como um atributo o formato circular.

Os algoritmos de classificação são diferentes na forma que em que eles aprendem a solução. Entretanto, na essência, todos realizam a mesma tarefa que consiste em encontrar uma regra que consegue separar diferentes classes (LEBAN et al., 2006). Dadas as características usadas como entradas para os classificadores, precisa-se identificar alguns valores dentre aquelas características que serão capazes de dizer qual classe aquela entrada pertence.

A Figura 1 mostra um cenário em que o conjunto de descritores é composto por duas características, representadas nos eixos X e Y. O problema contém exemplares de três diferentes classes, representadas por diferentes marcadores. Na representação à esquerda temos linhas retas, que são exemplares mais simples de classificação, porém não categorizam cada classe como uma curva não-linear da direita.

Figura 1 – Na direita um conjunto de linhas de bordas retas para um problema de classificação, na direita linhas de bordas curvadas que permite uma separação melhor das classes (MARSLAND, 2014)



2.2 Algoritmos de Aprendizado de Máquina

Segundo Hastie, Tibshirani e Friedman (2009), uma grande variedade de métodos foi desenvolvida para a aprendizagem preditiva a partir de informações e, para cada método em particular, existem situações em que ele é mais adequado e cenários em que o mesmo algoritmo pode ter um desempenho ruim.

Neste sentido, apresentamos os Sistemas de Múltiplos Classificadores mais conhecidos como SMC's. Essa abordagem visa aumentar a eficiência da classificação dos classificadores "fracos", ou seja, aqueles que apresentam uma taxa de precisão baixa, pois Ko, Sabourin e Britto Jr (2008) afirmam que classificadores diferentes geralmente cometem erros diferentes em amostras diferentes. Isso significa que a combinação de classificadores gera uma decisão mais precisa.

Primeiramente serão apresentados os classificadores individuais abordados neste trabalho que são Naïve Bayes, Árvore de Decisão (*Decision Tree* - DT), Máquina de Vetores de Suporte (*Support Vector Machines* - SVM) e Redes Neurais Artificiais (*Artificial Neural Networks* - ANNs).

Na sequência, será apresentado a abordagem com SMC's e as técnicas para a combinação destes classificadores individuais sendo a Regra do produto, Regra do produto, Regra do Máximo e do Mínimo, Regra da Média e a Regra do Voto Majoritário, onde Jain, Duin e Mao (2000) cita alguns motivos para combinar classificadores para resolver problemas de classificação:

1. Um projetista tem acesso a classificadores diferentes, onde cada um foi desenvolvido em um contexto diferente e com uma representação completa do mesmo problema. Um exemplo disso, seria a identificação de uma pessoa pela voz, pelo rosto, ou pela forma de escrever.

2. Possibilidade de mais de um conjunto de treinamento, cada um coletado em momento e ambientes diferentes. Esses conjuntos podem usar atributos diferentes.

3. Classificadores diferentes treinados com os mesmos dados podem divergir no desempenho global e também no desempenho local, onde cada classificador pode possuir uma região no espaço de características onde possui um desempenho melhor.

4. Em alguns classificadores tal como redes neurais mostram resultados diferente devido a aleatoriedade da inicialização do processo de treinamento. Assim em vez de selecionar a melhor rede neural e descartar as outras, a combinação poderia tirar vantagem de cada uma delas.

2.3 Sistemas Monolíticos

Nessa seção serão apresentados alguns sistemas de classificação monolíticos, que utilizam uma determinada estratégia individual, tendo como objetivo definir a qual classe um objeto pertence.

2.3.1 Naïve Bayes

O classificador Naïve Bayes se aplica da seguinte forma: cada instância x é descrita por uma conjunção de valores de atributos e onde a função alvo *f(x)* pode assumir qualquer valor de algum conjunto finito C. Assim, um conjunto de exemplares de treinamento da função alvo é fornecido e uma nova instância é apresentada e descrita pela tupla de valores de atributo X = (a_1, a_2, a_3, ... a_n). Desta maneira, o classificador então é solicitado a prever o valor alvo, ou classificação, para esta nova instância.

A abordagem Bayesiana para classificar a nova instância consiste em atribuir o valor alvo mais provável P(C_j), dados os valores de atributo X = (a_1, a_2, a_3, ... a_n) que descrevem o novo exemplar.

A equação que representa o teorema de Bayes pode ser visto na Equação 2.1.

$$P(C_i|X) = \frac{P(X|C_i)P(C_i)}{P(X)} = P(X|C_i)P(C_i) \tag{2.1}$$

Onde:

- P(C_i | X): representa a probabilidade de uma classe C_i dado uma entrada X = (a_1, a_2, a_3, ... a_n).
- P(X | C_i): probabilidade da entrada X = (a_1, a_2, a_3, ... a_n) dada a classe C_i.
- P(C_i): probabilidade a *priori* da classe C_i.
- P(X): probabilidade a *priori* da entrada de treinamento X= (a_1, a_2, a_3, ... a_n).

É possível estimar cada um dos P(C_i) da Equação 2.1 com base nos dados de treinamento, simplesmente contando a frequência com que cada valor alvo C_i ocorre nos dados de treinamento (MITCHELL, 1997).

O classificador Naïve Bayes é baseado na suposição simplificadora de que os valores dos atributos são condicionalmente independentes, dado o valor alvo. Em outras palavras, a suposição é que dado o valor alvo da instância, a probabilidade de observar a conjunção X = (a_1, a_2, a_3, ... a_n), é apenas o produto das probabilidades para os atributos individuais, conforme detalhado na Equação 2.2.

$$P(a_1, a_2, a_3 ... a_n) = \prod_i P(a_i|C_j) \tag{2.2}$$

Substituindo o produto das probabilidades na Equação 2.1, temos a abordagem utilizada pelo classificador Naïve Bayes.

$$V_{nb} = argmax\, P(C_j) \prod_i P(a_i|C_j) \tag{2.3}$$

Onde V_{nb} denota a saída do valor alvo pelo classificador. Observe que em um classificador Naïve Bayes o número de termos $\mathrm{P}(X_i \mid C_j)$ distintos que devem ser estimados a partir dos dados de treinamento é apenas o número de valores de atributos distintos multiplicado pelo número de valores alvo distintos - um número muito menor do que se estivéssemos estimando os termos $\mathrm{P}(X_i \mid C_j)$ como primeiro contemplados (MITCHELL, 1997).

É possível perceber que método de aprendizagem Naïve Bayes envolve uma etapa de aprendizagem na qual os vários termos $\mathrm{P}(C_j)$ e $\mathrm{P}(X_i \mid C_j)$ são estimados. Com base em suas frequências sobre os dados de treinamento. O conjunto dessas estimativas corresponde à hipótese aprendida a qual é então usada para classificar cada nova instância aplicando a regra definida pela Equação 2.3. Sempre que a suposição de independência condicional de Naïve Bayes é satisfeita, essa classificação V_{nb} é semelhante à classificação de hipótese máxima a posteriori (MAP) (MITCHELL, 1997).

Uma diferença interessante entre o método de aprendizado Naïve Bayes e outras estratégias de aprendizado é que não há busca explícita através do espaço de hipóteses possíveis (neste caso, o espaço de hipóteses possíveis é o espaço de valores possíveis que podem ser atribuídos a os vários termos $\mathrm{P}(C_j)$ e $\mathrm{P}(X_i, C_j)$). Em vez disso, a hipótese é formada sem pesquisa, simplesmente contando a frequência de várias combinações de dados nos exemplos de treinamento (MITCHELL, 1997).

Segundo Mitchell (1997), em alguns casos, principalmente quando o problema possui características discretas, o desempenho do classificador Naïve Bayes tem se mostrado comparável aos métodos que utilizam redes neurais artificiais e as árvores de decisão.

2.3.2 Árvores de Decisões (*Decision Trees* - DTs)

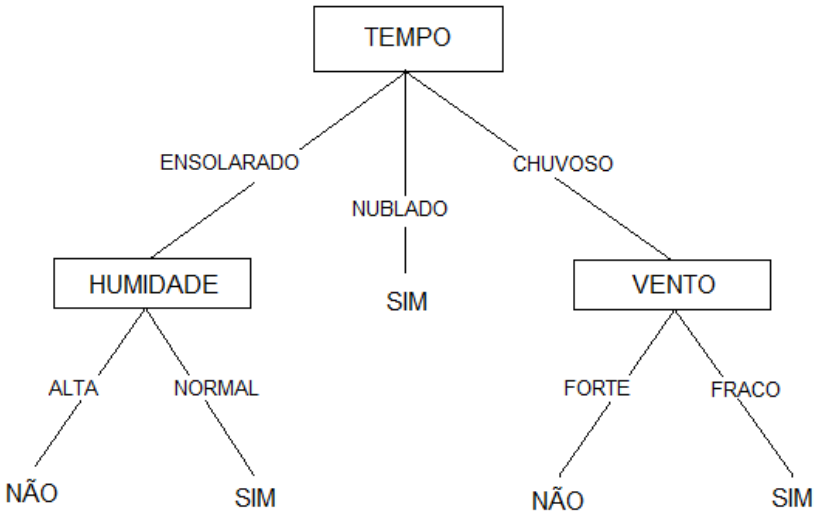
Segundo Mitchell (1997), o aprendizado utilizando Árvore de Decisão é um dos métodos mais comuns e práticos para a inferência indutiva. Além disso, é um método robusto e capaz de lidar com o ruído dos dados. Algumas versões deste algoritmo foram desenvolvidas das quais pode-se citar o ID3 proposto por Quinlan (1986), C4.5 proposto Quinlan (1993), ASSISTANT proposto por Cestnik, Kononenko e Bratko (1987) e CART proposto por Friedman, Bentley e Finkel (1977) e Breiman et al. (1984). Esses algoritmos

estão entre os mais populares, sendo aplicados com sucesso a uma grande gama de tarefas de aprendizado como diagnóstico médico e avaliação de risco de crédito em instituições bancárias.

A ideia básica do funcionamento da Árvore de Decisão é classificar as instâncias separando-as da raiz da árvore até algum nó folha testando o atributo especificado para o nó atual, que fornecerá a classificação daquela amostra. Este processo é repetido para a subárvore tendo o nó atual como raiz. Podemos visualizar essa representação na Figura 2, onde temos uma árvore de decisão que avalia dado algumas condições climáticas se o jogador deve ou não jogar tênis.

Na Figura 2 cada nó da árvore representa um atributo, onde cada um dos ramos determina um valor possível para este nó, este processo termina quando chega nas folhas da árvore que contém as saídas para o problema de classificação.

Figura 2 – Árvore de decisão que determina se o jogador deve o não jogar tênis dado alguns atributos do clima (MITCHELL, 1997).



Segundo Mitchell (1997), apesar de uma grande variedade de métodos baseados em árvores de decisão tenha sido desenvolvida, elas normalmente apresentam um desempenho melhor em problemas com as seguintes características:

- Quando as instâncias são representadas por pares atributo-valor. Ou seja, essas instâncias têm um conjunto de atributos possíveis (espaço discreto). Além disso, problemas onde cada atributo tenha um número pequeno de valores facilita o trabalho do algoritmo. Como no exemplo da Figura 2, o atributo "tempo" possui apenas os valores "ensolarado, nublado e chuvoso" possível.
- Quando a função objetivo possui valores discretos como saída, como no exemplo da Figura 2, onde a saída é booleana, ou seja, cada entrada tem como resposta os valores "sim" ou "não".

- Quando o conjunto de treinamento contém erros ou com valores faltantes, pois neste sentido as árvores de decisão são robustas.

Muitos problemas práticos têm essas características. Murata et al. (2019) utiliza como base um método de aprendizado de máquina baseado em árvore de decisão para identificar pacientes com câncer de mama. Nasseri, Tucker e Cesare (2015) analisam o comportamento do mercado financeiro baseado em previsão de sentimento, técnicas de mineração de texto, utilizando o algoritmo de árvore de decisão, buscando o melhor momento para vender, comprar ou "segurar" o ativo. Wang e Kong (2019) trabalham com o conceito de cidades inteligentes, onde previsões eficientes e precisas sobre os níveis de qualidade do ar podem fornecer uma base confiável para decisões sociais, neste sentido um método de árvore de decisão é proposto. Todos esses problemas a classificação a função objetivo é escolhida de acordo com um grupo discreto de possibilidades.

Dentre os algoritmos que implementam as árvores de decisão podemos destacar o ID3 propostos por Quinlan (1986) e o seu sucessor, o C4.5, proposto por Quinlan (1993). A ideia básica destes algoritmos é escolher o melhor atributo para colocar na raiz da árvore, ou seja, aquele atributo que melhor divide o conjunto de treinamento, assim em cada atributo é calculado um valor estatístico. Todo esse processo e repetido até o algoritmo chegar a uma conclusão.

Para isso é definido uma propriedade estatística chamada de ganho de informação, que mede o quanto um atributo separa o conjunto de treinamento de acordo com a função objetivo. Para definir o ganho de informação é preciso calcular a entropia que caracteriza a pureza de um conjunto de exemplos. Dado um atributo que pode conter c valores diferentes, a entropia do conjunto S relativo ao atributo pode ser calculado por:

$$Entropia(S) = \sum_{i=1}^{c} -p_i \log_2 \ p_i \tag{2.4}$$

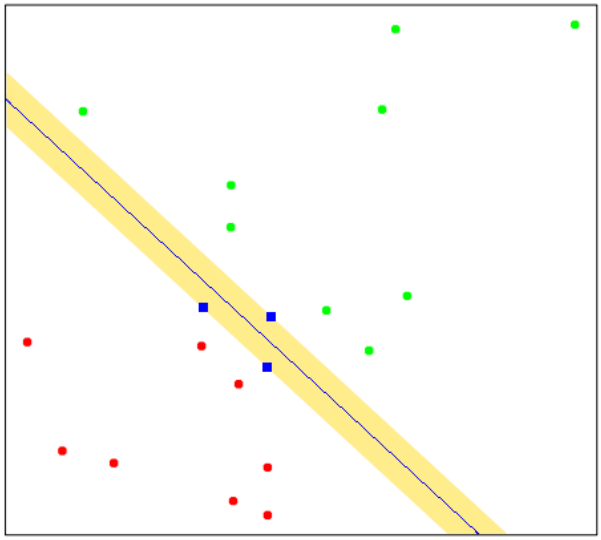
Onde p_i é a proporção de S que pertence a classe i. Dessa forma para cada nó da árvore começando pela raiz o algoritmo divide o conjunto de dados de acordo com a variável que produz o melhor ganho de informação.

2.3.3 Máquina de Vetores de Suporte (*Support Vector Machines* - SVM)

De acordo com Vapnik, Golowich e Smola (1996) o método de vetores de suporte é uma ferramenta universal para resolver problemas de estimação de funções multidimensional, ela foi inicialmente desenvolvida para solucionar problemas de reconhecimento de padrão, que é capaz de encontrar uma boa regra de generalização selecionando um grupo pequeno de dados de treinamento chamados de vetores de suporte (SVs), sendo que a separação ótima dos SVs é equivalente a separação ótima do conjunto inteiro dos dados.

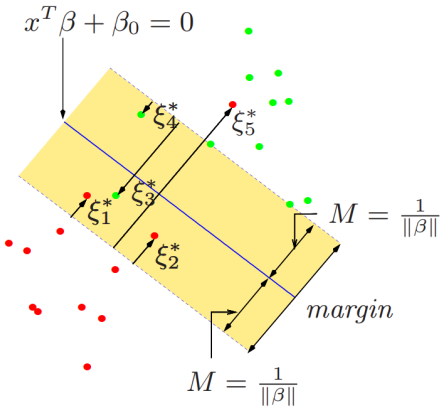
Neste sentido Hastie, Tibshirani e Friedman (2009) argumentam que a SVM consegue produzir limites não lineares aplicando limites lineares em uma transformação do espaço de características. Essa técnica consegue construir um hiperplano ótimo em um problema onde duas classes são perfeitamente separáveis. Além disso, maximiza a margem entre duas classes no conjunto de treinamento, isso leva a um melhor desempenho na classificação (Figura 3).

Figura 3 – O região sombreada determina a margem máxima de separação de duas classes. Existem três pontos de suporte, que ficam no limite da margem, onde a separação ótima, ou seja, o hiperplano divide a faixa exatamente ao meio (HASTIE; TIBSHIRANI; FRIEDMAN, 2009).



No entanto para construir um hiperplano ótimo quando os dados não são linearmente separáveis, Vapnik (1995) acrescenta uma variável $\xi \geq 0$:

Figura 4 – A figura mostra um problema de classificação não separável, ou seja, possui dados sobrepostos, onde ξ_i é calculado de acordo com a distância com a sua margem. Portanto $\sum \xi_i$ é a distância total dos pontos que estão do lado errado da sua margem. (HASTIE; TIBSHIRANI; FRIEDMAN, 2009).



Onde a solução para esse problema de otimização é um hiperplano que minimiza o valor de erro do treinamento (Equação 2.5).

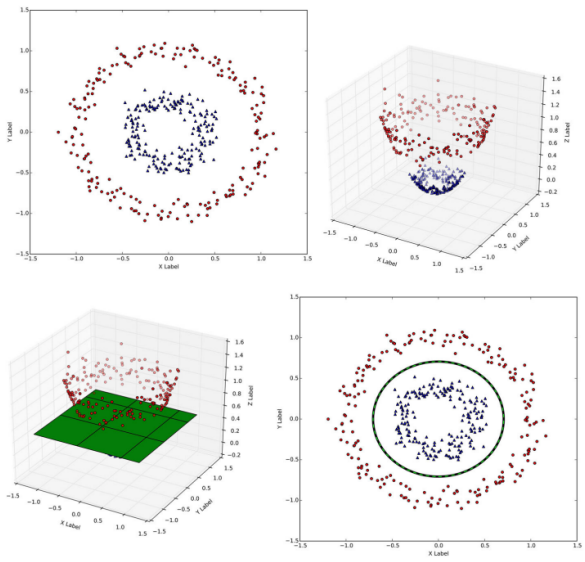
$$F(\xi) = \sum_{i=0}^{n} \xi_i^{\sigma} \tag{2.5}$$

Desta maneira quanto menor for o valor erro permitido na construção do hiperplano, o modelo ficará mais sujeito a um *overfitting*, ou seja, o modelo gerado se ajusta muito aos dados de treinamento se mostrando ineficiente ao classificar novas informações de entrada.

Pelo contrário se o erro permitido na construção do modelo for maior isso tornará o modelo mais generalista capaz de classificar corretamente outros dados de entrada. O valor do erro deve ser ajustado de acordo com cada problema, pois se esse valo for muito grande o classificador vai perder desempenho em relação a taxa de acertos.

Para auxiliar no processo de classificação Vapnik (1995) apresenta o conceito de kernel, que se trata de uma função que é aplicada a um espaço de características, com o objetivo deixar mais claro a divisão entre as classes. Entre alguns Kernels utilizados podemos citar Polinomial, BRF (*Radial-Basis Function*) e Sigmoidal. Na Figura 5 podemos ver um exemplo de uma transformação nas informações utilizando um Kernel polinomial.

Figura 5 – Nesta figura temos um exemplo de como um kernel polinomial é usado para transformar o espaço dos dados em uma dimensão superior. No quadro superior esquerdo mostra os dados originais no plano bidimensional, já o quadro superior direito mostra os dados transformados em um novo espaço de características. O quadro inferior esquerdo mostra o hiperplano que divide as duas classes, já o quadro inferior direito mostra a projeção do hiperplano no espaço de características original (THARWAT, 2019).



2.3.4 Redes Neurais Artificiais – Multilayer Perceptron (MLP)

De acordo com Haykin (2009) o trabalho em "redes neurais" foi motivado pelo reconhecimento de que o cérebro humano computa de maneira diferente que um computador digital tradicional, pois o cérebro é complexo capaz de processar vários tipos de informação, que possui a capacidade de se organizar em estruturas chamada de "neurônio". Além disso, o cérebro é capaz de aprender por meio de "experiências" ao longo da vida

Deforma geral Haykin (2009) define um rede neural como uma máquina desenhada para modelar a forma que o cérebro resolver uma tarefa em particular e para alcançar um bom desempenho a rede neural emprega uma conexão densa de células computacionais chamados de "neurônios" ou unidades de processamento.

As redes neurais entregam um grande poder computacional, primeiro por sua estrutura distribuída e paralela e segundo pela sua habilidade de aprender algo e depois generalizar. A abordagem utilizada pelas redes neurais tem como princípio de funcionamento resolver um problema complexo, decompondo-o em tarefas menores e mais simples e assim resolve-los para encontrar a solução completa.

A utilização da redes neurais oferece os seguinte benefícios e capacidades, resolução de problemas não lineares, mapeamento de entrada e saída, capacidade de se adaptar ao ambiente, capacidade de dizer sobre o grau de confiança do resultado encontrado, levar em consideração a informação do contexto, tolerante a falhas, adequada para a implementação em grande escala, uniformidade de análise e analogia ao funcionamento ao cérebro humano (HAYKIN, 2009).

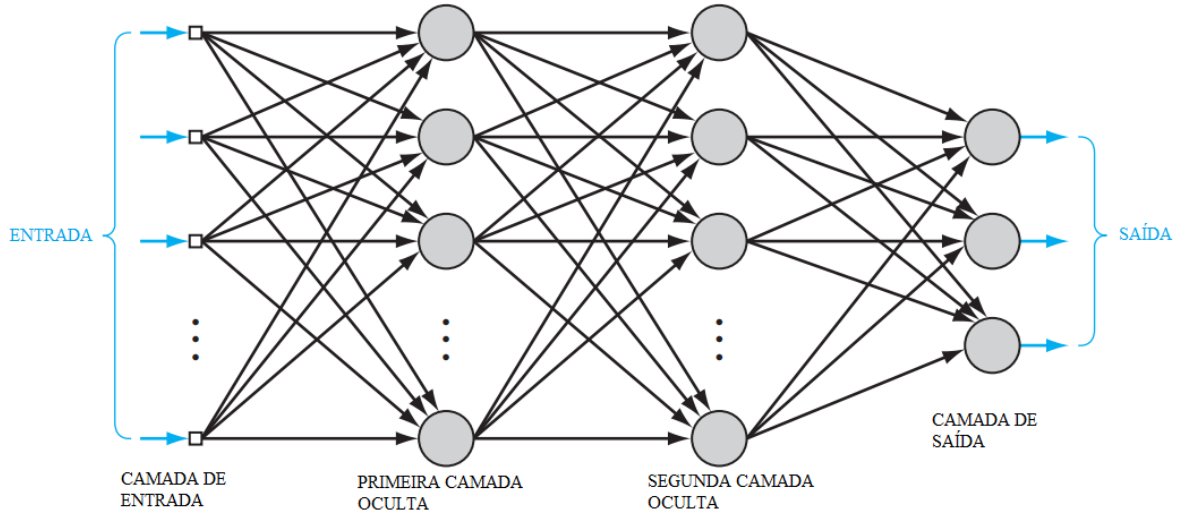
Os três pontos a seguir destacam as característica básicas do *multilayer perceptron*:

- O modelo de cada neurônio na rede inclui uma função de ativação não linear.
- A rede contém uma ou mais camada que são ocultas tanto dos nós de entradas quando dos nós de saída.
- A rede possui um alto grau de conectividade.

A arquitetura do *multilayer perceptron* consiste basicamente no sinal de entradas, em seguida uma ou mais camadas ocultas e uma camada de saída (Figura 6). Em geral a rede formada pelo classificador é totalmente conectada. Isto significa que um neurônio (nó) em qualquer camada é conectado a todos os neurônios (nós) da camada anterior. O fluxo de processamento segue iniciando da esquerda até a direita de camada em camada.

Ao receber a entrada o classificador inicializa os pesos das arestas (inicialmente valores aleatórios) e calcula o valor para cada nó. Esse processo segue por cada uma das camadas até chegar a última camada. Um ponto importante das redes neurais é a

Figura 6 – Arquitetura representada graficamente o classificador *multilayer perceptron* com duas camadas ocultas(HAYKIN, 2009)



utilização das funções de ativação, onde para cada neurônio uma transformação é aplicada decidindo basicamente se o neurônio será ativado ou não, alguns exemplos mais comuns de funções de ativação são função linear, sigmoide, tangente hiperbólica e função ReLU.

Por fim o algoritmo responsável pelo treinamento da rede neural chamado de *Backpropagation*, cuja a função é ajustar o valor dos pesos das arestas afim de alcançar a classificação de acordo com o conjunto de treinamento. Segundo MIGUEZ, MACULAN e XAVIER (2011) esse tipo de rede apresenta solução para problemas que não são linearmente separáveis, desta maneira é necessário um algoritmo capaz de ajustar os pesos automaticamente.

Na fase de treinamento o algoritmo *Backpropagation* atua em duas etapas. Na primeira uma entrada é fornecida a rede, que avança através das camadas até chegar à camada de saída. Já na segunda etapa a saída gerada pela rede neural é comparada a saída desejada e caso não esteja correta um erro será calculado, desta maneira o erro é propagado a partir da camada de saída até a camada de entrada, assim o peso das arestas de todas as camadas vão sendo modificados de acordo com a retro propagação do erro (MIGUEZ; MACULAN; XAVIER, 2011).

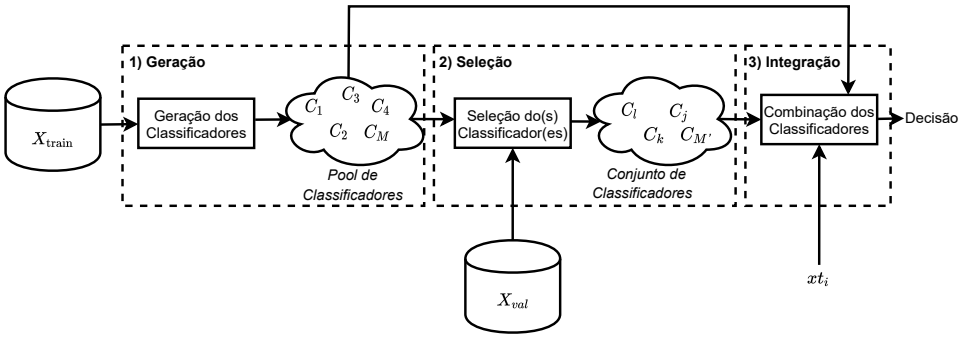
2.4 Sistemas de Múltiplos Classificadores (SMC)

Segundo Kittler et al. (1998) o principal objetivos dos sistemas de classificação é ter a melhor ferramenta de classificação a disposição e por conta disso diferentes esquemas de classificação foram desenvolvidos. Os autores sugerem que diferentes classificadores não se sobrepõem, ou seja, eles têm o potencial para oferecer informações complementares

sobre um padrão que será classificado.

A Figura 7 mostra o esquema de funcionamento de um sistema de múltiplos classificadores e suas fases. Na primeira, o conjunto de classificadores é gerado. Na segunda fase, um subconjunto destes classificadores é selecionado. Na última fase, a decisão é feita baseada na predição dos classificadores selecionados (BRITTO Jr; SABOURIN; OLIVEIRA, 2014).

Figura 7 – Esquema de Sistema de Múltiplos Classificadores (adaptado de Britto Jr, Sabourin e Oliveira (2014))



Kheradpisheh, Behjati-Ardakani e Ebrahimpour (2013) dizem que a ideia básica por trás da combinação de classificadores é inspirada na tendência das pessoas em procurar várias opiniões para tomar decisões importantes.

Em geral sistemas que utilizam combinação de classificadores são mais generalistas e precisos que um classificador único (PARVIN et al., 2008).

A possibilidade de interação entre vários módulos de um sistema de classificação no qual os vetores de características escolhidos para os classificadores podem ser diferentes, desde que eles se tratem do mesmo padrão a ser reconhecido (GUNES et al., 2003).

Kittler et al. (1998) propõe alguns métodos para realizar a combinação das opiniões dos classificadores que serão vistos a seguir.

2.4.1 Regra do produto

Dada a representação das probabilidades extraídas dos classificadores $p(x_1, ..., x_n|w_k)$, temos que a probabilidade de todos os classificadores concordarem em uma classe é produto de todas as probabilidades conforme a Equação 2.6.

$$p(x_1, ..., x_n)|w_k) = \prod_{i=1}^{n} p(x_i|w_k) \tag{2.6}$$

Consequentemente é possível obter a regra geral de decisão conforme a Equação 2.7:

$$P(w_j)\prod_{i=1}^{n} p(x_i|w_j) = max_{k=1}^{m}\ P(w_k)\prod_{i=1}^{n} p(x_i|w_k) \tag{2.7}$$

Desta maneira, esta abordagem quantifica a probabilidade de uma hipótese combinando as probabilidades geradas pelos classificadores individuais. Efetivamente este classificador utiliza uma regra muito rígida, pois apenas um classificador pode diminuir significativamente a probabilidade de uma hipótese caso este classificador tenha uma probabilidade próxima a zero (KITTLER et al., 1998).

2.4.2 Regra da soma

Considerando a Regra do Produto é desejável que as probabilidades calculadas pelos classificadores não se afastem demasiadamente quando um ou poucos classificadores tiverem opiniões diferentes. Kittler et al. (1998) argumenta que esta suposição é bastante forte, porém ela é satisfeita quando as caraterísticas das informações demostram uma natureza ambígua devido aos altos níveis de ruídos.

Temos que a probabilidade de todos os classificadores concordarem em uma classe é soma de todas as probabilidades (Equação 2.8).

$$max_{k=1}^{m}\left[(1-n)P(w_k) + \sum_{i=1}^{n} P(w_k|x_i)\right] \tag{2.8}$$

2.4.3 Regra do Máximo e do Mínimo

A regra do máximo busca dentre todos os classificadores aquele que possui a maior probabilidade (grau de certeza) da classe analisada (Equação 2.9).

$$max_{i=1}^{n} P(w_j|x_i) \tag{2.9}$$

A regra do mínimo busca dentre todos os classificadores aquele que possui a menor probabilidade (grau de certeza) da classe analisada (Equação 2.10).

$$min_{i=1}^{n} P(w_j|x_i) \tag{2.10}$$

2.4.4 Regra da Média

A Regra da Média retorna o resultado médio de todos os classificadores conforme Equação 2.11. Segundo Salvadeo (2009) a Regra da Soma poder ser visto como uma Regra da Média, considerando que as probabilidades a priori são iguais.

$$med_{i=1}^{n} P(w_j|x_i) \tag{2.11}$$

2.4.5 Regra do Voto Majoritário

Na regra do voto majoritário cada classe w_k simplesmente quantifica os votos recebidos para esta hipótese dos classificadores de forma individual. Desta maneira a classe que recebeu o maior número de votos será a classe selecionada, ou seja, os classificadores entram em um consenso onde é respeitada a decisão da maioria (Equação 2.12)(KITTLER et al., 1998).

$$\sum_{i=1}^{n} \Delta_{ji} = max_{k=1}^{m} \sum_{i=1}^{n} \Delta_{ki} \tag{2.12}$$

3 Aprendizado Distribuído

Como visto no capítulo anterior, o conceito de aprendizagem costuma ser tratado como um processo local, considerando a disponibilidade completa dos dados para treinamento e avaliação dos modelos. No entanto, devido às características atuais, muitas vezes as informações obtidas podem estar distribuídas geograficamente, demandando abordagens evoluídas para a construção de um modelo de classificação.

Segundo Chen (2018) os dados já são naturalmente distribuídos, principalmente em razão da privacidade ou da segurança em alguns casos, pois se uma coleta central das informações for necessária existe a possibilidade de vazamento de informações neste processo. Neste sentido, Lu et al. (2020) complementam que a coleta de grande quantidade de informação de fontes de dados distribuídas, apresentam alguns obstáculos incluindo limitação na conectividade de rede e largura de banda, limitação no consumo de energia e a necessidade de preservar a privacidade dos dados brutos.

Outro fator importante é que um gargalo que geralmente impede o desenvolvimento de sistemas mais inteligentes é a quantidade limitada de dados. Sem dados suficientes, os sistemas encontram dificuldades para tomar decisões inteligentes como os humanos (GUO; ZHANG, 2016). Em seu trabalho, Peteiro-Barral e Guijarro-Berdiñas (2013) afirmam que, tradicionalmente, um gargalo que impedia o desenvolvimento de sistemas mais robustos era a quantidade limitada de dados disponíveis. Nesta situação, desafios são levantados em relação à escalabilidade e eficiência dos algoritmos de aprendizagem no que diz respeito aos recursos computacionais e a maioria das implementações existentes de algoritmos operam com o conjunto de treinamento inteiramente na memória principal.

Este cenário foi sendo redefinido com o crescimento das tecnologias de detecção e coleta de dados desenvolvidas e adotadas nos últimos anos. Porém, muitos dos algoritmos tradicionais de aprendizado de máquina exigem o carregamento de todo o conjunto de dados em um computador, o que torna essa etapa quase impossível, pois o volume de dados é impeditivo (GUO; ZHANG, 2016). Dados tais desafios, uma abordagem para ser utilizada nestes casos é o aprendizado de máquina distribuído. Tsoumakas e Vlahavas (2009) apresenta algumas vantagens em manter os dados distribuídos ao invés de agrupá-los para um processo de classificação.

- O custo de armazenamento de um conjunto de dados central é muito maior do que a soma do custo de armazenamento de partes menores do conjunto de dados;
- O custo computacional de mineração de um banco de dados central é muito maior do que a soma do custo de análise de partes menores dos dados;

- A transferência de grandes volumes de dados pela rede pode levar muito tempo e também requer um custo financeiro alto;
- Os dados podem ser privados ou confidenciais, como os registros médicos e financeiros das pessoas.

Como a demanda por processamento de dados de treinamento dos algoritmos de classificação ultrapassou o poder de computação das máquinas, é necessário distribuir a carga de trabalho de aprendizado realizado por uma máquina de forma local para várias máquinas e transformar o sistema centralizado em um sistema distribuído (VERBRAEKEN et al., 2020).

Segundo Chen (2018), o objetivo do aprendizado distribuído é produzir um resultado de aprendizado com base em todos os conjuntos de dados. os resultados do aprendizado local precisam ser combinados de alguma forma, pois na aprendizagem distribuída não existe um classificador global para receber a mensagem de todos os classificadores locais nem a transmissão de dados para um ponto central da rede. Para isso, geralmente, existem dois métodos diferentes que podem ser utilizados. O primeiro consiste em mesclar as previsões de classificadores locais. A segunda estratégia dá-se pela combinação dos modelos de classificação.

3.1 Informações que podem ser compartilhadas

Neste processo de aprendizado distribuído é necessário definir que tipo de informação será compartilhada entre os nós do sistema uma vez que os dados não serão repassados pela rede. Desta maneira Lu et al. (2020), apresenta duas principais abordagens. A primeira consiste em construir modelos globais a partir de modelos individuais derivados de dados locais. A segunda se baseia na ideia de agregar globalmente as saídas de modelos locais.

Uma das abordagens possíveis se resume basicamente em compartilhar os modelos de classificação gerados por cada nó da rede de forma individual. No entanto, os algoritmos de aprendizagem levam em conta os atributos locais para formar uma opinião, isso seria uma desvantagem deste método. Além disso, se o tipo de técnica de aprendizagem aplicada em um nó da rede for diferente da empregada em outro, os algoritmos a serem combinados podem ter diferentes representações e, ao combinar os classificadores gerados podem gerar divergências. Desta maneira, seria preciso definir uma representação uniforme na qual classificadores diferentes seriam traduzidos. Porém, não é uma tarefa fácil definir uma representação para encapsular todas as outras representações sem perder uma quantidade relevante de informações. Além disso, uma consequência provável e indesejável dessa tradução seria principalmente a restrição das informações suportadas pelo classificador (PETEIRO-BARRAL; GUIJARRO-BERDIÑAS, 2013).

A combinação de classificadores pode seguir de outra maneira também, pois em vez de combinar seus modelos, seria combinado suas saídas (ou seja suas opiniões). Dessa forma, não haveria necessidade de traduzir um modelo de classificação, pois sua organização interna não é levada em consideração, já que a informação compartilhada é baseada nas previsões que são as saídas dos classificadores para um determinado conjunto de treinamento. Nesta abordagem, a padronização diz respeito às previsões visto que estas podem ser categóricas ou numéricas. Sendo assim, a complexidade de definir uma estrutura uniforme para combinar as saídas é muito menor do que a de combinar classificadores (PETEIRO-BARRAL; GUIJARRO-BERDIÑAS, 2013). Isso envolve a aplicação do mesmo algoritmo em diferentes subconjuntos de dados em paralelo e desta maneira combinar os resultados parciais (CHAN; STOLFO et al., 1993). No entanto, uma desvantagem desta estratégia é que toda nova entrada a ser classificada deve ser enviada a cada nó da rede, para que este nó envie sua opinião, sendo inviável em casos em que as informações não podem trafegar pela rede.

3.2 Topologias de aprendizado distribuído

Verbraeken et al. (2020) apresentam algumas topologias nas quais os nós dentro da rede podem ser organizados. Segundo os autores, o processo pode de aprendizado distribuído pode ser realizado de quatro formas: Centralizado, baseado em Árvore, Servidor de Parâmetros (em nuvem) e Totalmente distribuído (*peer-to-Peer*). A seguir são apresentados detalhes de cada estratégia.

3.2.1 Topologia centralizada

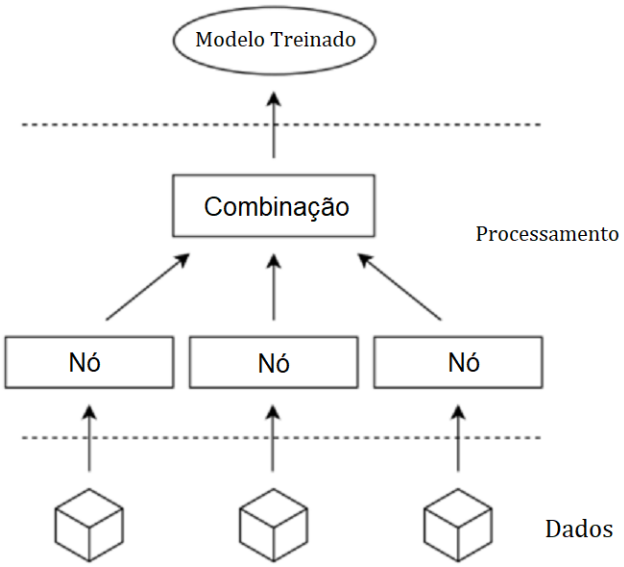
De acordo com Verbraeken et al. (2020), os sistemas de classificação centralizados empregam uma abordagem estritamente hierárquica para agregação, que ocorre em um único local central. Desta maneira, em cada nó é executado um algoritmo de aprendizado sobre os dados locais e, em seguida, o modelo obtido é encaminhado a um nó central para a realização da combinação destes algoritmos, resultando no modelo de classificação final, conforme a Figura 8.

Após a obtenção do modelo central este deve, de alguma forma, ser compartilhado aos demais nós da rede para que eles possam realizar a classificação localmente, sem a necessidade de enviar os dados de novos registros através da rede até o nó central, aumentando o tráfego na rede.

3.2.2 Topologia em árvore

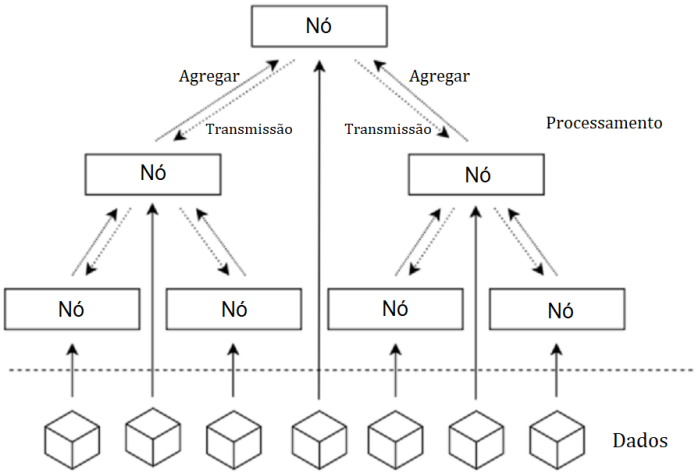
De acordo com Verbraeken et al. (2020) a topologia em árvore tem a vantagem de ser fácil de escalar e gerenciar, pois cada nó só precisa se comunicar com seu nó pai e nós

Figura 8 – Abordagem de aprendizado centralizado (VERBRAEKEN et al., 2020)



filhos. A desvantagem desta topologia é que se um nó pai cair se perde todo o treinamento realizados pelos nós filhos, conforme a Figura 9.

Figura 9 – Topologia em árvore (VERBRAEKEN et al., 2020)



Como pode ser visto na ilustração, cada nó realiza o aprendizado localmente, com base nas informações que possui e o modelo obtido é então enviado ao nó pai. Este é então responsável por repassar os modelos enviados pelos seus nós filhos e os encaminha ao nível hierárquico superior. A raiz da estrutura é então responsável por combinar os modelos e repassá-lo aos elementos presentes na rede.

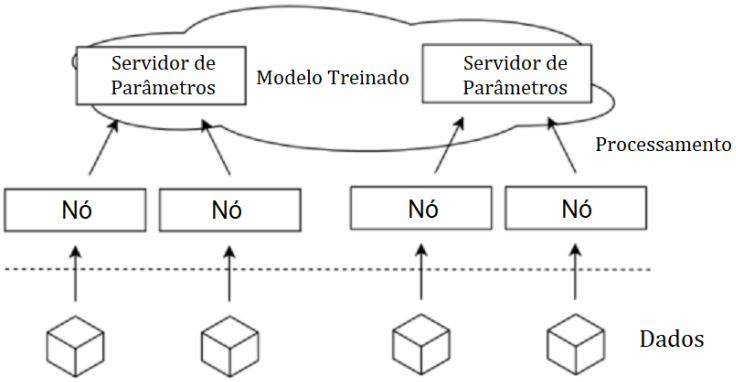
3.2.3 Topologia servidor de parâmetros (em nuvem)

A topologia servidor de parâmetros usa um conjunto de nós descentralizado de aprendizado de máquina, onde as informações estão armazenadas (nós folhas) juntamente

com um conjunto nós centralizado mestres (em nuvem) que mantém todo o compartilhamento do sistema. Desta maneira, todos os parâmetros do modelo ficam armazenados de forma dividida entra os servidores da nuvem (VERBRAEKEN et al., 2020).

Uma vantagem é que todos os parâmetros do modelo estão em uma memória compartilhada global, o que facilita a inspeção do modelo. No entanto, como a este conjunto de nós mestres acabam se comunicando com toda a rede, é possível que haja um gargalo de comunicação, dependendo do tamanho do conjunto de nós folhas. Pois quanto maior for o número de nós folhas, a tendência é que o desempenho se torne parecido com a abordagem centralizada. A Figura 10 ilustra a forma de comunicação desta topologia.

Figura 10 – Topologia do Servidor de Parâmetros(VERBRAEKEN et al., 2020)



3.2.4 Topologia totalmente distribuída

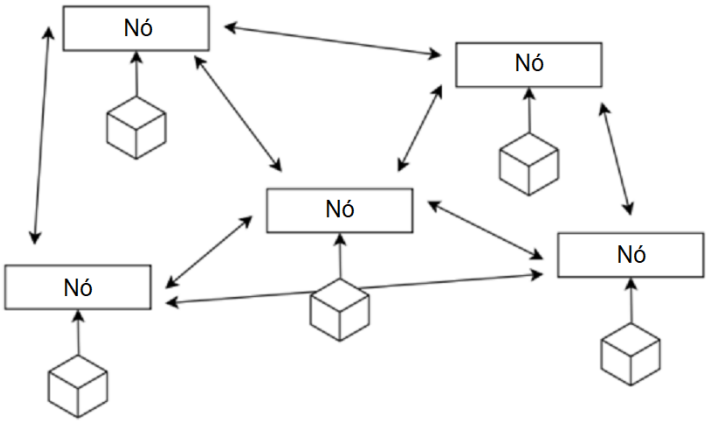
A topologia *peer-to-peer* apresentada por Verbraeken et al. (2020) demostra uma abordagem contrária a um modelo centralizado, pois cada nó tem sua própria cópia dos parâmetros e os nós trabalhadores se comunicam diretamente uns com os outros. Isso oferece a vantagem de escalabilidade tipicamente maior do que um modelo centralizado e a eliminação de pontos únicos de falha no sistema, conforme a Figura 11.

Neste trabalho será adotada a topologia totalmente distribuída (*peer-to-peer*) para a construção do modelo de aprendizado, dados os fatores positivos que oferece. Entretanto, dada sua característica distribuída, se faz necessária alguma estratégia de tolerância à falhas para manter os modelos de classificação atuais e coesos. Para tanto será adotada a estratégia VCube, apresentando mais adiante.

3.3 Trabalhos Correlatos

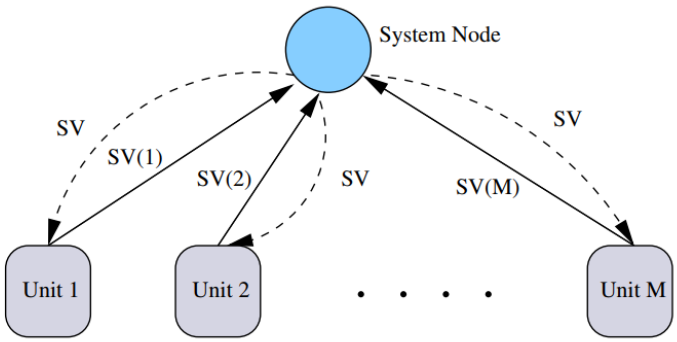
Alpcan e Bauckhage (2009) apresentam um *framework* de aprendizagem distribuída para o classificador SVM (seção 2.3.3), onde o problema de classificação é decomposto em múltiplos subproblemas. Em seguida, cada um deles são processados por unidades

Figura 11 – Topologia totalmente distribuído - *peer-to-peer* (VERBRAEKEN et al., 2020)



individuais, como computadores separados ou núcleos de processador, em paralelo e com acesso a apenas um subconjunto dos dados. Assim uma atualização paralela síncrona converge para a solução aproximada geometricamente. A Figura 12 mostra o fluxo de trabalho do algoritmo de aprendizagem distribuída utilizando o classificador SVM.

Figura 12 – O fluxo de informações dentro da estrutura distribuída (ALPCAN; BAUCKHAGE, 2009).



Uma das ferramentas mais utilizadas para armazenar e processar dados distribuídos é o Apache Hadoop (APACHE, 2022). Uma das estruturas do Hadoop é o MapReduce que ajuda a escrever programas que processam grandes conjuntos de dados usando algoritmos distribuídos e paralelos dentro do ambiente Hadoop. O MapReduce é o componente principal do processamento em um ecossistema Hadoop, pois fornece a lógica do processamento. Em um programa MapReduce, *Map* e *Reduce* são duas funções distintas. A função *Map* realiza ações como filtragem, agrupamento e classificação. Enquanto a função *Reduce* agrega e resume o resultado produzido pela função *Map* (MERLA; LIANG, 2017).

Mizukoshi et al. (2013) apresentam um Sistema de Detecção de Intrusão (SDI) utilizando técnica de aprendizado de máquina executadas por um sistema de múltiplos classificadores simultaneamente (paralelos) no *framework* Hadoop MapReduce. Neste

sistema apresentado, os classificadores são treinados individualmente na função Map enquanto o processamento e a união das respostas são realizados pela função Reduce.

Hu, Wang e Wu (2020) apresentam um modelo de sincronização de parâmetros para aprendizado de máquina distribuído, onde na borda estão sistemas heterogêneos. Nesta sincronização dos parâmetro o tempo de espera é minimizado significativamente. A ideia deste modelo é que os dispositivos de borda mais rápidos continuem o processo de treinamento, enquanto envia as atualizações dos seus modelos em intervalos estratégicos, este intervalo é definido por um algoritmo que garante uma convergência mais rápida e veloz. Segundo os autores este modelo supera significativamente os modelos de sincronização de parâmetros que existem.

Wang, Niu e Li (2019) propõe um *framework* de aprendizado distribuído assíncrono baseado em uma arquitetura sem servidor chamado *SIREN*, onde as funções são executadas na nuvem. Entre as vantagens da ferramenta são o paralelismo e elasticidade. Além disso, autores propõe ainda um escalonador que aprende com o próprio processo de treinamento que busca o menor tempo possível para este treinamento. A partir de testes experimentais os autores chegaram à conclusão que utilizando o *SIREN* foi possível reduzir o tempo de treinamento em até 44%.

Sparks et al. (2013) desenvolveram uma API (*Application Programming Interface*) cujo objetivo é facilitar e simplificar o desenvolvimento de algoritmo distribuídos para o aprendizado de máquina com um desempenho alto e escalável. Essa API auxilia o desenvolvedor a carregar os dados e extrair características. Os resultados mostram que com a ferramenta é possível construir algoritmos de aprendizagens distribuídos de maneira simples que mantém uma alto desempenho e escalabilidade.

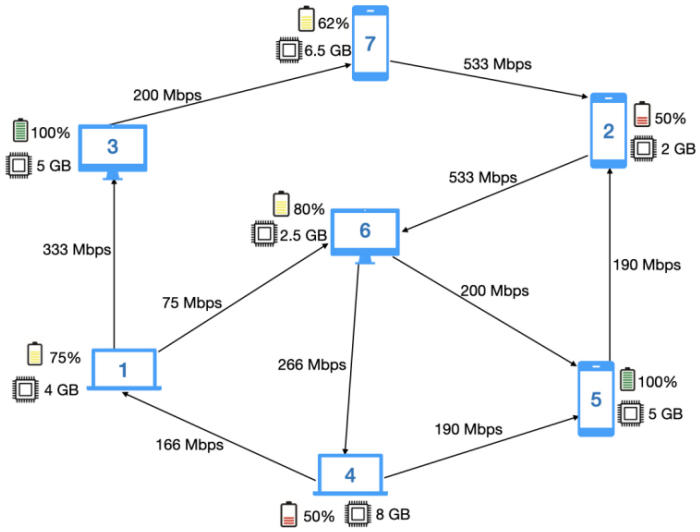
Gupta et al. (2022) abordam a aprendizagem federada utilizando a topologia *peer-to-peer* mostrando que é possível utilizar grafos para construir um algoritmo de comunicação eficiente em uma rede totalmente conectada. A Figura 12 mostra uma rede com 7 dispositivos iniciados aleatoriamente para indicar o custo de comunicação, onde o custo da comunicação é calculado como a soma dos pesos das arestas percorridas de um nó até o outro.

3.4 Algoritmo distribuído VCube

Nesta seção será apresentado o algoritmo utilizado para garantir a comunicação de todos nós da rede com tolerância a falhas.

Segundo Duarte, Bona e Ruoso (2014a), o algoritmo VCube pressupõe que exista uma rede totalmente conectada para o seu funcionamento, ou seja, onde um nó consegue se comunicar com qualquer outro desta rede, direta ou indiretamente. Cada nó pode assumir

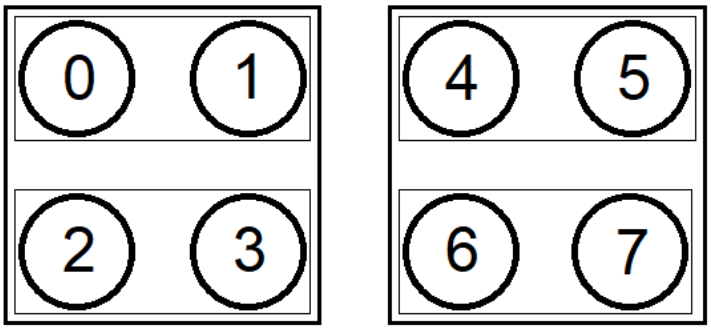
Figura 13 – Modelo do aprendizado federado com 7 dispositivos, cada com seus próprios dados locais (GUPTA et al., 2022).



um estado dos dois estados possíveis: falho e sem-falha.

O algoritmo trabalha com a ideia de agrupamento dos nós em *clusters*, onde o tamanho destes são sempre uma potência de dois. A Figura 14 mostra uma rede com 8 nós organizados em *clusters*. Na ilustração é apresentada uma estrutura hierárquica, em que grupos menores podem ser unidos para compor clusters mais abrangentes. Os nós 0 e 1 compõem um primeiro cluster que, junto com o grupo formado pelos nós 2 e 3, formaram um agrupamento maior. O mesmo fato é observado para os nós 4, 5, 6 e 7.

Figura 14 – *Cluster* com 8 nós (DUARTE; BONA; RUOSO, 2014a).



3.4.1 Funcionamento do VCube

Nesta seção é apresentado como o funcionamento do algoritmo ocorre para um conjunto composto por N nós. Os nós realizam teste em outros nós em intervalos de testes, vale lembrar que nó sem falha sempre realizam testes coretos.

1. Na primeira rodada de testes o *nó i* realiza os testes nos *clusters* que contém um nó.

2. Na segunda rodada de testes em *clusters* que contém dois nós.

3. Na terceira rodada de testes *cluster* que contém 4 nós e assim sucessivamente, até que os nós do *clusters* contenham $\frac{N}{2}$ elementos.

4. Uma função é responsável por retornar uma lista ordenada dos nós que podem ser testado em um determinada rodada, esse operação usa como base operador lógico XOR (OU exclusivo) que garante que todos os nós sem-falha irão realizar o teste pelo menos um vez.

5. Quando o nó for testado como sem-falha, o nó testador obtém informações de todos os outros nós do sistema por meio dele.

Duarte, Bona e Ruoso (2014a) afirmam ainda que o VCube é baseado na regra em que antes de um *nó i* executar um teste em um *nó j*, o *nó i* deve ser o primeiro nó sem falha que pode testar o *nó j* na rodada. Caso o *nó j* seja testado como sem falha, o *nó i* obtém novas informações de diagnóstico do *nó j* sobre todo os nós da rede. Desta maneira o processo é repetido para todos os nós do *cluster* que participa da rodada de testes.

Já que o *nó i* testador pode receber informações de outros nós por meio do *nó j* que está sendo testado. O algoritmo utiliza o campo chamado *timestamps* para guardar a informação do estado do nó na rede. Inicialmente os nós são considerados sem falha e recebem o valor 0 (zero). Assim, quando um evento é detectado, seja ele quando um nó sem-falha se torna falho ou vice-versa, o valor do *timestamps* é incrementado. Desta maneira se um *timestamps* com o valor par corresponde a um nó sem-falha, enquanto um valor ímpar corresponde a um nó falho.

Com a utilização do *timestamps* o algoritmo é capaz de diagnosticar falhas e recuperações de forma dinâmica. Já que o nó testador possui um *array* de *timestamps* correspondente ao estado de cada nó da rede e ao receber informações de diagnóstico de outro nó, só atualizará o *timestamps* local se o valor do *timestamps* recebido for maior que o valor armazenado localmente (Figura 15).

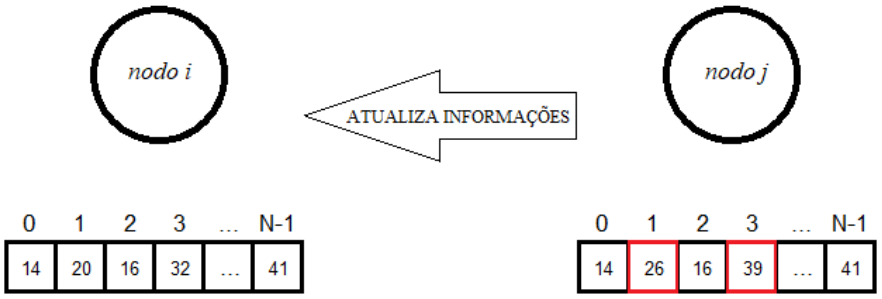
3.4.2 Detecção de Falha do VCube

Durante o processo de testagem quando o testador testa um nó falho, ele continua testando os nós daquele cluster até que um nó sem falha seja encontrado ou então até testar todos os nós do cluster como falhos.

Por exemplo (Figura 16) em uma rede com 8 nós este processo funciona da seguinte forma:

1. Inicialmente na primeira rodada o nó 0 testa o nó 1.

Figura 15 – Cada nó possui um *array* que contém informações de todos os nós do sistema, assim o *nó i* obtém informações de outros nós da rede por meio do *nó j* que foi testado como sem falha, atualizando apenas as posições do *array* onde o valor do *timestamps* é maior do que o valor local, conforme o destaque em vermelho (Elaborada pelo autor).

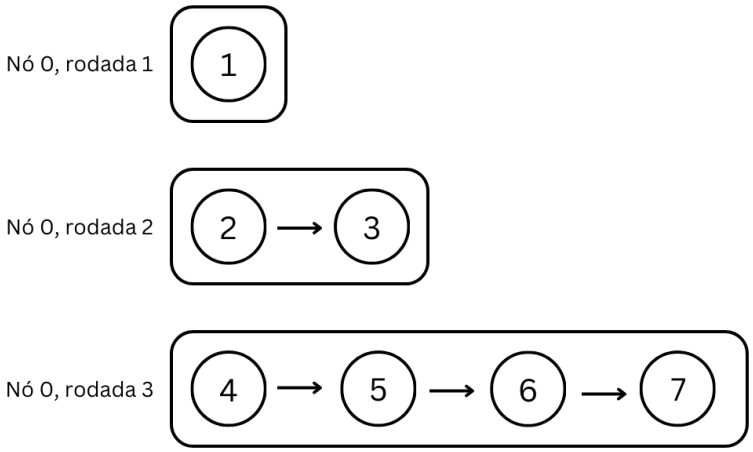


2. Na segunda rodada o nó 0 testará o *cluster* formados pelos nós 2 e 3.

3. Na terceira (última) rodada o nó 0 testará o *cluster* formados pelos nós 4, 5, 6 e 7.

Neste caso na terceira rodada se o nó 0 testar o nó 4 como falho, ele não obtém informações dele e pula para o próximo nó, repetindo o processo até que acabe todos os nós do *cluster*.

Figura 16 – *Cluster* testados pelo nó 0 (DUARTE; BONA; RUOSO, 2014a)



3.4.3 Topologia de Dados Utilizada

Com as diversas maneiras de aplicar o processo de aprendizagem de máquina de forma distribuídas apresentadas neste capítulo, será utilizada o método *peer-to-peer*, onde é possível gerar um modelo descentralizado em que cada nó possui informações de outros nós da rede sem a necessidade de um ponto central.

Já o VCube, será utilizando para garantir que o nosso sistema de aprendizado distribuído seja tolerante a falhas, ou seja que informe se um nó (ou mais) da rede conseguiu enviar seu modelo de classificação para os outros. Consequentemente o sistema de classificação pode avaliar se executa o algoritmo de classificação sem o modelo que não foi enviado ou aguarda o nó se recuperar para enviar o modelo e desta maneira iniciar o processo de classificação.

4 Materiais e métodos

Foram apresentados alguns algoritmos de classificação e estratégias de combinação de classificadores no capítulo 2 que serão utilizados em conjunto com a topologia de aprendizado distribuído *peer-to-peer* visto na seção 3.2 para treinamento do conjunto de dados da rede. Toda essa comunicação e verificação de possíveis falhas dos nós da rede será monitorado pelo algoritmo VCube visto na seção 3.4.

Inicialmente, para treinar os dados no processo de aprendizagem será utilizado o algoritmo Perceptron de Múltiplas Camadas (seção 2.3.4). Já como estratégia de combinação dos modelos gerados serão avaliadas as abordagens propostas por Kittler et al. (1998), que são a regra da soma (seção 2.4.2), regra do voto majoritário (seção 2.4.5) e a regra do produto (seção 2.4.1).

4.1 Linguagem de programação utilizada

No aprendizado de máquina foram utilizados historicamente várias linguagens de programação diferentes. Dentre ela a linguagem de programação que mais se destacou foi Python, pois dentro da comunidade de aprendizado de máquina houve um crescimento intenso da sua popularidade, tanto que as bibliotecas mais recentes de aprendizado profundo em sua maioria são desenvolvidos na linguagem Python (RASCHKA; PATTERSON; NOLET, 2020).

No entanto a linguagem de programação Java também é muito utilizada na área de aprendizagem de máquina, segundo Abeel, Peer e Saeys (2009) a linguagem contém uma coleção de algoritmos de aprendizado de máquina e mineração de dados, cuja interface é simples facilitado o trabalho de desenvolvedores e cientistas de dados, além disso, as implementações dos algoritmos estão claramente escritas e devidamente documentadas

Desta forma, será utilizado a linguagem de programação Java, pois outra vantagem de utilizar essa linguagem é o fato de que o algoritmo de comunicação e tolerância a falhas VCube foi desenvolvido em Java, facilitando assim a comunicação entre os processos de aprendizagem e de troca de modelos entre os nós. Além disso, a linguagem conta suporte de bibliotecas de aprendizagem de máquina capaz de armazenar de forma fácil e ágil os modelos de treinamento gerado pelos algoritmos de classificação.

4.1.1 Weka

Para implementar os algoritmos de classificação será utilizado a plataforma WEKA que possui uma biblioteca em Java que fornece uma gama de algoritmos de aprendizado,

que pode facilmente ser utilizado e aplicado a qualquer conjunto de dados. Outra vantagem é que a biblioteca fornece ferramentas para pré-processar um conjunto de dados, analisar o classificador resultante e seu desempenho (EIBE; HALL; WITTEN, 2016).

Além disso, o WEKA consegue facilmente salvar e carregar rapidamente os modelos de treinamento gerado pelos algoritmos de classificação, facilitando o envio e recebimento dos modelos entre os nós da rede.

4.2 Algoritmo Perceptron de Múltiplas Camadas (MLP)

Como classificador base optamos pelo Perceptron de Múltiplas Camadas (MLP), visto que é um classificador robusto, que pode ser aplicado em problemas multiclasse e que é bastante utilizado em pesquisas e comercialmente. A estrutura do MLP utilizada é descrita a seguir.

- Uma camada de entrada, onde cada neurônio representa uma dimensão do conjunto de características de 12 dimensões. Portanto, há 12 neurônios nessa camada;
- A primeira camada oculta é composta por 16 neurônios. Ela processa os dados da camada de entrada, onde cada neurônio calcula uma combinação linear das entradas e passa o resultado através de uma função de ativação sigmoid.
- A segunda camada oculta é composta por 32 neurônios, essa camada continua o processamento dos dados. Novamente, cada neurônio calcula uma combinação linear das saídas da camada anterior e aplica a função de ativação.
- A terceira camada oculta e composta por 64 neurônios, essa camada continua a extração de características dos dados. Cada neurônio realiza operações semelhantes às camadas anteriores.
- A camada de saída é composta por 6 neurônios, onde cada neurônio na camada de saída representa uma das classes de saída. A saída de cada neurônio pode ser interpretada como a probabilidade de pertencer à classe correspondente.

4.2.1 Hiperparâmetros Utilizados

Para este trabalho foram utilizados os parâmetros padrões que a biblioteca Weka oferece, sendo elas:

- Taxa de aprendizagem para o algoritmo de retropropagação de 0.3 (mínimo 0 e máximo 1)
- Taxa de impulso para o algoritmo de retropropagação de 0.2 (mínimo 0 e máximo 1).

- Número de épocas para treinar de 500.
- Tamanho percentual do conjunto de validação a ser usado para encerrar o treinamento de 0.
- Valor para propagar o gerador de números aleatórios de 0.
- Valor de aumentos consecutivos de erro permitidos para testes de validação antes do término do treinamento de 20.

4.3 Repositório de Dados

Tendo em vista os desafios em se coletar informações suficientes para testar o método de aprendizado distribuído, será optado por utilizar dados de repositórios públicos e especializados em problemas de classificação. Segundo Brickley, Burgess e Noy (2019), existem vários repositórios de dados na internet que fornecem milhões de conjunto de dados, entre eles governamentais, científicos, consórcios e comerciais. Assim, o acesso a esses dados é fundamental para a facilitar a reprodutibilidade dos resultados de uma pesquisa.

Dentre os repositórios mais conhecidos podemos citar Kaggle Dataset[1], Amazon Datasets[2], Google's Datasets Search Engine[3] e UCI Machine Learning Repository[4].

4.3.1 Repositório Kaggle

Neste trabalho serão utilizados dados do repositório Kaggle, segundo Badr (2019) neste repositório cada conjunto de dados é uma pequena comunidade onde é possível discutir sobre os dados, ou então encontrar algum código público relacionado àquele problema. Neste repositório existe uma grande quantidade de conjuntos de dados da vida real de várias formas, tamanhos e em muitos formatos diferentes. Muitos cientistas de dados diferentes forneceram blocos de anotações para analisar o conjunto de dados, em alguns casos é possível encontrar blocos de anotações com algoritmos que resolvem o problema de previsão nesse conjunto de dados específico.

4.4 Dados utilizados

Para avaliar o desempenho do aprendizado distribuído foi utilizado o conjunto de dados "**MotionSense:** Atividade humana baseada em sensor e reconhecimento de

[1] https://www.kaggle.com/datasets
[2] https://registry.opendata.aws/
[3] https://datasetsearch.research.google.com/
[4] https://archive.ics.uci.edu/ml/index.php

atributos" disponível no repositório Kaggle (MALEKZADEH et al., 2019). Esse conjunto é composto por dados de séries temporais gerados por acelerômetros e sensores de giroscópio (altitude, gravidade, aceleração do usuário e taxa de rotação), que geraram ao todo 12 (doze) atributos conforme a Tabela 1 (APPLE, 2024).

Estas informações foram coletados com um *smatphone Iphone 6s* mantido no bolso frontal de cada participante usando o *SensingKit* que coleta informações do *framework Core Motion* em dispositivos com o sistema operacional iOS.

Tabela 1 – Detalhamento dos dados utilizado no treinamento do algoritmo de classificação

Tipo de sensor	Nome do Atributo	Descrição	Tipo
Altitude	attitude.roll	A rotação do dispositivo, em radianos	Double
	attitude.pitch	A inclinação do dispositivo, em radianos	Double
	attitude.yaw	A guinada do dispositivo, em radianos	Double
Gravidade	gravity.x	Aceleração do eixo X em G's (força gravitacional)	Double
	gravity.y	Aceleração do eixo Y em G's (força gravitacional)	Double
	gravity.z	Aceleração do eixo Z em G's (força gravitacional)	Double
Taxa de Rotação	rotationRate.x	O valor para o eixo X	Double
	rotationRate.y	O valor para o eixo Y	Double
	rotationRate.z	O valor para o eixo Z	Double
Aceleração do Usuário	userAcceleration.x	Aceleração do eixo X em G's (força gravitacional	Double
	userAcceleration.y	Aceleração do eixo Y em G's (força gravitacional)	Double
	userAcceleration.z	Aceleração do eixo Z em G's (força gravitacional)	Double

Foram coletados dados de 6 atividade realizadas por 24 participantes de diferentes gêneros, idades, pesos e alturas. Cada participante realizou 3 tentativas da atividade "andar para baixo", 3 tentativas da atividade "andar para cima", 3 tentativas da atividade "caminhar", 2 tentativas da atividade "correr", 2 tentativas da atividade "sentar" e 2 tentativas da atividade "ficar em pé" (MALEKZADEH et al., 2019).

Todas essas tentativas foram realizadas no mesmo ambiente e condições. Ao todo foram gerados 1.412.865 (um milhão e quatrocentos e doze mil e oitocentos e sessenta e cinco) registros sendo distribuídos da seguinte forma:

1. 131.856 pertencentes à classe "andar para baixo".
2. 157.285 exemplares da classe "andar para cima".
3. 344.288 amostras da classe "caminhar".
4. 134.231 exemplares da classe "correr".
5. 338.778 amostras da classe "sentar".
6. 306.427 pertencentes à classe "ficar em pé".

Desta maneira o conjunto de dados se mostra adequado ao trabalho proposto, pois contém um número considerável de registros, algo comum em cenários com dados distribuídos.

Uma forma de expressar a distribuição das instâncias dentro das classes presentes em cada conjunto de dados é o *Imbalance Ratio* (IR) ou Taxa de Desequilíbrio. Seu valor obtido pela divisão da quantidade de instâncias da classe mais frequente pela menos frequente. Valores próximos de 1 indicam que certa proporcionalidade é respeitada. Por outro lado, valores maiores indicam que há classes com maior presença no conjunto de dados do que outras (SILVA et al., 2020).

Analisando-se a distribuição das classes percebe-se um favorecimento às classes "caminhar" e "sentar" uma vez que estas são as mais frequentes. Por outro lado, as classes "andar para baixo" e "correr" são compostas por um número bastante inferior de amostras.

Para o conjunto de dados empregado no estudo, o valor de IR é 2,61, o que indica que a classe mais frequente ("caminhar") tem quase três vezes mais representantes do que a classe "andar para baixo", que é a menos frequente.

4.5 Topologia de Aprendizado Distribuído

Considerando as abordagens tratadas na seção 3.2, será implementada a topologia *peer-to-peer*. Uma das vantagens da utilização deste método é a ausência de gargalo, pois não existe um nó central que recebe todos modelos gerados pelos outros nós.

Além disso, se um nó da rede falhar não compromete toda a estrutura como ocorre em uma abordagem centralizada ou em árvore.

A estratégia implementada é aquela em que cada nó treina com seu próprio conjunto de dados e compartilha o seu modelo de classificação com os outros nós da rede. Essa abordagem evita a necessidade de transmissão de conjuntos robustos de dados através da rede, além de não sujeitar informações sigilosas a possíveis ataques durante o processo de transmissão.

O modelo base utilizado no treinamento é um Perceptron de Múltiplas Camadas, empregando em seu aprendizado, a configuração default oferecida pela biblioteca Weka conforme descrito na Seção 4.2.

Nesta implementação cada nó conhece o modelo de classificação de todos os outros nós rede e realiza o processo de combinação dos classificadores localmente, seguindo uma abordagem pré-estabelecida.

O responsável para que essa comunicação funcione e que garanta o monitoramento da rede para verificação se um nós está com falha ou não será o algoritmo VCube (Seção 3.4).

4.6 Configuração dos Experimentos

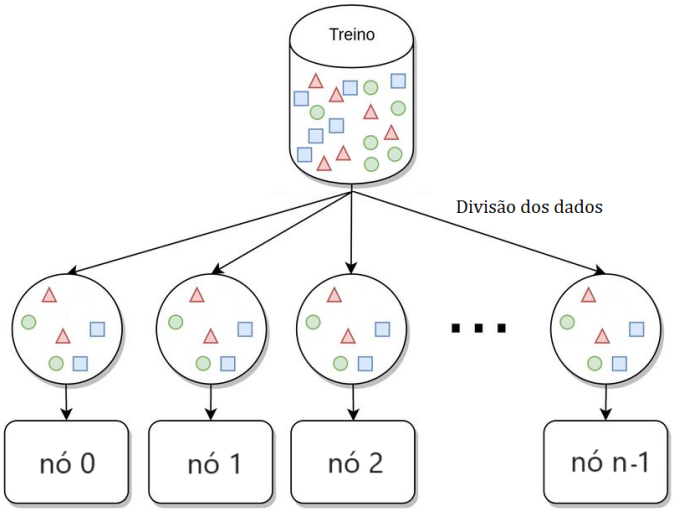
Inicialmente realizou-se um processo de classificação centralizado, nomeado "Classificação Global". Nesta etapa todos os dados estavam presentes em um único ponto e foram divididos em dois conjuntos. O primeiro, contendo 70% da base original (989.004 amostras) foi destinado ao treinamento de um modelo de aprendizado de máquina. Já o segundo conjunto, usado para avaliar o desempenho do modelo treinado, contém os 30% restantes dos dados (423.861 registros).

Em uma segunda etapa do experimento foram definidos oito nós, os quais receberam dados do conjunto de treino para construção de seus modelos individualmente. Ou seja nesta segunda etapa de testes o conjunto de treino foi "fatiado" e uma porção dos dados foi destinado a cada um dos nós de acordo com alguns cenários preestabelecidos.

A distribuição dos dados entre os oito nós foi realizada a partir de três cenários distintos, descritos a seguir.

1. No primeiro cenário todos os nós receberam uma quantidade de registros similar. Neste caso, o conjunto de dados possui aproximadamente 990 mil registros, logo, cada nó recebeu em torno de 123 mil registros mantendo a proporção original das classes (Figura 17).

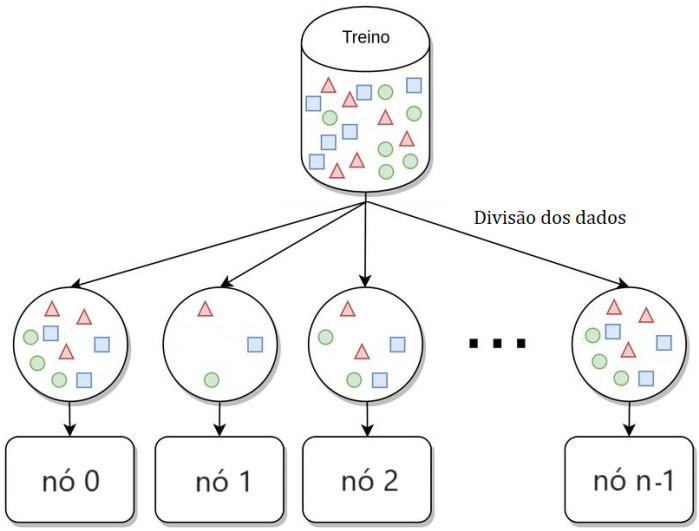
Figura 17 – Divisão com a mesma quantidade de registros e a estratificação das classes



2. No segundo cenário alguns nós têm mais registros que outros. Porém, a proporção entre as classes se mantém, ou seja, ao diminuir o tamanho de registros do conjunto são retirados registros de todas as classes de forma proporcional (Figura 18).

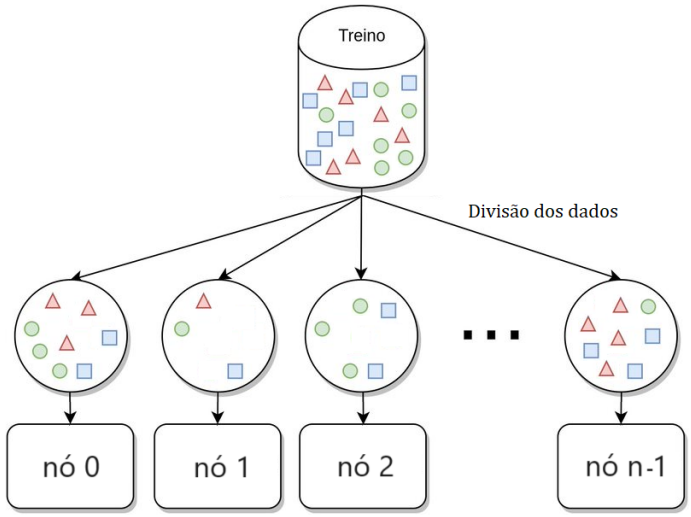
3. No terceiro cenário a distribuição do conjunto de registros do treino é feito de forma desbalanceada, sem respeitar a estratificação das classes. A ideia experimento é tentar simular casos em que um nó da rede contém mais informação sobre determinada

Figura 18 – Cenário 2 - Alguns nós com mais registro que outros



classe do que outro. A ideia desta divisão é representada na Figura 19. Na ilustração percebe-se, por exemplo, que o Nó 3 não recebe amostras da classe triângulo, logo ele não será capaz de classificar (sozinho) elementos desse grupo.

Figura 19 – Cenário 3 - Quantidade de dados e proporções desbalanceadas.



No caso do aprendizado distribuído, somente o modelo de classificação será compartilhado entre os nós, o que implica na necessidade de uma estratégia para combinação da opinião dos vários modelos construídos. Desta maneira, para fundir tais modelos foram utilizadas a Regra do Produto, a Regra da Soma e a Regra do Voto Majoritário (apresentadas no Capitulo 2), em cada um dos três cenários propostos.

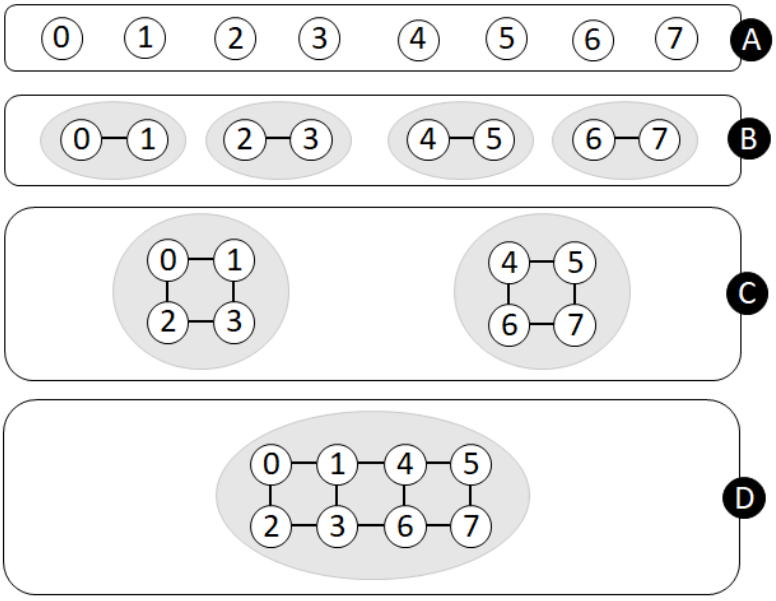
Por fim, para validar a acurácia tanto da classificação global quanto das abordagens distribuídas foi utilizado o mesmo conjunto de dados de teste que contém 30% dos registros da base original (aproximadamente 424 mil amostras).

Para os cenários distribuídos será adotada a topologia VCube como estratégia de compartilhamento dos modelos individuais de forma a permitir que cada nó tenha conhecimento dos modelos treinados em cada um dos outros vértices da rede. A partir dos oito modelos construídos espera-se obter um SMC capaz de consolidar a opinião de todos os classificadores treinados em cada nó.

A Figura 20 ilustra de que forma ocorre a comunicação entre os nós de acordo com a topologia adotada. Inicialmente (etapa A), cada modelo é treinado localmente apenas com os dados contido em seu nó, desconhecendo os demais classificadores. Em um segundo momento (etapa B), tem início o compartilhamento dos modelos entre os nós. Conforme definido pelo VCube cada nó envia seu modelo a apena outro nó. No exemplo, os nós 0 e 1 trocam modelos, assim como os pares 2 e 3, 4 e 5 e, por fim, 6 e 7.

Na terceira etapa (C), os pares formados no passo anterior são combinados de forma que agora os modelos são compartilhados entre todos os nós presentes em cada conjunto composto por quatro elementos (clusters $0-3$ e $4-7$). No passo final (etapa D), todos os elementos fazem parte do mesmo grupo, o que implica que todos os nós da rede conhecem todos os oito modelos treinados.

Figura 20 – Topologia do VCube aplicada na gestão de um SMC



Em posse de todos os modelos treinados em cada um dos oito nós, cada ponto da rede é capaz de combinar a opinião destes indutores e classificar qualquer nova instância.

Os resultados alcançados durante a etapa de execução dos experimentos para cada um dos quatro cenários são apresentados no próximo capítulo.

4.7 Ambiente de execução dos testes

Tanto os algoritmo de treinamento do MLP quanto o algoritmo da comunicação (VCube) foram executados em JAVA versão 8 (64 bit).

Todos os algoritmos deste trabalho foram rodados em uma máquina exclusiva para os testes, ou seja, não foi executado nenhum outro programa em paralelo.

As configurações da máquina de testes segue as seguintes especificações: Sistema Operacional Microsoft Windows 11 arquitetura de 64 bits, memória RAM de 12GB (*Gigabytes*), processador Intel Core i7 de 8ª (oitava) geração e armazenamento de 1TB (*Terabyte*) utilizando a tecnologia HDD (*Hard Disk Drive*).

5 Resultados

Neste capítulo são apresentados os resultados dos experimentos realizados, envolvendo uma classificação de dados global, onde todos os dados são utilizados para o treino do modelo, além de três cenários alternativos, de forma que os dados são distribuídos de maneiras distintas entre os nós da rede.

5.1 Classificação global

Inicialmente foi realizada a classificação global, utilizando 70% dos dados para treinamento e 30% dados para realizar os testes. Neste experimento simulamos um cenário em que todas as instâncias da base estão presentes no mesmo nó da rede.

Esta primeira alternativa tem a vantagem de que ao treinar o modelo ele tem conhecimento de todos os exemplares disponíveis para o aprendizado. No entanto, esta abordagem tem como pontos fracos a necessidade de consolidação dos registros em um único ponto, podendo causar gargalo na rede de transmissão e sujeitar as informações (que podem ser sigilosas) a possíveis ataques de captura.

Nesta abordagem a classificação foi testada 3 (três) vezes tendo resultados semelhantes, o classificador gastou uma média de 20 horas para treinar o modelo, obtendo acurácia média de 82,15%, que pode ser considerado um bom resultado, pois como este problema possui 6 classes, isso significa que para acertar uma classe aleatoriamente a probabilidade é de aproximadamente 16% (conforme dados apresentados na Tabela 2.

Tabela 2 – Tempo de treinamento para a estratégia de Classificação Global

Modelo treinado	Tempo de treinamento (média)	Acurácia (média)
MLP	20 horas	82,15%

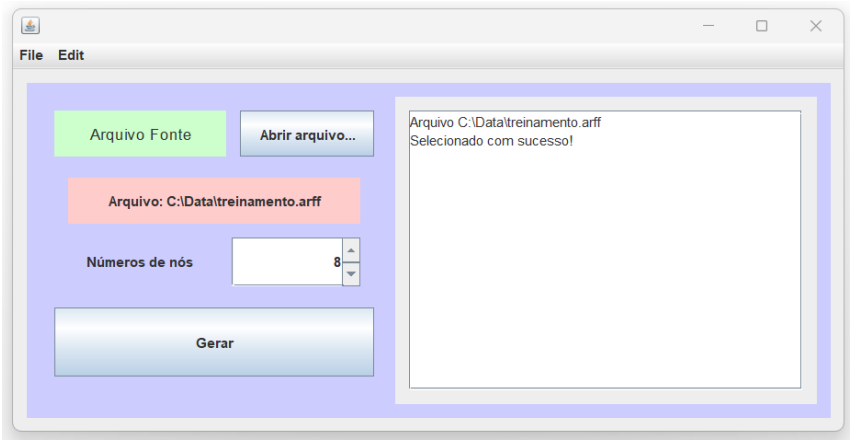
5.2 Configuração dos cenários

Para a elaboração dos três cenários alternativos foi desenvolvida uma ferramenta capaz de realizar essa tarefa de forma mais eficiente e ágil, permitindo a execução dos experimentos, coleta e representação dos resultados.

Na tela inicial da ferramenta (Figura 21), o usuário seleciona o arquivo que contém todos os dados de treinamento. O conjunto a ser carregado refere-se exclusivamente às instâncias de treino, uma vez que os dados de teste (correspondentes a 30% da base original) já foram separados.

Selecionado o arquivo com os dados de treino o passo seguinte é a definição do número de nós que o sistema terá, ou seja, em quantas partes o conjunto de treinamento será dividido.

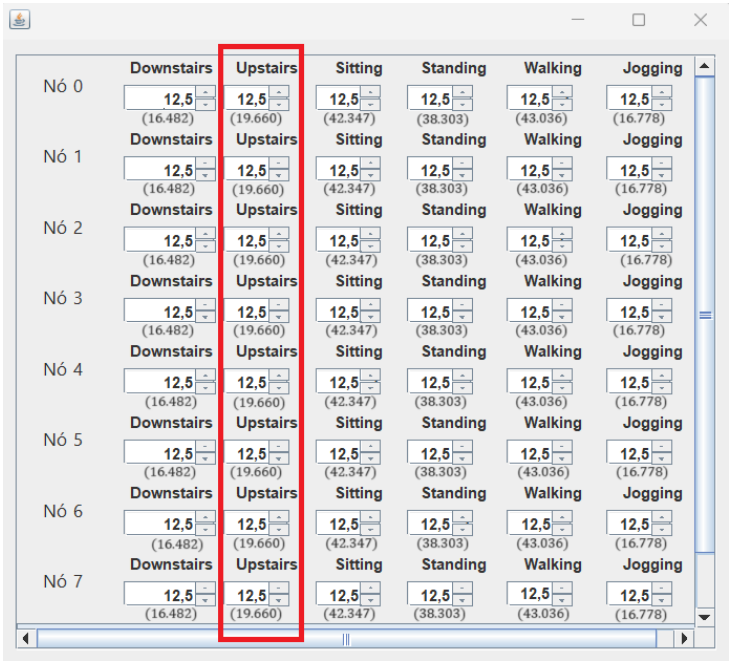
Figura 21 – Tela para seleção dos dados de treino e dimensionamento da rede



Logo após essa configuração a ferramenta vai separar todas as classes (“andar para baixo”, “andar para cima”, “caminhar”, “correr”, “sentar” e “ficar em pé”) presentes no arquivo de entrada para distribuir aos nós de acordo com a configuração especificada.

Com cada uma das classes isolada é possível definir o percentual que cada nó da rede terá de cada classe. Por exemplo, ao dividir os dados da classe “andar para cima” em 8 nós, cada um receberá 12,5%, ou seja, 19.660 (dezenove mil e seiscentos e sessenta) exemplares desta classe, conforme mostra a Figura 22.

Figura 22 – Divisão dos dados entre os nós. Em destaque é mostrada a configuração da classe em cada nó, onde o valor presente no botão *spinner* representa a porcentagem daquela classe e o valor abaixo o número de exemplares

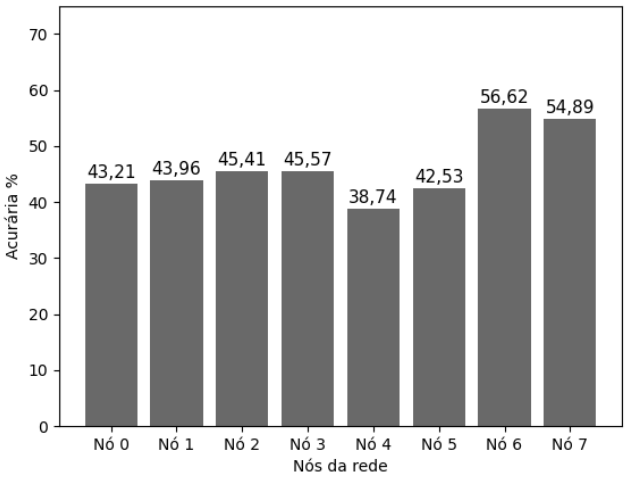


5.3 Cenário 1

Com o auxílio da ferramenta foi montado o primeiro cenário, onde cada nó recebeu a mesma quantidade de dados de cada classe. Para fins de nomenclatura, os modelos de classificação serão nomeados C_i em que i corresponde ao nó em que o classificador foi treinado. Dessa forma, o SMC construído é composto por 8 classificadores, nomeados de C_0 (treinado no nó 0) a C_7 (treinado no nó 7).

Primeiramente foi feito um treinamento local, onde cada nó da rede utilizou apenas os dados disponíveis localmente para o treinamento do modelo de predição. Na Figura 23 é possível visualizar o desempenho em relação a acurácia dos classificadores treinados em cada nó.

Figura 23 – Acurácia individual dos classificadores treinados em cada nó para o Cenário 1



Foi possível perceber que na maioria dos nós (5 no total) os classificadores tiveram uma acurácia muito parecida, algo em torno de 43%. Por outro lado, 3 modelos apresentaram resultados diferentes da média, o classificador C_4 por exemplo, apresentou uma acurácia um pouco abaixo dos demais com o valor de 38,74% de acerto, já os classificadores C_6 e C_7 observou-se uma acurácia acima da média, representando 56,62% e 54,89%, respectivamente.

5.3.1 Tempo de treinamento

A Tabela 3 mostra o tempo necessário para o treinamento dos classificadores sobre os conjuntos de dados individuais.

Em relação ao tempo de treinamento percebeu-se que os nós tiveram um tempo bem parecido, já que todos tinham praticamente a mesma quantidade de informações. Como esperado, frente ao modelo monolítico que gastou mais de 20 horas, o consumo de tempo diminuiu consideravelmente.

Tabela 3 – Tempo de treinamento de cada nó no cenário 1

Nó	Tempo
0	2h50m
1	2h36m
2	2h29m
3	2h29m
4	2h40m
5	2h29m
6	2h38m
7	2h55m

5.3.2 Execução da primeira rodada do VCube (sem falha)

Após o treinamento dos nós de forma individual, o modelo de classificação é compartilhado por cada nó por meio do VCube que é responsável por definir quais nós terão comunicação direta. Na primeira rodada cada nó troca mensagens com seu vizinho mais próximo, compartilhando seu modelo de classificação.

Desta forma, na primeira execução do VCube são formados 4 grupos (*clusters*), cada um contendo dois modelos de classificação, da seguinte maneira:

- Grupo 1 - Composto pelos classificadores C_0 e C_1.
- Grupo 2 - Composto pelos classificadores C_2 e C_3.
- Grupo 3 - Composto pelos classificadores C_4 e C_5.
- Grupo 4 - Composto pelos classificadores C_6 e C_7.

Assim foi possível perceber que já na primeira rodada de execução do VCube, houve um aumento da acurácia na abordagem que utiliza a regra da soma e na regra do voto majoritário. Já quando é utilizado a regra do produto o aumento da acurácia é pequena e em alguns casos existe uma queda no percentual de acertos. (Figura 24).

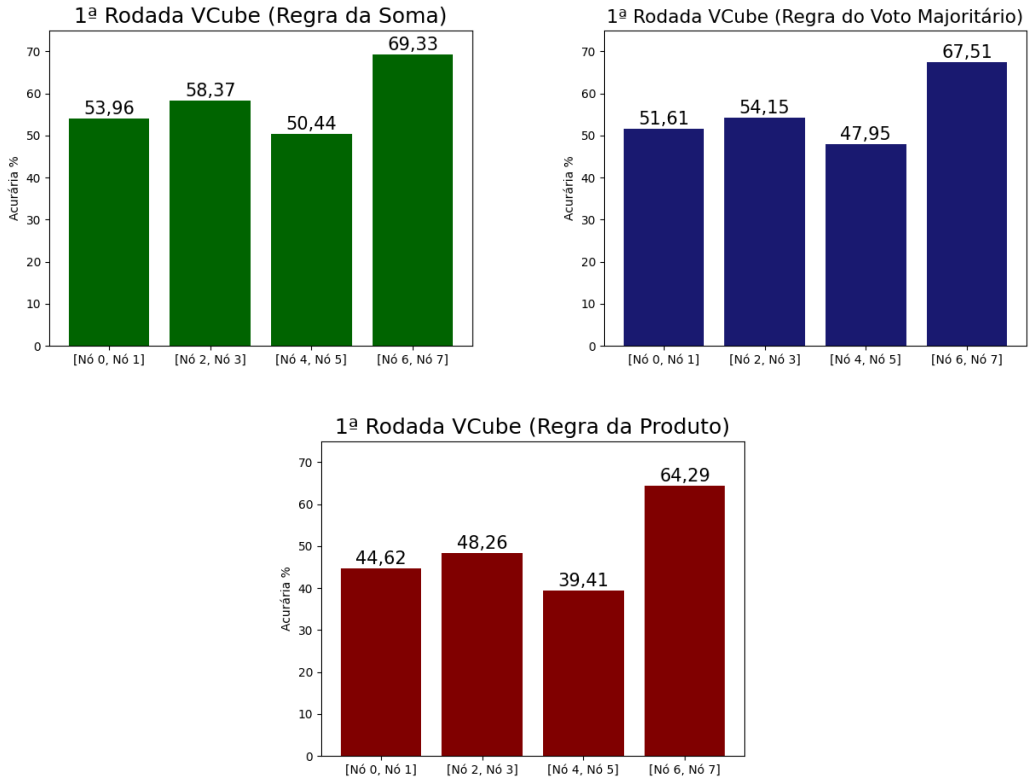
É possível perceber que o Grupo 4 formado pelos classificadores C_6 e C_7 obteve um desempenho maior em relação aos outros grupos, isso se deve ao fato de que esses classificadores obtiveram as maiores acurácias no treinamento individual.

5.3.3 Execução da segunda rodada do VCube (sem falha)

Na segunda rodada de execução do VCube foram formados 2 grupos (*clusters*) (cada um contendo a metade dos classificadores da rede) da seguinte forma:

- Grupo 1 - Composto pelos classificadores C_0, C_1, C_2 e C_3, .

Figura 24 – Acurácia na primeira rodada do VCube.



- Grupo 2 - Composto pelos classificadores C_4, C_5, C_6 e C_7.

Ao combinar os classificadores de cada grupo foi possível perceber que na regra da soma a acurácia do Grupo 1 foi de 60,02%, obtendo um valor da acurácia um pouco acima do grupo formado pelos classificadores C_2 e C_3 (na rodada anterior), obtendo pouco ganho. Já na regra do voto majoritário o ganho foi maior, pois o Grupo obteve uma acurácia de 58,46%, enquanto a acurácia dos classificadores C_2 e C_3 foi de 54,15% na rodada anterior, ou seja, obteve um ganho significativo. Na regra do produto a acurácia do Grupo 1 foi de 48,46%, houve um ganho mínimo de acurácia em relação a rodada anterior, além disso, foi possível notar que essa abordagem possui um desempenho muito inferior a regra do produto e do voto majoritário.

Já em relação ao Grupo 2 percebemos que nos três cenários de combinação foi o grupo que obteve a melhor acurácia, isso deve ao fato de que os classificadores com os melhores desempenho individuais (C_6 e C_7) estavam neste grupo. Na regra da soma a acurácia foi de 61,35%, obtendo um desempenho inferior comparado ao grupo dos classificadores C_6 e C_7 (na rodada anterior) que foi de 69,33%, ou seja, o desempenho dos classificadores C_4 e C_5 afetaram de forma negativa a acurácia do Grupo 2. Na regra do voto majoritário a acurácia do Grupo foi de 60,84%, tendo também uma queda da acurácia comparado com os grupos formados na rodada anterior. Por fim na regra do produto a acurácia do Grupo foi de 51,51%, sendo um desempenho bem abaixo dos resultados

obtidos na rodada anterior.

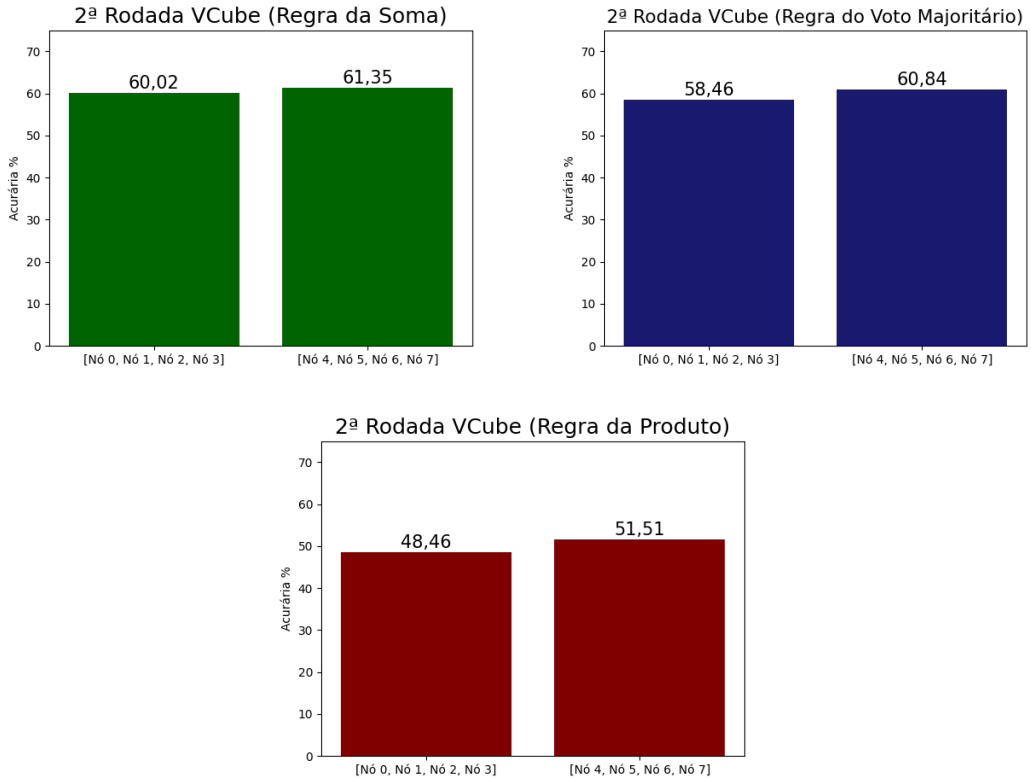


Figura 25 – Acurácia na segunda rodada do VCube.

5.3.4 Execução da terceira (última) rodada do VCube (sem falha)

Como nesta rede temos 8 nós, na terceira rodada de execução do VCube cada nó possui todos os modelos de classificação da rede, sendo possível fazer um modelo de combinação utilizando todos os classificadores. Desta maneira foi possível perceber que a abordagem que utilizou a regra da soma foi a que obteve a melhor acurácia com 66,23% de acertos, seguida pela regra do voto majoritário com a acurácia de 64,66% e com o pior desempenho a regra do produto com 51,65%, podemos observar que a Regra do Produto teve um desempenho pior que alguns classificadores individual (Figura 26).

5.3.5 Cenário 1 - Com falha

Neste cenário foi realizada uma simulação onde cada um dos nós do sistema não conseguiu mandar seu modelo de classificação para os outros nós da rede. Foi utilizado o Cenário 1 como base, ou seja, onde todos os nós tem uma quantidade de registros semelhantes.

Foi realizado um teste simulando a falha em cada um dos 8 nós do sistema, resultando na acurácia final conforme valores da Tabela 4:

Figura 26 – Acurácia da combinação de todos os nós, após a rodada 3 do VCube (sem falha)

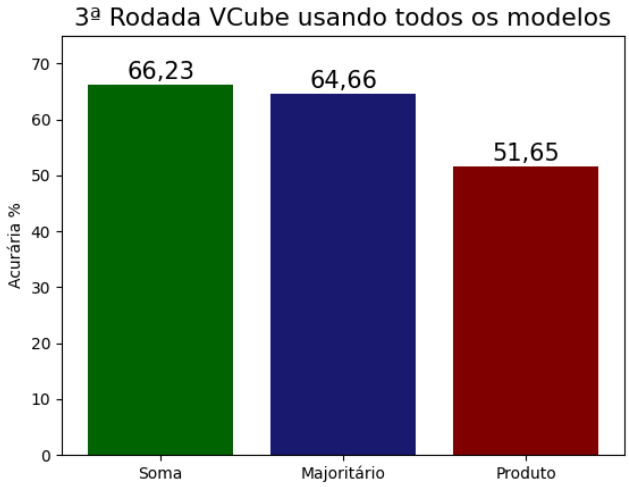


Tabela 4 – Cenário 1 - Falha em cada um dos nós da rede

	Regra de Combinação		
Nó em falha	Soma	Voto	Produto
0	66,43	65,39	51,88
1	64,32	63,39	52,69
2	64,31	63,34	51,79
3	64,30	62,86	50,70
4	68,03	67,32	55,10
5	65,89	63,99	51,85
6	63,22	61,77	48,62
7	64,04	62,40	48,95

A Tabela 5 mostra o desempenho do SMC quando o nó com o pior (menor acurácia) classificador entra em falha, neste caso foi o $nó_4$ com o classificador C_4 que obteve 38,74% de acurácia. Na primeira rodada do VCube o $nó_5$ não recebeu o modelo do seu vizinho mantendo a taxa de acertos classificador C_5. Na segunda rodada dois grupo são formados o primeiro constituído classificadores C_0, C_1, C_2 e C_3 e o segundo com um classificador a menos constituído pelos classificadores C_5, C_6 e C_7. Podemos perceber que este segundo grupo tem uma acurácia maior mesmo sem o classificador C_4 compor o grupo. Na terceira rodada quando os nós possuem o classificador dos outros nós da rede (exceção do classificador C_4), temos o melhor desempenho do SMC em todas as abordagem de combinação, principalmente utilizando a regra da soma com acurácia de 68,03%.

A Tabela 5 mostra o desempenho do SMC quando o oposto acontece, ou seja, quando o nó com o melhor (maior acurácia) classificador da rede entra em falha, neste caso foi o nó 6 com o classificador C_6 que obteve 56,62% de acurácia. Na primeira rodada do VCube o $nó_7$ não recebeu o modelo do seu vizinho mantendo a taxa de acertos do

Tabela 5 – Acurácia (%) do SMC baseado na VCube para o **primeiro cenário** com falha do **nó 4** (menor desempenho) ao longo das três etapas do processo

		Regra de Combinação		
Etapa	Grupos	Soma	Voto	Produto
Rodada 1	1. C_0, C_1	53,96	51,61	44,62
	2. C_2, C_3	58,37	54,15	48,26
	3. C_5	42,53	42,53	42,53
	4. C_6, C_7	69,33	67,51	64,29
Rodada 2	1. C_0, C_1, C_2, C_3	60,02	58,46	48,46
	2. C_5, C_6, C_7	62,81	61,08	57,49
Rodada 3	1. C_0, C_1, C_2, C_3, C_5, C_6, C_7	68,03	67,32	55,10

classificador C_7. Na segunda rodada podemos perceber novamente que um dos grupos (C_4, C_5 e C_7) possui um classificador a menos, prejudicando a sua acurácia. Por fim, na terceira rodada quando os nós possuem o classificador dos outros nós da rede (exceção do classificador C_6), temos o melhor pior desempenho do SMC em todas as abordagem de combinação, isto está ligado diretamente a falha do $nó_6$ em enviar o seu modelo para a rede.

Tabela 6 – Acurácia (%) do SMC baseado na VCube para o **primeiro cenário** com falha do **nó 6** (maior desempenho) ao longo das três etapas do processo

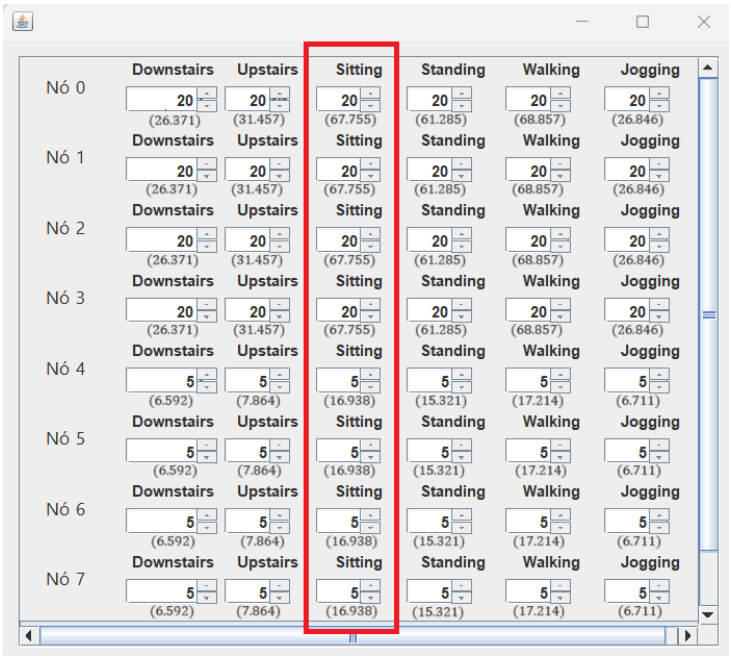
		Regra de Combinação		
Etapa	Grupos	Soma	Voto	Produto
Rodada 1	1. C_0, C_1	53,96	51,61	44,62
	2. C_2, C_3	58,37	54,15	48,26
	3. C_4, C_5	50,44	47,95	39,40
	4. C_7	54,89	54,89	54,89
Rodada 2	1. C_0, C_1, C_2, C_3	60,02	58,46	48,46
	2. C_4, C_5, C_7	55,42	52,98	44,33
Rodada 3	1. C_0, C_1, C_2, C_3, C_4, C_5, C_7	63,22	61,77	48,62

5.4 Cenário 2

Neste cenário a divisão foi realizada da seguinte forma os nós 0, 1, 2 e 3 receberam ao todo 80% do volume de dados, ou seja, cada um recebeu 20% dos registros, enquanto os nós 4, 5, 6 e 7 receberam 20% do volume de dados, ou seja, cada um recebeu 5% das dos registros, conforme mostra a Figura 27. Apesar de haver desbalanço no número de instâncias presentes em cada nó da rede, a proporção entre as classes é mantida conforme o conjunto original.

Neste cenário foi possível observar que quanto mais informações estavam concentradas em um nó da rede, melhor foi a sua acurácia. Podemos observar a taxa de acertos na Figura 28, onde o classificador C_0 obteve 51,80% de acertos, o classificador C_1 obteve

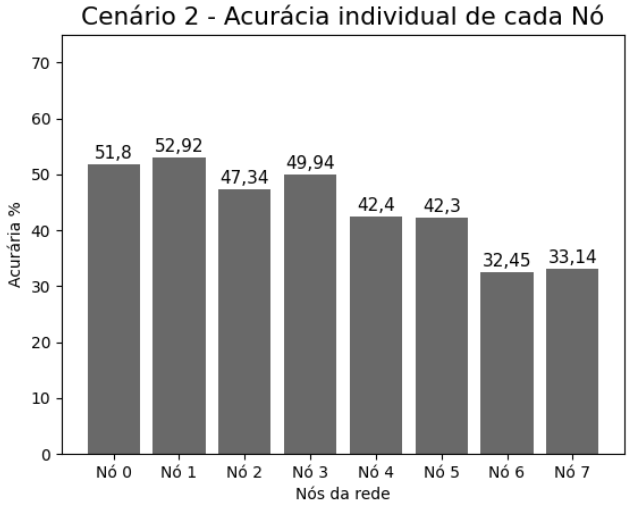
Figura 27 – Divisão do cenário 2, em destaque é mostrado a configuração da classe em cada nó, onde o valor presente no botão *spinner* representa a porcentagem daquela classe e o valor abaixo o número de exemplares.



52,92% de acertos, o classificador C_2 obteve 47,34% de acertos e o classificador C_3 obteve 49,94% de acertos.

Por outro lado, quanto menos informações nós temos sobre o problema a ser classificado menor é a taxa de acerto do classificador, como ocorreu nos classificadores C_4, C_5, C_6 e C_7 que continham apenas 5% do total das amostras de treino. Observou-se uma queda mais acentuada para nos classificadores C_6, C_7, os quais obtiveram acurácia de 32,45% e 33,14%, respectivamente.

Figura 28 – Acurácia de cada nó individual no cenário 2



5.4.1 Tempo de treinamento

A Tabela 7 mostra o tempo que cada nó levou para realizar o treinamento do seu conjunto de dados.

Tabela 7 – Tempo de treinamento de cada nó no cenário 2

Nó	Tempo
0	4h11m
1	4h16m
2	3h57m
3	4h09m
4	1h02m
5	0h53m
6	0h54m
7	1h01m

Neste novo cenário percebeu-se mudanças no tempo de treinamento dos modelos. Analisando os valores foi possível perceber que quanto mais informações um nó possui, mais tempo para o treinamento do algoritmo é necessário. Da mesma forma, quanto menor a quantidade de dados presente no nó, menor é tempo de treinamento.

5.4.2 Execução da primeira rodada do VCube (sem falha)

Da mesma forma como aconteceu no primeiro cenário, após o treinamento dos nós de forma individual, na primeira rodada de execução do VCube o modelo de classificação é compartilhado por cada nó com seu vizinho mais próximo.

Assim são formados 4 grupos com dois classificadores cada, onde foi possível combiná-los e calcular sua acurácia, sendo eles:

- Grupo 1 - Composto pelos classificadores C_0 e C_1.
- Grupo 2 - Composto pelos classificadores C_2 e C_3.
- Grupo 3 - Composto pelos classificadores C_4 e C_5.
- Grupo 4 - Composto pelos classificadores C_6 e C_7.

Pode-se perceber que os Grupos 1 e 2 tiveram um resultado superior aos Grupos 3 e 4 em todas as abordagens (Figura 29). Isso se deve ao fato de que neste cenário os classificadores C_0, C_1, C_2 e C_3 contavam com 80% dos registros enquanto os classificadores C_4, C_5, C_6 e C_7 contavam com apenas 20% das amostras de treino.

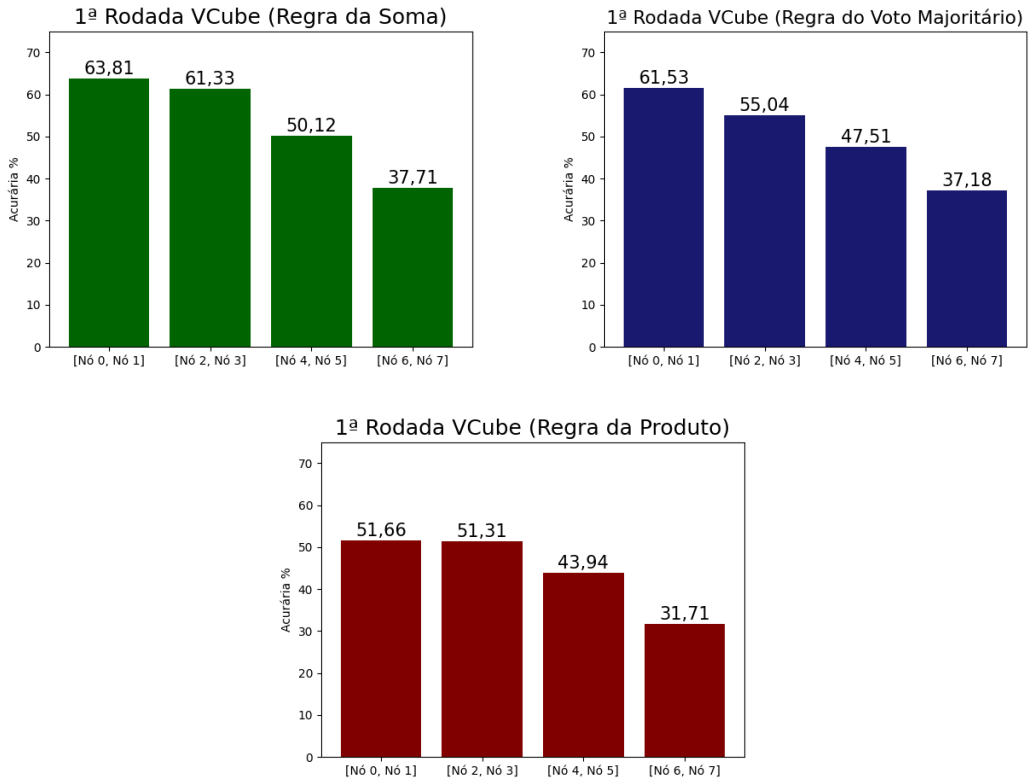


Figura 29 – Acurácia na primeira rodada do VCube.

5.4.3 Execução da segunda rodada do VCube (sem falha)

Assim como no primeiro cenário, novamente na segunda rodada de execução do VCube foram formados 2 grupos cada um contendo a metade dos classificadores da rede separados da seguinte forma:

- Grupo 1 - Composto pelos classificadores C_0, C_1, C_2 e C_3, .
- Grupo 2 - Composto pelos classificadores C_4, C_5, C_6 e C_7.

Após a combinação dos classificadores de cada grupo, as acurácias do Grupo 1 a partir da regra da soma, voto majoritário e regra do produto foram 66,79%, 65,01% e 56,22%, respectivamente. Em todas as abordagens houve um aumento considerável na acurácia quando combinado os 4 classificadores do grupo.

Já em relação ao Grupo 2, as acurácias partir da regra da soma, voto majoritário e regra do produto foram 45,19%, 42,93% e 37,48%, respectivamente. Na avaliação da acurácia deste grupo foi possível perceber que quando acontece a combinação de modelos de classificação com baixo desempenho, o modelo de classificação resultante tende a apresentar desempenho inferior.

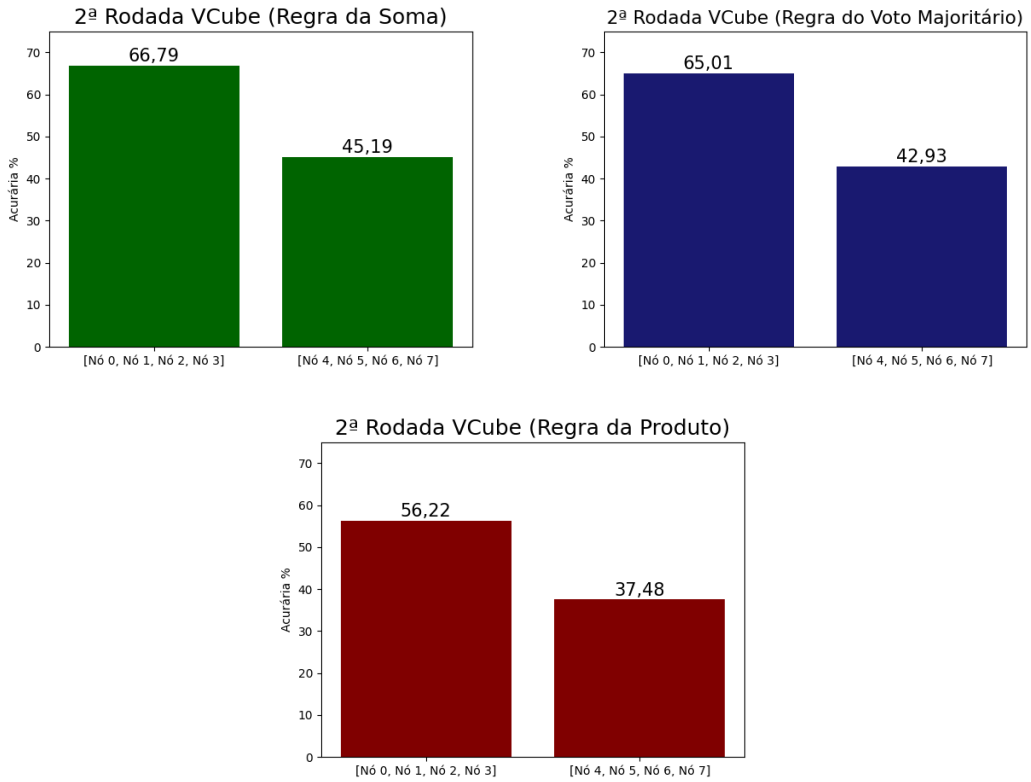


Figura 30 – Acurácia na segunda rodada do VCube.

5.4.4 Execução da terceira (última) rodada do VCube (sem falha)

Novamente, na terceira rodada de execução do VCube, cada nó possui todos os classificadores da rede. Desta maneira, conseguimos realizar a combinação de todos os nós. Neste cenário a abordagem que obteve o melhor desempenho foi a que utilizou regra da soma com uma acurácia de 61,21%. Apesar do desempenho melhor, o resultado foi muito semelhante à abordagem que utilizou a regra do voto majoritário cuja acurácia foi de 61,12%. Por fim, a regra do produto obteve um desempenho de 46,71% penalizado principalmente pelos nós com baixa acurácia (Figura 31).

5.4.5 Cenário 2 - Com falha

Novamente foi realizada uma simulação onde cada um dos nós do sistema não conseguiu mandar seu modelo de classificação para os outros nós da rede. Foi utilizado o Cenário 2 como base.

A Tabela 8 é possível observar a acurácia do modelo final quando cada um dos seus nós falham. Neste cenário ficou perceptível que quando um nó com muitos registros (nós 0, 1, 2, e 3) cai, o desempenho do SMC diminui e o contrário também acontece quando um nó com pouco registros cai o desempenho do SMC apresenta um aumento na acurácia.

A Tabela 9 mostra o desempenho do SMC quando o nó com o pior (menor acurácia)

Figura 31 – Acurácia da combinação de todos os nós, após a rodada 3 do VCube (sem falha)

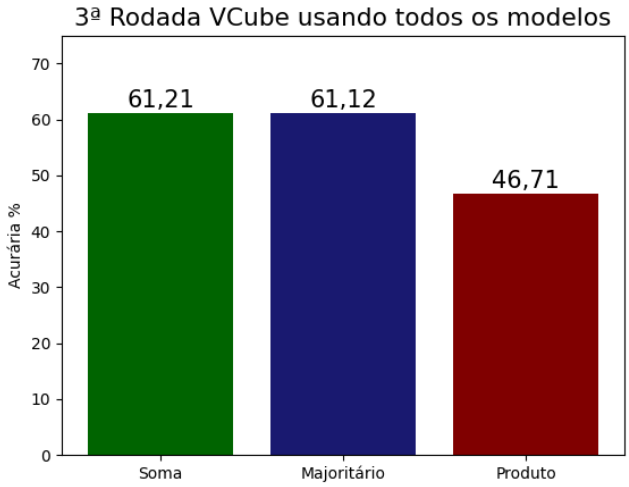


Tabela 8 – Cenário 2 - Falha em cada um dos nós da rede

	Regra de Combinação		
Nó em falha	Soma	Voto	Produto
0	59,01	58,62	45,99
1	57,45	56,77	45,30
2	58,28	58,31	44,89
3	60,14	58,99	45,32
4	61,72	61,03	46,50
5	61,82	60,36	46,36
6	61,22	61,34	49,91
7	62,18	61,20	48,61

classificador da rede entra em falha, neste caso foi o classificador C_6 que obteve 32,45% de acurácia. Podemos perceber que em cada rodada do VCube a o desempenho aumenta apesar da falha do nó, porém como este nó representa apenas 5% dos registro, o ganho na acurácia do SMC final é praticamente o mesmo que no cenário sem falha.

Tabela 9 – Acurácia (%) do SMC baseado na VCube para o **segundo cenário** com falha do **nó 6** (menor desempenho) ao longo das três etapas do processo

		Regra de Combinação		
Etapa	Grupos	Soma	Voto	Produto
Rodada 1	1. C_0, C_1	63,81	61,53	51,66
	2. C_2, C_3	61,33	55,04	51,31
	3. C_4, C_5	50,12	47,50	43,94
	4. C_7	33,14	33,14	33,14
Rodada 2	1. C_0, C_1, C_2, C_3	66,79	65,01	56,22
	2. C_4, C_5, C_7	44,70	44,38	43,56
Rodada 3	1. C_0, C_1, C_2, C_3, C_4, C_5, C_7	61,22	61,34	49,91

Por outro lado ao quando um nó que possui um classificador com uma boa acurácia falha, o desempenho do SMC final cai significativamente, conforme observado na Tabela 10.

Tabela 10 – Acurácia (%) do SMC baseado na VCube para o **segundo cenário** com falha do **nó 1** (maior desempenho) ao longo das três etapas do processo

		Regra de Combinação		
Etapa	Grupos	Soma	Voto	Produto
Rodada 1	1. C_0	51,80	51,80	51,80
	2. C_2, C_3	61,33	55,04	51,31
	3. C_4, C_5	50,12	47,50	43,94
	4. C_6, C_7	37,71	37,18	31,70
Rodada 2	1. C_0, C_2, C_3	63,77	60,68	56,18
	2. C_4, C_5, C_6, C_7	45,19	42,93	37,48
Rodada 3	1. C_0, C_2, C_3, C_4, C_5, C_6, C_7	57,45	56,77	45,30

Quando o Modelo de Classificação Combinado não possui informações de todos os nós, o desempenho diminuiu independentemente da abordagem de Combinação utilizada.

Diante disso, novamente é válido ressaltar a importância de um sistema de tolerância a falha como o VCube para garantir que o processo execute apenas se todos os nós da rede consigam transmitirem seus modelos, já que a falta de um classificador pode prejudica a acurácia do modelo final.

5.5 Cenário 3

Neste cenário a divisão dos dados foi realizada de forma desbalanceada, onde tanto a quantidade de registros quanto a proporção das classes presentes em cada nó possuem variação, conforme mostrado na Figura 32.

A partir dos valores pode-se perceber que a metade dos classificadores tiveram um desempenho abaixo dos 40%, sendo o classificador C_0 com 32,83%, o classificador C_1 com 39,16%, o classificador C_4 com 38,37% e o classificador C_5 com 33,24%. Já os classificadores C_2 e C_3 tiveram uma taxa de acertos de 45% e 47,72%, por fim os classificador C_6 e C_7 que foram os que mais receberam dados tiveram uma acurácia acima de 55%, sendo 58,77% e 55,64% respectivamente (Figura 33.).

5.5.1 Tempo de treinamento

A Tabela 11 mostra o tempo que cada nó levou para realizar o treinamento do seu conjunto de dados no Cenário 3.

Como esperado, os nós com quantidade similar de registros tiveram um tempo de treinamento semelhante, novamente indicando uma relação direta entre o número de

Figura 32 – Divisão do cenário 3

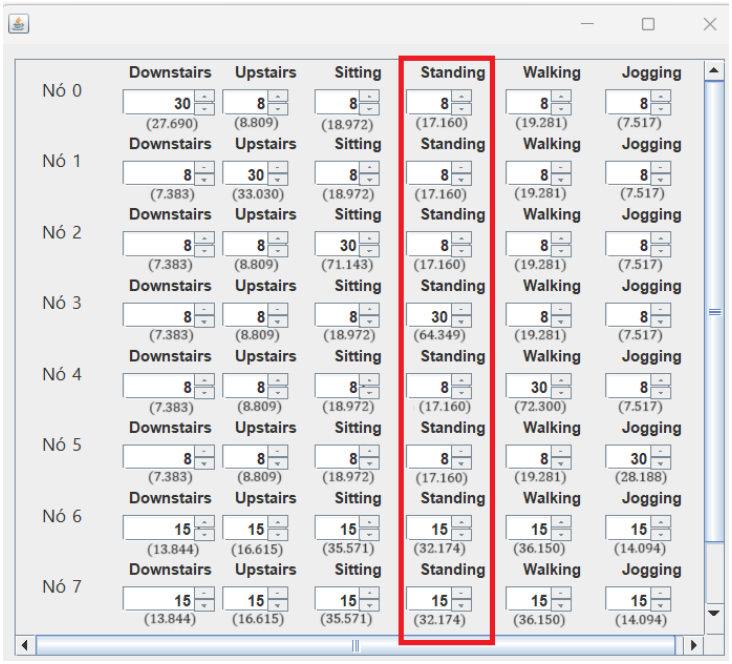
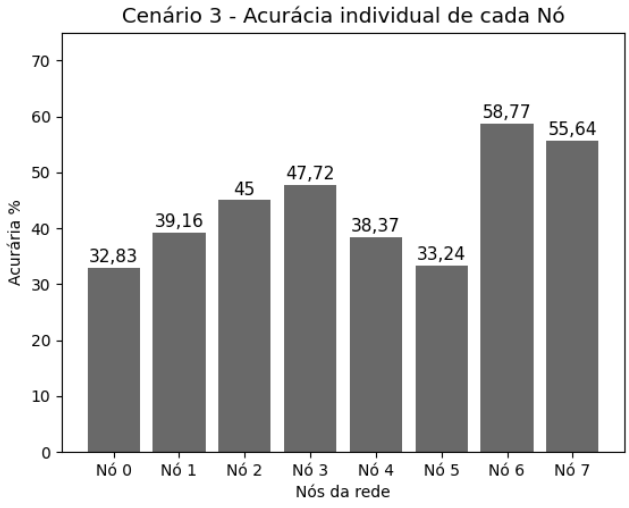


Figura 33 – Acurácia de cada nó individual no cenário 3



amostras e o tempo gasto no aprendizado. Podemos perceber que os nós com classificadores com uma acurácia baixa (C_0 e C_1) treinaram o modelo mais rapidamente enquanto nós com acurácia mais elevada (C_6 e C_7) demoraram mais.

5.5.2 Execução da primeira rodada do VCube (sem falha)

Da mesma forma como aconteceu nos cenários anteriores, após o treinamento dos nós de forma individual, o modelo de treinamento é compartilhado para toda a rede usando o VCube.

Assim, na primeira rodada de execução do VCube o modelo é compartilhado com

Tabela 11 – Tempo gasto para o cenário 3

Nó	Tempo
0	1h52m
1	1h56m
2	2h57m
3	2h39m
4	2h43m
5	2h04m
6	3h15m
7	3h01m

seu vizinho mais próximo, formando assim 4 grupos com dois classificadores cada, onde foi possível combiná-los e calcular sua acurácia, sendo eles:

- Grupo 1 - Composto pelos classificadores C_0 e C_1.
- Grupo 2 - Composto pelos classificadores C_2 e C_3.
- Grupo 3 - Composto pelos classificadores C_4 e C_5.
- Grupo 4 - Composto pelos classificadores C_6 e C_7.

Pode-se perceber que utilizando a regra da soma, o grupo 1 obteve uma taxa de acerto de 48,21%, refletindo o baixo desempenho dos classificadores que formaram este grupo. Do mesmo modo, o Grupo 3 alcançou 43,21% de acurácia. Já o Grupo 2 obteve uma taxa de acerto de 59,91%, sendo maior que a dos grupos anteriores. Por fim, o Grupo 4 obteve uma taxa de acerto de 67,61% refletindo o alto desempenho dos classificadores que formaram este grupo (principalmente em função de serem os classificadores que contém o maior volume de instâncias de treino).

Em relação às outras abordagens de combinação de classificadores neste cenário, pode-se perceber que o comportamento foi semelhante a regra da soma, ou seja, tanto no voto majoritário quanto na regra do produto os Grupos 1 e 3 apresentaram desempenho inferior aos demais. O segundo Grupo obteve desempenho intermediário enquanto o quarto grupo obteve o melhor desempenho (Figura 34).

5.5.3 Execução da segunda rodada do VCube (sem falha)

Na segunda rodada de execução do VCube foram formados 2 grupos com cada um contendo a metade dos classificadores da rede separados da seguinte forma:

- Grupo 1 - Composto pelos classificadores C_0, C_1, C_2 e C_3, .
- Grupo 2 - Composto pelos classificadores C_4, C_5, C_6 e C_7.

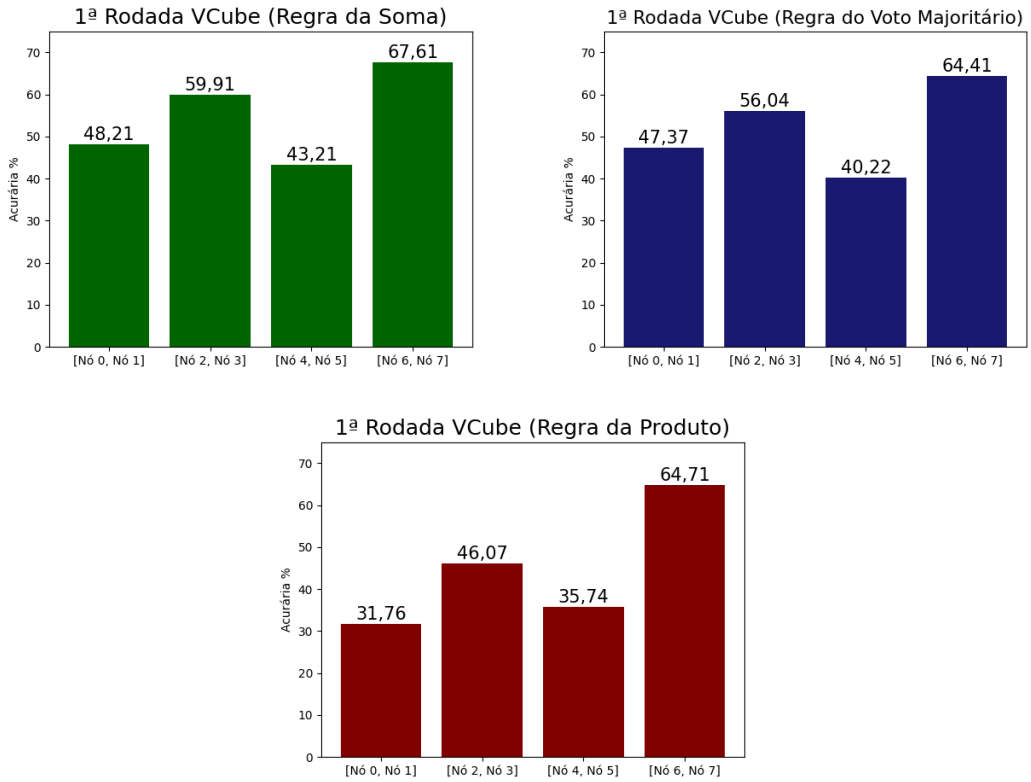


Figura 34 – Acurácia na primeira rodada do VCube.

Após o segundo compartilhamento dos modelos (segunda iteração do VCube), a acurácia do primeiro grupo para a regra da soma foi de 52,31%, para o voto majoritário foi de 51,76% e utilizando a regra do produto foi de 42,49%. Neste grupo foi possível perceber de forma mais clara que os nós que possuem um desempenho fraco tendem a puxar a taxa de acerto geral para baixo.

Já combinando os classificadores do grupo 2, a acurácia utilizando a regra da soma foi de 61,04%, da regra do voto majoritário foi de 58,11% e utilizando a regra do produto foi de 52,16%. Da mesma forma que no grupo anterior, foi possível perceber que apesar de haver classificadores com bom desempenho a acurácia geral do grupo tende a baixar devido a influência de alguns classificadores mais fracos.

5.5.4 Execução da terceira (última) rodada do VCube (sem falha)

Ao término da terceira iteração do VCube percebe-se que apesar da baixa taxa de acertos da maioria dos nós da rede, podemos perceber que ao combinar os modelos de classificação é possível verificar que a acurácia aumenta significativamente, isso ocorre pelo fato de que os nós podem oferecer informações complementares a respeito dos dados do sistema todo. Tal fato é constatado ao perceber-se que a taxa de acerto do Sistema de Múltiplos Classificadores foi de 66,04% para a Regra da Soma, 64,93% na Regra do Voto

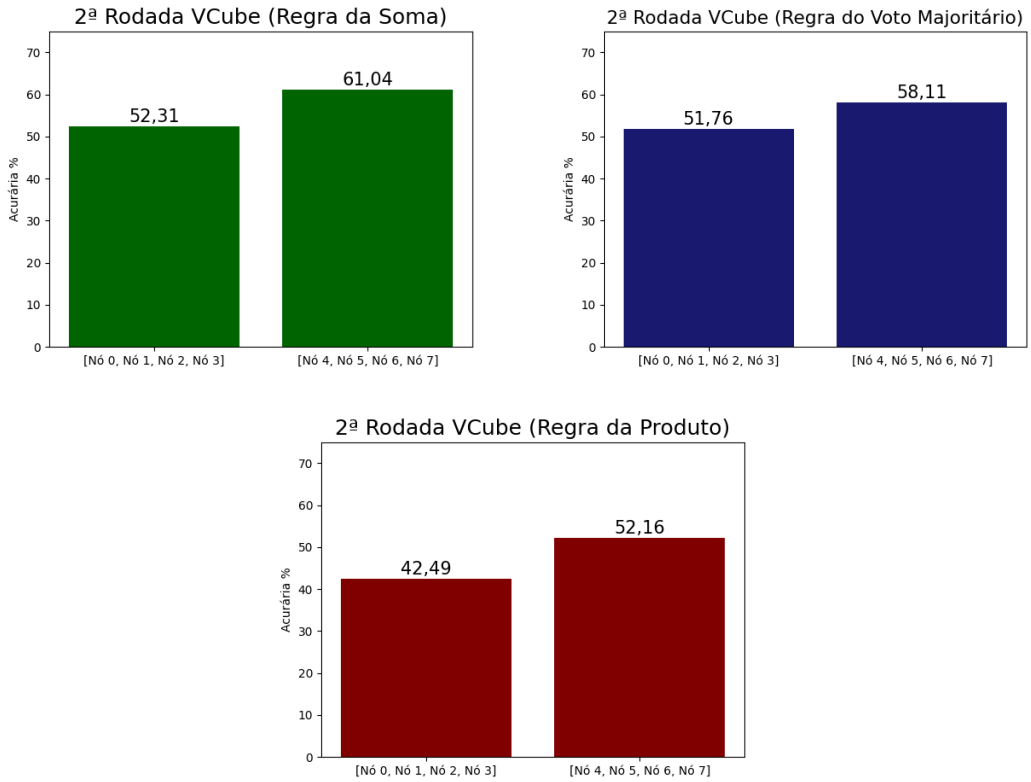
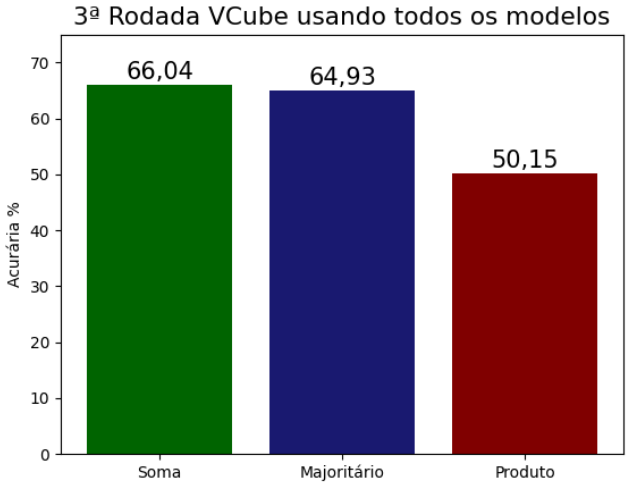


Figura 35 – Acurácia na segunda rodada do VCube.

Majoritário e 50,15% para a Regra do Produto (Figura 31).

Figura 36 – Acurácia da combinação de todos os nós, após a rodada 3 do VCube (sem falha)



5.5.5 Cenário 3 - Com falha

Por fim, utilizando o cenário 3, também foi realizada uma simulação onde cada um dos nós do sistema apresenta falha e não consegue mandar seu modelo de classificação para os outros nós da rede.

A Tabela 12 é possível observar a acurácia do SMC quando ocorre falha de cada um dos nós. Neste cenário também ficou perceptível que quando um nó com um classificador com baixa acurácia falha a tendencia do SMC final é ter um bom desempenho na classificação enquanto que quando nós que possuem classificadores com alta acurácia falham o desempenho do SMC final diminui.

Tabela 12 – Cenário 3 - Falha em cada um dos nós da rede

	Regra de Combinação		
Nó em falha	Soma	Voto	Produto
0	65,82	64,41	52,43
1	67,75	66,28	50,61
2	65,37	64,88	52,47
3	61,93	60,65	51,09
4	64,38	63,58	50,91
5	66,10	64,73	50,10
6	63,41	62,02	45,46
7	62,11	60,97	47,55

A Tabela 13 mostra o desempenho do SMC quando o nó com um classificador com desempenho mais baixo (menor acurácia) não consegue enviar seu modelo para ou outros nós, neste caso o classificador C_0 com 32,83%. Percebemos que em toda as as abordagens de combinação de classificadores os desempenho do SMC final ficou entre os melhores resultados considerando todas as possibilidades de falha dos nós.

Tabela 13 – Acurácia (%) do SMC baseado na VCube para o **segundo cenário** com falha do **nó 0** (menor desempenho) ao longo das três etapas do processo

		Regra de Combinação		
Etapa	Grupos	Soma	Voto	Produto
Rodada 1	1. C_1	39,16	39,16	39,16
	2. C_2, C_3	59,90	56,04	46,07
	3. C_4, C_5	43,21	40,22	35,74
	4. C_6, C_7	67,60	64,41	64,71
Rodada 2	1. C_1, C_2, C_3	53,12	51,13	42,78
	2. C_4, C_5, C_6, C_7	61,04	58,11	52,16
Rodada 3	1. C_1, C_2, C_3, C_4, C_5, C_7	65,82	54,41	52,43

Já a Tabela 14 mostra o desempenho do SMC quando o oposto acontece, ou seja, quando o nó com melhor (maior acurácia) classificador falha, neste caso o classificador C_6 que obteve 58,77% de acurácia, percebe-se que a acurácia do SMC final diminui.

Tabela 14 – Acurácia (%) do SMC baseado na VCube para o **segundo cenário** com falha do **nó 6** (maior desempenho) ao longo das três etapas do processo

		Regra de Combinação		
Etapa	Grupos	Soma	Voto	Produto
Rodada 1	1. C_0, C_1	48,21	47,37	31,76
	2. C_2, C_3	59,90	56,04	46,07
	3. C_4, C_5	43,21	40,22	35,74
	4. C_7	55,64	55,64	55,64
Rodada 2	1. C_0, C_1, C_2, C_3	52,30	51,76	42,49
	2. C_4, C_5, C_7	54,91	50,18	44,91
Rodada 3	1. C_0, C_1, C_2, C_3, C_4, C_5, C_7	63,41	62,02	45,46

5.6 Nós individuais com baixo desempenho

Durante a execução dos testes foram mantidos os nós com um desempenho baixo, pois o modelo de classificação gerado por ele pode representar a classificação de uma saída específica.

A combinação desse modelo com outro, pode funcionar com um complemento ajudando a classificar corretamente uma instância que um modelo sozinho não seja capaz de acertar.

5.7 Desempenho de carga

O modelo de classificação gerado em cada nó teve média 203KB (*Kilobyte*), ou seja, em um cenário com 8 nós no sistema, cada um pode receber 7 modelos de classificação ao mesmo tempo, gerando uma carga total de 1,4MB.

Por outro lado, se a opção fosse movimentar os dados para um determinado nó para que fosse realizado treinamento do modelo de classificação, seria necessário movimentar aproximadamente 99,3MB (*Megabyte*) de informação, uma quantidade de dados aproximadamente 70 vezes maior.

6 Conclusão

Algoritmos de classificação são utilizados em diversas áreas para encontrar soluções e resolver problemas, porém existem casos que não é possível obter todos os dados para o treinamento de um classificador robusto, pois esses dados podem estar em uma rede distribuída, onde a movimentação dessas informações se torna inviável devido ao volume de dados ou então devido ao próprio sigilo da informação.

Neste sentido a estratégia de aprendizado distribuído usando a topologia virtual VCube apresentado neste trabalho apresentou resultados significativos para a resolução deste problema, onde podemos perceber que após o treinamento individual de cada nó a melhor estratégia de combinação de classificadores é a que utiliza a regra da soma, pois foi a que obteve o melhor desempenho em todos os cenários, ou seja, futuras pesquisas podem levar em consideração utilizar apenas a regra da soma para melhorias nesta estratégia de aprendizado distribuído. Por meio do método de comunicação utilizado pelo VCube pode-se perceber em geral que a cada rodada de execução do algoritmo o desempenho do SMC aumenta em relação a taxa de acertos, desta maneira é essencial usar uma topologia de comunicação entre nós e tolerância a falha que garanta a comunicação rápida (logarítmica) e capaz de detectar se um nó está em falha ou não, pois a falta de um nó pode afetar diretamente na acurácia do SMC final.

Outro aspecto importante que é valido ressaltar a partir dos resultados obtidos é que o desempenho do SMC é dependente de cada modelo individual obtido, ou seja, se o SMC é composto por um modelo individual com alto desempenho o seu desempenho também é aumentado. No entanto, se o SMC é composto por um modelo individual com baixo desempenho o seu desempenho é diminuído. Sendo assim é possível aumentar (otimizar) o desempenho do SMC excluindo modelos individuais com desempenho baixo.

Quando foi realizado o treinamento com todos os dados (treinamento global) obtemos uma acurácia de 82,15%, enquanto na abordagem de aprendizado distribuído o melhor resultado que obtemos foi de 66,23% que foi no cenário 1 utilizando a regra da soma, ou seja, uma diferença de 16,75 pontos percentuais. Entretanto percebemos que na primeira rodada de execução do VCube no cenário 1 o SMC formado pelos modelos dos nós 6 e 7 obteve uma acurácia de 69,33%, ou seja, durante a construção do VCube é possível que sejam formados SMC intermediários melhores que o SMC final.

Desta maneira apontamos que outros classificadores, assim como outros métodos de combinação devem ser testados a fim de diminuir essa diferença percentual da aprendizagem global e da aprendizagem distribuída. Além disso, seria importante testar uma estratégia que ao combinar os classificadores houvesse uma escolha do melhor ou melhores

classificadores individuais de cada classe, independentemente do desempenho geral dele, para verificar se há o aumento de desempenho no sistema de classificação final.

Referências

ABEEL, T.; PEER, Y. Van de; SAEYS, Y. Java-ml: A machine learning library. *Journal of Machine Learning Research*, v. 10, p. 931–934, 2009. Citado na página 45.

ALPCAN, T.; BAUCKHAGE, C. A distributed machine learning framework. In: IEEE. *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*. [S.l.], 2009. p. 2546–2551. Citado 3 vezes nas páginas 9, 38 e 39.

APACHE, S. F. 2022. Disponível em: <https://hadoop.apache.org/>. Citado na página 39.

APPLE. *Core Motion - Process accelerometer, gyroscope, pedometer, and environment-related events.* [S.l.], 2024. Disponível em <https://developer.apple.com/documentation/coremotion>. Citado na página 48.

BADR, W. *Top sources for machine learning datasets.* Towards Data Science, 2019. Disponível em: <https://towardsdatascience.com/top-sources-for-machine-learning-datasets-bb6d0dc3378b>. Citado na página 47.

BREIMAN, L. et al. *Classification and regression trees*. [S.l.]: Routledge, 1984. Citado na página 24.

BRICKLEY, D.; BURGESS, M.; NOY, N. Google dataset search: Building a search engine for datasets in an open web ecosystem. In: *The World Wide Web Conference*. [s.n.], 2019. p. 1365–1375. Disponível em: <https://doi.org/10.1145/3308558.3313685>. Citado na página 47.

BRITTO Jr, A. S.; SABOURIN, R.; OLIVEIRA, L. E. Dynamic selection of classifiers—a comprehensive review. *Pattern recognition*, Elsevier, v. 47, n. 11, p. 3665–3680, 2014. Citado 2 vezes nas páginas 9 e 31.

BRUN, A. L. *Geração e Seleção de Classificadores com base na Complexidade do Problema*. Tese (Doutorado) — Pontifícia Universidade Católica do Paraná, Programa de Pós Graduação em Informática, Curitiba, 2017. Citado 2 vezes nas páginas 16 e 20.

CESTNIK, B.; KONONENKO, I.; BRATKO, I. A knowledge-elicitation tool for sophisticated users. In: *Proceedings of the 2nd European Conference on European Working Session on Learning EWSL*. [S.l.: s.n.], 1987. v. 87. Citado na página 24.

CHAN, P. K.; STOLFO, S. J. et al. Toward parallel and distributed learning by meta-learning. In: *AAAI workshop in Knowledge Discovery in Databases*. [S.l.: s.n.], 1993. v. 227, p. 240. Citado na página 36.

CHEN, S. A distributed algorithm for machine learning. *AIP Conference Proceedings*, v. 1955, n. 1, p. 040079, 2018. Disponível em: <https://aip.scitation.org/doi/abs/10.1063/1.5033743>. Citado 2 vezes nas páginas 34 e 35.

DORNADULA, V. N.; GEETHA, S. Credit card fraud detection using machine learning algorithms. *Procedia Computer Science*, v. 165, p. 631–641, 2019. ISSN 1877-0509. 2nd International Conference on Recent Trends in Advanced Computing ICRTAC -DISRUP - TIV INNOVATION , 2019 November 11-12, 2019. Disponível em: <https://www.sciencedirect.com/science/article/pii/S187705092030065X>. Citado na página 19.

DUARTE, E. P.; BONA, L. C. E.; RUOSO, V. K. Vcube: A provably scalable distributed diagnosis algorithm. In: *2014 5th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems*. [S.l.: s.n.], 2014. p. 17–22. Citado 6 vezes nas páginas 9, 10, 40, 41, 42 e 43.

DUARTE, E. P.; BONA, L. C. E.; RUOSO, V. K. VCube: A provably scalable distributed diagnosis algorithm. In: *2014 5th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems*. [S.l.: s.n.], 2014. p. 17–22. Citado na página 17.

EIBE, F.; HALL, M. A.; WITTEN, I. H. The weka workbench. online appendix for data mining: practical machine learning tools and techniques. In: *Morgan Kaufmann*. [S.l.]: Morgan Kaufmann Publishers San Francisco, California, 2016. Citado na página 46.

FRIEDMAN, J. H.; BENTLEY, J. L.; FINKEL, R. A. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software (TOMS)*, ACM New York, NY, USA, v. 3, n. 3, p. 209–226, 1977. Citado na página 24.

GUNES, V. et al. Combination, cooperation and selection of classifiers: A state of the art. *Int. J. Pattern Recognit. Artif. Intell.*, v. 17, p. 1303–1324, 2003. Citado 2 vezes nas páginas 16 e 31.

GUO, H.; ZHANG, J. A distributed and scalable machine learning approach for big data. In: *IJCAI*. [S.l.: s.n.], 2016. p. 1512–1518. Citado na página 34.

GUPTA, V. et al. Travellingfl: Communication efficient peer-to-peer federated learning. TechRxiv, 2022. Citado 3 vezes nas páginas 9, 40 e 41.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. *The elements of statistical learning: data mining, inference and prediction*. 2. ed. Springer, 2009. Disponível em: <http://www-stat.stanford.edu/~tibs/ElemStatLearn/>. Citado 3 vezes nas páginas 9, 22 e 27.

HAYKIN, S. S. *Neural networks and learning machines*. Third. Upper Saddle River, NJ: Pearson Education, 2009. Citado 3 vezes nas páginas 9, 29 e 30.

HU, H.; WANG, D.; WU, C. Distributed machine learning through heterogeneous edge systems. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. [S.l.: s.n.], 2020. v. 34, n. 05, p. 7179–7186. Citado na página 40.

JAIN, A. K.; DUIN, R. P. W.; MAO, J. Statistical pattern recognition: A review. *IEEE Transactions on pattern analysis and machine intelligence*, Ieee, v. 22, n. 1, p. 4–37, 2000. Citado na página 22.

JIAO, K. et al. A mobile application-based tower network digital twin management. In: SPRINGER. *Cyber Security Intelligence and Analytics: The 5th International Conference on Cyber Security Intelligence and Analytics (CSIA 2023), Volume 1*. [S.l.], 2023. p. 369–377. Citado na página 16.

KHERADPISHEH, S. R.; BEHJATI-ARDAKANI, F.; EBRAHIMPOUR, R. Combining classifiers using nearest decision prototypes. *Applied Soft Computing*, v. 13, n. 12, p. 4570–4578, 2013. ISSN 1568-4946. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1568494613002834>. Citado na página 31.

KITTLER, J. et al. On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 20, n. 3, p. 226–239, 1998. Citado 7 vezes nas páginas 16, 17, 30, 31, 32, 33 e 45.

KO, A. H.; SABOURIN, R.; Britto Jr, A. S. From dynamic classifier selection to dynamic ensemble selection. *Pattern recognition*, Elsevier, v. 41, n. 5, p. 1718–1731, 2008. Citado na página 22.

LEBAN, G. et al. Vizrank: Data visualization guided by machine learning. *Data Mining and Knowledge Discovery*, Springer, v. 13, n. 2, p. 119–136, 2006. Citado na página 21.

LU, H. et al. Robust coreset construction for distributed machine learning. *IEEE Journal on Selected Areas in Communications*, IEEE, v. 38, n. 10, p. 2400–2417, 2020. Citado 2 vezes nas páginas 34 e 35.

MALEKZADEH, M. et al. Mobile sensor data anonymization. In: *Proceedings of the International Conference on Internet of Things Design and Implementation*. New York, NY, USA: ACM, 2019. (IoTDI '19), p. 49–58. ISBN 978-1-4503-6283-2. Disponível em: <http://doi.acm.org/10.1145/3302505.3310068>. Citado na página 48.

MARSLAND, S. *MACHINE LEARNING*: An algorithmic perspective. New York: Chapman & Hall/CRC, 2014. ISBN 978-1-4665-8333-7. Citado 4 vezes nas páginas 9, 19, 20 e 21.

MATISA, N. A.; MAMAT, W. M. F. W. Clustered-hybrid multilayer perceptron network for pattern recognition application. *Applied Soft Computing*, v. 11, n. 1, p. 1457–1466, 2011. ISSN 1568-4946. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1568494610000931>. Citado 2 vezes nas páginas 16 e 19.

MERLA, P.; LIANG, Y. Data analysis using hadoop mapreduce environment. In: IEEE. *2017 IEEE International Conference on Big Data (Big Data)*. [S.l.], 2017. p. 4783–4785. Citado na página 39.

MIGUEZ, G. A.; MACULAN, N.; XAVIER, A. E. OtimizaÇÃo do algorÍtmo de backpropagation pelo uso da funÇÃo de ativaÇÃo bi-hiperbÓlica. In: *XIV Simpósio de Pesquisa Operacional e Logística da Marinha - SPOLM 2011*. Rio de Janeiro: A Pesquisa Operacional e a Logística na Amazônia Azul, 2011. Citado na página 30.

MITCHELL, T. M. *Machine Learning*. New York: McGraw-Hill, 1997. ISBN 978-0-07-042807-2. Citado 5 vezes nas páginas 9, 19, 23, 24 e 25.

MIZUKOSHI, M. et al. Implementation of multiple classifier system on mapreduce framework for intrusion detection. In: THE STEERING COMMITTEE OF THE WORLD CONGRESS IN COMPUTER SCIENCE, COMPUTER .... *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA)*. [S.l.], 2013. p. 185. Citado na página 39.

MU, Z. Optimization of inter-network bandwidth resources for large-scale data transmission. *JOURNAL OF NETWORKS*, v. 9, n. 3, p. 689–694, 2014. Citado na página 17.

MURATA, T. et al. Salivary metabolomics with alternative decision tree-based machine learning methods for breast cancer discrimination. *Breast cancer research and treatment*, Springer, v. 177, n. 3, p. 591–601, 2019. Citado na página 26.

NASSERI, A. A.; TUCKER, A.; CESARE, S. D. Quantifying stocktwits semantic terms' trading behavior in financial markets: An effective application of decision tree algorithms. *Expert systems with applications*, Elsevier, v. 42, n. 23, p. 9192–9210, 2015. Citado na página 26.

PARVIN, H. et al. Cchr: Combination of classifiers using heuristic retraining. In: *2008 Fourth International Conference on Networked Computing and Advanced Information Management*. [S.l.: s.n.], 2008. v. 2, p. 302–305. Citado na página 31.

PETEIRO-BARRAL, D.; GUIJARRO-BERDIÑAS, B. A survey of methods for distributed machine learning. *Progress in Artificial Intelligence*, Springer, v. 2, n. 1, p. 1–11, 2013. Citado 3 vezes nas páginas 34, 35 e 36.

POPOVIC, J. R. Distributed data networks: a blueprint for big data sharing and healthcare analytics. *Ann N Y Acad Sci*, v. 1387, n. 1, p. 105–111, 2017. Citado na página 16.

QUINLAN, J. R. Induction of decision trees. *Mach. Learn.*, Kluwer Academic Publishers, USA, v. 1, n. 1, p. 81–106, mar 1986. ISSN 0885-6125. Disponível em: <https://doi.org/10.1023/A:1022643204877>. Citado 2 vezes nas páginas 24 e 26.

QUINLAN, J. R. *C4.5: programs for machine learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993. ISBN 1-55860-238-0. Disponível em: <http://portal.acm.org/citation.cfm?id=152181>. Citado 2 vezes nas páginas 24 e 26.

RASCHKA, S.; PATTERSON, J.; NOLET, C. Machine learning in python: Main developments and technology trends in data science, machine learning, and artificial intelligence. *Information*, Multidisciplinary Digital Publishing Institute, v. 11, n. 4, p. 193, 2020. Citado na página 45.

SALVADEO, D. H. P. Combinaçao de múltiplos classificadores para reconhecimento de face humana. Universidade Federal de São Carlos, 2009. Citado na página 32.

SILVA, R. A. et al. CSBF: A static ensemble fusion method based on the centrality score of complex networks. *Computational Intelligence*, v. 36, n. 2, p. 522–556, 2020. https://doi.org/10.1111/coin.12249. Citado na página 49.

SPARKS, E. R. et al. Mli: An api for distributed machine learning. In: IEEE. *2013 IEEE 13th International Conference on Data Mining*. [S.l.], 2013. p. 1187–1192. Citado na página 40.

THARWAT, A. Parameter investigation of support vector machine classifier with kernel functions. *Knowledge and information systems*, Springer London, London, v. 61, n. 3, p. 1269–1302, 2019. ISSN 0219-1377. Citado 2 vezes nas páginas 9 e 28.

TSOUMAKAS, G.; VLAHAVAS, I. Distributed data mining. In: *Database technologies: Concepts, methodologies, tools, and applications*. [S.l.]: IGI Global, 2009. p. 157–164. Citado na página 34.

VAPNIK, V.; GOLOWICH, S.; SMOLA, A. Support vector method for function approximation, regression estimation and signal processing. In: MOZER, M.; JORDAN, M.; PETSCHE, T. (Ed.). *Advances in Neural Information Processing Systems*. MIT Press, 1996. v. 9. Disponível em: <https://dl.acm.org/doi/10.5555/2998981.2999021>. Citado na página 26.

VAPNIK, V. N. *The nature of statistical learning theory*. [S.l.]: Springer-Verlag New York, Inc., 1995. ISBN 0-387-94559-8. Citado 2 vezes nas páginas 27 e 28.

VERBRAEKEN, J. et al. A survey on distributed machine learning. *Acm computing surveys (csur)*, ACM New York, NY, USA, v. 53, n. 2, p. 1–33, 2020. Citado 6 vezes nas páginas 9, 35, 36, 37, 38 e 39.

WANG, H.; NIU, D.; LI, B. Distributed machine learning with a serverless architecture. In: IEEE. *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. [S.l.], 2019. p. 1288–1296. Citado na página 40.

WANG, Y.; KONG, T. Air quality predictive modeling based on an improved decision tree in a weather-smart grid. *IEEE Access*, IEEE, v. 7, p. 172892–172901, 2019. Citado na página 26.

ZAHARIA, M. et al. Apache spark: A unified engine for big data processing. *Communications of the ACM*, v. 59, n. 11, p. 56 – 65, 2016. ISSN 00010782. Disponível em: <https://doi.org/10.1145/2934664>. Citado na página 16.

ZHONG, N.; DONG, J.; OHSUGA, S. Using rough sets with heuristics for feature selection. *Journal of intelligent information systems*, Springer Nature, DORDRECHT, v. 16, n. 3, p. 199–214, 2001. ISSN 0925-9902. Citado na página 20.