



UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO, DE CIÊNCIAS EXATAS E EDUCAÇÃO
DEPARTAMENTO DE ENG. DE CONTROLE, AUTOMAÇÃO E COMPUTAÇÃO
CURSO DE GRADUAÇÃO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Leonardo dos Santos Schmitt

**Avaliação do Uso de *Metalabels* na Chain para Melhorar o Desempenho do
Algoritmo *Ensemble of Classifier Chains* (ECC)**

Blumenau
2025

Leonardo dos Santos Schmitt

**Avaliação do Uso de *Metalabels* na Chain para Melhorar o Desempenho do
Algoritmo *Ensemble of Classifier Chains* (ECC)**

Trabalho de Conclusão de Curso de Graduação em Engenharia de Controle e Automação do Centro Tecnológico, de Ciências Exatas e Educação da Universidade Federal de Santa Catarina como requisito para a obtenção do título de Engenheiro de Controle e Automação.
Orientador: Prof. Mauri Ferrandin, Dr.

Ficha catalográfica gerada por meio de sistema automatizado gerenciado pela BU/UFSC.
Dados inseridos pelo próprio autor.

Schmitt, Leonardo dos Santos
Avaliação do Uso de Metalabels na Chain para Melhorar o
Desempenho do Algoritmo Ensemble of Classifier Chains
(ECC) / Leonardo dos Santos Schmitt ; orientadora, Mauri
Ferrandin, 2025.
73 p.

Trabalho de Conclusão de Curso (graduação) -
Universidade Federal de Santa Catarina, Campus Blumenau,
Graduação em Engenharia de Controle e Automação, Blumenau,
2025.

Inclui referências.

1. Engenharia de Controle e Automação. 2. Classificação
Multirótulo. 3. Cadeia de Classificadores. 4. Aprendizado
de Máquina. I. Ferrandin, Mauri. II. Universidade Federal
de Santa Catarina. Graduação em Engenharia de Controle e
Automação. III. Título.

Leonardo dos Santos Schmitt

Avaliação do Uso de *Metalabels* na Chain para Melhorar o Desempenho do Algoritmo *Ensemble of Classifier Chains* (ECC)

Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do Título de “Engenheiro de Controle e Automação” e aprovado em sua forma final pelo Curso de Graduação em Engenharia de Controle e Automação.

Blumenau, 3 de Dezembro de 2025.

Banca Examinadora:



Documento assinado digitalmente
Mauri Ferrandin
Data: 12/12/2025 15:17:00-0300
CPF: ***.580.869-**
Verifique as assinaturas em <https://v.ufsc.br>

Prof. Mauri Ferrandin, Dr.
Universidade Federal de Santa Catarina



Documento assinado digitalmente
Carlos Roberto Moratelli
Data: 12/12/2025 15:19:21-0300
CPF: ***.999.369-**
Verifique as assinaturas em <https://v.ufsc.br>

Prof. Carlos Roberto Moratelli, Dr.
Universidade Federal de Santa Catarina



Documento assinado digitalmente
Maiquel de Brito
Data: 12/12/2025 15:27:01-0300
CPF: ***.657.150-**
Verifique as assinaturas em <https://v.ufsc.br>

Prof. Maiquel de Brito, Dr.
Universidade Federal de Santa Catarina

Este trabalho é dedicado a minha mãe, namorada, família e a Deus.

AGRADECIMENTOS

Gostaria de agradecer primeiramente a Deus por toda força e coragem a qual foi me dada durante a jornada. Agradecer a minha namorada, mãe e restante da família pelo apoio e incentivo. Aos meus professores, em especial professor Mauri Ferrandin pela orientação de todo o projeto.

RESUMO

Problemas de classificação são comuns no estudo do aprendizado de máquina, especialmente em situações em que é necessário atribuir múltiplos rótulos a uma única instância. Nesse contexto, destaca-se a classificação multirrótulo, em inglês *Multi-label Classification* (MLC), na qual instâncias podem estar associadas a diversos rótulos, sendo aplicada em áreas como categorização de músicas, identificação de funções proteicas e reconhecimento de imagens. Diversas abordagens têm sido propostas para resolver problemas de MLC, sendo o conjunto de classificadores em cadeia, em inglês *Ensemble of Classifier Chains* (ECC), uma das técnicas utilizadas. Pesquisadores têm desenvolvido variações do ECC com o objetivo de melhorar os resultados, seja otimizando a ordem dos classificadores na cadeia, seja reduzindo o número de rótulos no conjunto de dados. Diferentemente de abordagens anteriores, este trabalho tem como objetivo inserir *metalabels*, combinações de dois rótulos derivados de medidas de similaridade, no ECC e investigar se a presença desses *metalabels*, bem como a ordem em que são inseridos na cadeia, influencia o desempenho preditivo. Por fim, são apresentados os resultados obtidos nas simulações realizadas, avaliados por oito métricas amplamente utilizadas em problemas multirrótulo: perda de Hamming, F1 macro, F1 micro, acurácia de subconjunto, AUC macro, AUC micro, AUPRC macro e AUPRC micro. Os experimentos mostram que a abordagem proposta proporciona melhorias consistentes em diversas dessas medidas.

Palavras-chave: Classificação Multirrótulo. Metalabels. Cadeia de Classificadores.

ABSTRACT

Multi-label classification (MLC) problems, in which instances may be associated with multiple labels, are common in machine learning applications such as music categorization, protein function identification, and image recognition. Several approaches have been proposed to address MLC problems, and the *Ensemble of Classifier Chains* (ECC) is one of the methods frequently adopted. Researchers have developed variations of ECC with the aim of improving predictive performance, either by optimizing the label ordering within the chain or by reducing the number of labels in the dataset. Unlike previous approaches, this work introduces metalabels pairs of labels combined based on similarity measures into the ECC framework and investigates whether the presence of these metalabels, as well as the position in which they are inserted, affects predictive performance. Finally, the results obtained from the simulations, evaluated using eight widely adopted multi-label metrics: Hamming-Loss, Macro-F1, Micro-F1, Subset Accuracy, Macro-AUC, Micro-AUC, Macro-AUPRC, and Micro-AUPRC, show that the proposed approach yields consistent improvements across several of these measures.

Keywords: Multi-Label Classifier. Metalabels, Classifier Chain.

LISTA DE FIGURAS

Figura 1 – Relação entre inteligência artificial e o aprendizado de máquina.	15
Figura 2 – Processo de classificação em aprendizado de máquina.	18
Figura 3 – Taxonomia dos problemas de classificação.	18
Figura 4 – Classificação de um novo dado utilizando o algoritmo kNN.	19
Figura 5 – Representação esquemática da margem e dos vetores de suporte.	22
Figura 6 – Estrutura de categorização dos algoritmos de aprendizado multi-rótulo.	24
Figura 7 – Decomposição do problema multi-rótulo pelo BR.	25
Figura 8 – Representação do processo de classificação no CC.	26
Figura 9 – Arquitetura do ECC.	27
Figura 10 – Processo de classificação após criação e inserção do <i>metalabels</i>	35
Figura 11 – Seleção dos melhores rótulos para criação do <i>metalabel</i>	36
Figura 12 – Estrutura final da metodologia.	38
Figura 13 – Exemplo de matriz de similaridade.	39
Figura 14 – Fluxograma do funcionamento da função <code>verifica_best metas</code>	40
Figura 15 – Fluxograma do funcionamento da função <code>verifica_pred</code>	42
Figura 16 – Arquitetura do ECCx com <i>metalabels</i> no início da cadeia.	43
Figura 17 – Esquema da Validação Cruzada <i>k-Fold</i>	44
Figura 18 – <i>Workflow</i> para execução em <i>Shell Script</i>	45
Figura 19 – Resultado do teste de <i>Wilcoxon</i> aplicado em dois métodos aleatórios.	47
Figura 20 – Comparação de vitórias, derrotas e empates — SVM (lógica AND).	49
Figura 21 – Comparação de vitórias, derrotas e empates — SVM (lógica OR).	49
Figura 22 – Comparação de vitórias, derrotas e empates — SVM (lógica XNOR).	50
Figura 23 – Comparação de vitórias, derrotas e empates — C5.0 (lógica AND).	51
Figura 24 – Comparação de vitórias, derrotas e empates — C5.0 (lógica OR).	51
Figura 25 – Comparação de vitórias, derrotas e empates — C5.0 (lógica XNOR).	52
Figura 26 – Comparação de vitórias, derrotas e empates — kNN (lógica AND).	53
Figura 27 – Comparação de vitórias, derrotas e empates — kNN (lógica OR).	53
Figura 28 – Comparação de vitórias, derrotas e empates — kNN (lógica XNOR).	54
Figura 29 – Comparação de vitórias, derrotas e empates — NB (lógica AND).	55
Figura 30 – Comparação de vitórias, derrotas e empates — NB (lógica OR).	55
Figura 31 – Comparação de vitórias, derrotas e empates — NB (lógica XNOR).	56
Figura 32 – Comparação de vitórias, derrotas e empates — ECC.	58
Figura 33 – Comparação do ECC-SVM com os melhores métodos.	59
Figura 34 – Comparação do ECC-C5.0 com os melhores métodos.	60

LISTA DE TABELAS

Tabela 1	– Predição de Y_1 , Y_2 e <i>metalabel</i> formado apartir da lógica AND	33
Tabela 2	– Predição de Y_1 , Y_2 e <i>metalabel</i> formado apartir da lógica OR	34
Tabela 3	– Predição de Y_1 , Y_2 e <i>metalabel</i> formado apartir da lógica XNOR	34
Tabela 4	– Tabela de probabilidades antes da regra de propagação.	41
Tabela 5	– Tabela de probabilidades depois da regra de propagação.	41
Tabela 6	– Conjuntos de dados utilizados no experimento (obtidos em: https://cometa.ujaen.es)	46
Tabela 7	– Melhores métodos por lógica e classificador (ordenados por %vitórias) .	57
Tabela 8	– Piores métodos por lógica e classificador (ordenados por %vitórias dentro de cada lógica)	57
Tabela 9	– Resultados de <i>Hamming-Loss</i> — melhores e piores valores por algoritmo e fator	61
Tabela 10	– Resultados de <i>Macro-F1</i> — melhores e piores valores por algoritmo e fator	62
Tabela 11	– Resultados de <i>Micro-F1</i> — melhores e piores valores por algoritmo e fator	62
Tabela 12	– Resultados de <i>Subset Accuracy</i> — melhores e piores valores por algoritmo e fator	63
Tabela 13	– Resultados de <i>Macro-AUC</i> — melhores e piores valores por algoritmo e fator	63
Tabela 14	– Resultados de <i>Micro-AUC</i> — melhores e piores valores por algoritmo e fator	64
Tabela 15	– Resultados de <i>Macro-AUPRC</i> — melhores e piores valores por algoritmo e fator	64
Tabela 16	– Resultados de <i>Micro-AUPRC</i> — melhores e piores valores por algoritmo e fator	65

LISTA DE ABREVIATURAS E SIGLAS

AUC	<i>Area Under the ROC (Receiver Operating Characteristic) Curve</i>
AUPRC	<i>Area Under the Precision–Recall Curve</i>
BR	<i>Binary Relevance</i>
CC	<i>Classifier Chain</i>
CML	<i>Collective Multi-Label Classifier</i>
CSMLTB	<i>Correção Sem Metalabel Tail Block</i>
CXSMMLTB	<i>Correção Sem Metalabel Tail Block Exclusive</i>
ECC	<i>Ensemble of Classifier Chains</i>
ECCx	<i>Ensemble of Classifier Chains Exclusivo</i>
FN	<i>False Negative</i>
FP	<i>False Positive</i>
IoT	<i>Internet of Things</i>
JSON	<i>JavaScript Object Notation</i>
kNN	<i>k-Nearest Neighbor</i>
LP	<i>Label Powerset</i>
MLC	<i>Multi-label Classification</i>
ML-DT	<i>Multi-Label Decision Tree</i>
ML-kNN	<i>Multi-Label k-Nearest Neighbor</i>
NB	<i>Naive Bayes</i>
NoSQL	<i>Not Only Structured Query Language</i>
PR	<i>Precision–Recall</i>
RAkEL	<i>Random k-Labelsets</i>
Rank-SVM	<i>Ranking Support Vector Machine</i>
SMLTB	<i>Sem Metalabel Tail Block</i>
SVM	<i>Support Vector Machine</i>
TN	<i>True Negative</i>
TP	<i>True Positive</i>
XML	<i>Extensible Markup Language</i>
XSMMLTB	<i>Sem Metalabel Tail Block Exclusive</i>

SUMÁRIO

1	INTRODUÇÃO	12
1.1	OBJETIVOS	13
1.1.1	Objetivo Geral	13
1.1.2	Objetivos Específicos	13
2	FUNDAMENTAÇÃO TEÓRICA	15
2.1	APRENDIZADO DE MÁQUINA	15
2.2	CLASSIFICAÇÃO	17
2.2.1	<i>k-Nearest Neighbor</i> (kNN)	18
2.2.2	<i>Naive Bayes</i> (NB)	20
2.2.3	<i>C5.0</i>	20
2.2.4	<i>Support Vector Machine</i> (SVM)	21
2.3	CLASSIFICAÇÃO MULTIRRÓTULO	22
2.3.1	<i>Binary Relevance</i> (BR)	24
2.3.2	<i>Classifier Chain</i> (CC)	26
2.3.3	<i>Ensemble Classifier Chain</i> (ECC)	26
2.4	MÉTRICA PARA AVALIAÇÃO DA CLASSIFICAÇÃO MULTIRRÓ- TULO	27
2.4.1	Métricas Baseadas em Exemplos	28
2.4.2	Métricas Baseadas em Rótulos	29
2.4.3	Métricas Baseadas em Ranking	30
2.5	SIMILARIDADE DE <i>JACCARD</i>	31
3	PROJETO E IMPLEMENTAÇÃO	33
3.1	ESTRUTURA DO MÉTODO PROPOSTO	33
3.2	APLICANDO SIMILARIDADE DE JACCARD	35
3.3	MÉTODOS RESULTANTES	37
3.4	IMPLEMENTAÇÃO DO MÉTODO PROPOSTO	37
3.4.1	Seleção das <i>Metalabels</i>	38
3.4.2	Atribuição de Probabilidade nos Rótulos Originais	40
3.4.3	Inserindo os Metalabels no Início da Chain	41
3.4.4	Implementação em Shell Script	44
4	RESULTADO	46
4.1	COMPARAÇÃO GERAL	47
4.2	COMPARAÇÃO GERAL ENTRE MÉTRICAS	60
5	CONCLUSÃO	66
	REFERÊNCIAS	68

1 INTRODUÇÃO

O aprendizado de máquina é uma área da inteligência artificial capaz de resolver problemas a partir de experiências passadas. Dentre os tipos de problemas que podem ser tratados com aprendizado de máquina, destacam-se os problemas de classificação, que consistem em relacionar um atributo x a uma classe y , dada uma função alvo f que mapeia os valores de cada conjunto (TAN; STEINBACH; KUMAR, 2009). Os problemas de classificação podem ser monorrótulo, nos quais apenas um rótulo (ou classe) é atribuído a uma instância, e multirrótulo, nos quais n classes podem ser atribuídas a uma instância (SAJID *et al.*, 2023). O trabalho em questão visa apresentar soluções na área da classificação multirrótulo.

Diversas abordagens foram desenvolvidas para resolver problemas multirrótulo, como, por exemplo, a *Binary Relevance* (BR), que decompõe todos os multirrótulos em rótulos únicos e tenta prever cada um de forma independente. No entanto, o BR não considera a correlação entre rótulos. Para resolver esse problema, foram propostas diversas abordagens, sendo uma delas a *Classifier Chain* (CC). O CC tem como princípio o método BR, particionando o conjunto de dados em n classificadores binários, um para cada rótulo, e organizando-os em uma cadeia, onde a previsão de saída do classificador anterior é usada como um recurso de entrada adicional para os classificadores subsequentes (READ *et al.*, 2009). Entretanto, assim como BR, o CC tem problemas relacionados à performance, visto que depende da seleção randômica da cadeia. Ou seja, dependendo da seleção e da ordem da cadeia, a predição pode ser afetada negativamente. Por fim, para mitigar essas problemáticas, desenvolveu-se o *Ensemble of Classifier Chains* (ECC). O ECC cria m modelos de CC, cada um com uma cadeia aleatória diferente, construída a partir de uma fração de instâncias e atributos. A previsão final é baseada na média dos valores de confiança de cada rótulo obtido a partir desses modelos (MOYANO *et al.*, 2018).

Em vista do que foi previamente apresentado, o presente trabalho tem como objetivo estender o estudo publicado pelo autor em Schmitt e Ferrandin (2024), no qual foi realizada uma avaliação inicial da performance do algoritmo ECC quando o espaço de rótulos é ampliado por meio de rótulos artificiais, chamados de *metalabels*, gerados a partir da combinação entre dois rótulos originais. No artigo, é apresentado que a criação dos *metalabels* requer um método para medir a semelhança ou distância entre os rótulos originais, pois isso constitui a base para sua construção, haja vista que, caso o número de rótulos inseridos na cadeia aumente, o algoritmo pode ser afetado negativamente e consumir uma quantidade significativa de memória computacional. Além disso, outro fator que pode impactar negativamente é a presença de muitos zeros nos *metalabels*. Isso ocorre porque, em problemas multirrótulo, os zeros indicam a ausência de uma classe em particular. O excesso de zeros pode ser prejudicial, pois o modelo terá que lidar com muitas instâncias onde nenhum rótulo é atribuído. Na etapa de criação e simulação

apresentada no estudo anterior, utilizou-se a lógica AND e a similaridade de Jaccard para gerar os *metalabels*, avaliando-se o impacto dessa estratégia no ECC por meio dos classificadores C5.0 e *Support Vector Machine* (SVM). Os resultados mostraram que algumas métricas, como *Micro-Precision*, *Precision*, *Hamming-Loss* e *Subset-Accuracy*, apresentaram melhora com a presença dos *metalabels*, indicando que, em certos casos, essas combinações artificiais conseguem capturar relações úteis entre os rótulos. Contudo, outras métricas apresentaram desempenho superior sem os *metalabels*, evidenciando que sua inclusão também pode introduzir redundância e prejudicar o modelo.

Diante disso, este trabalho propõe a mesma forma controlada de gerar *metalabels*, utilizando apenas os pares de rótulos que apresentam maior similaridade segundo a medida de Jaccard. Entretanto, não é utilizada apenas a lógica AND, também são avaliadas outras lógicas para a formação dos *metalabels*, como OR e XNOR, uma vez que cada uma delas pode gerar padrões distintos e influenciar a densidade final dos novos rótulos. Além disso, considerou-se, além dos classificadores-base C5.0 e SVM, os classificadores *k-Nearest Neighbor* (kNN) e *Naive Bayes* (NB), tanto para a lógica já utilizada quanto para as demais.

Outra parte importante da proposta é analisar como a ordem dos rótulos influencia diretamente o processo de predição. Nesse sentido, o trabalho estuda tanto a inserção dos *metalabels* no final da cadeia quanto no início, além de uma versão que ajusta as probabilidades dos rótulos originais com base nos *metalabels*. Essas variações permitem observar se os *metalabels* de fato auxiliam o ECC ou se podem prejudicar o desempenho, dependendo da forma como são integrados.

1.1 OBJETIVOS

1.1.1 Objetivo Geral

O objetivo geral deste projeto é avaliar o impacto da inserção de *metalabels* no algoritmo ECC e investigar como a expansão controlada do espaço de rótulos influencia o desempenho em tarefas de classificação multirrótulo.

1.1.2 Objetivos Específicos

Norteados pelo objetivo geral, os objetivos específicos serão:

- Revisar a literatura a respeito de classificação multirrótulo, métodos baseados em cadeias e técnicas de expansão de espaço de rótulos;
- Especificar e implementar uma versão estendida do ECC que permita a criação e inserção de *metalabels* na cadeia, considerando diferentes lógicas e diferentes posições na estrutura;

- Definir e avaliar métodos de seleção de *metalabels*, incluindo técnicas baseadas em medidas de correlação (como similaridade de Jaccard) e abordagens aleatórias para fins comparativos;
- Explorar estratégias adicionais, como propagação de probabilidades e inserção dos *metalabels* no início ou no final da cadeia;
- Avaliar exaustivamente o desempenho dos métodos propostos utilizando 8 medidas de desempenho disponíveis no pacote `utiml`, considerando 4 classificadores base;
- Comparar os resultados obtidos com métodos representativos do estado da arte em classificação multirrótulo;

2 FUNDAMENTAÇÃO TEÓRICA

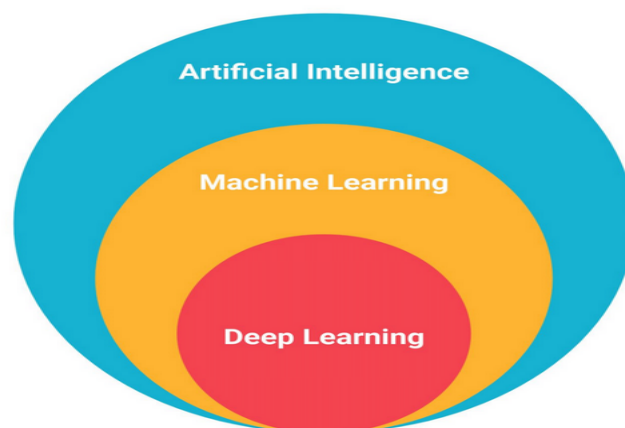
O presente capítulo apresenta a fundamentação teórica que dá suporte ao desenvolvimento deste trabalho, abordando conceitos, técnicas e estratégias relacionadas à área de aprendizado de máquina. Inicialmente, são discutidos os princípios gerais do aprendizado de máquina, seguidos pela apresentação das técnicas de classificação, com destaque para os modelos de Regressão Linear e Regressão Logística, amplamente utilizados em tarefas de previsão e categorização de dados.

Além disso, o capítulo aborda o conceito de classificação multirrótulo, um tema de relevância crescente em aplicações práticas, no qual uma mesma instância pode estar associada a múltiplas classes simultaneamente. São discutidas as principais abordagens para tratar esse tipo de problema, como os métodos de transformação do problema e de adaptação de algoritmos, bem como suas vantagens e limitações.

2.1 APRENDIZADO DE MÁQUINA

O Aprendizado de Máquina é uma área da Inteligência Artificial, conforme apresentado na Figura 1, e tem crescido significativamente nos últimos anos devido à sua ampla aplicação na Indústria 4.0 e em diversos outros setores (SARKER, 2021). Essa área busca desenvolver métodos computacionais capazes de identificar padrões e extrair conhecimento a partir de dados, permitindo que sistemas melhorem seu desempenho com a experiência, sem necessidade de programação explícita (FACELI *et al.*, 2011).

Figura 1 – Relação entre inteligência artificial e o aprendizado de máquina.



Fonte: (KOUL; GANJU; KASAM, 2020)

Os algoritmos de aprendizado de máquina podem ser classificados em quatro categorias principais: supervisionado, não supervisionado, semi-supervisionado e por reforço. No aprendizado supervisionado, o modelo é treinado com dados rotulados para aprender

relações entre variáveis e realizar previsões em novos conjuntos de dados. Já o aprendizado não supervisionado utiliza dados sem rótulos, buscando identificar padrões ou agrupamentos de forma autônoma. O aprendizado semi-supervisionado combina ambas as abordagens, aproveitando um pequeno conjunto de dados rotulados e um grande volume de dados não rotulados, sendo útil quando o processo de rotulagem é limitado. Por fim, o aprendizado por reforço ocorre a partir da interação de um agente com o ambiente, no qual ele aprende, por meio de recompensas e penalidades, a tomar decisões que maximizem seu desempenho ao longo do tempo (HSSINA *et al.*, 2014).

Além dos tipos de aprendizado, um fator essencial para a construção de modelos é a disponibilidade e o formato dos dados, que podem ser estruturados, semiestruturados, não estruturados ou metadados (HAN; KAMBER; PEI, 2011). Os dados estruturados possuem formato tabular e organização definida, como nomes, datas e endereços, sendo facilmente armazenados e processados em bancos de dados relacionais. Os dados não estruturados, por sua vez, não seguem um padrão fixo, o que dificulta sua análise, englobando textos, imagens, vídeos, áudios e outros conteúdos multimídia. Já os semiestruturados apresentam alguma organização, embora não sigam esquemas rígidos, como é o caso de formatos *Extensible Markup Language* (XML), *JavaScript Object Notation* (JSON) e bancos *Not Only Structured Query Language* (NoSQL). Por fim, os metadados descrevem informações sobre os próprios dados, como autor, data de criação, tamanho e palavras-chave, fornecendo contexto adicional para sua interpretação (SARKER, 2021).

De modo geral, a escolha do tipo de aprendizado e o formato dos dados influenciam diretamente as técnicas empregadas na construção dos modelos. Entre as principais abordagens e aplicações, destacam-se as tarefas preditivas e descritivas, que diferem quanto ao objetivo e à forma de análise dos dados. As tarefas preditivas têm como finalidade estimar ou antecipar valores ou categorias desconhecidas a partir de dados históricos, buscando generalizar padrões para novos casos. São exemplos a classificação, que prevê rótulos ou categorias como “sim” ou “não” para concessão de empréstimos, e a regressão, voltada à previsão de valores numéricos contínuos, como preços de ações ou volume de chuva. Já as tarefas descritivas possuem caráter exploratório e visam identificar estruturas, padrões e relações ocultas nos dados, sem que haja um valor de saída previamente definido. Nessa categoria enquadra-se o agrupamento (*clustering*), que organiza dados semelhantes em grupos, como no caso de textos que tratam de assuntos relacionados (CERRI; LEON FERREIRA DE CARVALHO, 2019).

Apesar dos avanços e da ampla utilização do aprendizado de máquina em diferentes domínios, diversos desafios ainda comprometem sua aplicação prática. Entre eles, destaca-se a coleta de dados em áreas como cibersegurança, *Internet of Things* (IoT), saúde e agricultura, que nem sempre é simples, mesmo diante do grande volume de informações geradas continuamente (ŚLUSARCZYK, 2018). Além disso, os dados disponíveis para o treinamento dos modelos podem conter valores ambíguos, ausentes ou inconsistentes, o que

afeta diretamente a qualidade do treinamento e o desempenho dos modelos. Diante disso, o processo de pré-processamento e limpeza de dados permanece uma tarefa complexa, exigindo o aprimoramento de métodos existentes ou o desenvolvimento de novas abordagens que melhorem a preparação dos dados para análise. Outro desafio relevante está na seleção do algoritmo mais adequado, pois resultados distintos podem surgir de acordo com as características dos dados e do problema a ser resolvido. Nesse contexto, o uso de modelos híbridos e técnicas de ensemble, bem como o aperfeiçoamento de métodos de aprendizado existentes, representa uma direção promissora de pesquisa (SARKER *et al.*, 2020).

2.2 CLASSIFICAÇÃO

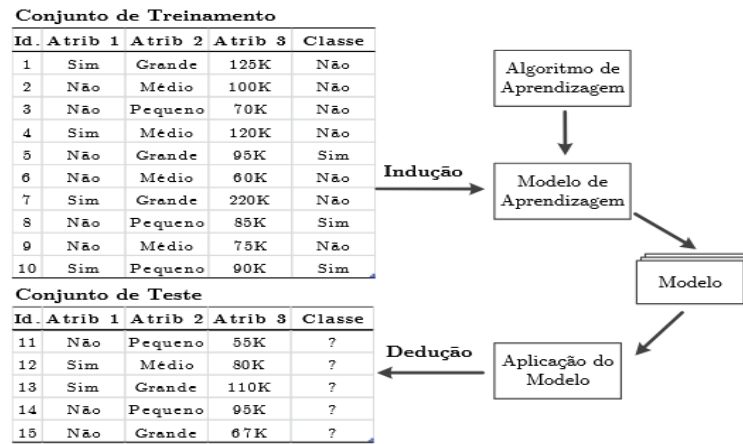
A classificação é um dos problemas mais importantes do aprendizado de máquina e da mineração de dados, pois está presente em uma ampla variedade de aplicações práticas, como o diagnóstico médico, a filtragem de mensagens de *spam*, o reconhecimento de padrões e a análise de imagens. A classificação é a tarefa de aprender uma função-alvo f que mapeia cada conjunto de atributos x para um dos rótulos de classe y pré-determinados, dos quais a instância faz parte (TAN; STEINBACH; KUMAR, 2009).

Em outras palavras, o principal objetivo da classificação é construir um modelo capaz de aprender padrões a partir dos dados de conhecidos e, assim, generalizar o conhecimento adquirido para prever corretamente a classe de novas instâncias ainda não observadas. Para isso, o conjunto de dados é normalmente dividido em duas partes: treinamento e teste. No conjunto de treinamento, os rótulos das classes são utilizados, em conjunto com um algoritmo de aprendizado de máquina, para induzir o modelo. Em seguida, o modelo gerado é aplicado ao conjunto de teste, no qual os rótulos são desconhecidos, a fim de avaliar sua capacidade de generalização. Esse processo é essencial para garantir a qualidade das previsões e reforça o papel da classificação na tomada de decisão automatizada e na descoberta de conhecimento em grandes volumes de dados, conforme apresentado na Figura 2. Entretanto, em alguns casos, como no caso de classificadores preguiçosos (*lazy learning*) o treinamento é omitido, e a classificação é realizada diretamente a partir da relação entre as instâncias de treinamento e a instância de teste. Exemplos disso são métodos baseado em instância como classificador de vizinhos mais próximo (*nearest neighbor classifiers*), mesmo assim a fase pré-processamento ainda é necessária (AGGARWAL, 2014).

Os problemas de classificação de dados podem ser divididos em dois tipos principais: classificação plana e hierárquica. No caso da classificação plana, consiste na predição de classe sem considerar hierarquia entre as classes, o que difere da classificação hierárquica (JR.; FREITAS, 2011). Além disso dois tipos principais podem ser subdivididos em dois subtipos: monorrótulo, no qual apenas um rótulo (ou classe) é atribuído a uma instância, e multirrótulo, no qual n classes podem ser atribuídas a uma instância (SAJID *et al.*, 2023). É importante ressaltar que a classificação plana monorrótulo pode ser binária,

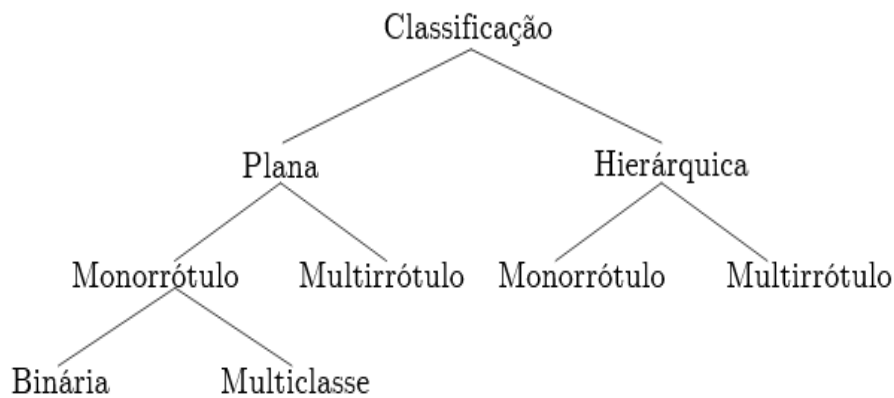
quando existem dois rótulos, ou multiclasse, quando o número de rótulos é maior que dois (TSOUMAKAS; KATAKIS, 2007). Essa hierarquia pode ser visualizada na Figura 3.

Figura 2 – Processo de classificação em aprendizado de máquina.



Fonte: (TAN; STEINBACH; KUMAR, 2009)

Figura 3 – Taxonomia dos problemas de classificação.



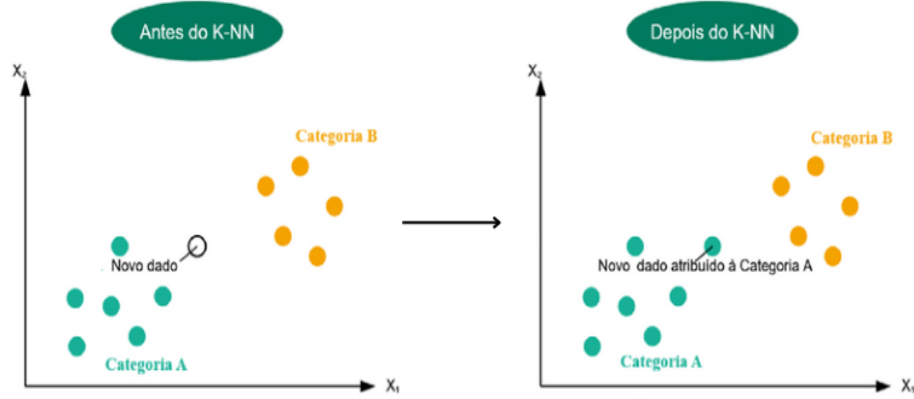
Fonte: (FERRANDIN; ÁVILA; NIEVOLA, 2012)

2.2.1 *k*-Nearest Neighbor (kNN)

O kNN é um método de aprendizado não paramétrico e baseado em instância, fazendo parte da técnica de aprendizado de máquina supervisionado. Conforme mencionado anteriormente, o kNN é um algoritmo de aprendizado preguiçoso, que formula previsões analisando a estrutura dos dados em tempo real, à medida que novas instâncias são apresentadas, sem exigir uma fase prévia explícita de treinamento (HALDER; UDDIN, M. N.;

UDDIN, M. A. *et al.*, 2024). Em outras palavras suponha que tenhamos duas categorias, isto é, Categoria A e Categoria B, e recebamos um novo ponto de dados X . O objetivo é determinar a qual dessas categorias X pertence, conforme apresentado na Figura 4.

Figura 4 – Classificação de um novo dado utilizando o algoritmo kNN.



Fonte: Elaborado pelo autor baseado em (SUYAL; GOYAL, 2022).

Segundo Suyal e Goyal (2022), o algoritmo kNN pode ser implementado por meio de uma sequência de etapas bem definidas. Primeiramente, seleciona-se o número de vizinhos (k), isto é, a quantidade de pontos mais próximos que serão considerados para a classificação. Em seguida, calcula-se a distância entre o novo ponto de dados e todos os pontos existentes no conjunto de treinamento. A métrica mais comumente utilizada é a distância Euclidiana, definida pela Equação (1), em que A e B representam os vetores de características de dimensão n .

$$d(A, B) = \sqrt{\sum_{i=1}^n (A_i - B_i)^2} \quad (1)$$

Após o cálculo das distâncias, identificam-se os k vizinhos mais próximos do novo dado, considerando aqueles que apresentam os menores valores de distância Euclidiana. Entre esses k vizinhos selecionados, contabiliza-se a frequência das classes, isto é, o número de pontos pertencentes a cada categoria. Em seguida, o novo ponto de dados é atribuído à classe majoritária, ou seja, à categoria que possuir o maior número de vizinhos dentre os k selecionados. Por fim, forma-se o modelo de classificação kNN, que é capaz de classificar novas instâncias com base na proximidade entre os dados de entrada e os exemplos previamente armazenados no conjunto de treinamento.

Em relação a aplicação do kNN, existem algumas questões chaves, como por exemplo a escolha de k vizinhos. Caso k for muito pequeno, o resultado pode ser sensível a pontos de ruído. Por outro lado, se k for muito grande, a vizinhança pode incluir muitos

pontos de outras classes. Destaca-se ainda a escolha da equação de medidas de distância que pode ser afetada na alta dimensionalidade. Por exemplo, se o kNN for aplicado à classificação de documentos, pode ser melhor usar a medida do cosseno em vez da distância euclidiana (WU, X. *et al.*, 2008)

2.2.2 Naive Bayes (NB)

O NB é um classificador probabilístico simples, porém amplamente utilizado em tarefas de aprendizado de máquina supervisionado. O princípio do NB deriva do Teorema de Bayes, que combina a probabilidade *a priori* de cada classe com a probabilidade dos dados serem observados sob essa classe, conforme apresentado na Equação (2) (HECKERMAN, 1997). Em outras palavras, o classificador estima a probabilidade de uma amostra X pertencer a uma determinada classe C_i com base nas evidências observadas, atualizando as crenças iniciais sobre cada classe à luz das novas informações contidas nos atributos da amostra (CHEN *et al.*, 2021).

$$P(C_i | X) = \frac{P(X | C_i) P(C_i)}{P(X)} \quad (2)$$

No caso específico do classificador NB, o objetivo é maximizar a probabilidade posterior, dado o conjunto de treinamento. A regra de decisão é, portanto, formulada conforme a Equação (3), onde o rótulo previsto para uma amostra é aquele que maximiza $P(C_i | X)$. Por exemplo, ao classificar se uma pessoa possui diabetes ou não, a decisão é tomada de acordo com a condição $P(\text{diabetes} | x_i) \geq P(\text{não-diabetes} | x_i)$; caso contrário, a pessoa é classificada como saudável (RASCHKA, 2014).

$$\hat{C} = \arg \max_{i=1, \dots, m} P(C_i | X) \quad (3)$$

2.2.3 C5.0

O C5.0 é versão melhorada do algoritmo o C4.5, que foi proposto por J. Ross Quinlan's, capaz de lidar tanto com atributos categóricos quanto contínuos, além de tratar valores ausentes durante o processo de construção da árvore. Para atributos contínuos, o C4.5 determina um limiar e divide os dados em duas partições, criando ramos distintos na árvore. Assim como outros algoritmos de árvore de decisão, o C4.5 utiliza o método de dividir e conquistar: dado um conjunto S , se todos os exemplos pertencem à mesma classe ou se S é pequeno, o nó torna-se uma folha rotulada pela classe mais frequente. Caso contrário, seleciona-se o atributo que maximiza o ganho de informação normalizado (*Gain Ratio*) para particionar o conjunto em subconjuntos e aplica-se o mesmo procedimento recursivamente até que toda a árvore seja construída (SALZBERG, 1994).

Em termos práticos, segundo Hssina *et al.* (2014) o processo envolve inicialmente o cálculo da entropia do conjunto de dados, definido por:

$$Entropy(S) = - \sum_{i=1}^k p_i \log_2(p_i)$$

onde p_i representa a proporção de exemplos pertencentes à classe i em S . Em seguida, calcula-se o ganho de informação para cada atributo A :

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} \cdot Entropy(S_v)$$

e, por fim, o $C4.5$ utiliza o índice de divisão (*SplitInfo*) para normalizar o ganho:

$$SplitInfo(S, A) = - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} \log_2 \left(\frac{|S_v|}{|S|} \right)$$

resultando na razão de ganho (*Gain Ratio*), métrica utilizada para selecionar o melhor atributo de divisão:

$$GainRatio(S, A) = \frac{Gain(S, A)}{SplitInfo(S, A)}$$

No $C5.0$, a árvore de decisão também é construída a partir de um conjunto de dados de treinamento, porém com melhorias importantes em relação ao $C4.5$, como a incorporação do *boosting*, que combina diversas árvores menores para formar um classificador mais estável e com menor taxa de erro; suporte ampliado a novos tipos de dados, incluindo datas e valores não aplicáveis, além da possibilidade de definir custos distintos para diferentes tipos de erro de classificação, tornando o modelo mais adequado a cenários reais; geração de conjuntos de regras não ordenadas, permitindo avaliar todas as regras aplicáveis a cada instância e favorecendo modelos mais compactos e interpretáveis; e otimização para execução em ambientes com múltiplos núcleos, o que melhora significativamente a eficiência e a escalabilidade em bases de dados maiores (WU, X. *et al.*, 2008).

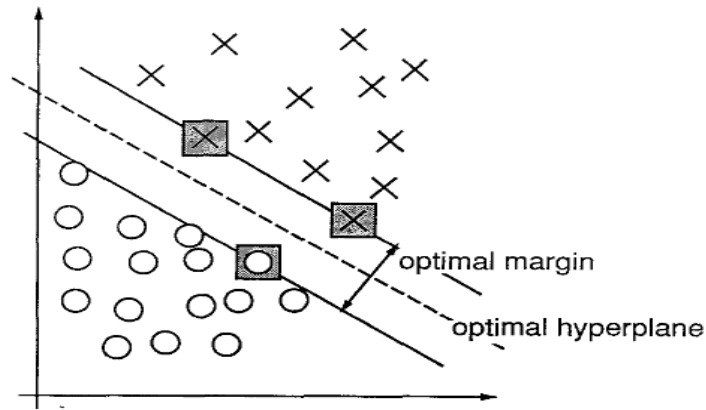
2.2.4 Support Vector Machine(SVM)

O SVM é um algoritmo de aprendizado supervisionado conhecido como classificador de “margem máxima”, pois busca reduzir o erro de generalização por meio da maximização da margem entre conjuntos de dados linearmente separáveis. Para determinar o hiperplano ótimo, o SVM resolve um problema de otimização do tipo *soft margin*, no qual o parâmetro de penalidade C , definido pelo usuário, estabelece um equilíbrio entre dois objetivos: ampliar a margem e minimizar a soma das variáveis de folga. Assim, C é interpretado como um parâmetro de regularização que controla o grau de tolerância a erros durante o treinamento (GUIDO *et al.*, 2024).

Como podem existir diversos hiperplanos capazes de separar linearmente as classes, o SVM seleciona aquele que maximiza a distância entre os exemplos de treinamento mais

próximos de cada classe e o hiperplano separador. Essa distância é denominada margem e, geometricamente, corresponde ao espaço mínimo entre os vetores de suporte e o hiperplano, conforme ilustrado na Figura 5. A escolha do hiperplano de maior margem não é arbitrária: essa estratégia tende a proporcionar melhor capacidade de generalização. Assim, além de apresentar bom desempenho nos dados utilizados durante o treinamento, o modelo possui maior probabilidade de manter precisão quando aplicado a novos dados que não fizeram parte do processo de aprendizagem (WU, X. *et al.*, 2008).

Figura 5 – Representação esquemática da margem e dos vetores de suporte.



Fonte: (CORTES; VAPNIK, 1995).

2.3 CLASSIFICAÇÃO MULTIRRÓTULO

A classificação multirrótulo é amplamente utilizada em aplicações modernas, como na categorização musical ou na classificação semântica de cenas (TSOUMAKAS; KATAKIS, 2009). Por definição, a classificação multirrótulo ocorre quando uma instância pode ser atribuída simultaneamente a vários rótulos, diferentemente da classificação monorrótulo, em que cada instância pertence a apenas uma categoria (GATTO, 2023). Formalmente, um conjunto de dados multirrótulo é definido como $\mathcal{D} = \{(x_i, Y_i) \mid 1 \leq i \leq m\}$, onde:

- m representa o número total de instâncias no conjunto de dados;
- cada instância i está associada a um vetor de atributos $x_i = \{x_{i1}, \dots, x_{id}\}$, onde d é o número de atributos;
- cada instância possui um subconjunto de rótulos $Y_i \subseteq L$;
- o espaço de rótulos é definido como $L = \{l_1, \dots, l_q\}$, com $q \geq 2$;
- $Y_i = \{l_{i1}, \dots, l_{ik}\}$ representa o conjunto de rótulos atribuídos à instância x_i .

conforme descrito em Silva *et al.* (2014).

Ao que se refere à construção do modelo, este deve ser capaz de prever, para cada nova instância, o conjunto de rótulos que melhor a descreve. Esse modelo pode gerar

predições de duas maneiras: atribuindo diretamente os rótulos relevantes (bipartição) ou ordenando todos os rótulos conhecidos de acordo com sua relevância. Na abordagem de bipartição, define-se uma função preditiva $h : X \rightarrow 2^L$, responsável por mapear uma instância x_i ao conjunto de rótulos previstos $\hat{Y}_i = h(x_i)$, onde:

- 2^L denota o conjunto de todos os possíveis subconjuntos de rótulos;
- $\hat{Y}_i = \{\hat{y}_1, \hat{y}_2, \dots, \hat{y}_p\}$ representa o conjunto de rótulos previstos para x_i ;
- $Y_i = \{y_1, \dots, y_k\}$ corresponde ao conjunto de rótulos verdadeiros associados à instância.

conforme descrito em Gatto (2023).

Na abordagem baseada em ranqueamento, o objetivo é aprender uma função $f : \mathcal{X} \times \mathcal{L} \rightarrow \mathcal{R}$ capaz de atribuir um grau de relevância a cada rótulo possível para uma instância x_i . Dessa forma, o modelo produz uma lista ordenada de rótulos, refletindo a probabilidade ou importância de cada um em relação à instância.

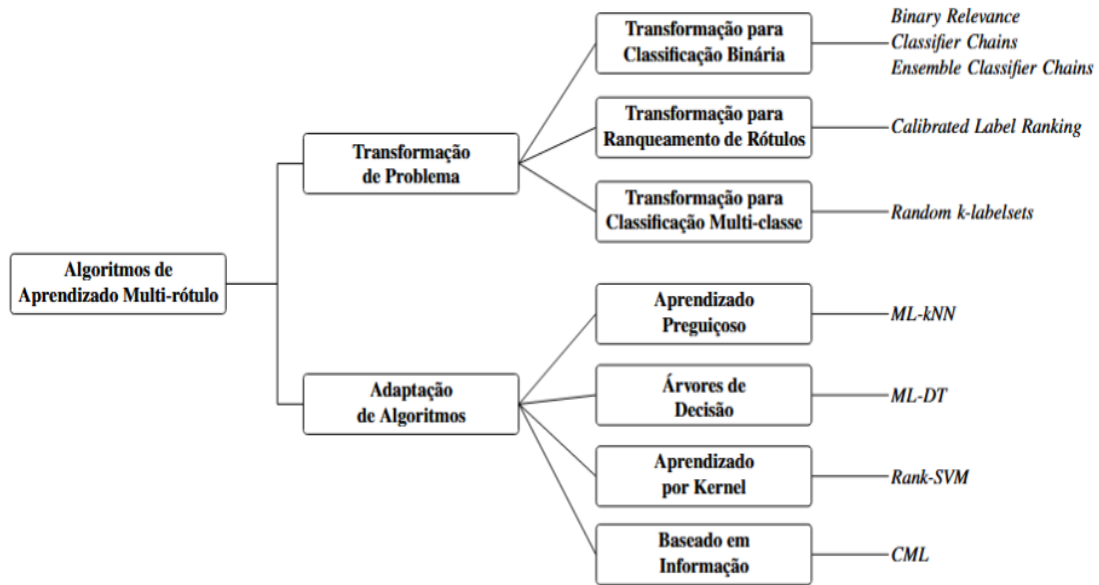
Esse ranqueamento pode ser obtido por diferentes estratégias, como métodos de votação ou ponderação. Em classificadores binários independentes, por exemplo, a saída de cada modelo pode ser usada como voto para compor a ordem final dos rótulos. Alternativamente, cada classificador pode gerar uma estimativa probabilística da relevância dos rótulos, consultando outros classificadores caso o valor esteja abaixo de um limiar. Diversas variações dessas abordagens são discutidas em estudos como (GIBAJA; VENTURA, 2014; MADJAROV *et al.*, 2012).

Tradicionalmente, os métodos de classificação multirrótulo podem ser divididos em duas categorias principais: adaptação de algoritmos e transformação de problema, conforme apresentado na Figura 6. Na abordagem de adaptação de algoritmos, novos algoritmos são desenvolvidos, ou algoritmos existentes são adaptados para resolver o problema multirrótulo original. Esses algoritmos tratam todos os rótulos do problema ao mesmo tempo, e treinam apenas um único classificador multirrótulos. No entanto informações individuais de cada rótulo que podem ser úteis para explorar diferentes padrões são ignorados (SILLA; FREITAS, 2011).

Entre os exemplos, destacam-se o *Multi-Label k-Nearest Neighbor* (ML-kNN), *Multi-Label Decision Tree* (ML-DT), *Ranking Support Vector Machine* (Rank-SVM) e *Collective Multi-Label Classifier* (CML). Todos esses algoritmos foram adaptados para lidar com classificação multi-rótulo (ZHANG; ZHOU, 2014).

Já na abordagem de transformação de problema, os métodos buscam converter o problema multirrótulo em um ou mais subproblemas mais simples, de forma que algoritmos de classificação convencionais possam ser aplicados (CARVALHO; FREITAS, 2009). Em muitos casos, essa transformação resulta em um conjunto de problemas binários, nos quais é necessário treinar um classificador independente para cada rótulo individualmente, como ocorre em algoritmos representativos como *Binary Relevance* (BR), *Classifier Chain* (CC) e *Ensemble of Classifier Chains* (ECC) (READ *et al.*, 2009).

Figura 6 – Estrutura de categorização dos algoritmos de aprendizado multi-rótulo.



Fonte: Elaborado pelo autor com base (ZHANG; ZHOU, 2014).

Entretanto, existem variações dessa abordagem que realizam a transformação de maneira diferente. Na transformação para classificação multiclasse, o problema multirrótulo é convertido em um único problema multiclasse, no qual cada classe representa uma combinação possível de rótulos, como ocorre no método *Label Powerset* (LP) e em sua variação *Random k-Labelsets* (RAkEL) (TSOUMAKAS; VLAHAVAS, Ioannis, 2007). Já na transformação para ranqueamento de rótulos, o objetivo é aprender uma ordenação de relevância entre os rótulos a partir de comparações entre pares, como implementado no *Calibrated Label Ranking* (GHAMRAWI; MCCALLUM, 2005).

Por fim, dentre os desafios da classificação multirrótulo, destacam-se a alta dimensionalidade do espaço de rótulos, especialmente quando o número de rótulos é muito grande e pode superar o número de atributos de entrada (TSOUMAKAS; KATAKIS; VLAHAVAS, I., 2008), o desbalanceamento, no qual a maioria dos rótulos possui apenas alguns exemplos positivos (MIRZA; ASGHAR; NOOR, 2019), e a complexidade em identificar e explorar dependências e correlações entre rótulos (CHANG *et al.*, 2019). Entretanto, estudos têm mostrado que o desempenho preditivo de classificadores multirrótulo pode ser melhorado quando tais correlações são exploradas e diversas abordagens têm sido propostas com esse objetivo.

2.3.1 *Binary Relevance* (BR)

O BR tem como ideia principal a decomposição do problema de classificação multirrótulo em n problemas de classificação binária, no qual neste tipo de classificação cada

problema corresponde apenas um rótulo possível (BOUTELL *et al.*, 2004). Ou seja, para cada rótulo λ_j , o BR deriva um conjunto de treinamento binário D_j a partir do conjunto multirrótulo original D , conforme a Equação (4).

$$D_j = \{(x_i, y_{ij}) \mid 1 \leq i \leq m\}. \quad (4)$$

Em outras palavras, cada exemplo multirrótulo (x_i, y_i) é transformado em um exemplo binário com base na relevância do rótulo λ_j para a instância x_i (ZHANG *et al.*, 2018). A Figura 7 ilustra esse processo de decomposição, no qual o conjunto multirrótulo original é dividido em três subconjuntos binários independentes (D_1 , D_2 , D_3), cada um correspondente a um rótulo específico (Y_1 , Y_2 , Y_3). Assim, o problema multirrótulo é transformado em múltiplos problemas de classificação binária, permitindo o uso de classificadores tradicionais para cada rótulo individualmente.

Figura 7 – Decomposição do problema multi-rótulo pelo BR.

X	Y₁	Y₂	Y₃
x_1	0	1	1
x_2	1	0	0
x_3	0	0	0
x_4	1	1	0
x_5	0	0	0

X	Y₁	X	Y₂	X	Y₃
x_1	0	x_1	1	x_1	1
x_2	1	x_2	0	x_2	0
x_3	0	x_3	0	x_3	0
x_4	1	x_4	1	x_4	0
x_5	0	x_5	0	x_5	0

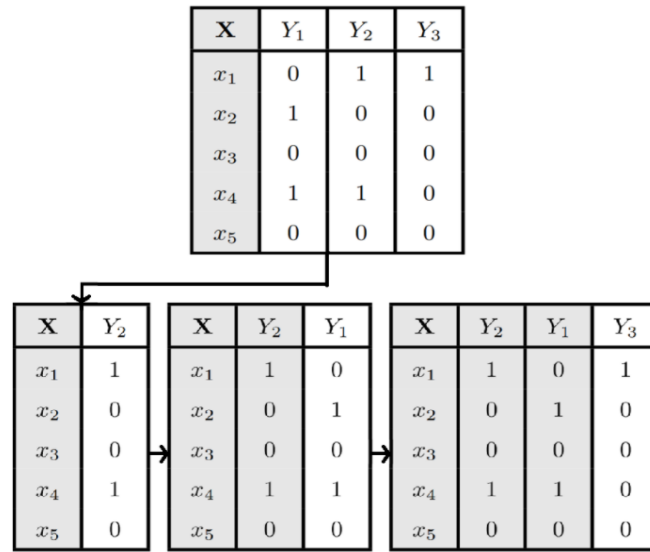
Fonte: Elaborado pelo autor.

Entretanto, como os rótulos são tratados individualmente pelo BR, as correlações existentes entre os rótulos não são modeladas, e isso impacta diretamente na generalização do modelo. Para contornar essa limitação, foram propostas diversas extensões do BR que incorporam dependências entre os rótulos, mantendo a complexidade linear em relação ao número de classes. Uma dessas extensões é *Chaining Structure* ou denominado *Classifier Chains* que encadeia classificadores binários (READ *et al.*, 2009).

2.3.2 Classifier Chain(CC)

O algoritmo CC é uma extensão do BR. A principal diferença é que, no CC, a saída prevista de cada classificador é usada como entrada adicional para o próximo da cadeia. Dessa forma, o método leva em conta que os rótulos podem estar relacionados entre si, ou seja, a presença de um rótulo pode influenciar a probabilidade de outro. Cada classificador h_j é treinado não apenas com as variáveis originais $\mathbf{x}$, mas também com as previsões dos rótulos anteriores $(y_1, y_2, \dots, y_{j-1})$, conforme mostrado na Figura 8 (READ *et al.*, 2009).

Figura 8 – Representação do processo de classificação no CC.



Fonte: Elaborado pelo autor.

Assim, o modelo aprende a estimar a probabilidade de cada rótulo y_j ser relevante, considerando as decisões anteriores, conforme mostra a Equação (5). Entretanto, o CC apresenta problemas de desempenho porque depende da seleção aleatória da ordem da cadeia. Se a escolha sequencial for subótima, a previsão pode ser afetada negativamente. Para mitigar esse problema foi desenvolvido o ECC, que cria m modelos de CC, cada um com uma cadeia aleatória diferente, construída a partir de uma fração de instâncias e atributos (READ *et al.*, 2019).

$$P(\mathbf{y} \mid \mathbf{x}) = P(y_1 \mid \mathbf{x}) \prod_{j=2}^L P(y_j \mid \mathbf{x}, y_1, \dots, y_{j-1}) \quad (5)$$

2.3.3 Ensemble Classifier Chain (ECC)

O ECC surgiu mediante a dificuldade relacionada a seleção aleatória do CC. Em suma, o ECC cria m modelos do CC, conforme apresentado na Figura 9, cada um com uma

cadeia diferente, construído a partir de uma fração de instâncias e atributos. A previsão final é baseada na média valores de confiança de cada rótulo obtido a partir desses modelos (MOYANO *et al.*, 2018).

Figura 9 – Arquitetura do ECC.

CC1				CC2				CC3			
X	Y ₁	Y ₂	Y ₃	X	Y ₂	Y ₃	Y ₁	X	Y ₃	Y ₁	Y ₂
x_1	0	1	1	x_1	1	1	0	x_1	1	0	1
x_2	1	0	0	x_2	0	0	1	x_2	0	1	0
x_3	0	0	0	x_3	0	0	0	x_3	0	0	0
x_4	1	1	0	x_4	1	0	1	x_4	0	1	1
x_5	0	0	0	x_5	0	0	0	x_5	0	0	0

Fonte: Elaborado pelo autor.

Segundo (READ *et al.*, 2009) o ECC combina as saídas de múltiplas cadeias de classificadores por meio de um vetor de confianças $\hat{\mathbf{w}} = [\hat{w}_1, \dots, \hat{w}_L]$, onde cada $\hat{w}_j$ indica o grau médio de confiança do j -ésimo rótulo, calculado conforme a Equação (6). Em seguida, aplica-se uma função de limiar $f_t(\cdot)$ para converter o vetor de confianças em uma predição binária de rótulos relevantes e irrelevantes, conforme a Equação (7).

$$\hat{w}_j = \frac{1}{m} \sum_{k=1}^m \hat{y}_{j,k} \quad (6)$$

$$\hat{\mathbf{y}} = f_t(\hat{\mathbf{w}}) \quad (7)$$

2.4 MÉTRICA PARA AVALIAÇÃO DA CLASSIFICAÇÃO MULTIRÓTULO

Ao construir um modelo de classificação, pode-se utilizar métricas de avaliação como *Accuracy*, *Recall*, *Precision* e *F1-score*. Contudo, na classificação multi-rótulo a avaliação se torna mais complexa, pois cada exemplo pode estar associado simultaneamente a múltiplos rótulos e não apenas a um. Em vista disso, foram desenvolvidas métricas específicas para o aprendizado multirrótulo, as quais são divididas em duas categorias: métricas baseadas em exemplos e métricas baseadas em rótulos, incluindo também métricas de *ranking* (ZHANG; ZHOU, 2014).

2.4.1 Métricas Baseadas em Exemplos

As métricas baseadas em exemplos, segundo Krstinić *et al.* (2020), são amplamente utilizadas na avaliação de modelos de classificação multirrótulo. Elas consideram o desempenho do classificador individualmente para cada instância e, em seguida, calculam a média sobre todo o conjunto de testes. Dessa forma, avaliam a taxa média de acertos e erros de um modelo de maneira equilibrada, independentemente da quantidade de rótulos por instância. As principais métricas desse grupo incluem a *Precision*, o *Recall* e a *Accuracy*, além de outras derivadas, como o *F1-score*, a *Hamming-Loss* e a *Subset-Accuracy*.

A *Precision* representa a proporção de rótulos corretamente previstos em relação ao total de rótulos previstos. Já o *Recall* mede a proporção de rótulos verdadeiros que foram corretamente recuperados pelo modelo. O *F1-score* corresponde à média harmônica entre *Precision* e *Recall*, fornecendo uma métrica equilibrada entre ambas. A *Hamming-Loss* quantifica a fração média de rótulos incorretamente classificados, enquanto a *Subset-Accuracy* indica a proporção de instâncias para as quais o conjunto de rótulos previstos coincide exatamente com o conjunto verdadeiro. Por fim, a *Accuracy* mede a sobreposição média entre os conjuntos de rótulos verdadeiros e previstos, sendo mais tolerante a erros parciais. As Equações (8) a (13) apresentam formalmente o cálculo dessas métricas.

$$\text{Precision} = \frac{1}{N} \sum_{i=1}^N \frac{|Y_i \cap \hat{Y}_i|}{|\hat{Y}_i|} \quad (8)$$

$$\text{Recall} = \frac{1}{N} \sum_{i=1}^N \frac{|Y_i \cap \hat{Y}_i|}{|Y_i|} \quad (9)$$

$$F_1 = \frac{1}{N} \sum_{i=1}^N \frac{2|Y_i \cap \hat{Y}_i|}{|\hat{Y}_i| + |Y_i|} \quad (10)$$

$$\text{Hamming-Loss} = \frac{1}{N} \sum_{i=1}^N \frac{|Y_i \Delta \hat{Y}_i|}{L} \quad (11)$$

$$\text{Subset-Accuracy} = \frac{1}{N} \sum_{i=1}^N \mathbb{I}(\hat{Y}_i = Y_i) \quad (12)$$

$$\text{Accuracy} = \frac{1}{N} \sum_{i=1}^N \frac{|Y_i \cap \hat{Y}_i|}{|Y_i \cup \hat{Y}_i|} \quad (13)$$

onde:

- Y_i representa o conjunto de rótulos verdadeiros da instância i ;
- $\hat{Y}_i$ representa o conjunto de rótulos previstos para a instância i ;
- N é o número total de instâncias do conjunto de dados;
- L é o número total de rótulos possíveis;

- Δ denota a diferença simétrica entre conjuntos;
- $\mathbb{I}(\cdot)$ é a função indicadora, que retorna 1 se a condição for verdadeira e 0 caso contrário.

2.4.2 Métricas Baseadas em Rótulos

As métricas baseadas em rótulos, conforme descrito por Zhang e Zhou (2014), avaliam o desempenho do modelo considerando cada classe individualmente, tratando o problema multirrótulo como uma sequência de classificações binárias para cada rótulo. Nesse contexto, o desempenho de cada rótulo pode ser descrito por quatro possíveis resultados: *True Positive* (TP), *False Positive* (FP), *True Negative* (TN) e *False Negative* (FN). Essas medidas indicam, respectivamente, quando o modelo acerta ou erra ao prever a presença ou ausência de determinado rótulo.

Para o rótulo y_j , correspondente à classe j -ésima, podem ser definidas quatro quantidades fundamentais que caracterizam o desempenho da classificação binária associada a esse rótulo, com base na função preditora $h(\cdot)$:

- $TP_j = |\{X_i \mid y_j \in \mathcal{Y}_i \wedge y_j \in h_{\text{Multi-label Classification}(MLC)}(X_i), 1 \leq i \leq m\}|$
- $FP_j = |\{X_i \mid y_j \notin \mathcal{Y}_i \wedge y_j \in h_{MLC}(X_i), 1 \leq i \leq m\}|$
- $TN_j = |\{X_i \mid y_j \notin \mathcal{Y}_i \wedge y_j \notin h_{MLC}(X_i), 1 \leq i \leq m\}|$
- $FN_j = |\{X_i \mid y_j \in \mathcal{Y}_i \wedge y_j \notin h_{MLC}(X_i), 1 \leq i \leq m\}|$

A partir dessas quatro quantidades, diversas métricas binárias podem ser derivadas para cada rótulo, como *Accuracy*, *Precision*, *Recall* e F_β -score. O desempenho global do modelo pode, então, ser agregado de duas formas principais:

- *Macro-averaging*: calcula a métrica para cada rótulo e, em seguida, obtém a média aritmética entre todas as classes:

$$B_{\text{macro}}(h) = \frac{1}{q} \sum_{j=1}^q B(TP_j, FP_j, TN_j, FN_j)$$

- *Micro-averaging*: soma todas as quantidades (TP, FP, TN, FN) antes de calcular a métrica:

$$B_{\text{micro}}(h) = B\left(\sum_{j=1}^q TP_j, \sum_{j=1}^q FP_j, \sum_{j=1}^q TN_j, \sum_{j=1}^q FN_j\right)$$

Conceitualmente, o *macro-averaging* atribui pesos iguais a todos os rótulos, enquanto o *micro-averaging* atribui pesos iguais a todas as instâncias. Ambas as abordagens são amplamente utilizadas para avaliar classificadores multirrótulo, sendo que valores mais altos indicam melhor desempenho do modelo. As métricas macro são apresentadas nas Equações (14) a (16), enquanto as métricas micro são mostradas nas Equações (17) a (19).

$$\text{Macro-Precision} = \frac{1}{q} \sum_{j=1}^q \frac{TP_j}{TP_j + FP_j} \quad (14)$$

$$\text{Macro-Recall} = \frac{1}{q} \sum_{j=1}^q \frac{TP_j}{TP_j + FN_j} \quad (15)$$

$$\text{Macro-F1} = \frac{2 \times \text{Macro-Precision} \times \text{Macro-Recall}}{\text{Macro-Precision} + \text{Macro-Recall}} \quad (16)$$

$$\text{Micro-Precision} = \frac{\sum_{j=1}^q TP_j}{\sum_{j=1}^q TP_j + \sum_{j=1}^q FP_j} \quad (17)$$

$$\text{Micro-Recall} = \frac{\sum_{j=1}^q TP_j}{\sum_{j=1}^q TP_j + \sum_{j=1}^q FN_j} \quad (18)$$

$$\text{Micro-F1} = \frac{2 \times \text{Micro-Precision} \times \text{Micro-Recall}}{\text{Micro-Precision} + \text{Micro-Recall}} \quad (19)$$

2.4.3 Métricas Baseadas em Ranking

Além dessas medidas, destacam-se as métricas *Macro-AUC*, *Micro-AUC*, *Macro-AUPRC* e *Micro-AUPRC*, utilizadas para avaliar classificadores multirrótulo com saídas probabilísticas ou contínuas. As métricas baseadas na AUC derivam do conceito de *Area Under the Receiver Operating Characteristic Curve*, que quantifica a capacidade do modelo em ranquear instâncias positivas acima das negativas. Enquanto a *Macro-AUC* reflete o desempenho médio equilibrado entre os rótulos, a *Micro-AUC* avalia o desempenho global considerando conjuntamente todos os pares positivo–negativo. Valores próximos de 1 indicam ranqueamento quase perfeito, enquanto valores próximos de 0,5 indicam desempenho aleatório (WU, X.-Z.; ZHOU, 2017).

A *Macro-AUC* é calculada individualmente para cada rótulo e, em seguida, obtém-se a média entre todas as classes, atribuindo peso igual a cada uma, conforme apresentado na Equação (20). No termo $\mathcal{S}_j^{\text{macro}} = \{(a, b) \in Y_{\cdot j}^+ \times Y_{\cdot j}^- \mid f_j(x_a) \geq f_j(x_b)\}$, estão os pares de instâncias corretamente ordenados pelo modelo para o rótulo j . Em outras palavras, $\mathcal{S}_j^{\text{macro}}$ é o conjunto de pares de instâncias (a, b) em que a instância positiva a é corretamente classificada acima da instância negativa b , ou seja, a previsão do modelo para a instância positiva é maior do que para a instância negativa.

$$\text{Macro-AUC} = \frac{1}{q} \sum_{j=1}^q \frac{|\mathcal{S}_j^{\text{macro}}|}{|Y_{\cdot j}^+| |Y_{\cdot j}^-|}. \quad (20)$$

No caso da *Micro-AUC*, todas as instâncias e rótulos são considerados simultaneamente. Em vez de calcular a AUC por classe, somam-se todos os pares positivo–negativo

e todos os pares corretamente ordenados, conforme a Equação (21). Assim, $\mathcal{S}^{\text{micro}} = \bigcup_{j=1}^q \mathcal{S}_j^{\text{macro}}$ representa o conjunto global de pares ordenados corretamente.

$$\text{Micro-AUC} = \frac{|\mathcal{S}^{\text{micro}}|}{\sum_{j=1}^q |Y_{\cdot,j}^+| |Y_{\cdot,j}^-|}. \quad (21)$$

Já no caso das métricas baseadas em *Area Under the Precision–Recall Curve* (AUPRC), o objetivo é avaliar o desempenho do modelo considerando a relação entre *Precision* e *Recall*. Essas métricas são particularmente relevantes em cenários desbalanceados, nos quais as classes positivas são raras (DAVIS; GOADRIC, 2006).

Para a *Macro-AUPRC*, calcula-se a área sob a curva *Precision–Recall* (PR) de cada rótulo individualmente e, em seguida, obtém-se a média entre as classes, atribuindo peso igual a cada uma delas, conforme a Equação (22). No termo $P_j(R)$, tem-se a função de *Precision* em função do *Recall* para o rótulo j , e q representa o número total de rótulos. Essa métrica reflete um desempenho equilibrado entre as classes, independentemente do desbalanceamento presente em cada rótulo (YANG, 1999).

$$\text{Macro-AUPRC} = \frac{1}{q} \sum_{j=1}^q \int_0^1 P_j(R) dR. \quad (22)$$

Já a *Micro-AUPRC* agrega todas as instâncias e rótulos antes da construção da curva PR, avaliando o desempenho global ponderado pela frequência das classes. Nesse caso, *Precision* e *Recall* são calculados utilizando o total combinado de TP, FP e FN ao longo de todos os rótulos, como apresentado na Equação (24).

$$\text{Micro-AUPRC} = \int_0^1 \frac{\sum_{j=1}^q TP_j(R)}{\sum_{j=1}^q (TP_j(R) + FP_j(R))} dR. \quad (23)$$

$$\text{Micro-AUPRC} = \int_0^1 \frac{\sum_{j=1}^q TP_j(R)}{\sum_{j=1}^q (TP_j(R) + FP_j(R))} dR. \quad (24)$$

Valores elevados de AUPRC indicam que o modelo mantém alta *Precision* mesmo com o aumento do *Recall*, evidenciando uma boa capacidade de identificar corretamente instâncias positivas em cenários com forte desbalanceamento (SAITO; REHMSMEIER, 2015; BOYD; ENG; PAGE, 2013).

2.5 SIMILARIDADE DE JACCARD

A similaridade de *Jaccard* é uma medida estatística que quantifica o grau de semelhança entre dois conjuntos finitos, considerando apenas a ocorrência simultânea de elementos (presenças) em ambos. Originalmente proposta por Paul Jaccard no artigo de 1901 intitulado *Étude comparative de la distribution florale dans une portion des Alpes et du Jura* (JACCARD, 1901), a métrica foi empregada para comparar a composição de espécies

entre diferentes regiões ecológicas, buscando refletir o quanto duas amostras compartilham características em comum, desconsiderando a ausência conjunta de elementos.

Quando se tratam de atributos binários assimétricos, de acordo com Albuquerque *et al.* (2016), a concordância de dois valores iguais a 1 (correspondência positiva) é considerada mais relevante do que a concordância de dois valores iguais a 0 (correspondência negativa). Assim, a similaridade baseada nesses atributos é chamada de *similaridade não invariante*, pois ignora o número de correspondências negativas. Nesse contexto, o coeficiente de *Jaccard* se destaca exatamente por não considerar a ausência simultânea de atributos como evidência de similaridade.

A Equação (25) apresenta o cálculo do coeficiente de similaridade de *Jaccard*, definido como a razão entre o número de elementos presentes na interseção e o número total de elementos presentes na união dos conjuntos A e B . O coeficiente também pode ser expresso considerando probabilidades ou frequências relativas, em que $P(A)$ e $P(B)$ representam as probabilidades de ocorrência dos elementos nos conjuntos A e B , respectivamente, e $P(A \cap B)$ representa a probabilidade de ocorrência simultânea.

$$J_{ac} = \frac{|A \cap B|}{|A \cup B|} = \frac{P(A \cap B)}{P(A) + P(B) - P(A \cap B)} = \frac{p_1 p_2}{p_1 + p_2 - p_1 p_2} = \frac{a}{a + b + c} \quad (25)$$

onde, segundo Rinjeni, Indriawan e Rakhmawati (2024):

- a representa o número de atributos em que ambos os objetos possuem valor 1 (concordância positiva);
- b representa o número de atributos em que apenas o primeiro objeto possui valor 1;
- c representa o número de atributos em que apenas o segundo objeto possui valor 1;
- correspondências negativas (0,0) são ignoradas no cálculo.

3 PROJETO E IMPLEMENTAÇÃO

Este capítulo descreve o processo de implementação do método proposto para a criação de *metalabels* no contexto do algoritmo ECC. Serão apresentados os detalhes da construção do modelo, incluindo a escolha dos parâmetros e as adaptações necessárias para o processamento dos dados, com o objetivo de melhorar o desempenho do modelo em tarefas de classificação multirrótulo.

A proposta visa avaliar a inserção controlada de metalabels e investigar diferentes formas de integração desses rótulos ao modelo. O capítulo também discute as estratégias de seleção e a análise da ordem dos rótulos dentro da cadeia, com o intuito de explorar como essas modificações afetam o desempenho do algoritmo ECC.

3.1 ESTRUTURA DO MÉTODO PROPOSTO

O método proposto consiste em utilizar a base do algoritmo ECC, apresentado no capítulo anterior, adicionando novos rótulos (*labels*) ao CC, denominados *metalabels*. As *metalabels* são formadas pela combinação de dois rótulos originais do conjunto de dados, de modo que cada instância do *metalabel* depende da relação binária entre os valores dos rótulos originais. Para exemplificar o método, podem ser analisadas as Tabelas 1 a 3, nas quais, além dos dois rótulos originais, há um *metalabel* criado a partir da combinação de Y_1 e Y_2 . Observa-se que os valores das instâncias de $MY_1_Y_2$ são diretamente influenciados pelos rótulos originais. Nesse contexto, três lógicas podem ser utilizadas para a formação das *metalabels*: AND, OR e XNOR.

X	Y_1	Y_2	$MY_1_Y_2$
x_1	0	1	0
x_2	1	0	0
x_3	0	0	0
x_4	1	1	1
x_5	0	0	0

Tabela 1 – Predição de Y_1 , Y_2 e *metalabel* formado apartir da lógica AND

No caso da lógica AND, apresentada na Tabela 1, quando os valores dos rótulos originais forem $(0, 1)$, $(1, 0)$ ou $(0, 0)$, o *metalabel* assume valor 0. Apenas quando ambos forem iguais a 1 o *metalabel* assume valor 1. Considerando dois rótulos originais r_a e r_b , os respectivos valores das instâncias podem ser representados como vetores binários y_{ia} e y_{ib} , onde i é o índice da instância. O *metalabel*, denotado por μ_{ab} , é então definido pela operação lógica $\wedge$, conforme a Equação (26).

$$\mu_{ab}(x_i) = \begin{cases} 1 & \text{se } y_{ia} = 1 \text{ e } y_{ib} = 1, \\ 0 & \text{caso contrário.} \end{cases} \quad (26)$$

A lógica OR, apresentada na Tabela 2, atribui valor 1 ao *metatlabel* sempre que pelo menos um dos rótulos originais possuir valor igual a 1. Caso ambos sejam iguais a 0, o *metatlabel* assume valor 0.

X	Y_1	Y_2	$MY_1_Y_2$
x_1	0	1	1
x_2	1	0	1
x_3	0	0	0
x_4	1	1	1
x_5	0	0	0

Tabela 2 – Predição de Y_1 , Y_2 e *metatlabel* formado apartir da lógica OR

$$\mu_{ab}(x_i) = \begin{cases} 1, & \text{se } y_{ia} = 1 \text{ ou } y_{ib} = 1, \\ 0, & \text{caso contrário.} \end{cases} \quad (27)$$

Por fim, na lógica XNOR, o *metatlabel* assume valor 1 quando ambos os rótulos originais apresentam o mesmo valor (ambos 0 ou ambos 1). Caso os valores sejam diferentes, o *metatlabel* assume valor 0, conforme mostrado na Tabela 3.

X	Y_1	Y_2	$MY_1_Y_2$
x_1	0	1	0
x_2	1	0	0
x_3	0	0	1
x_4	1	1	1
x_5	0	0	1

Tabela 3 – Predição de Y_1 , Y_2 e *metatlabel* formado apartir da lógica XNOR

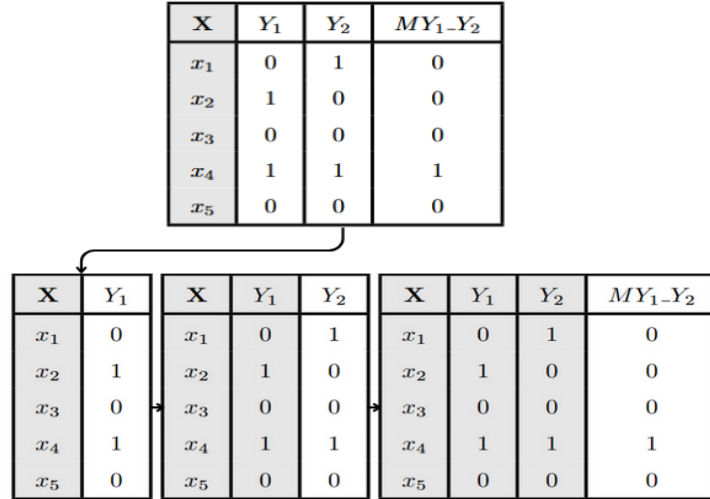
$$\mu_{ab}(x_i) = \begin{cases} 1, & \text{se } y_{ia} = y_{ib}, \\ 0, & \text{caso contrário.} \end{cases} \quad (28)$$

Os *metatlabels* são incorporados à cadeia juntamente com os rótulos originais, conforme ilustrado na Figura 10, com o objetivo de avaliar seu impacto no desempenho do

método ECC. Entretanto, dois aspectos críticos devem ser considerados na sua construção. O primeiro refere-se à predominância de valores zero nos *metalabels*, decorrente da baixa densidade e cardinalidade do conjunto de dados. Quando há concentração excessiva de zeros, isso indica fraca correlação entre os rótulos originais, reduzindo a utilidade informacional do *metalabel*.

O segundo aspecto diz respeito ao número potencialmente elevado de *metalabels* gerados a partir dos rótulos originais. Em conjuntos de dados com muitos rótulos, a quantidade de combinações possíveis pode alcançar 2^q , onde q representa o número de rótulos no conjunto de dados, o que eleva substancialmente a complexidade e os custos computacionais. Por outro lado, em conjuntos com poucos rótulos, pode não haver correlação suficiente para justificar a criação de *metalabels*. Experimentos realizados pelo autor indicam que, quando o número de rótulos é inferior a dez, podem ocorrer problemas devido à falta de variabilidade e dependência adequadas, o que compromete a formação de *metalabels* informativos. Contudo, esse número pode variar de acordo com a cardinalidade dos conjuntos de dados utilizados, sendo possível ajustar o valor mínimo conforme as características específicas de cada conjunto.

Figura 10 – Processo de classificação após criação e inserção do *metalabels*.



Fonte: Elaborado pelo autor.

3.2 APLICANDO SIMILARIDADE DE JACCARD

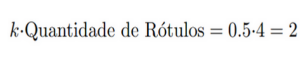
Para mitigar o problema mencionado na Seção 3.1 na criação dos *metalabels*, utilizou-se a similaridade de Jaccard (JACCARD, 1901) em conjunto com o ranqueamento dos rótulos. Inicialmente, conforme apresentado no capítulo anterior, a similaridade de Jaccard pode ser aplicada ao contexto de classificação multirrótulo para quantificar

a correlação entre dois rótulos. Ao considerar os rótulos r_a e r_b e aplicar a similaridade, obtém-se a Equação (29). Essa métrica permite determinar o grau de proximidade entre os rótulos originais, possibilitando a criação de *metalabels* apenas para aqueles que apresentam maior correlação.

$$J(l_a, l_b) = \frac{|\{i \mid l_a \in Y_i \wedge l_b \in Y_i\}|}{|\{i \mid l_a \in Y_i\} \cup \{i \mid l_b \in Y_i\}|} \quad (29)$$

Já para selecionar os melhores rótulos que serão utilizados para gerar os *metalabels*, utiliza-se um fator multiplicativo (k), definido como um valor flutuante aplicado ao número original de rótulos. Quando multiplicado, esse fator indica a quantidade de *metalabels* que será gerada e inserida no conjunto de dados. O valor de k é definido manualmente conforme a necessidade do modelo. Valores menores de k geram menos *metalabels*, enquanto valores maiores geram mais, podendo ultrapassar o número total de rótulos disponíveis. Quando k é maior que 1, a quantidade de *metalabels* começa a aumentar a complexidade computacional devido ao maior número de combinações geradas. Uma vez determinado o número de *metalabels* a serem criados, realiza-se uma busca simples pelos rótulos mais correlacionados, considerando os valores de similaridade de Jaccard, a fim de identificar quais rótulos serão utilizados na criação dos *metalabels*, conforme ilustrado na Figura 11.

Figura 11 – Seleção dos melhores rótulos para criação do *metalabel*.



Rótulos Relacionados	Similaridade de Jaccard
$\{Y_1, Y_2\}$	0.72
$\{Y_1, Y_3\}$	0.48
$\{Y_1, Y_4\}$	0.18
$\{Y_2, Y_3\}$	0.45
$\{Y_2, Y_4\}$	0.22
$\{Y_3, Y_4\}$	0.12

Fonte: Elaborado pelo autor

Finalmente, um modelo pode ser criado utilizando os rótulos originais em conjunto com os *metalabels* gerados. Após a realização da predição no conjunto de teste, as previsões correspondentes aos *metalabels* são removidas do resultado final. Essa versão do método, na qual os *metalabels* são utilizados apenas durante o treinamento e totalmente descartados após a predição, é denominada Sem *Metalabel Tail Block* (SMLTB).

3.3 MÉTODOS RESULTANTES

A metodologia desenvolvida apresenta duas versões distintas. A primeira segue exatamente o processo descrito anteriormente, no qual os *metalabels* são utilizados apenas durante o treinamento e, posteriormente, removidos das predições. Já a segunda versão introduz um passo adicional antes da remoção dos *metalabels*. Nessa abordagem, verifica-se se a probabilidade atribuída a um determinado *metalabel* é maior que as probabilidades dos dois rótulos originais que o compõem. Caso a probabilidade do *metalabel* seja superior à probabilidade de ambos os rótulos originais, o valor do *metalabel* é propagado para esses dois rótulos. Se apenas um dos rótulos apresentar probabilidade inferior à do *metalabel*, o valor é atribuído somente a ele. Essa versão modificada, que incorpora esse passo de correção baseado em probabilidades, é denominada Correção Sem *Metalabel Tail Block* (CSMLTB). Em teoria, a probabilidade atribuída aos rótulos originais deveria ser igual ou superior à probabilidade atribuída ao *metalabel*; entretanto, conforme será discutido posteriormente, essa condição nem sempre é observada.

Além das duas versões originalmente propostas, também se investigou o impacto da posição dos *metalabels* dentro da cadeia de rótulos, uma vez que classificadores do tipo CC são sensíveis à ordem em que os rótulos são apresentados. Como, em CC, cada rótulo previsto serve de entrada para a predição do próximo, alterar a sequência pode modificar as dependências consideradas pelo modelo e, conseqüentemente, os resultados obtidos.

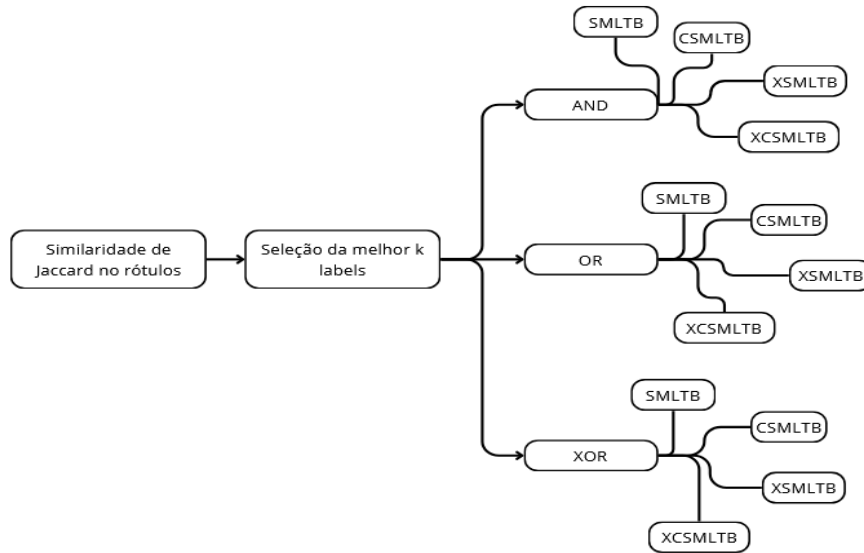
Nesse contexto, além das versões em que os *metalabels* são inseridos ao final do conjunto de rótulos (SMLTB e CSMLTB), foram desenvolvidas as variantes Sem *Metalabel Tail Block Exclusive* (XSMLTB) e Correção Sem *Metalabel Tail Block Exclusive* (CXSMMLTB), nas quais os *metalabels* são deslocados para o início da cadeia. Nessas siglas, o prefixo “X” é utilizado por analogia ao comportamento exclusivo presente em operadores lógicos, como o XNOR (NÃO exclusivo), indicando uma variação estrutural em relação às versões originais. A ideia por trás dessas versões é avaliar se a antecipação dos *metalabels* e, portanto, sua influência direta sobre todos os rótulos subsequentes pode modificar o processo de propagação das predições dentro da cadeia. A Figura 12 apresenta, de forma integrada, a lógica das quatro abordagens, destacando como a simples mudança na posição dos *metalabels* gera fluxos distintos de predição e, assim, permite analisar de maneira mais abrangente o efeito da ordem na modelagem em CC.

3.4 IMPLEMENTAÇÃO DO MÉTODO PROPOSTO

A implementação do método proposto foi realizada no software R, utilizando as bibliotecas `utiml` e `proxy`. Em conjunto com o R, empregou-se *shell script* para automatizar a execução do algoritmo nos 19 conjuntos de dados apresentados na Tabela 3, cujos códigos estão disponíveis em: <https://github.com/spttleonardo/Augmented-Chains-ECC.git>. As execuções foram realizadas em um ambiente composto por processadores Intel(R) Xeon(R)

Platinum 8480C, cada um contendo 56 núcleos físicos (112 threads) a 2,9 GHz, e 2 TB de memória RAM. Ressalta-se que o tempo de processamento dos algoritmos não foi considerado na análise.

Figura 12 – Estrutura final da metodologia.



Fonte: Elaborado pelo autor.

3.4.1 Seleção das *Metalabels*

Conforme mencionado anteriormente, um dos principais desafios do CC está relacionado ao tamanho do espaço de rótulos. Dessa forma, não é viável criar *metalabels* diretamente a partir de todos os rótulos originais, tanto devido ao problema do aumento exponencial da dimensionalidade quanto pela questão da cardinalidade. Em outras palavras, além de expandir ainda mais o espaço de rótulos, corre-se o risco de gerar *metalabels* que não apresentam correlação significativa entre si, tornando o modelo menos eficiente.

Na Figura 11, é ilustrado o processo utilizado para controlar a criação dessas *metalabels*. Nesse procedimento, define-se um fator multiplicativo (0,3, 0,5 ou 0,8, escolhidos com base em experimentos realizados neste trabalho para balancear a quantidade e precisão das *metalabels*), que, ao ser aplicado sobre o número total de rótulos, determina a quantidade de *metalabels* a serem geradas. Em seguida, é construída uma matriz de similaridade entre os rótulos originais, a partir da qual são selecionados aqueles com maior correlação. Por fim, realiza-se o processo de seleção das *metalabels* mais adequadas.

No Quadro 1, apresenta-se o trecho de código responsável por esse procedimento, incluindo o cálculo da quantidade de *k-rótulos*, a criação da matriz de similaridade por meio da função `cria_matriz_sim()`, e, posteriormente, a seleção das melhores *metalabels*.

por meio da função `verifica_best_metas()`, que recebe como parâmetros a matriz de similaridade e o número de rótulos selecionados.

Quadro 1 – Trecho de Código da Implementação das k-labels e Criação da Matriz de Similaridade.

```

1      k_labels <- as.integer(ncol(dslab) * nfactor)
2
3      # Criando a matriz de similaridade
4      matriz_sim <- cria_matriz_sim(dslab)
5
6      # Selecionando as melhores metalabels
7      best_metas <- verifica_best_metas(matriz_sim, k_labels)

```

Fonte: Autor, 2024.

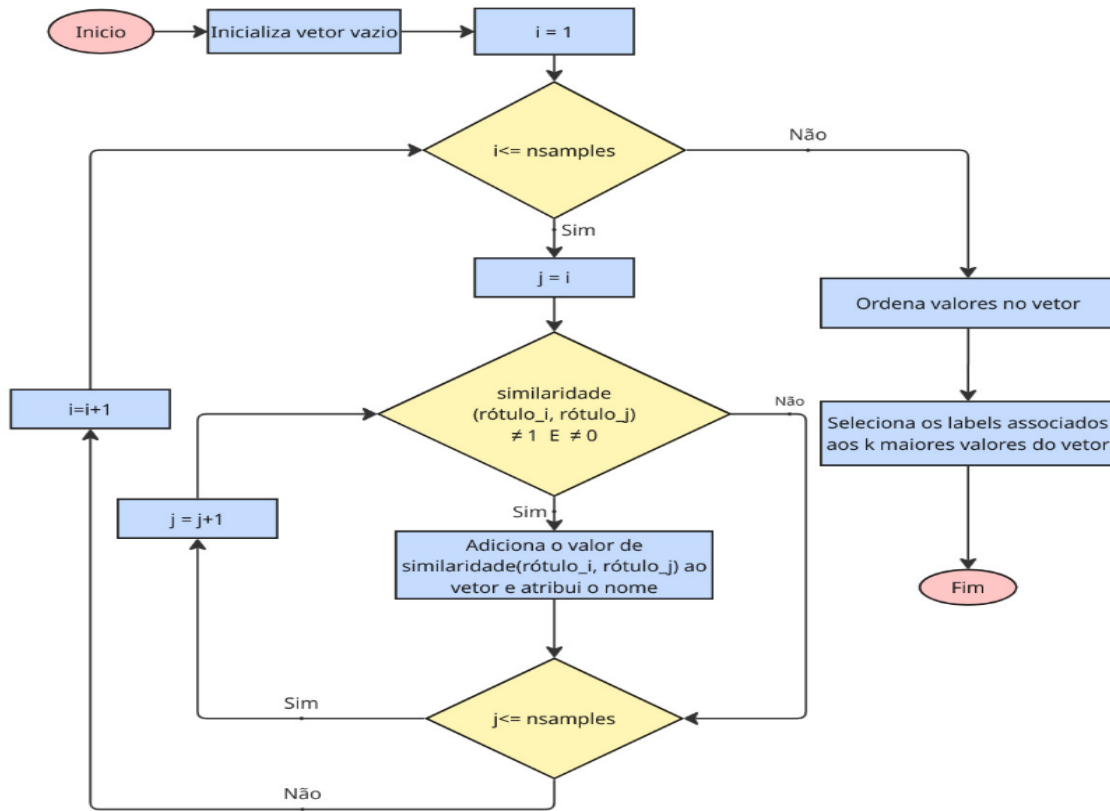
Além disso, no Quadro 1 é possível observar que a similaridade de Jaccard é utilizada para criar uma matriz de similaridade. Essa matriz, por sua vez, armazena em cada índice o grau de correlação entre os rótulos originais. Um exemplo desse cálculo é apresentado na Figura 13, considerando os seguintes rótulos: $l_1 = (1, 0, 1, 0, 1)$, $l_2 = (1, 1, 0, 0, 1)$ e $l_3 = (1, 0, 1, 0, 1)$. Nessa estrutura, verifica-se que os rótulos l_1 e l_3 são idênticos e, portanto, o valor resultante da similaridade é igual a 1, indicando a máxima semelhança entre eles.

Figura 13 – Exemplo de matriz de similaridade.

	l1	l2	l3
l1	0.0	0.5	1.0
l2	0.5	0.0	0.5
l3	1.0	0.5	0.0

Fonte: Elaborado pelo autor.

Já em relação à seleção dos melhores rótulos, é possível compreender a lógica do algoritmo por meio do fluxograma apresentado na Figura 14. A partir da figura, nota-se a existência de dois laços: o laço externo, responsável por percorrer as linhas, e o laço interno, que percorre as colunas, representados pelas variáveis i e j , respectivamente. Para cada par (i, j) , caso o valor da similaridade na posição $[i, j]$ seja diferente de 0 e de 1, esse valor é adicionado a um vetor. Ao final da execução dos laços, o vetor é ordenado de forma decrescente, possibilitando a seleção dos k rótulos mais similares, ou seja, daqueles associados aos maiores valores de similaridade.

Figura 14 – Fluxograma do funcionamento da função `verifica_best_metas`.

Fonte: Elaborado pelo autor.

3.4.2 Atribuição de Probabilidade nos Rótulos Originais

Conforme elucidado anteriormente na Seção 3.3, a atribuição de probabilidade nos dois rótulos originais, ou em apenas um deles, é realizada caso o valor de probabilidade do *metabel* seja maior, e somente se essa condição for atendida. Através das Tabelas 4 e 5, é possível visualizar um exemplo hipotético, construído apenas para fins de explicação, que demonstra de forma clara a aplicação dessa regra de propagação aos rótulos originais.

Ao analisar a Tabela 4, observa-se que, em determinadas instâncias, o valor de probabilidade atribuído ao *metabel* é superior às probabilidades dos rótulos que o compõem. Nesses casos, conforme descrito pela metodologia, o valor do *metabel* é propagado para ambos os rótulos originais (Instância 1 para o *metabel* 1-2) ou apenas para aquele que apresenta probabilidade inferior (Instância 2 para o *metabel* 3-4). Em situações nas quais o *metabel* não apresenta probabilidade superior a nenhum dos seus rótulos correspondentes, nenhuma atualização é realizada, como ocorre parcialmente na Instância 3. A Tabela 5 apresenta os valores após a aplicação da regra de propagação. É possível verificar claramente o efeito da metodologia: valores em negrito indicam a substituição da probabilidade do rótulo original pela probabilidade do *metabel*, nos casos em que esta

última é superior.

Tabela 4 – Tabela de probabilidades antes da regra de propagação.

Instância	1-2	3-4	L1	L2	L3	L4
1	0.8	0.2	0.3	0.5	0.3	0.6
2	0.4	0.7	0.4	0.6	0.8	0.4
3	0.6	0.5	0.7	0.3	0.6	0.8

Tabela 5 – Tabela de probabilidades depois da regra de propagação.

Instância	1-2	3-4	L1	L2	L3	L4
1	0.8	0.2	0.8	0.8	0.3	0.6
2	0.4	0.7	0.4	0.6	0.8	0.7
3	0.6	0.5	0.7	0.6	0.6	0.8

Já na Figura 15, é possível verificar o fluxograma que representa o funcionamento da função `verifica_pred()`, a qual recebe como parâmetros a matriz de predição contendo as probabilidades dos *metalabels* gerados na chain, tanto no início quanto no final, além das melhores *metalabels* selecionadas e dos rótulos originais.

No fluxograma, observa-se que inicialmente são extraídos apenas os *metalabels* que de fato estão presentes na tabela de predição. Esse passo é necessário porque nem todos os *metalabels* gerados previamente aparecem como colunas no conjunto de predições e, portanto, é preciso identificar quais deles são válidos para o processo de atualização.

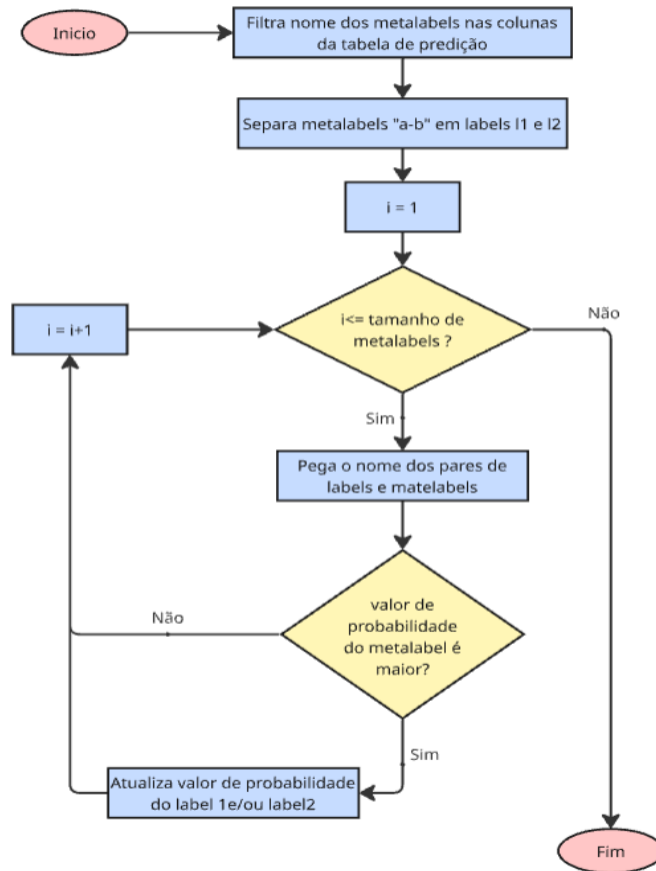
Após essa filtragem, cada *metalabel* é decomposto em dois rótulos originais, possibilitando recuperar a relação entre o *metalabel* e os rótulos que o compõem. Esse processo garante que, para cada *metalabel* no formato “a–b”, seja possível identificar corretamente quais rótulos originais estão associados aos índices *a* e *b*.

Em seguida, a função percorre cada *metalabel* válido e, para cada um deles, verifica seus dois rótulos correspondentes. Nesse estágio, a lógica consiste em comparar a probabilidade do *metalabel* com as probabilidades dos respectivos rótulos originais. Caso a probabilidade do *metalabel* seja superior à de um rótulo original, o valor desse rótulo é atualizado para o maior entre os dois, assegurando consistência entre as probabilidades individuais e aquelas atribuídas às combinações de rótulos.

Ao final desse processo iterativo, a função retorna a matriz de predições atualizada, agora refletindo possíveis correções provenientes dos *metalabels*, o que contribui para uma representação mais consistente das probabilidades finais atribuídas aos rótulos originais.

3.4.3 Inserindo os Metalabels no Início da Chain

A inserção dos *metalabels* no início da cadeia foi implementada por meio de uma função manual denominada *Ensemble of Classifier Chains Exclusivo* (ECCx), uma adaptação do método ECC projetada para operar com rótulos originais e *metalabels* de forma

Figura 15 – Fluxograma do funcionamento da função `verifica_pred`.

Fonte: Elaborado pelo autor.

integrada. Diferentemente do ECC tradicional, que trata apenas rótulos individualmente, o ECCx incorpora *metalabels* como elementos estruturais capazes de representar dependências de ordem superior entre os rótulos. Nesse contexto, os métodos XSMLTB e CXSMLTB correspondem às versões modificadas de SMLTB e CSMLTB, diferenciando-se apenas pela realocação dos *metalabels* para o início da cadeia e pelo uso dessa lógica estendida do ECC. Enquanto o XSMLTB aplica apenas essa reorganização, o CXSMLTB combina a nova ordem com a etapa adicional de propagação de probabilidades, permitindo analisar como a alteração estrutural da cadeia influencia o processo de predição.

Em cada iteração do *ensemble*, conforme apresentado na Algoritmo 1, o método executa uma sequência de etapas. Começa com a **seleção aleatória de subconjuntos** de instâncias e atributos, seguida pela **identificação dos metalabels** dentro do conjunto de rótulos e pela **definição de uma cadeia de rótulos**, na qual os *metalabels* são posicionados no início, seguidos pelos rótulos originais, conforme ilustrado na Figura 16. O modelo gerado é então **treinado** e **adicionado ao ensemble**.

Cada iteração resulta em uma cadeia de rótulos independente, treinada com dife-

rentes subconjuntos de dados e atributos, além de diferentes ordens de rótulos. Após m iterações, o *ensemble* formado por múltiplos modelos proporciona maior **robustez**, menor **variância** e uma melhor capacidade de capturar **dependências complexas** entre rótulos e **metalabels**.

O ECCx, portanto, amplia a capacidade do método ECC ao integrar informações estruturais provenientes dos *metalabels* e ao explorar uma diversidade sistemática entre os modelos aprendidos. Dessa forma, o método é capaz de aprender dependências *top-down*, do mais complexo para o mais simples, uma vez que o *metalabel* carrega informação de ordem superior, enquanto o rótulo representa informação de primeiro nível. Além disso, destaca-se que a implementação da etapa de atribuição de probabilidade também é realizada nesse caso, pois ainda podem existir situações em que a probabilidade do *metalabel* seja maior que a dos rótulos originais.

Algoritmo 1: ECCx: Visão Estruturada

Entrada: Conjunto de dados com *metalabels* e rótulos originais, m iterações.

Saída: Ensemble ECCx.

- 1 $ECCx \leftarrow \emptyset$;
 - 2 **para** $k \leftarrow 1$ **até** m
 - 3 Selecionar subconjunto aleatório $D_k \subseteq D$, $A_k \subseteq A$;
 - 4 Identificar *metalabels* $M \subseteq R$, rótulos R ;
 - 5 Construir cadeia $C_k \leftarrow [M, R]$;
 - 6 Treinar $f_k \leftarrow \mathcal{T}(C_k)$;
 - 7 Adicionar modelo ao ensemble: $ECCx \leftarrow ECCx \cup \{f_k\}$;
 - 8 **retorna** $ECCx$;
-

Figura 16 – Arquitetura do ECCx com *metalabels* no início da cadeia.

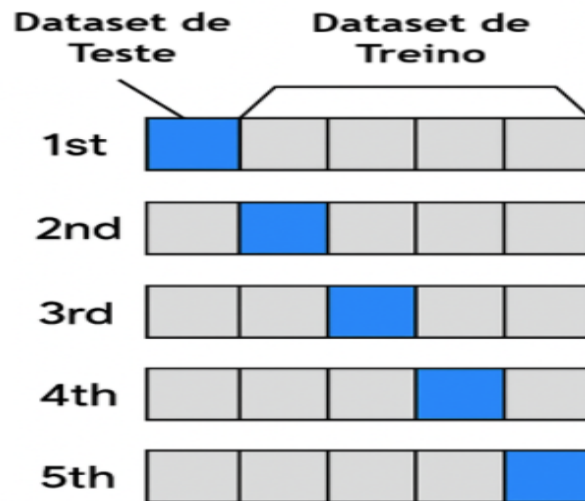
CC1							CC2						
X	MLTB.12	MLTB.13	MLTB.23	Y ₁	Y ₂	Y ₃	X	MLTB.13	MLTB.12	MLTB.23	Y ₃	Y ₂	Y ₁
x_1	0	1	0	1	0	1	x_1	1	0	0	1	0	1
x_2	0	0	1	0	1	1	x_2	0	0	1	1	1	0
x_3	1	0	0	1	1	0	x_3	0	1	0	0	1	1
x_4	1	1	1	1	1	1	x_4	1	1	1	1	1	1
x_5	0	0	0	0	0	0	x_5	0	0	0	0	0	0

Fonte: Elaborado pelo autor.

3.4.4 Implementação em Shell Script

Para executar o script do método proposto em suas cinco variações (as quatro mencionadas anteriormente, além do ECC puro para o fator 0,3), utilizando três valores de k e 19 conjuntos de dados, foi necessário desenvolver um script em *shell* para permitir a execução automatizada de todas as combinações. Além disso, é válido mencionar que o padrão utilizado para os conjuntos de dados consiste na *5-fold cross-validation*, na qual cada *fold* possui um conjunto de treino e outro de teste. Cada *fold* é executado separadamente e, ao final, as métricas são agregadas individualmente, conforme apresentado na Figura 17.

Figura 17 – Esquema da Validação Cruzada k -Fold.



Fonte: Elaborado pelo autor baseado em (MALI; MISHRA; VIJAYALAKSHMI, 2022).

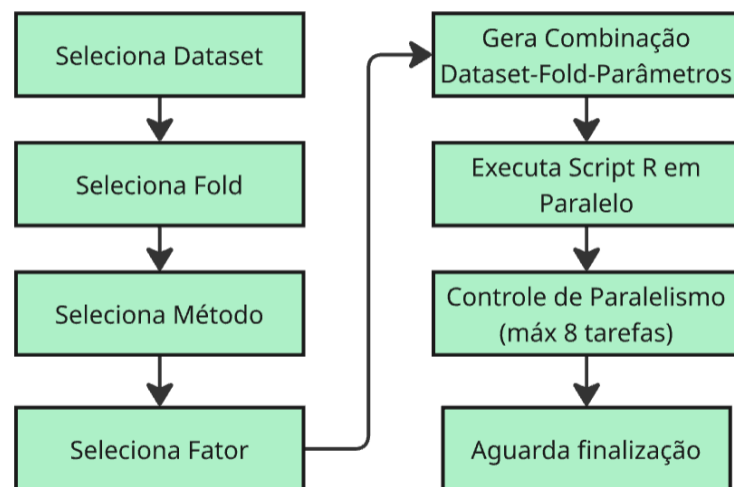
Nesse caso, como a implementação do código controla um experimento massivo em lote, é necessário utilizar quatro laços de repetição: um para iterar sobre os *folds* da validação cruzada, outro para percorrer os fatores k , o seguinte para varrer todos os *datasets* utilizados e, por fim, um laço destinado à execução de cada método empregado (ECC, SMLTB, CSMLTB, XSMLTB e CXSMLTB). Cada iteração desses quatro níveis gera uma combinação distinta de parâmetros, resultando em uma tarefa individual de execução.

Devido ao fato de o trecho de código não representar um algoritmo científico propriamente dito, mas sim uma rotina operacional responsável por organizar e distribuir as execuções, optou-se por apresentar, na Figura 18, um *workflow*. Esse diagrama ilustra a sequência lógica das etapas envolvidas, desde a seleção dos parâmetros experimentais até o controle de paralelismo e a finalização das tarefas, permitindo uma compreensão clara e objetiva do funcionamento global da rotina.

Além da organização das combinações experimentais, o script também gerencia a execução dos modelos no ambiente R. Cada tarefa gerada é enviada para o interpretador R instalado no sistema Windows, sendo chamada de forma assíncrona por meio do comando `Rscript`. Essa chamada recebe como parâmetros os caminhos dos arquivos de treino e teste correspondentes ao *fold*, bem como o fator de expansão, o método selecionado e o número de núcleos a ser utilizado no processamento interno da cadeia de classificadores.

Como várias combinações precisam ser avaliadas, a execução é realizada em paralelo, permitindo que múltiplas instâncias do script em R sejam iniciadas simultaneamente. No entanto, para evitar sobrecarga da máquina e garantir estabilidade durante o experimento, foi implementado um mecanismo de controle de paralelismo no próprio script *Bash*. Esse mecanismo monitora continuamente a quantidade de processos ativos em execução e impõe um limite máximo de oito tarefas simultâneas. Caso esse limite seja atingido, o script aguarda até que algum processo finalize, liberando espaço para que novas tarefas sejam iniciadas.

Figura 18 – *Workflow* para execução em *Shell Script*.



Fonte: Elaborado pelo autor.

4 RESULTADOS

Os resultados apresentados referem-se aos experimentos conduzidos utilizando 19 conjuntos de dados, conforme mostrado na Tabela 3. Nessa tabela, é possível visualizar todos os conjuntos utilizados, bem como seus respectivos metadados: número de instâncias, atributos, rótulos, cardinalidade, densidade e IR médio (a proporção média de desequilíbrio (CHARTE *et al.*, 2013)). A seleção desses conjuntos de dados levou em consideração a existência de, no mínimo, dez rótulos, a fim de evitar problemas na geração dos *metalabels*.

Tabela 6 – Conjuntos de dados utilizados no experimento (obtidos em: <https://cometa.ujaen.es>)

Name	Domain	m	d	$ \mathcal{L} $	Card	Dens	MeanIR
birds	Audio	645	279	19	1.01	0.05	5.40
cal500	Audio	502	242	174	260.44	0.15	20.57
enron	Text	1702	1054	53	33.78	0.06	73.95
EukPAAC	Biology	7766	462	22	11.46	0.05	45.01
foodtruck	Other	407	33	12	22.90	0.19	7.09
genbase	Biology	662	1213	27	12.52	0.05	37.31
HumanGO	Biology	3106	9858	14	11.85	0.08	15.28
HumPAAC	Biology	3106	454	14	11.85	0.08	15.28
langlog	Text	1460	1079	75	11.80	0.02	39.26
medical	Text	978	1494	45	12.45	0.03	89.50
ng20	Text	19300	1026	20	10.29	0.05	1.00
ohsumed	Text	13929	1025	23	16.63	0.07	7.86
PlantGO	Biology	978	3103	12	10.79	0.09	6.69
PlanPAAC	Biology	978	452	12	10.79	0.09	6.69
reutersk500	Text	6000	603	103	14.62	0.01	51.98
slashdot	Text	3782	1101	22	11.81	0.05	17.69
stackex_chess	Text	1675	812	227	24.11	0.01	85.78
tmc2007_500	Text	28596	522	22	22.20	0.10	15.15
yeast	Biology	2417	117	14	42.37	0.30	7.19

A seleção dos k melhores rótulos para a criação dos *metalabels*, na qual o usuário informa o valor desejado como fator multiplicativo, foi realizada utilizando três valores distintos de k : 0,3, 0,5 e 0,8, que correspondem a 30%, 50% e 80% do total de rótulos. No caso e que $k = 1$, o número de *metalabels* gerados é igual ao total de rótulos. Quando k é maior que 1, o número de *metalabels* ultrapassa o total de rótulos, o que permite gerar mais combinações possíveis entre os rótulos. Conforme discutido em Schmitt e Ferrandin (2024), valores iguais ou superiores a 1,0 deixam de oferecer ganhos nas métricas de avaliação, motivo pelo qual não foram considerados neste estudo.

No que se refere às métricas, não foram utilizadas todas as disponíveis, sobretudo devido à grande quantidade existente em conjunto com os *datasets* empregados. Assim, optou-se por adotar apenas as métricas tradicionalmente empregadas na avaliação de classificadores multirrótulo, sendo elas: *Hamming-Loss*, *Macro-F1*, *Micro-F1*, *Subset Accuracy*, *Macro-AUC*, *Micro-AUC*, *Macro-AUPRC* e *Micro-AUPRC*.

Já na execução dos algoritmos ECC e ECCx, alguns parâmetros foram previamente definidos com o objetivo de evitar variações entre diferentes execuções. Dessa forma, estabeleceu-se o número de modelos CC como $m = 10$, além de $subsample = 0.75$ e

`attr.space = 0.5`, que representam, respectivamente, a fração de amostras e a fração de atributos utilizadas em cada modelo da cadeia. Ademais, para a seleção aleatória das instâncias, adotou-se a configuração `replacement = TRUE`, permitindo a amostragem com reposição. Além disso, todos os experimentos utilizaram validação cruzada de cinco *folds*. Para fins de comparação e validação da metodologia proposta, foram empregados quatro algoritmos base amplamente utilizados na literatura: C5.0, SVM, kNN e NB.

4.1 COMPARAÇÃO GERAL

A comparação geral entre os métodos propostos e o ECC foi realizada por meio do teste de Wilcoxon (WILCOXON, 1945), utilizando as oito métricas avaliadas. Em cada confronto, analisou-se o *p-value* para verificar a significância estatística das diferenças, adotando-se como critério um valor inferior a 0,05. A Figura 19 apresenta um exemplo desses resultados, mostrando, para cada métrica, o número de vitórias de cada método e o respectivo *p-value*. A notação $< (01 \times 10)$ indica qual método obteve melhor desempenho, representado pelo símbolo relacional, e a quantidade de vitórias de cada um, sendo o primeiro valor referente ao método da esquerda e o segundo ao da direita. Assim, $< (01 \times 10)$ significa que o segundo método venceu dez vezes, enquanto o primeiro venceu apenas uma. No exemplo mostrado, o método CXSMLTB_0.3_NB-0_AND supera o XSMLTB_0.3_NB-0_AND na maioria das métricas, o que é confirmado pelo resumo inferior da tabela: *wins c1* = 1 e *wins c2* = 7, sem empates. Considerando apenas vitórias significativas, o primeiro método não apresenta nenhuma (*wins c1 sig* = 0), enquanto o segundo mantém vantagem, reduzindo de 7 para 5 vitórias significativas.

Figura 19 – Resultado do teste de Wilcoxon aplicado em dois métodos aleatórios.

	XSMLTB_0.3_NB-0_AND x CXSMLTB_0.3_NB-0_AND	XSMLTB_0.3_NB-0_AND x CXSMLTB_0.3_NB-0_AND p-value
"macro-F1"	"< (01x10)"	"0.0559296803251122"
"micro-F1"	"< (02x09)"	"0.0366712945648324"
"subset-accuracy"	"> (06x02)"	"0.362726506485098"
"hamming-loss"	"< (03x07)"	"0.540817904242618"
"micro-AUC"	"< (02x15)"	"0.000920809429065008"
"macro-AUC"	"< (05x12)"	"0.015767922944396"
"micro-AUCPR"	"< (04x14)"	"0.0186868755048476"
"macro-AUCPR"	"< (06x12)"	"0.0451433426811758"
"Summary"	"	"
"wins c1"	"XSMLTB_0.3_NB-0_AND"	"1"
"wins c2"	"CXSMLTB_0.3_NB-0_AND"	"7"
"ties"	"Tie"	"0"
"wins c1 sig"	"XSMLTB_0.3_NB-0_AND"	"0"
"win c2 sig"	"CXSMLTB_0.3_NB-0_AND"	"5"

Fonte: Elaborado pelo autor.

Para fins práticos, foi realizada uma comparação entre as três lógicas e os quatro algoritmos avaliados. Essa análise está apresentada da Figura 20 até a Figura 31. A comparação funciona da seguinte forma: para cada lógica, em conjunto com um determinado

classificador base, verifica-se como a variação do fator afeta o desempenho de cada método, contabilizando quantas vezes ele vence, empata ou perde com significância estatística. A Figura 20 é um exemplo dessa análise e mostra, para cada método, quantas vitórias, empates e derrotas ele obteve nas comparações diretas com os demais. Observando a primeira coluna do gráfico, nota-se que o método XSMLTB, com fator 0,3, classificador SVM e lógica AND, venceu 52,7% das comparações diretas com os outros métodos que utilizam o mesmo classificador e a mesma lógica.

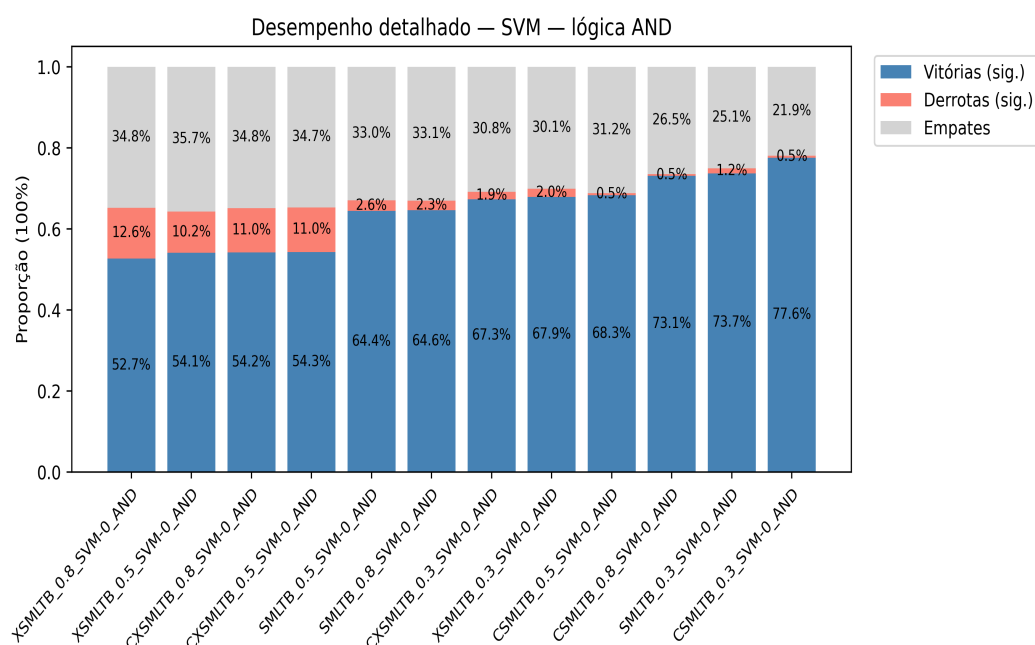
Nas Figuras 20 a 22, é possível observar a comparação detalhada do desempenho dos métodos quando combinados ao algoritmo SVM sob as três lógicas avaliadas: AND, OR e XNOR. As figuras apresentam a proporção de vitórias significativas, derrotas significativas e empates, permitindo analisar como o comportamento dos métodos varia conforme a lógica utilizada. Iniciando pela Figura 20, correspondente à lógica AND, nota-se que os métodos com fator 0,3, em especial o CSMLTB, obtêm a maior proporção de vitórias significativas, alcançando aproximadamente 77,6%. Esse comportamento indica um cenário claramente favorável ao SVM quando a lógica AND é utilizada, apresentando baixas proporções de derrotas e uma quantidade moderada de empates.

Diferentemente do que é mostrado nas Figuras 21 e 22, referentes às lógicas OR e XNOR, observa-se uma mudança progressiva na distribuição dos resultados. Na Figura 21, que apresenta a lógica OR, os métodos com fator 0,3 continuam obtendo desempenhos relativamente superiores, embora com valores mais modestos em comparação à lógica AND. O destaque está associado à variante SMLTB, que atinge cerca de 63,2% de vitórias significativas. Embora ainda expressivo, esse valor já indica uma queda considerável em relação ao caso anterior.

Ao avançar para a Figura 22, correspondente à lógica XNOR, essa tendência de redução torna-se ainda mais evidente. O melhor desempenho, também associado ao fator 0,3, atinge apenas 31,5% de vitórias significativas. Além disso, nota-se um aumento substancial na proporção de derrotas significativas, refletindo um comportamento menos favorável sob essa lógica. Essa transição da lógica AND para a lógica XNOR evidencia uma resposta sensível do SVM à estrutura lógica empregada, com desempenho sistematicamente reduzido à medida que se avança para lógicas menos restritivas ou estruturalmente diferentes.

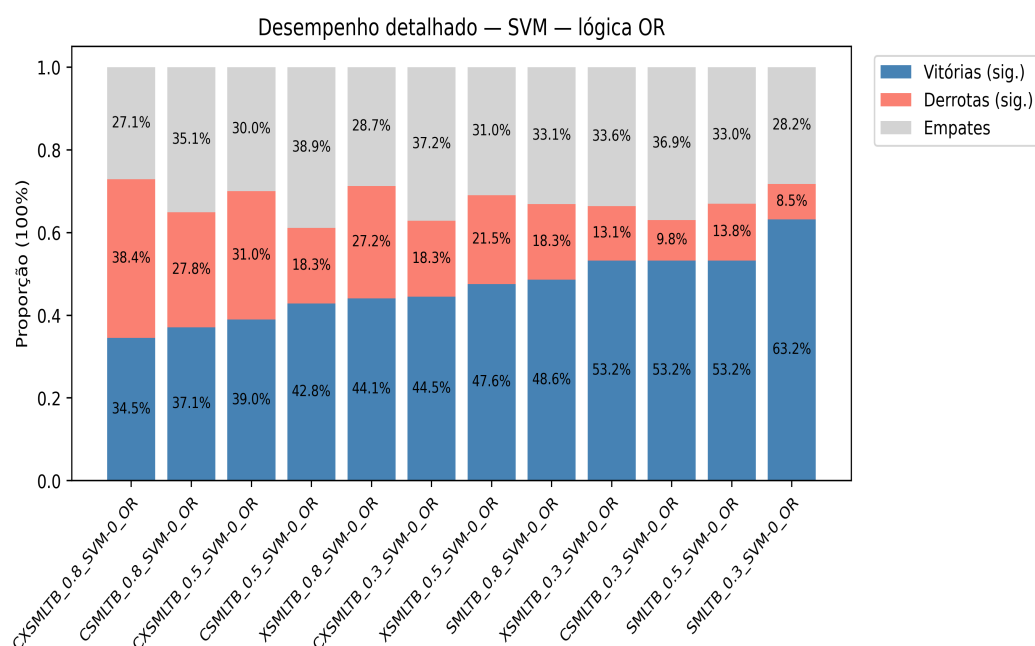
Quanto aos piores desempenhos, observa-se uma progressão clara entre as três lógicas. Na Figura 20 (lógica AND), o método menos favorável é o XSMLTB com fator 0,8 que, apesar de ocupar a última posição, apresenta um comportamento relativamente estável, com predominância de empates e poucas derrotas significativas. Já na Figura 21 (lógica OR) e na Figura 22 (lógica XNOR), o pior resultado passa a ser o CXSMLTB com fator 0,3, cuja proporção de vitórias é consideravelmente menor (34,5% e 0,2%, respectivamente) que a dos melhores métodos dessas lógicas, indicando um desempenho mais instável.

Figura 20 – Comparação de vitórias, derrotas e empates — SVM (lógica AND).



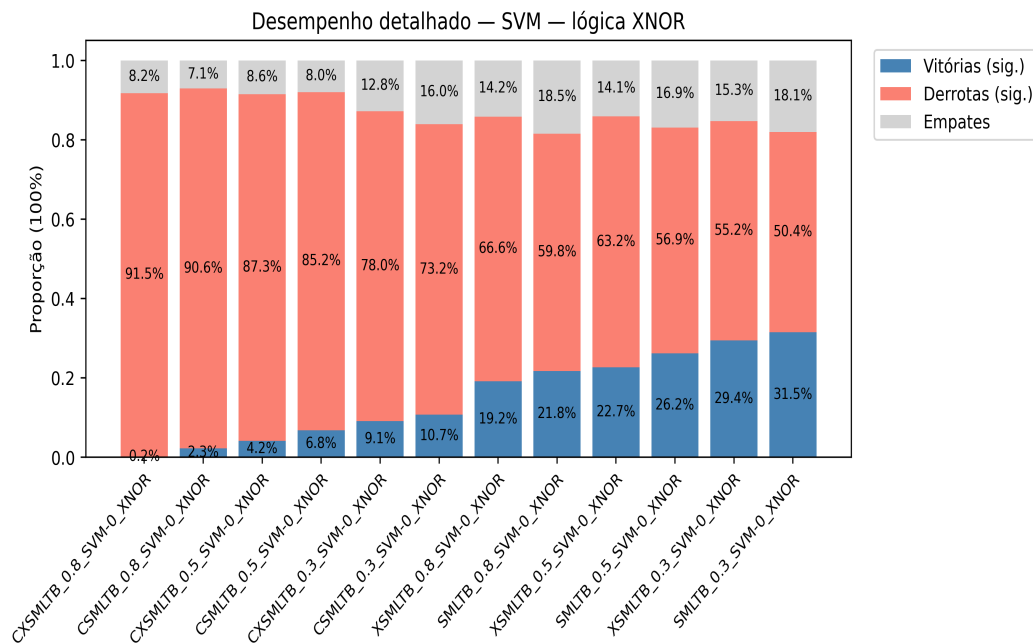
Fonte: Elaborado pelo autor.

Figura 21 – Comparação de vitórias, derrotas e empates — SVM (lógica OR).



Fonte: Elaborado pelo autor.

Figura 22 – Comparação de vitórias, derrotas e empates — SVM (lógica XNOR).

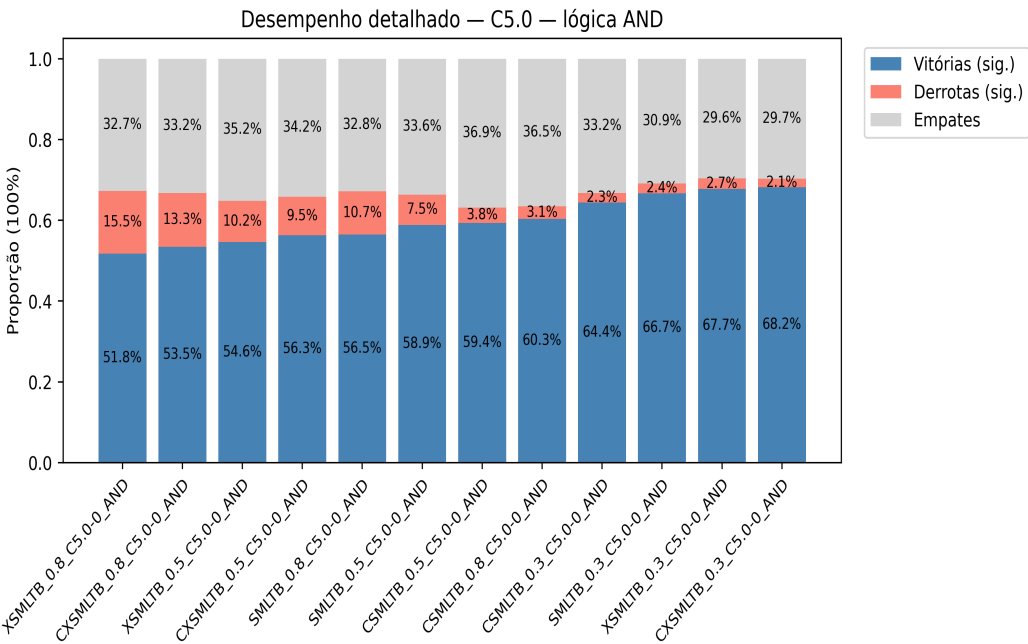


Fonte: Elaborado pelo autor.

Ao analisar em conjunto as três lógicas aplicadas ao algoritmo C5.0 nas Figuras 23 a 25, torna-se evidente que certos métodos se destacam de forma recorrente, independentemente da lógica considerada. No caso da lógica AND (Figura 23), o melhor desempenho está associado novamente ao CXSMLTB com fator 0,3, que atinge aproximadamente 68,2% de vitórias significativas, superando com folga os demais métodos. Já nas lógicas OR e XNOR (Figura 24 e Figura 25), observa-se que o método que mais se sobressai é o SMLTB, também com fator 0,3, alcançando cerca de 54,2% de vitórias na lógica OR e aproximadamente 29,1% na lógica XNOR. Embora o desempenho absoluto varie entre as lógicas, o padrão comum é que o fator 0,3 se mostra o mais consistente entre todas as combinações avaliadas, apresentando superioridade tanto em cenários favoráveis (nas lógicas AND e OR) quanto nos contextos mais adversos (na lógica XNOR).

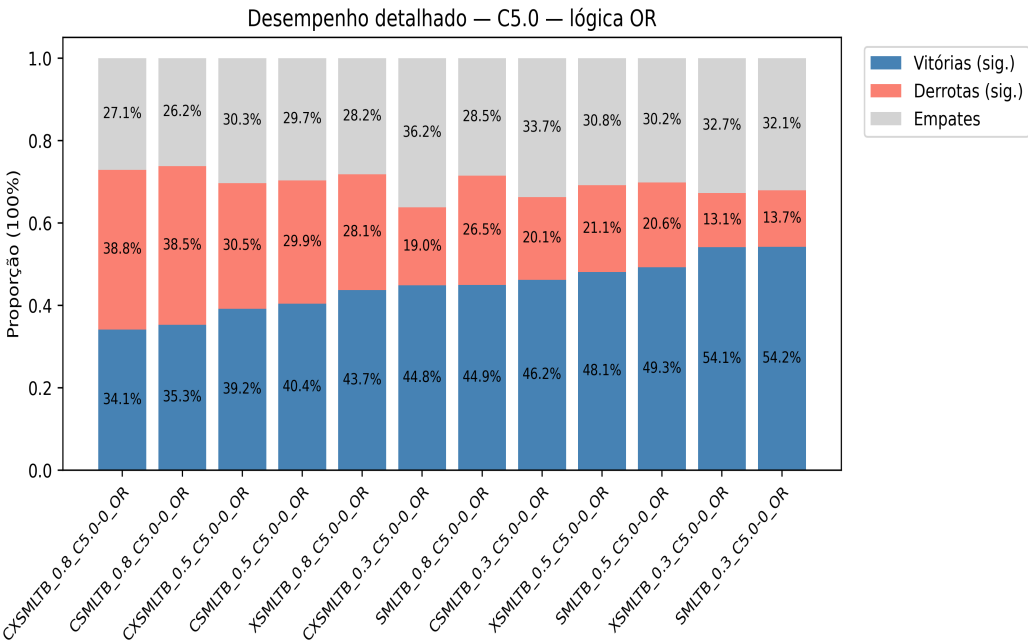
No sentido oposto, os piores desempenhos também seguem uma tendência sistemática entre as três lógicas. Na lógica AND, a menor proporção de vitórias ocorre com o método XSMLTB com fator 0,8, enquanto na lógica OR o pior resultado é encontrado no CXSMLTB com fator 0,8. Já na lógica XNOR, a queda é ainda mais acentuada, com o CXSMLTB com fator 0,8 apresentando apenas 0,7% de vitórias significativas. Assim, de forma semelhante ao observado para o SVM, os métodos da família CXSMLTB/XSMLTB voltam a compor, de maneira consistente, os piores cenários com fator 0,8, especialmente quando submetidos à lógica XNOR.

Figura 23 – Comparação de vitórias, derrotas e empates — C5.0 (lógica AND).



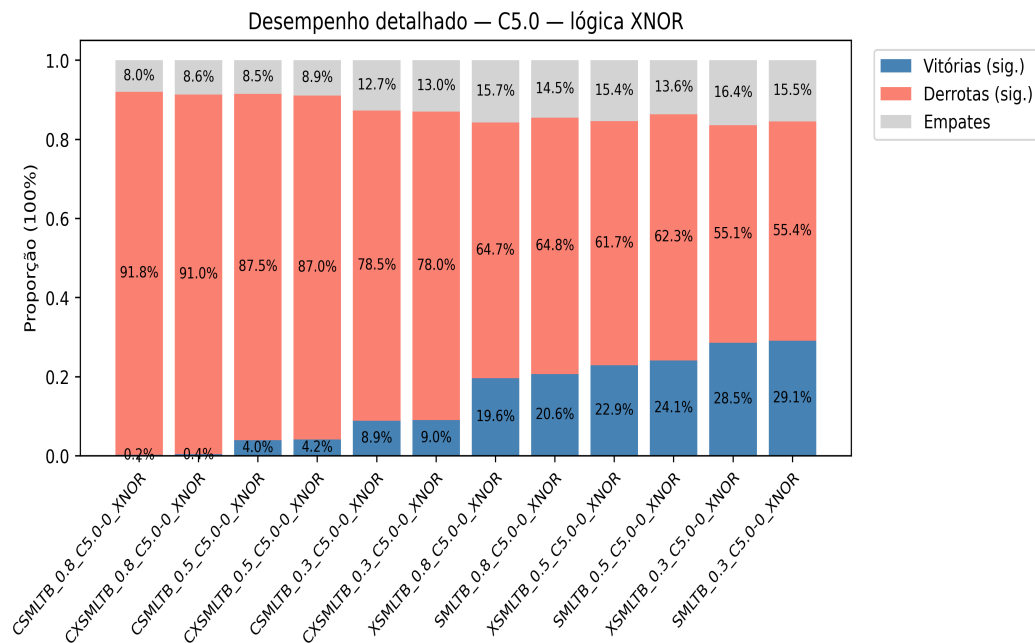
Fonte: Elaborado pelo autor.

Figura 24 – Comparação de vitórias, derrotas e empates — C5.0 (lógica OR).



Fonte: Elaborado pelo autor.

Figura 25 – Comparação de vitórias, derrotas e empates — C5.0 (lógica XNOR).



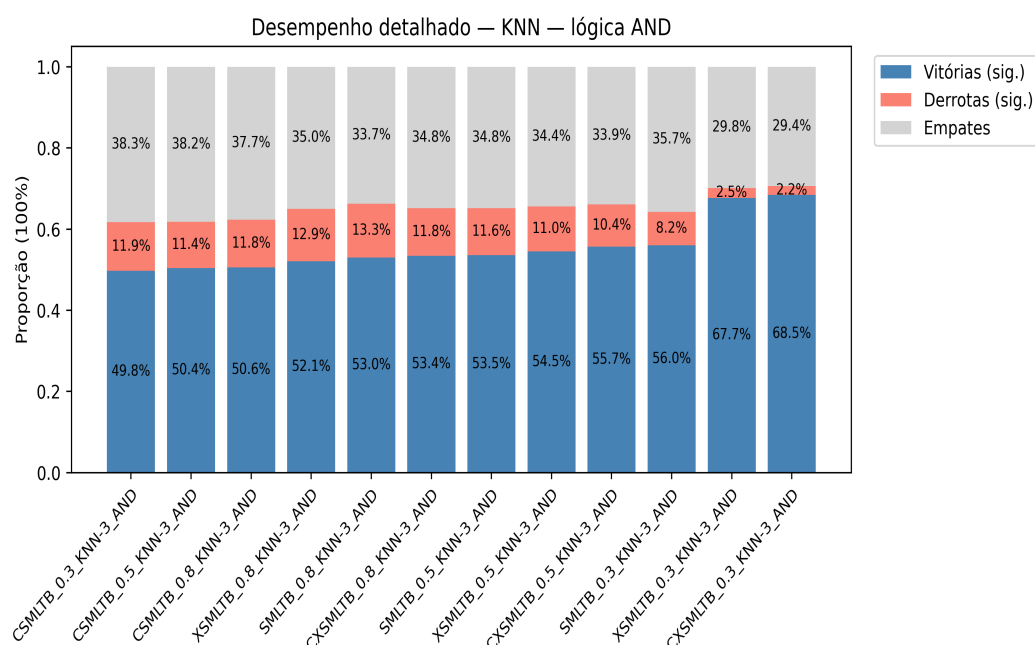
Fonte: Elaborado pelo autor.

Diferentemente do que foi observado para os classificadores SVM e C5.0, ao analisar o comportamento do kNN nas três lógicas, nas Figuras 26 a 28, obtêm-se padrões de desempenho mais suaves. Na Figura 26, correspondente à lógica AND, o kNN apresenta um desempenho globalmente favorável: as proporções de vitórias significativas situam-se em torno de 50% para a maior parte dos métodos, chegando a aproximadamente 68,5% no melhor caso, associado ao método *CXSMILTB* com fator 0,3. Além disso, as derrotas significativas permanecem baixas (em geral, abaixo de 13%), e os empates ocupam valores entre 30% e 38%.

Ao passar para a lógica OR, apresentada na Figura 27, observa-se uma leve redução nas proporções de vitórias, que agora variam de cerca de 33% a 52,7%. O melhor desempenho é obtido pelo método *XSMILTB* com fator 0,3, atingindo 54,2%. As derrotas situam-se em patamares um pouco mais elevados que no caso AND (entre aproximadamente 17% e 39%, dependendo do método), enquanto os empates permanecem em torno de 25% a 33%.

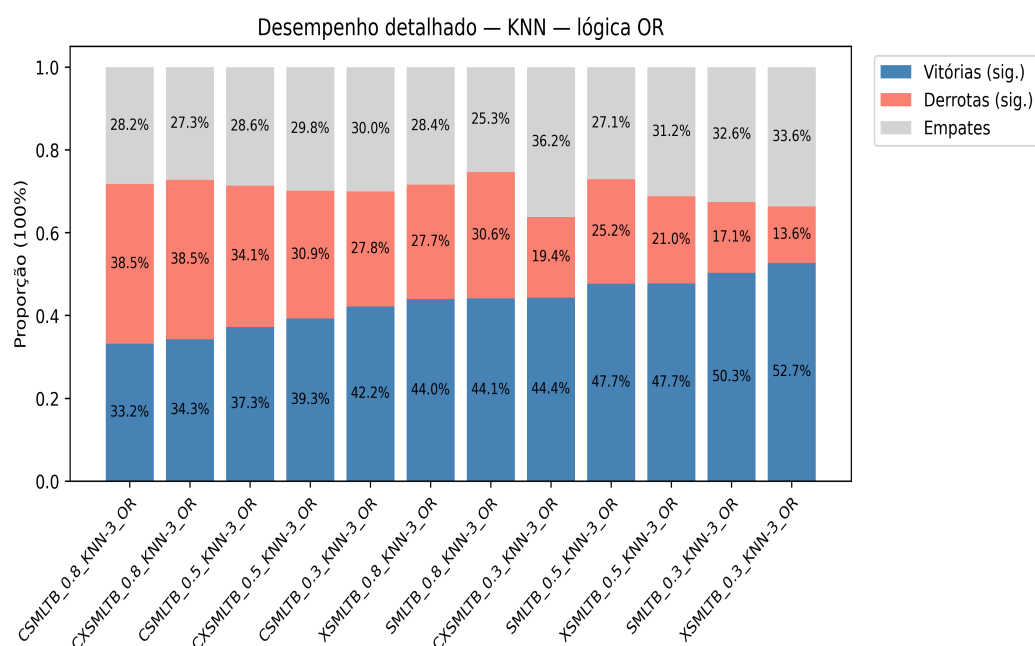
Por fim, a Figura 28, correspondente à lógica XNOR, evidencia um cenário bem mais desafiador para o kNN, com derrotas atingindo valores superiores a 60% em vários casos e ultrapassando 90% no pior desempenho, associado ao fator 0,8. As vitórias significativas tornam-se bem mais raras: apenas nas combinações mais favoráveis, como o *SMLTB* com fator 0,3, elas se aproximam de 28,7%, enquanto nos piores casos (por exemplo, *CSMLTB*/*CXSMILTB* com fator 0,8) ficam próximas de 0,5%.

Figura 26 – Comparação de vitórias, derrotas e empates — kNN (lógica AND).



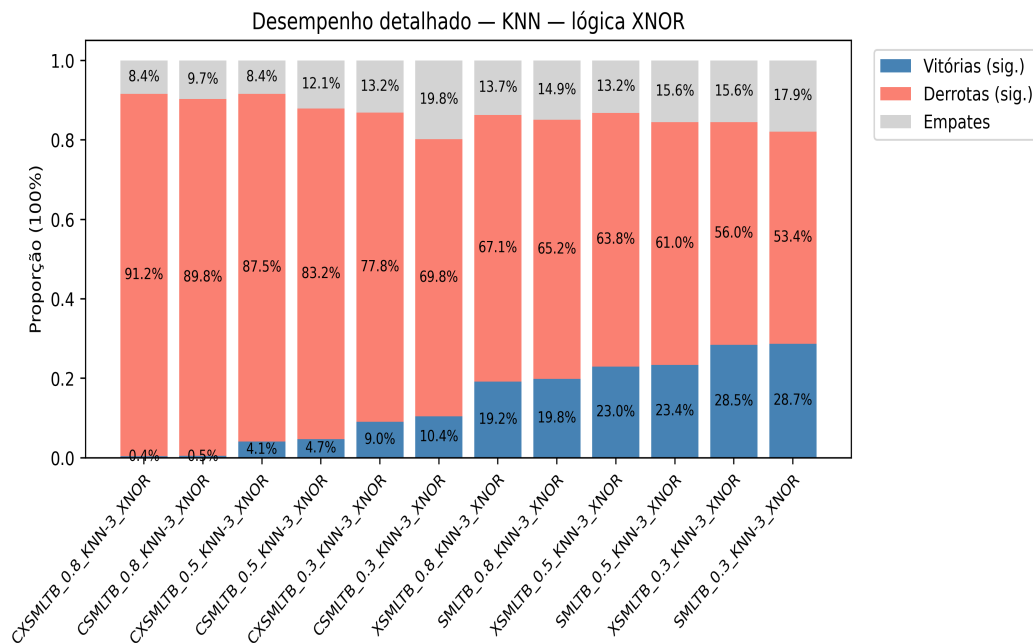
Fonte: Elaborado pelo autor.

Figura 27 – Comparação de vitórias, derrotas e empates — kNN (lógica OR).



Fonte: Elaborado pelo autor.

Figura 28 – Comparação de vitórias, derrotas e empates — kNN (lógica XNOR).



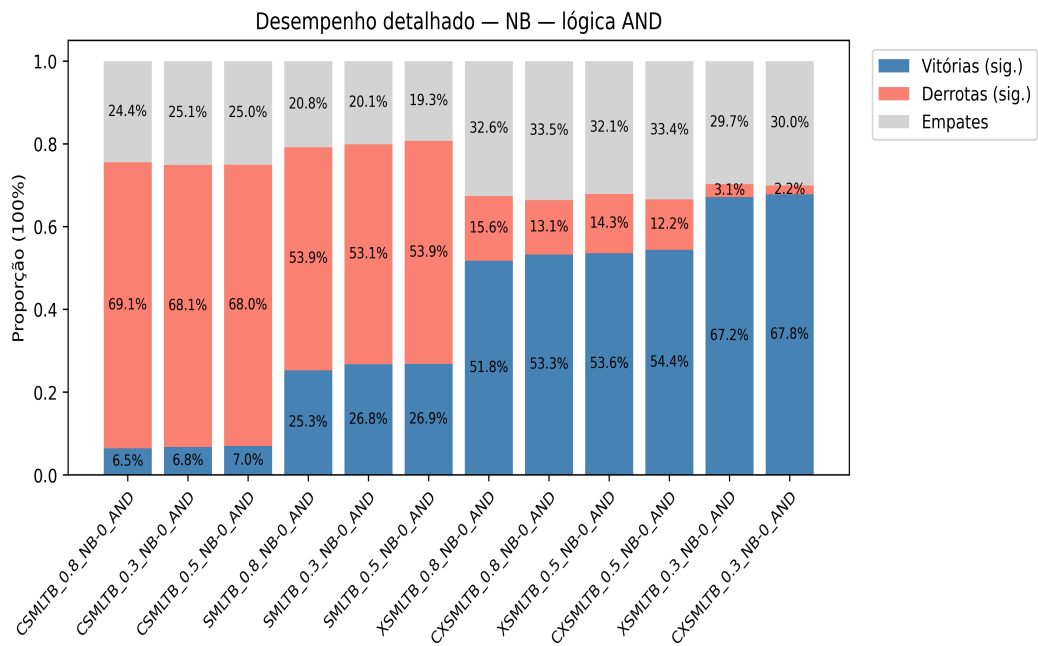
Fonte: Elaborado pelo autor.

Por fim, ao analisar o comportamento do algoritmo NB nas três lógicas, apresentadas nas Figuras 29 a 31, observa-se um padrão diferente daquele identificado nos classificadores SVM, C5.0 e kNN. No caso da lógica AND (Figura 29), o NB apresenta um comportamento relativamente equilibrado, embora ainda abaixo do desempenho observado para os demais classificadores. As vitórias significativas variam entre aproximadamente 25% e 67,8%, com os melhores resultados associados às combinações XSMILTB e SMLTB com fator 0,3.

Ao avançar para a lógica OR (Figura 30), verifica-se uma queda clara no desempenho do NB: várias combinações apresentam mais de 60% de derrotas significativas, e as vitórias se mantêm majoritariamente abaixo de 30%. Apenas nas variantes XSMILTB/CXSMILTB com fator 0,3 e 0,8 o desempenho se torna mais favorável, alcançando cerca de 44% a 54% de vitórias.

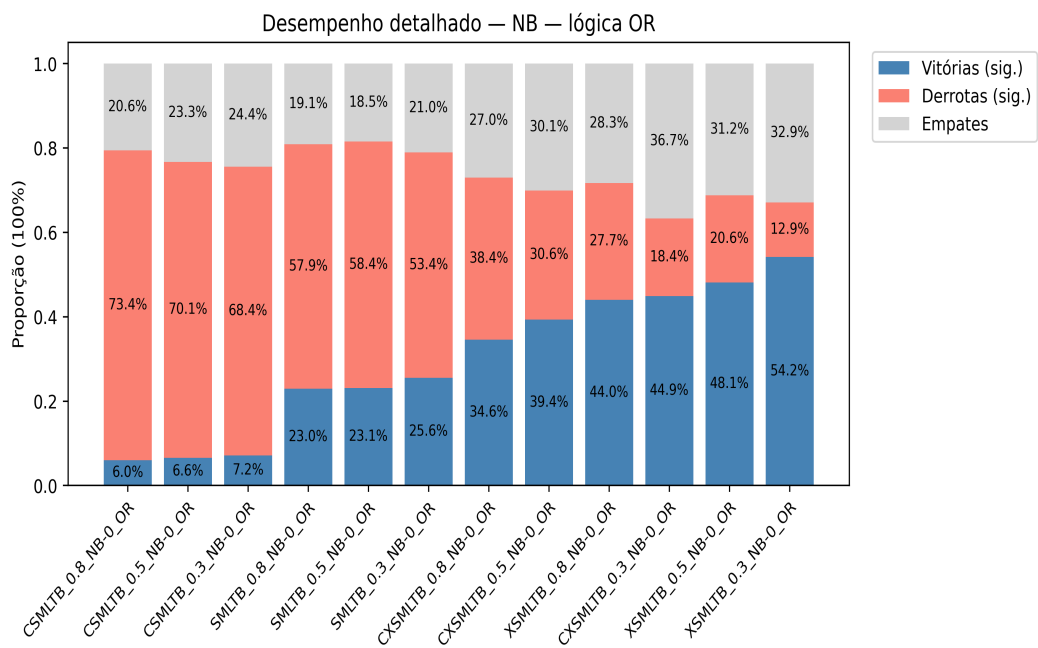
Por sua vez, a lógica XNOR (Figura 31) representa o cenário mais adverso para o NB. As derrotas significativas chegam a ultrapassar 90% no pior caso (como no CXSMILTB com fator 0,8), e as vitórias tornam-se praticamente inexpressivas para a maioria dos métodos. Apenas nas combinações do XSMILTB com fator 0,3 e 0,8 nota-se algum nível de robustez, atingindo aproximadamente 22% a 29% de vitórias significativas, ainda assim muito abaixo do observado nas lógicas AND e OR.

Figura 29 – Comparação de vitórias, derrotas e empates — NB (lógica AND).



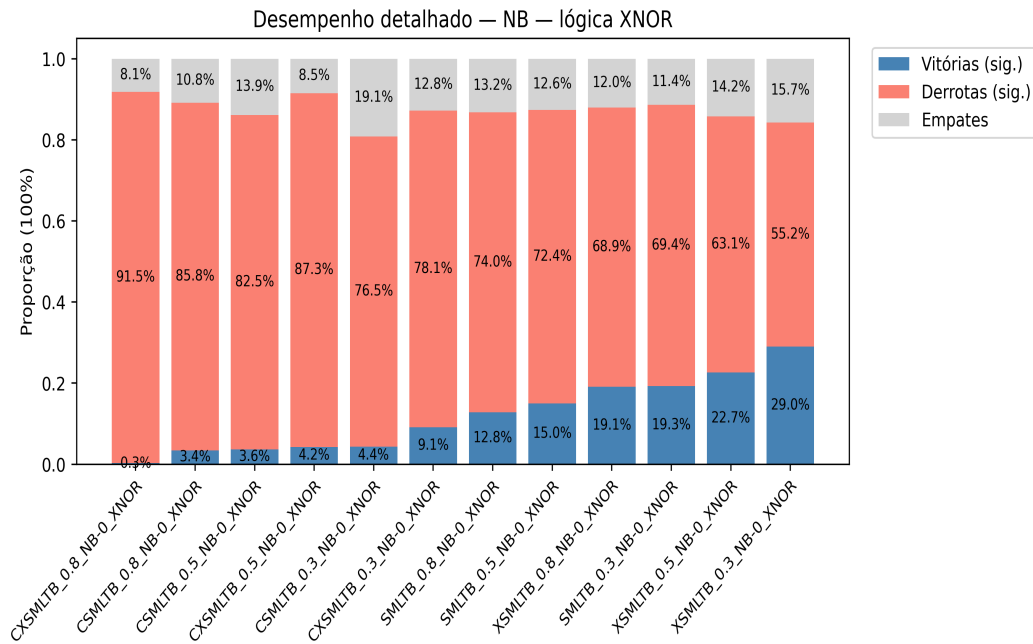
Fonte: Elaborado pelo autor.

Figura 30 – Comparação de vitórias, derrotas e empates — NB (lógica OR).



Fonte: Elaborado pelo autor.

Figura 31 – Comparação de vitórias, derrotas e empates — NB (lógica XNOR).



Fonte: Elaborado pelo autor.

A partir de todas as análises realizadas, e do resumo dos melhores e piores métodos apresentados nas Tabelas 7 e 8, torna-se possível identificar padrões consistentes de comportamento entre os classificadores, os fatores e as lógicas utilizadas. De forma geral, observa-se que a lógica AND concentra os melhores desempenhos em praticamente todos os classificadores avaliados. Em especial, destacam-se os métodos derivados das famílias CSMLTB/CXSMLTB com fator 0,3, que apresentam níveis elevados de vitórias estatisticamente significativas. O caso mais expressivo ocorre no classificador SVM, no qual o método CSMLTB com fator 0,3 atinge aproximadamente 77,6% de vitórias, a maior proporção registrada entre todas as combinações analisadas.

Em contraste, a lógica XNOR concentra os piores desempenhos globais, tanto ao se observarem os melhores resultados quanto ao analisar os métodos mais fracos, cujas taxas de vitória situam-se entre aproximadamente 28% e 31%. Já a lógica OR apresenta um desempenho intermediário: superior ao da lógica XNOR, porém inferior ao da lógica AND. Os melhores resultados sob a lógica OR também surgem majoritariamente com o fator 0,3, cujas taxas de vitória variam entre 52% e 63%. Ainda assim, tais valores permanecem inferiores aos obtidos com a lógica AND.

Outro ponto relevante é a consistência dos métodos com fator 0,3. Independentemente da lógica ou do classificador, esse fator se destaca como aquele que mais favorece a obtenção de relações significativas entre os rótulos. Em contraposição, os métodos com fator 0,8, principalmente das famílias CSMLTB e CXSMLTB, apresentam proporções de

vitórias próximas de zero, acompanhadas de taxas extremamente elevadas de derrotas significativas, frequentemente acima de 90%. Esse comportamento reforça que altos níveis de variação no processo de transformação dos rótulos prejudicam a estabilidade do classificador final, motivo pelo qual tais métodos figuram recorrentemente entre os piores.

Tabela 7 – Melhores métodos por lógica e classificador (ordenados por %vitórias)

Lógica	Classificador	Método-base	Fator	%Vitórias	%Derrotas	%Empates
AND	SVM	CSMLTB	0.3	77.58	0.48	21.94
AND	C5.0	CXSMILTB	0.3	68.23	2.10	29.68
AND	KNN	CXSMILTB	0.3	68.47	2.18	29.35
AND	NB	CXSMILTB	0.3	67.82	2.18	30.00
OR	SVM	SMLTB	0.3	63.23	8.55	28.23
OR	NB	XSMILTB	0.3	54.19	12.90	32.90
OR	C5.0	SMLTB	0.3	54.19	13.71	32.10
OR	KNN	XSMILTB	0.3	52.74	13.63	33.63
XNOR	SVM	SMLTB	0.3	31.53	50.40	18.06
XNOR	C5.0	SMLTB	0.3	29.11	55.40	15.48
XNOR	NB	XSMILTB	0.3	29.03	55.24	15.73
XNOR	KNN	SMLTB	0.3	28.71	53.39	17.90

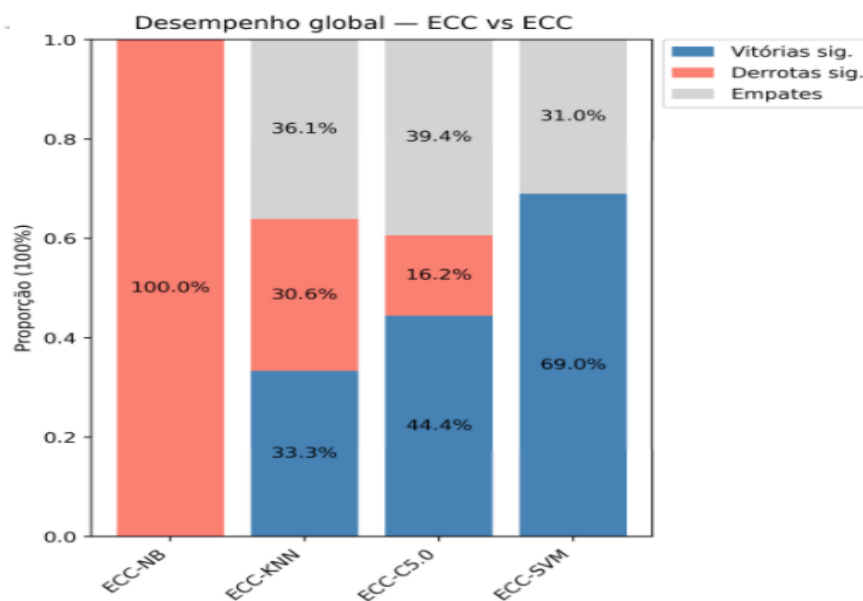
Tabela 8 – Piores métodos por lógica e classificador (ordenados por %vitórias dentro de cada lógica)

Lógica	Classificador	Método-base	Fator	%Vitórias	%Derrotas	%Empates
AND	NB	CSMLTB	0.8	6.45	69.11	24.44
AND	KNN	CSMLTB	0.3	49.76	11.94	38.31
AND	C5.0	XSMILTB	0.8	51.77	15.48	32.74
AND	SVM	XSMILTB	0.8	52.66	12.58	34.76
OR	NB	CSMLTB	0.8	6.05	73.39	20.56
OR	KNN	CSMLTB	0.8	33.23	38.55	28.23
OR	C5.0	CXSMILTB	0.8	34.11	38.79	27.10
OR	SVM	CXSMILTB	0.8	34.52	38.39	27.10
XNOR	C5.0	CSMLTB	0.8	0.24	91.78	7.98
XNOR	SVM	CXSMILTB	0.8	0.24	91.53	8.22
XNOR	NB	CXSMILTB	0.8	0.33	91.53	8.14
XNOR	KNN	CXSMILTB	0.8	0.41	91.20	8.39

Outra análise relevante refere-se ao comportamento do método ECC puro quando combinado com diferentes classificadores, permitindo sua comparação direta com os melhores métodos do modelo proposto. A Figura 32 apresenta o desempenho global do ECC sob os classificadores NB, kNN, C5.0 e SVM, considerando a proporção de vitórias, derrotas e empates estatisticamente significativos segundo o teste de *Wilcoxon*.

Observa-se que o classificador SVM é aquele que mais se destaca, apresentando aproximadamente 69% de vitórias significativas, seguido pelo C5.0, com cerca de 44,4%. O desempenho do kNN é consideravelmente inferior, com 33,3% de vitórias e uma quantidade expressiva de empates (36,1%), enquanto o NB apresenta o pior resultado geral: 100% de derrotas significativas, não registrando qualquer vitória ou empate. Esses resultados confirmam que o ECC apresenta comportamento fortemente dependente do classificador utilizado, mostrando-se robusto apenas quando combinado com modelos mais expressivos, como SVM e C5.0.

Figura 32 – Comparação de vitórias, derrotas e empates — ECC.



Fonte: Elaborado pelo autor.

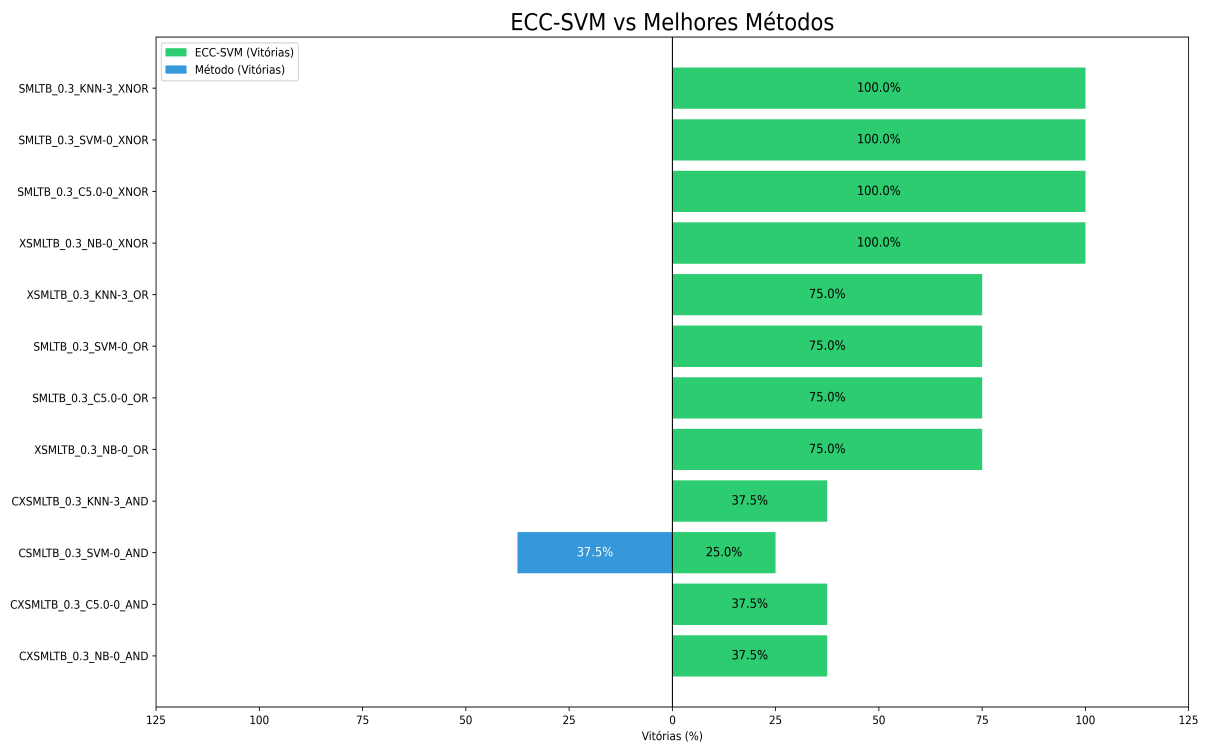
A partir da identificação das melhores configurações do ECC para cada classificador, torna-se possível compará-las diretamente com os melhores métodos do modelo proposto, apresentados na Tabela 7. Essa comparação é detalhada nas Figuras 33 e 34, nas quais cada barra horizontal representa um duelo entre o ECC e um dos melhores métodos propostos. A porção em verde indica a proporção de vitórias significativas do ECC, enquanto a porção em azul corresponde à proporção de vitórias significativas do método proposto.

Na Figura 33, que ilustra os confrontos entre o ECC-SVM e os melhores métodos propostos, observa-se que o ECC-SVM domina completamente os métodos associados à lógica XNOR e, em menor grau, aqueles baseados na lógica OR. Nesses cenários, praticamente todas as vitórias significativas ficam do lado do ECC (barras verdes próximas de 100%), enquanto o método proposto raramente supera o ECC. Entretanto, o comportamento muda quando são considerados os métodos com lógica AND e fator 0,3, justamente aqueles que se destacaram na Tabela 7. Nesses casos, o método proposto CSMLTB_0.3_SVM-0_AND

atinge proporções de vitória superiores às do ECC-SVM, apresentando maior proporção de vitórias (barra azul) do que o ECC-SVM (barra verde).

A Figura 34 apresenta análise análoga para o ECC-C5.0, o qual vence todos os duelos contra os melhores métodos associados à lógica XNOR e apresenta desempenho favorável na lógica OR. Contudo, nos confrontos envolvendo os métodos com lógica AND e fator 0,3, o cenário volta a se equilibrar. As barras indicam que o ECC-C5.0 e os métodos propostos obtêm proporções de vitórias muito próximas, frequentemente empatando em torno de 25% de vitórias para cada lado, exceto pelo método CSMLTB_0.3_SVM-0_AND, que supera o ECC-C5.0 com aproximadamente 37,5%.

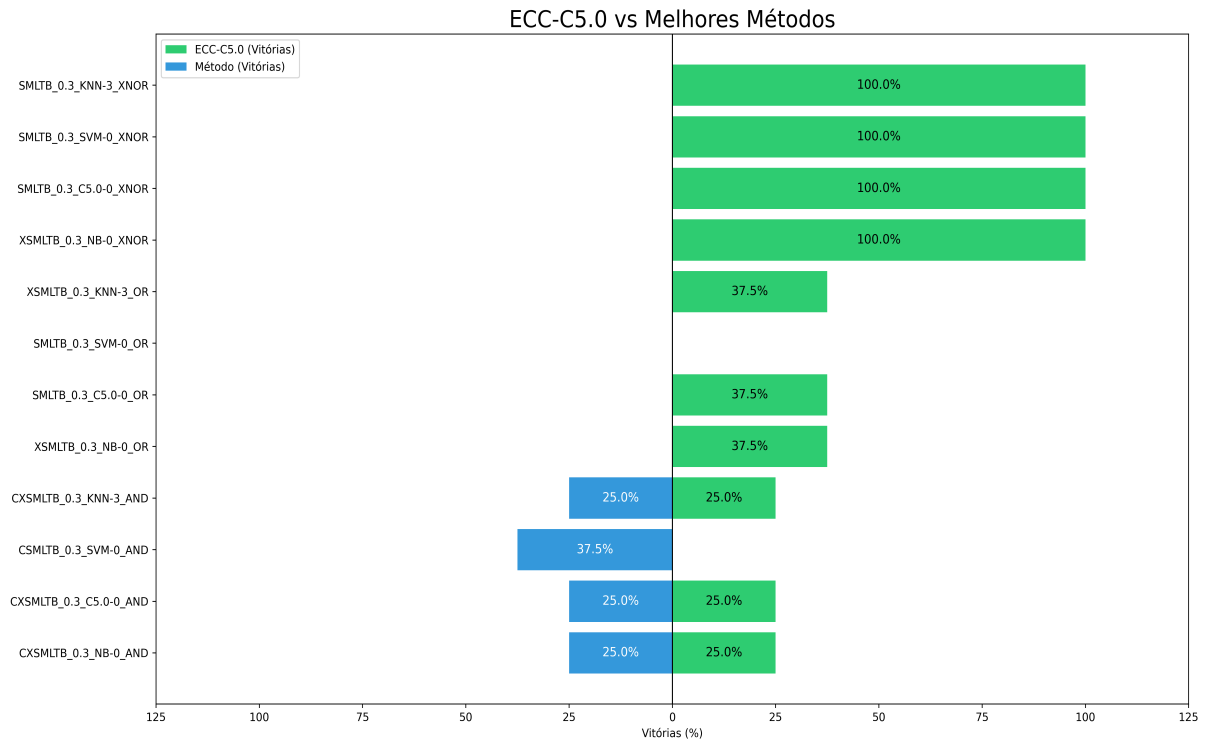
Figura 33 – Comparação do ECC-SVM com os melhores métodos.



Fonte: Elaborado pelo autor.

Em síntese, os resultados mostram que o ECC puro tende a se beneficiar sobretudo em cenários em que a lógica utilizada é menos adequada (como XNOR), enquanto o modelo proposto, ao explorar combinações específicas da lógica AND e do fator 0,3 nas famílias CSMLTB e CXSMLTB, é capaz de se igualar ou superar o ECC-C5.0 e o ECC-SVM. Esses achados reforçam que a estratégia de transformação de rótulos adotada neste trabalho não apenas se mantém competitiva frente ao ECC, como também se mostra particularmente vantajosa quando utilizada em suas configurações mais bem ajustadas.

Figura 34 – Comparação do ECC-C5.0 com os melhores métodos.



Fonte: Elaborado pelo autor.

4.2 COMPARAÇÃO GERAL ENTRE MÉTRICAS

Para realizar a comparação global entre os métodos avaliados, utilizou-se a média dos valores obtidos em cada métrica considerando todos os *datasets*. Em outras palavras, cada método foi executado individualmente em cada conjunto de dados, a métrica de interesse foi calculada e, ao final, computou-se a média sobre todos os resultados. No total, foram analisadas oito métricas: *Hamming-Loss*, *Macro-F1*, *Micro-F1*, *Subset Accuracy*, *Macro-AUC*, *Micro-AUC*, *Macro-AUPRC* e *Micro-AUPRC*. Ressalta-se que apenas a métrica *Hamming-Loss* possui caráter negativo, isto é, quanto menor o valor apresentado, melhor o desempenho do método. Todas as demais métricas são positivas, de modo que valores mais altos indicam melhor performance.

A partir das Tabelas 9 até 16, observa-se um padrão claro e consistente entre os melhores e piores desempenhos. De forma geral, a lógica AND aparece associada à maior parte dos melhores resultados, independentemente do algoritmo base (C5.0, kNN, NB ou SVM) e do fator analisado (0,3, 0,5 ou 0,8). Esse comportamento indica que a utilização da lógica AND tende a produzir classificações mais robustas e menos suscetíveis a ruídos, contribuindo para melhor desempenho nas métricas avaliadas.

Em contrapartida, a lógica XNOR concentra praticamente todos os piores resul-

tados, apresentando valores extremamente baixos em métricas como *Subset Accuracy*, *Macro-AUPRC* e *Micro-AUPRC*. Esse comportamento está alinhado com a natureza da operação XNOR, que impõe uma forte exigência de concordância total entre os rótulos que são zeros, o que é estatisticamente improvável em cenários reais de classificação multi-rótulo. A lógica OR, por sua vez, não aparece entre os extremos das tabelas. Em todas as métricas avaliadas, seus resultados mantiveram-se em desempenho intermediário, razão pela qual não integra as listas de melhores ou piores desempenhos. Esse comportamento é coerente, uma vez que OR é uma lógica permissiva, porém não tão restritiva quanto XNOR.

Além das lógicas, observa-se também um padrão relevante entre os métodos: as variantes CXSMLTB e CSMLTB destacam-se com frequência entre os melhores resultados quando combinadas com a lógica AND, enquanto as mesmas aparecem associadas à lógica XNOR na maior parte dos piores desempenhos. Métodos baseados no ECC apresentaram bom desempenho especialmente quando empregados com SVM, C5.0 e kNN no caso da métrica *Macro-F1*, e com SVM e C5.0 no caso da *Micro-F1*.

Por fim, destaca-se que a análise realizada para os fatores 0,3, 0,5 e 0,8 apresenta comportamento qualitativamente semelhante. Embora os valores numéricos variem conforme o fator, o padrão estrutural permanece inalterado: a lógica AND domina os melhores resultados, enquanto a lógica XNOR concentra os piores desempenhos. Esse comportamento reforça que a lógica de combinação exerce influência mais significativa sobre o desempenho final do que o algoritmo base utilizado.

Tabela 9 – Resultados de *Hamming-Loss* — melhores e piores valores por algoritmo e fator

MELHOR — Fator 0.3				PIOR — Fator 0.3			
Algoritmo	Método	Lógica	Average	Algoritmo	Método	Lógica	Average
C5.0	CXSMLTB	AND	0,067997	C5.0	CSMLTB	XNOR	0,381276
KNN	CXSMLTB	AND	0,068027	KNN	CSMLTB	XNOR	0,383026
NB	CXSMLTB	AND	0,068046	NB	CSMLTB	XNOR	0,713643
SVM	CSMLTB	AND	0,064168	SVM	CXSMLTB	XNOR	0,379037
MELHOR — Fator 0.5				PIOR — Fator 0.5			
Algoritmo	Método	Lógica	Average	Algoritmo	Método	Lógica	Average
C5.0	CSMLTB	AND	0,068437	C5.0	CXSMLTB	XNOR	0,531066
KNN	CXSMLTB	AND	0,068714	KNN	CXSMLTB	XNOR	0,532707
NB	CXSMLTB	AND	0,068975	NB	CSMLTB	XNOR	0,752697
SVM	CSMLTB	AND	0,064660	SVM	CXSMLTB	XNOR	0,532221
MELHOR — Fator 0.8				PIOR — Fator 0.8			
Algoritmo	Método	Lógica	Average	Algoritmo	Método	Lógica	Average
C5.0	CSMLTB	AND	0,068576	C5.0	CSMLTB	XNOR	0,622022
KNN	CXSMLTB	AND	0,069072	KNN	CSMLTB	XNOR	0,624194
NB	CXSMLTB	AND	0,069182	NB	CSMLTB	XNOR	0,772294
SVM	CSMLTB	AND	0,064335	SVM	CXSMLTB	XNOR	0,621799

Tabela 10 – Resultados de *Macro-F1* — melhores e piores valores por algoritmo e fator

MELHOR — Fator 0.3				PIOR — Fator 0.3			
Algoritmo	Método	Lógica	Average	Algoritmo	Método	Lógica	Average
C5.0	ECC	-	0,293976	C5.0	CSMLTB	XNOR	0,196026
KNN	ECC	-	0,304456	KNN	CXSMLTB	XNOR	0,199589
NB	CXSMLTB	AND	0,276304	NB	CSMLTB	XNOR	0,138656
SVM	ECC	-	0,331638	SVM	CXSMLTB	XNOR	0,198680
MELHOR — Fator 0.5				PIOR — Fator 0.5			
Algoritmo	Método	Lógica	Average	Algoritmo	Método	Lógica	Average
C5.0	SMLTB	AND	0,269308	C5.0	CSMLTB	XNOR	0,172223
KNN	SMLTB	AND	0,278344	KNN	CXSMLTB	XNOR	0,175068
NB	CXSMLTB	AND	0,267628	NB	CSMLTB	XNOR	0,136032
SVM	SMLTB	AND	0,283321	SVM	CXSMLTB	XNOR	0,174430
MELHOR — Fator 0.8				PIOR — Fator 0.8			
Algoritmo	Método	Lógica	Average	Algoritmo	Método	Lógica	Average
C5.0	CSMLTB	AND	0,271273	C5.0	CSMLTB	XNOR	0,166170
KNN	SMLTB	AND	0,275017	KNN	CXSMLTB	XNOR	0,169832
NB	CXSMLTB	AND	0,266128	NB	CSMLTB	XNOR	0,133617
SVM	CSMLTB	AND	0,300148	SVM	CXSMLTB	XNOR	0,168781

Tabela 11 – Resultados de *Micro-F1* — melhores e piores valores por algoritmo e fator

MELHOR — Fator 0.3				PIOR — Fator 0.3			
Algoritmo	Método	Lógica	Average	Algoritmo	Método	Lógica	Average
C5.0	ECC	-	0,529293	C5.0	CXSMLTB	XNOR	0,211461
KNN	CXSMLTB	AND	0,506906	KNN	CSMLTB	XNOR	0,207302
NB	CXSMLTB	AND	0,507111	NB	CSMLTB	XNOR	0,170344
SVM	ECC	-	0,557278	SVM	CXSMLTB	XNOR	0,211841
MELHOR — Fator 0.5				PIOR — Fator 0.5			
Algoritmo	Método	Lógica	Average	Algoritmo	Método	Lógica	Average
C5.0	CSMLTB	AND	0,505769	C5.0	CSMLTB	XNOR	0,183942
KNN	CXSMLTB	AND	0,498837	KNN	CSMLTB	XNOR	0,182228
NB	CXSMLTB	AND	0,496645	NB	CSMLTB	XNOR	0,163662
SVM	CSMLTB	AND	0,529195	SVM	CXSMLTB	XNOR	0,184386
MELHOR — Fator 0.8				PIOR — Fator 0.8			
Algoritmo	Método	Lógica	Average	Algoritmo	Método	Lógica	Average
C5.0	CSMLTB	AND	0,504689	C5.0	CSMLTB	XNOR	0,176624
KNN	CXSMLTB	AND	0,493276	KNN	CSMLTB	XNOR	0,174851
NB	CXSMLTB	AND	0,493257	NB	CSMLTB	XNOR	0,162674
SVM	CSMLTB	AND	0,530012	SVM	CXSMLTB	XNOR	0,176784

Tabela 12 – Resultados de *Subset Accuracy* — melhores e piores valores por algoritmo e fator

MELHOR — Fator 0.3				PIOR — Fator 0.3			
Algoritmo	Método	Lógica	Average	Algoritmo	Método	Lógica	Average
C5.0	XSMLTB	AND	0,344224	C5.0	CSMLTB	XNOR	0,000002
KNN	XSMLTB	AND	0,344107	KNN	CSMLTB	XNOR	0,000152
NB	XSMLTB	AND	0,344247	NB	CSMLTB	XNOR	0,000023
SVM	CSMLTB	AND	0,366336	SVM	CSMLTB	XNOR	0,000010
MELHOR — Fator 0.5				PIOR — Fator 0.5			
Algoritmo	Método	Lógica	Average	Algoritmo	Método	Lógica	Average
C5.0	CSMLTB	AND	0,338325	C5.0	CSMLTB	XNOR	0,000000
KNN	CXSMLTB	AND	0,339957	KNN	CSMLTB	XNOR	0,000002
NB	CXSMLTB	AND	0,337651	NB	CSMLTB	XNOR	0,000136
SVM	CSMLTB	AND	0,362603	SVM	CXSMLTB	XNOR	0,000000
MELHOR — Fator 0.8				PIOR — Fator 0.8			
Algoritmo	Método	Lógica	Average	Algoritmo	Método	Lógica	Average
C5.0	CSMLTB	AND	0,340063	C5.0	CSMLTB	XNOR	0,000000
KNN	CXSMLTB	AND	0,336450	KNN	CSMLTB	XNOR	0,000000
NB	CXSMLTB	AND	0,336056	NB	CSMLTB	XNOR	0,000023
SVM	CSMLTB	AND	0,368941	SVM	CSMLTB	XNOR	0,000000

Tabela 13 – Resultados de *Macro-AUC* — melhores e piores valores por algoritmo e fator

MELHOR — Fator 0.3				PIOR — Fator 0.3			
Algoritmo	Método	Lógica	Average	Algoritmo	Método	Lógica	Average
C5.0	XSMLTB	OR	0,708460	C5.0	CSMLTB	XNOR	0,545841
KNN	CSMLTB	AND	0,709182	KNN	CXSMLTB	XNOR	0,549091
NB	XSMLTB	OR	0,708779	NB	CXSMLTB	XNOR	0,548383
SVM	CSMLTB	AND	0,716746	SVM	CXSMLTB	XNOR	0,548860
MELHOR — Fator 0.5				PIOR — Fator 0.5			
Algoritmo	Método	Lógica	Average	Algoritmo	Método	Lógica	Average
C5.0	CSMLTB	AND	0,704362	C5.0	CSMLTB	XNOR	0,476891
KNN	CSMLTB	AND	0,708180	KNN	CXSMLTB	XNOR	0,479206
NB	XSMLTB	OR	0,704198	NB	CXSMLTB	XNOR	0,480232
SVM	CSMLTB	AND	0,717474	SVM	CXSMLTB	XNOR	0,479231
MELHOR — Fator 0.8				PIOR — Fator 0.8			
Algoritmo	Método	Lógica	Average	Algoritmo	Método	Lógica	Average
C5.0	XSMLTB	OR	0,704555	C5.0	CSMLTB	XNOR	0,439588
KNN	SMLTB	OR	0,707390	KNN	CXSMLTB	XNOR	0,442198
NB	XSMLTB	OR	0,705292	NB	CXSMLTB	XNOR	0,441737
SVM	CSMLTB	AND	0,714938	SVM	CXSMLTB	XNOR	0,440700

Tabela 14 – Resultados de *Micro-AUC* — melhores e piores valores por algoritmo e fator

MELHOR — Fator 0.3				PIOR — Fator 0.3			
Algoritmo	Método	Lógica	Average	Algoritmo	Método	Lógica	Average
C5.0	CSMLTB	AND	0,840762	C5.0	CXSMMLTB	XNOR	0,620727
KNN	CXSMMLTB	AND	0,840434	KNN	CXSMMLTB	XNOR	0,621225
NB	CXSMMLTB	AND	0,840235	NB	CXSMMLTB	XNOR	0,620309
SVM	CXSMMLTB	AND	0,840484	SVM	CXSMMLTB	XNOR	0,621409
MELHOR — Fator 0.5				PIOR — Fator 0.5			
Algoritmo	Método	Lógica	Average	Algoritmo	Método	Lógica	Average
C5.0	CSMLTB	AND	0,840503	C5.0	CSMLTB	XNOR	0,518885
KNN	XSMMLTB	AND	0,843239	KNN	CXSMMLTB	XNOR	0,523130
NB	CXSMMLTB	AND	0,835888	NB	CXSMMLTB	XNOR	0,523176
SVM	CSMLTB	AND	0,837710	SVM	CXSMMLTB	XNOR	0,522823
MELHOR — Fator 0.8				PIOR — Fator 0.8			
Algoritmo	Método	Lógica	Average	Algoritmo	Método	Lógica	Average
C5.0	CSMLTB	AND	0,840097	C5.0	CSMLTB	XNOR	0,464836
KNN	CXSMMLTB	AND	0,835237	KNN	CXSMMLTB	XNOR	0,468343
NB	CXSMMLTB	AND	0,835342	NB	CXSMMLTB	XNOR	0,467746
SVM	CSMLTB	AND	0,835626	SVM	CXSMMLTB	XNOR	0,468287

Tabela 15 – Resultados de *Macro-AUPRC* — melhores e piores valores por algoritmo e fator

MELHOR — Fator 0.3				PIOR — Fator 0.3			
Algoritmo	Método	Lógica	Average	Algoritmo	Método	Lógica	Average
C5.0	CXSMMLTB	AND	0,335650	C5.0	CSMLTB	XNOR	0,218072
KNN	CXSMMLTB	AND	0,336089	KNN	CXSMMLTB	XNOR	0,219141
NB	CXSMMLTB	AND	0,334460	NB	CSMLTB	XNOR	0,105339
SVM	CSMLTB	AND	0,361653	SVM	CXSMMLTB	XNOR	0,218591
MELHOR — Fator 0.5				PIOR — Fator 0.5			
Algoritmo	Método	Lógica	Average	Algoritmo	Método	Lógica	Average
C5.0	CXSMMLTB	AND	0,334081	C5.0	CSMLTB	XNOR	0,168784
KNN	XSMMLTB	AND	0,330374	KNN	CXSMMLTB	XNOR	0,169655
NB	CXSMMLTB	AND	0,330117	NB	CSMLTB	XNOR	0,098992
SVM	CSMLTB	AND	0,359668	SVM	CXSMMLTB	XNOR	0,176073
MELHOR — Fator 0.8				PIOR — Fator 0.8			
Algoritmo	Método	Lógica	Average	Algoritmo	Método	Lógica	Average
C5.0	CSMLTB	AND	0,332117	C5.0	CSMLTB	XNOR	0,144824
KNN	XSMMLTB	AND	0,329699	KNN	CSMLTB	XNOR	0,145776
NB	CXSMMLTB	AND	0,328335	NB	CSMLTB	XNOR	0,094526
SVM	CSMLTB	AND	0,358768	SVM	CXSMMLTB	XNOR	0,144297

Tabela 16 – Resultados de *Micro-AUPRC* — melhores e piores valores por algoritmo e fator

MELHOR — Fator 0.3				PIOR — Fator 0.3			
Algoritmo	Método	Lógica	Average	Algoritmo	Método	Lógica	Average
C5.0	CSMLT	AND	0,493996	C5.0	CSMLTB	XNOR	0,108887
KNN	CXSMLTB	AND	0,494067	KNN	CSMLTB	XNOR	0,109763
NB	CXSMLTB	AND	0,491736	NB	CSMLTB	XNOR	0,110181
SVM	ECC	-	0,512072	SVM	CXSMLTB	XNOR	0,108783
MELHOR — Fator 0.5				PIOR — Fator 0.5			
Algoritmo	Método	Lógica	Average	Algoritmo	Método	Lógica	Average
C5.0	CSMLT	AND	0,494234	C5.0	CSMLTB	XNOR	0,091020
KNN	XSMLTB	AND	0,484391	C5.0	CXSMLTB	XNOR	0,089581
NB	CXSMLTB	AND	0,482961	KNN	CXSMLTB	XNOR	0,089615
SVM	CSMLTB	AND	0,512110	NB	CSMLTB	XNOR	0,091698
MELHOR — Fator 0.8				PIOR — Fator 0.8			
Algoritmo	Método	Lógica	Average	Algoritmo	Método	Lógica	Average
C5.0	CSMLTB	AND	0,491947	C5.0	CSMLTB	XNOR	0,083079
KNN	XSMLTB	AND	0,481125	KNN	CSMLTB	XNOR	0,085002
NB	CXSMLTB	AND	0,480232	NB	CXSMLTB	XNOR	0,083272
SVM	SMLTB	AND	0,497257	SVM	CSMLTB	XNOR	0,085780

5 CONCLUSÃO

A metodologia proposta neste trabalho introduziu uma extensão ao algoritmo *Ensemble of Classifier Chains* (ECC) por meio da criação de *metalabels*, isto é, rótulos derivados da combinação lógica entre pares de rótulos originais. Essa abordagem permitiu explorar dependências de ordem superior que não são capturadas de forma explícita pelo ECC tradicional, especialmente em cenários com forte correlação entre rótulos. Para evitar a criação excessiva ou irrelevante desses *metalabels*, empregou-se a similaridade de Jaccard na seleção dos pares mais correlacionados, juntamente com um fator multiplicativo que determinou a quantidade final de *metalabels* inseridos em cada *dataset*. Além disso, investigou-se o impacto da ordem de inserção dos *metalabels* na cadeia, bem como uma versão alternativa do método que realiza a propagação da probabilidade do *metalabel* para suas labels constituintes.

Os resultados demonstraram, de maneira consistente, que a lógica AND é a mais eficaz entre as três lógicas avaliadas (AND, OR, XNOR), apresentando desempenho superior na maior parte das métricas e classificadores utilizados. O fator multiplicativo de 0.3 mostrou-se como a melhor escolha global, oferecendo o melhor equilíbrio entre ganho informativo e aumento de dimensionalidade. Entre os métodos testados, destacaram-se especialmente as variantes CSMLTB e CXSMILTB, que alcançaram proporções elevadas de vitórias estatísticas em comparação ao ECC puro, inclusive superando-o em casos específicos quando combinadas com SVM e C5.0. Por outro lado, a lógica XNOR apresentou desempenhos significativamente inferiores, reforçando sua inadequação para o processo de criação de *metalabels*. Observou-se também que o ECC puro mantém desempenho competitivo quando combinado a classificadores expressivos.

Apesar dos resultados promissores, algumas limitações foram identificadas. A metodologia depende de conjuntos de dados com número mínimo de rótulos, uma vez que a criação de *metalabels* se torna restrita ou inviável abaixo de aproximadamente dez rótulos. Além disso, o aumento do espaço de rótulos pode gerar custos computacionais adicionais, e o comportamento do método é sensível ao tipo de classificador base utilizado. Outro ponto refere-se à utilização de métricas agregadas por média, que, embora úteis para comparação geral, podem ocultar padrões específicos por *dataset*.

Como perspectivas futuras, destaca-se a possibilidade de empregar métodos adicionais de avaliação, como a análise do desvio-padrão e de intervalos de confiança, permitindo verificar a estabilidade dos resultados obtidos. Outra linha de continuidade envolve explorar formas alternativas de criação de *metalabels* em cenários com poucos rótulos, bem como testar outras medidas de similaridade, como o coeficiente de Tanimoto, na seleção das combinações mais representativas. Além disso, pode-se explorar a inserção dos *metalabels* em posições intermediárias da cadeia, utilizando critérios adaptativos de ordenação, ou ainda criar cadeias paralelas que integrem rótulos e *metalabels* de forma dinâmica.

Em síntese, o método proposto apresenta potencial para aprimorar o ECC em cenários específicos, especialmente quando há forte correlação entre rótulos, e abre espaço para novas investigações em classificação multirrótulo, contribuindo para o desenvolvimento de modelos mais robustos e sensíveis às dependências presentes nos dados.

REFERÊNCIAS

- AGGARWAL, Charu C. (Ed.). **Data Classification: Algorithms and Applications**. Boca Raton, FL: CRC Press, 2014.
- ALBUQUERQUE, M. A.; SILVA, E. L.; BARROS, K. N. N. O.; JUNIOR, S. F. A. X. Análise de Similaridade em Dados Binários Assimétricos: Um Estudo sobre o Coeficiente de Jaccard. **Revista Brasileira de Computação Aplicada**, Sociedade Brasileira de Computação, Porto Alegre, RS, v. 4, n. 2, p. 102–114, 2016. ISSN 2176-6649.
- BOUTELL, Matthew R.; LUO, Jiebo; SHEN, Xipeng; BROWN, Christopher M. Learning multi-label scene classification. **Pattern Recognition**, v. 37, n. 9, p. 1757–1771, 2004. ISSN 0031-3203.
- BOYD, Kendrick; ENG, Kevin H.; PAGE, David C. Area under the precision-recall curve: Point estimates and confidence intervals. **PLoS One**, Public Library of Science, v. 8, n. 2, e42250, 2013.
- CARVALHO, André C. P. L. F. de; FREITAS, Alex A. A tutorial on multi-label classification techniques. *In*: **STUDIES in Computational Intelligence**. [S.l.]: Springer Berlin Heidelberg, 2009. P. 177–195.
- CERRI, Ricardo; LEON FERREIRA DE CARVALHO, André Carlos Ponce de. APRENDIZADO DE MÁQUINA: BREVE INTRODUÇÃO E APLICAÇÕES. *In*:
- CHANG, Wei-Cheng; YU, Hsiang-Fu; ZHONG, Kai; YANG, Yiming; DHILLON, Inderjit. **A Modular Deep Learning Approach for Extreme Multi-label Text Classification**. [S.l.: s.n.], mai. 2019.
- CHARTE, F.; RIVERA RIVAS, Antonio; DEL JESUS, María José; HERRERA, Francisco. A First Approach to Deal with Imbalance in Multi-label Datasets. *In*:
- CHEN, H.; HU, S.; HUA, R.; YE, C.; TANG, Y. Improved Naive Bayes Classification Algorithm for Traffic Risk Management. **EURASIP Journal on Advances in Signal Processing**, v. 2021, n. 30, 2021. Received: 31 March 2021; Accepted: 08 June 2021; Published: 22 June 2021.

CORTES, Corinna; VAPNIK, Vladimir. Support-vector networks. **Machine Learning**, v. 20, n. 3, p. 273–297, 1995. ISSN 1573-0565.

DAVIS, Jesse; GOADRIC, Mark. The relationship between Precision-Recall and ROC curves. *In: PROCEEDINGS of the 23rd international conference on Machine learning (ICML)*. [S.l.]: ACM, 2006. P. 233–240.

FACELI, Katti; LORENA, Ana Carolina; GAMA, João; CARVALHO, André C. P. L. F. de. **Inteligência Artificial: Uma Abordagem de Aprendizado de Máquina**. Rio de Janeiro: LTC, 2011.

FERRANDIN, Mauri; ÁVILA, Bráulio Coelho; NIEVOLA, Júlio César. **Classificação hierárquica utilizando análise formal de conceitos**. 2012. F. 106. Tese (Doutorado) – Pontifícia Universidade Católica do Paraná, Curitiba. Acesso em: 29 maio 2014.

GATTO, Elaine Cecília. **Além do Aprendizado Local e Global: Particionando o Espaço de Classes em Problemas de Classificação Multirrótulo**. 2023. Tese (Doutorado em Ciência da Computação) – Universidade Federal de São Carlos, São Carlos, SP, Brasil.

GHAMRAWI, Nadia; MCCALLUM, Andrew. Collective multi-label classification. *In: PROCEEDINGS of the 14th ACM International Conference on Information and Knowledge Management*. Bremen, Germany: Association for Computing Machinery, 2005. (CIKM '05), p. 195–200.

GIBAJA, Enrique; VENTURA, Sebastián. Multi-label learning: A review of the state of the art and ongoing research. **Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery**, v. 4, n. 6, p. 411–444, 2014. ISSN 1942-4795.

GUIDO, Rosita; FERRISI, Stefania; LOFARO, Danilo; CONFORTI, Domenico. An Overview on the Advancements of Support Vector Machine Models in Healthcare Applications: A Review. **Information**, v. 15, n. 4, 2024. ISSN 2078-2489.

HALDER, R. K.; UDDIN, M. N.; UDDIN, M. A. *et al.* Enhancing K-nearest neighbor algorithm: a comprehensive review and performance analysis of modifications. **Journal of Big Data**, SpringerOpen, v. 11, n. 113, 2024. Version of record published August 11, 2024.

HAN, Jiawei; KAMBER, Micheline; PEI, Jian. **Data Mining: Concepts and Techniques**. 3rd. San Francisco / Waltham, MA: Morgan Kaufmann (Elsevier), 2011. (The Morgan Kaufmann Series in Data Management Systems). ISBN 978-0-12-381479-1.

HECKERMAN, David. Bayesian Networks for Data Mining. **Data Mining and Knowledge Discovery**, v. 1, n. 1, p. 79–119, 1997. Issue date: March 1997.

HSSINA, Badr; MERBOUHA, Abdelkarim; EZZIKOURI, Hanane; ERRITALI, Mohammed. A comparative study of decision tree ID3 and C4.5. (**IJACSA**) **International Journal of Advanced Computer Science and Applications**, Special Issue on Advances in Vehicular Ad Hoc Networking and Applications, jul. 2014.

JACCARD, Paul. Étude comparative de la distribution florale dans une portion des Alpes et des Jura. Versão francesa. **Bulletin de la Société Vaudoise des Sciences Naturelles**, v. 37, p. 547–579, 1901. Disponível em: <https://www.e-periodica.ch/digbib/view?pid=bsv-002:1901:37::547>.

JR., Carlos N. Silla; FREITAS, Alex A. A survey of hierarchical classification across different application domains. **Data Mining and Knowledge Discovery**, v. 22, n. 1-2, p. 31–72, 2011.

KOUL, Anirudh; GANJU, Siddha; KASAM, Meher. **Practical Deep Learning for Cloud, Mobile Edge: Real-World AI & Computer-Vision Projects Using Python, Keras & TensorFlow**. Sebastopol, CA: O'Reilly Media, 2020. ISBN 9781492034858.

KRSTINIĆ, Damir; BRAOVIĆ, Maja; ŠERIĆ, Ljiljana; BOŽIĆ-ŠTULIĆ, Dunja. Multi-label Classifier Performance Evaluation with Confusion Matrix. **International Journal of Data Science and Analytics**, v. 10, n. 3, p. 273–287, 2020. Available via ResearchGate: https://www.researchgate.net/publication/342873309_Multi-label_Classifier_Performance_Evaluation_with_Confusion_Matrix.

MADJAROV, Gjorgji; GJORGJEVIKJ, Dragi; DŽEROSKI, Dejan; MITOV, Sašo. An extensive experimental comparison of methods for multi-label learning. **Pattern Recognition**, v. 45, n. 9, p. 3084–3104, 2012. ISSN 0031-3203.

MALI, Mahesh; MISHRA, Dharendra; VIJAYALAKSHMI, M. **MF-RISE : Benchmarking for Multifaceted Recommender System Engine**. [S.l.: s.n.], dez. 2022.

- MIRZA, Amaad; ASGHAR, Sohail; NOOR, Muhammad. A Classification Model For Class Imbalance Dataset Using Genetic Programming. **IEEE Access**, v. 7, p. 71013–71037, mai. 2019.
- MOYANO, Jose; GIBAJA, Eva; CIOS, Krzysztof; VENTURA, Sebastian. Review of Ensembles of Multi-Label Classifiers: Models, Experimental Study and Prospects. **Information Fusion**, v. 44, p. 2018, nov. 2018.
- RASCHKA, Sebastian. Naive Bayes and Text Classification I - Introduction and Theory. **arXiv preprint arXiv:1410.5329**, 2014.
- READ, Jesse; PFAHRINGER, Bernhard; HOLMES, Geoff; FRANK, Eibe. Classifier Chains for Multi-label Classification. *In: MACHINE Learning and Knowledge Discovery in Databases*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. P. 254–269.
- READ, Jesse; PFAHRINGER, Bernhard; HOLMES, Geoff; FRANK, Eibe. **Classifier Chains: A Review and Perspectives**. [*S.l.: s.n.*], dez. 2019.
- RINJENI, Tri Puspa; INDRIAWAN, Ade; RAKHMAWATI, Nur Aini. Matching Scientific Article Titles using Cosine Similarity and Jaccard Similarity Algorithm. **Procedia Computer Science**, v. 234, p. 553–560, 2024. Seventh Information Systems International Conference (ISICO 2023). ISSN 1877-0509.
- SAITO, Takaya; REHMSMEIER, Marc. The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. **PloS one**, Public Library of Science, v. 10, n. 3, e0118432, 2015.
- SAJID, Muhammad; ABID, Adnan; ASIF, Muhammad; WAQAS, Muhammad; ARSHAD, M. Sheraz. Single vs. Multi-Label: The Issues, Challenges and Insights. **Applied Sciences**, MDPI, v. 13, n. 11, p. 6804, 2023.
- SALZBERG, Steven L. C4.5: Programs for Machine Learning by J. Ross Quinlan. Morgan Kaufmann Publishers, Inc., 1993. **Machine Learning**, v. 16, n. 3, p. 235–240, 1994.
- SARKER, Iqbal H. Machine Learning: Algorithms, Real-World Applications and Research Directions. **SN Computer Science**, Springer, v. 2, n. 3, p. 160, 2021.

SARKER, Iqbal H.; KAYES, A. S. M.; BADSHA, Shahriar; ALQAHTANI, Hamed; WATTERS, Paul; NG, Alex. Cybersecurity data science: an overview from machine learning perspective. **Journal of Big Data**, v. 7, n. 1, p. 1–29, 2020.

SCHMITT, Leonardo; FERRANDIN, Mauri. Augmented Chains to Ensemble of Classifier Chains. *In: ANAIS do XII Symposium on Knowledge Discovery, Mining and Learning*. Belém/PA: SBC, 2024. P. 17–24.

SILLA, Carlos N.; FREITAS, Alex A. A survey of hierarchical classification across different application domains. **Data Mining and Knowledge Discovery**, v. 22, n. 1, p. 31–72, jan. 2011.

SILVA, Pablo; GONÇALVES, Eduardo; PLASTINO, Alexandre; FREITAS, Alex. Distinct Chains for Different Instances: An Effective Strategy for Multi-label Classifier Chains. *In:*

ŚLUSARCZYK, Beata. Industry 4.0 – Are we ready? **Polish Journal of Management Studies**, v. 17, n. 1, p. 232–248, 2018.

SUYAL, Manish; GOYAL, Parul. A Review on Analysis of K-Nearest Neighbor Classification Machine Learning Algorithms based on Supervised Learning. **International Journal of Engineering Trends and Technology**, v. 70, n. 7, p. 43–48, 2022. Received: 22 May 2022; Revised: 21 June 2022; Accepted: 27 June 2022; Published: 18 July 2022.

TAN, Pang-Ning; STEINBACH, Michael; KUMAR, Vipin. **Introdução ao Data Mining - Mineração de Dados**. Rio de Janeiro: Ciência Moderna, 2009.

TSOUMAKAS, Grigorios; KATAKIS, Ioannis. Multi-Label Classification: An Overview. **International Journal of Data Warehousing and Mining**, IGI Global, v. 3, n. 3, p. 1–13, 2007.

TSOUMAKAS, Grigorios; KATAKIS, Ioannis. Multi-Label Classification: An Overview. **International Journal of Data Warehousing and Mining**, v. 3, p. 1–13, set. 2009.

TSOUMAKAS, Grigorios; KATAKIS, Ioannis; VLAHAVAS, I. Effective and efficient multilabel classification in domains with large number of labels, jan. 2008.

TSOUMAKAS, Grigorios; VLAHAVAS, Ioannis. Random k-Labelsets: An Ensemble Method for Multilabel Classification. *In*: KOK, Joost N.; KORONACKI, Jacek; MANTARAS, Raomon Lopez de; MATWIN, Stan; MLADENIČ, Dunja; SKOWRON, Andrzej (Ed.). **Machine Learning: ECML 2007**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007. P. 406–417.

WILCOXON, Frank. Individual Comparisons by Ranking Methods. **Biometrics Bulletin**, International Biometric Society; Wiley, v. 1, n. 6, p. 80–83, 1945. ISSN 00994987.

WU, Xindong *et al.* Top 10 algorithms in data mining. **Knowledge and Information Systems**, Springer, v. 14, n. 1, p. 1–37, 2008.

WU, Xi-Zhu; ZHOU, Zhi-Hua. A Unified View of Multi-Label Performance Measures. *In*: PROCEEDINGS of the 34th International Conference on Machine Learning (ICML). [S.l.: s.n.], 2017. v. 70. (Proceedings of Machine Learning Research), p. 3780–3788.

YANG, Yiming. A study of thresholding strategies for text categorization. *In*: PROCEEDINGS of the 24th annual international ACM SIGIR conference on Research and development in information retrieval. [S.l.]: ACM, 1999. P. 137–145.

ZHANG, Min-Ling; LI, Yu-Kun; LIU, Xu-Ying; GENG, Xin; ZHOU, Zhi-Hua. Binary relevance for multi-label learning: an overview. **Frontiers of Computer Science**, v. 12, n. 2, p. 191–202, 2018.

ZHANG, Min-Ling; ZHOU, Zhi-Hua. A Review on Multi-Label Learning Algorithms. **IEEE Transactions on Knowledge and Data Engineering**, v. 26, n. 8, p. 1819–1837, 2014.