Universidade Federal do Amazonas - UFAM
Instituto de Computação - ICOMP
Programa de Pós-Graduação em Informática - PPGI

Classificação de séries temporais com seleção dinâmica de representação por meta-aprendizado

Manaus
2025

Victor Beltrão Valente Duarte

Classificação de séries temporais com seleção dinâmica de representação por meta-aprendizado

Dissertação de mestrado apresentada ao Programa de Pós-Graduação em Informática do Instituto de Computação da Universidade Federal do Amazonas, como requisito para obtenção do título de Mestre em Informática.

Orientador: Prof. Dr. Rafael Giusti

Manaus

Ficha Catalográfica

Ficha catalográfica elaborada automaticamente de acordo com os dados fornecidos pelo(a) autor(a).

Duarte, Victor Beltrão Valente
D812c Classificação de séries temporais com seleção dinâmica de representação por meta-aprendizado / Victor Beltrão Valente Duarte . 2025
89 f.: il. color; 31 cm.

Orientador: Rafael Giusti
Dissertação (Mestrado em Informática) - Universidade Federal do Amazonas.

1. Aprendizagem de máquina. 2. Classificação. 3. Séries Temporais. 4. Seleção Dinâmica. 5. Meta-aprendizado. I. Giusti, Rafael. II. Universidade Federal do Amazonas III. Título

Ministério da Educação
Universidade Federal do Amazonas
Coordenação do Programa de Pós-Graduação em Informática

FOLHA DE APROVAÇÃO

"CLASSIFICAÇÃO DE SÉRIES TEMPORAIS COM SELEÇÃO DINÂMICA DE REPRESENTAÇÃO POR META-APRENDIZADO"

VICTOR BELTRÃO VALENTE DUARTE

Dissertação de Mestrado defendida e aprovada pela banca examinadora constituída pelos Professores:

Prof. Dr. Rafael Giusti - PRESIDENTE

Prof. Dr. Eduardo James Pereira Souto - MEMBRO INTERNO

Prof. Dr. Moisés Gomes de Carvalho - MEMBRO EXTERNO

MANAUS, 05 de fevereiro de 2025.

Documento assinado eletronicamente por **Rafael Giusti**, **Professor do Magistério Superior**, em 06/02/2025, às 17:14, conforme horário oficial de Manaus, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Moisés Gomes de Carvalho**, **Professor do Magistério Superior**, em 06/02/2025, às 18:37, conforme horário oficial de Manaus, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Eduardo James Pereira Souto**, **Professor do Magistério Superior**, em 10/02/2025, às 15:27, conforme horário oficial de Manaus, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://sei.ufam.edu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **2430528** e o código CRC **54469303**.

Avenida General Rodrigo Octávio, 6200 - Bairro Coroado I Campus Universitário Senador Arthur Virgílio Filho, Setor Norte - Telefone: (92) 3305-1181 / Ramal 1193
CEP 69080-900, Manaus/AM, coordenadorppgi@icomp.ufam.edu.br

Referência: Processo nº 23105.003844/2025-04 SEI nº 2430528

Resumo

Muitos métodos de classificação de séries temporais são baseados em similaridade, o que às vezes não é o suficiente para discriminar séries temporais a partir das observações originais. Uma maneira de contornar essa dificuldade é utilizar uma representação alternativa dos dados na qual as características mais importantes para a classificação são enfatizadas. Neste trabalho, emprega-se mudança de representação de dados temporais para a classificação de séries temporais, como foco na escolha de uma representação adequada e em tempo real para cada objeto de classificação. A premissa é que cada série pode ser melhor discriminada em um domínio de representação específico. Portanto, se for possível escolher uma representação adequada dinamicamente, então resultados mais satisfatórios de classificação podem ser alcançados.

Para tal fim, promove-se o estudo de domínios de representação de dados temporais, com o intuito de identificar como esses domínios fornecem uma visão alternativa de dados temporais que os diferem da visão tradicional. Os testes preliminares realizados focaram na avaliação de algoritmos k-NN e redes neurais convolucionais em 1D e 2D. O objetivo inicial era verificar como diferentes representações de dados podem afetar a qualidade de predição desses classificadores. Os resultados iniciais mostraram que utilizar diferentes representações melhora o desempenho da classificação em até 45%. Com base nisso, é criado um meta-classificador e um modelo de seleção dinâmica baseada em florestas aleatórias. Desta forma, para cada série é selecionada a representação mais adequada. Os resultados demonstram que a abordagem de escolha dinâmica de representações promove uma melhoria significativa na eficiência dos classificadores, tendo um ganho de até 16.21% contra os classificadores presentes na literatura.

Palavras Chave: Aprendizagem de máquina; Classificação; Séries temporais; Seleção dinâmica; Meta-Aprendizado.

Abstract

Many time series classification methods are based on similarity, which is sometimes not enough to discriminate time series from the original observations. One way around this difficulty is to use an alternative representation of the data in which the most important characteristics for classification are emphasized. In this work, a change in the representation of temporal data is employed for the classification of time series, focusing on the choice of a suitable, real-time representation for each classification object. The premise is that each series can be best discriminated in a specific representation domain. Therefore, if it is possible to choose a suitable representation dynamically, then more satisfactory classification results can be achieved.

To this end, the study of temporal data representation domains is promoted, with the aim of identifying how these domains provide an alternative view of temporal data that differs from the traditional view. The preliminary tests carried out focused on evaluating k-NN algorithms and convolutional neural networks in 1D and 2D. The initial objective was to verify how different data representations can affect the prediction quality of these classifiers. Initial results showed that using different representations improves classification performance by up to 45%. Based on this, a meta-classifier and a dynamic selection model based on random forests were created. In this way, the most appropriate representation is selected for each series. The results indicate that the dynamic representation selection approach promotes a significant improvement in classifier efficiency, with a gain of up to 16.21% against classifiers in the literature.

Keywords: Machine Learning; Time series classification; Dynamic selection; Meta-learning.

Agradecimentos

Primeiramente, agradeço a Deus pelo dom da vida e à minha mãe, meu maior exemplo de ser humano, sem seu auxílio, amor e compreensão eu jamais chegaria a ser a .

O apoio incondicional da minha família foi fundamental nos momentos mais desafiadores desta jornada.

Minha noiva Rebeca esteve ao meu lado em cada passo do caminho, sendo meu porto seguro.

Aos meus amigos, Eduardo, Pedro e Matheus, agradeço pelos momentos de descontração nas jogatinas, que trouxeram leveza a esta trajetória.

Aos meus colegas de trabalho do IFAM, em especial Priscilla, Ewerton, Goldema, André e Tatiana, que sempre me incentivaram a não desistir enquanto escrevia este trabalho.

Sou imensamente grato aos meus professores, sem os quais não teria chegado tão longe. Em especial, agradeço ao Felipe Pinagé, Luis (Lucho) Jorge, Teo Calvo, Jacilane Rabelo e Eduardo Souto por todo o conhecimento compartilhado.

Ao meu orientador, Rafael Giusti, expresso minha gratidão pelos quase três anos de convivência. Sua orientação foi crucial para meu desenvolvimento como aluno e pesquisador.

Agradeço à minha terapeuta, Simone Cerdeira, cujos conselhos e acompanhamento me auxiliaram a me tornar uma pessoa melhor e a compreender minhas limitações.

Um agradecimento especial à minha tia Cyani que infelizmente não pode me acompanhar no final desta trajetória, mas sinto que ficaria orgulhosa de ver a pessoa que eu me tornei, meus mais sinceros agradecimentos.

Por fim, o presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001. Este trabalho foi parcialmente financiado pela Fundação de Amparo à Pesquisa do Estado do Amazonas – FAPEAM – por meio do projeto POSGRAD.

Lista de Tabelas

Lista de Figuras

Sumário

Capítulo 1

Introdução

Neste primeiro capítulo é introduzido o problema de classificação de séries temporais e da utilização de diferentes formas de representação, assim como são apresentadas a motivação para o trabalho de pesquisa e os objetivos que se desejam alcançar. Para finalizar o capítulo, a organização da dissertação é exposta mediante uma breve descrição de cada capítulo.

1.1 Contextualização

Séries temporais são dados coletados ao longo do tempo e representam diversos fenômenos, podendo encontrar aplicações em diversas áreas do conhecimento. Por exemplo, na psicologia elas são comumente utilizadas para análises de padrões eletrofisiológicos no tratamento de desordens neurológicas, psicológicas e psiquiátricas (Catalano, 1981; Hammond, 2011; Voigt et al., 2024); em climatologia, são importantes na verificação das mudanças de estação e suas mais diversas mudanças climáticas, como a medição das temperaturas máximas e mínimas, velocidade dos ventos, queda e profundidade de neve (Applequist et al., 2024); em finanças, são usadas na análise dos dados não lineares e voláteis a fim de determinar um melhor momento de compra e venda de ações (Chen et al., 2011; Mazzonetto e Pigato, 2024).

Frequentemente considerada um subcampo da Inteligência Artificial (IA), a área de Aprendizagem de Máquina (AM) ou *Machine Learning* (ML) é um campo multidisciplinar que pode ser empregada como parte crucial do processo de várias tarefas envolvendo séries temporais, como a classificação de séries temporais (Fida et al., 2024; Jin et al., 2024) ou TSC (*Time Series Classification*). Com o aumento e a disponibilidade de dados temporais, centenas de algoritmos utilizados para a tarefa de classificação vêm sendo propostos (Middlehurst et al., 2024). Algoritmos de AM produzem modelos matemáticos que podem ser usados para explicar os dados ou para produzir previsões a partir deles (Casali et al., 2022). Vários trabalhos estão diretamente relacionados à classificação de séries temporais e utilizam diversas abordagens diferentes, buscando sempre a melhoria da acurácia dos modelos considerados Estado da Arte. Citando alguns exemplos, temos abordagens como extração e seleção de características (de Freitas Júnior, 2021), agrupamento de dados temporais (Zhu et al., 2012) e dicionarização

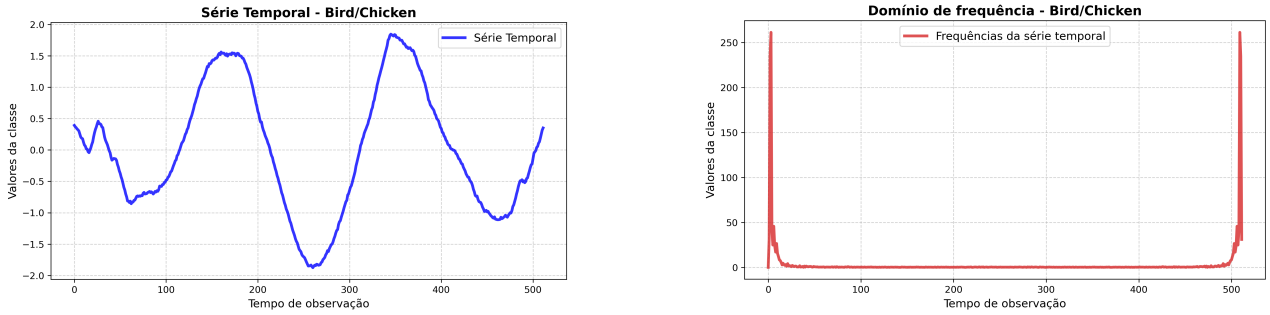
de dados temporais (Frank et al., 2021).

Existem diversos métodos utilizados para classificação de séries temporais, que podem ser divididos em categorias como: métodos baseados em distância (Guo et al., 2003), em intervalo e frequência (Lucas et al., 2019), dicionários (Schäfer, 2015), *shapelet* (Lines et al., 2012), *kernels* (Dempster et al., 2022), características (Lubba et al., 2019) e baseados em redes neurais profundas (Fawaz et al., 2020).

1.2 Motivação

Tradicionalmente, os dados temporais são analisados na chamada "representação temporal". Nessa representação, a série é uma sequência de valores numéricos que representa a observação de algum fenômeno (por exemplo, um valor da temperatura variando ao longo do tempo).

Para ilustrar a importância da representação de dados em classificação de séries temporais, consideremos o problema *BirdChicken*, que consiste em diferenciar entre imagens de contornos de pássaros e galinhas. O conjunto de dados é composto por imagens serializadas de duas classes distintas, que apresentam formas com características semelhantes entre si, tornando a classificação um desafio interessante. Cada classe contém 10 instâncias, totalizando 20 amostras para análise. A Figura 1 apresenta a representação da classe *Bird* utilizando duas abordagens distintas. Na Figura 1-(a) temos a representação da série temporal em seu domínio temporal, enquanto a Figura 1-(b) temos transformada no domínio de frequências.



(a) Série temporal em domínio temporal.

(b) Série temporal no domínio de frequências.

Figura 1: Ambas as imagens representam a classe *Bird*. Na figura (a) está no domínio temporal enquanto a figura (b) está apresentando apenas as frequências presentes na série.

Neste contexto, foi utilizado o classificador k-NN, um modelo de aprendizagem de máquina amplamente utilizado na literatura para tarefas de classificação de dados temporais para comparar o desempenho entre as representações temporal e de frequência (os detalhes deste classificador serão apresentados na Seção 2.3). Como demonstrado na Tabela 1, o classificador alcançou uma acurácia de **0.55** utilizando dados no domínio temporal, enquanto no domínio de frequência obteve uma acurácia de **0.80**. A utilização de uma abordagem alternativa na representação dos dados proporcionou uma melhoria significativa no desempenho

do classificador, destacada por um aumento de aproximadamente 45%. Este resultado indica que atributos relevantes não captados no domínio temporal tornaram-se mais evidentes após a transformação para o domínio de frequência, embora tal comportamento não seja universal para todos os domínios de aplicação.

Conjunto	**Resultado da classificação k-NN**		
	Domínio temporal	**Domínio de frequência**	**Ganhos**
BirdChicken	0.5500	0.8000	$\approx 45,45\%$

Tabela 1: Resultado da comparação de estratégias de representação.

De certeza, diversos métodos em TSC frequentemente ignoram atributos relevantes não visíveis no domínio temporal. O uso de representações inadequadas pode ter implicações significativas na tarefa de classificação, como perda de informação, viés e distorções.

Embora a seleção de uma representação mais adequada ajude a evitar tais complicações, utilizar apenas uma representação para todo o conjunto de dados pode não ser a abordagem mais eficaz.

A seleção dinâmica da melhor representação, que envolve a escolha adequada de uma representação para cada série temporal, apresenta um enorme potencial ainda pouco explorado em termos de eficácia na classificação. Esta pesquisa visa criar um modelo para selecionar uma representação de dados eficiente em séries temporais, considerando que a diversidade de representações para obter atributos discriminativos se mostra um assunto não tão explorado pelos pesquisadores da área.

1.3 Objetivos

O objetivo geral deste trabalho é elaborar um meta-modelo de aprendizagem de máquina capaz de selecionar a melhor representação de dados para cada série temporal a fim de tornar a atividade de classificação mais eficaz.

1.3.1 Objetivos Específicos

1. Investigar e quantificar o impacto de diferentes representações de dados no desempenho de classificadores de séries temporais, considerando múltiplos domínios de aplicação;

2. Desenvolver um método de seleção dinâmica de representações que identifique automaticamente a representação mais adequada para cada instância de série temporal;

3. Implementar e validar um meta-modelo que combine múltiplas representações de dados, avaliando seu desempenho em comparação com abordagens tradicionais de classificação de séries temporais;

1.4 Organização desta Dissertação

Esta dissertação está organizada da seguinte maneira: cada capítulo apresentará um breve contexto do que será abordado, apresentando os conceitos fundamentais para compreender o trabalho realizado, terminando com a etapa de considerações finais, onde serão discutidos os resultados e conclusões dos tópicos apresentados nas seções anteriores.

- Capítulo 2 - Fundamentação teórica: será apresentada a definição formal de séries temporais que serão usadas neste trabalho, unidas a conceitos formais de domínios de representações relevantes para esta pesquisa. Assim como os classificadores utilizados em aprendizado de máquina encontrados na literatura.

- Capítulo 3 - Trabalhos relacionados: é apresentada a contextualização da classificação de dados temporais, discorrendo sobre os demais trabalhos encontrados na literatura que servirão de base para esta pesquisa.

- Capítulo 4 - Proposta do trabalho: apresentará a metodologia e o escopo adotados para a realização deste trabalho. Assim como as principais propostas desenvolvidas.

- Capítulo 5 - Avaliação experimental: neste capítulo será apresentado aos experimentos realizados, incluindo o ambiente de teste, descrição dos conjuntos de dados, algoritmos avaliados e análise dos desempenhos dos experimentos realizados.

- Capítulo 6 - Conclusão: neste capítulo serão apresentadas as conclusões da dissertação, os pontos mais relevantes, suas contribuições, limitações encontradas e possíveis trabalhos futuros.

Capítulo 2

Fundamentação Teórica

Este capítulo apresenta a fundamentação teórica do trabalho. A Seção 2.1 discute a formulação matemática das séries temporais com base na literatura. Na Seção 2.2 são analisadas as diferentes transformações de dados junto a uma análise detalhada de cada representação empregada neste estudo. A Seção 2.3 discorre quais foram os classificadores utilizados. A Seção 2.4 destaca as características relevantes das diversas representações de séries temporais.

2.1 Definições

Na análise de dados de séries temporais, o objetivo é extrair informações estatísticas, resumidas e valiosas desses pontos de dados. Um dos principais objetivos da análise de séries temporais é identificar padrões nas variáveis de interesse. Ao observar comportamentos passados, é possível prever comportamentos futuros, o que pode melhorar os processos de tomada de decisão. Series temporais são conjuntos de dados coletados ao longo do tempo, nos quais cada observação está associada a um instante específico (Batista et al., 2014). A natureza e a estrutura de uma série temporal podem ser dadas pelas suas observações ao longo do tempo (Hyndman e Killick, 2023), podendo ser descritas como contínuas (Granger e Newbold, 1974) ou discretas (Davis e Nelson, 1935).

Neste trabalho, adotamos a definição de série temporal baseada nos estudos de (Chiu et al., 2003; Yankov et al., 2007; Wooldridge, 2015; Fawaz et al., 2019b; Mohammadi Foumani et al., 2024). Desse modo, uma série temporal é uma sequência $Z = (z_1, z_2, ..., z_n)$, $z_t \in \mathbb{R}$, onde Z será a função que descreve a série temporal em termos de t, n é o número de observações coletadas e t será o tempo observado.

No problema de classificação, a série temporal está associada a uma categoria ou classe. O objetivo é, dada uma série temporal desconhecida, identificar a qual classe ela pertence.

Para formalizar esse conceito, definimos $C = \{c_1, c_2, ..., c_M\}$ como sendo o conjunto das possíveis categorias ou classes. Seja um conjunto $D = \{(X_1, y_1), (X_2, y_2), ..., (X_N, y_N)\}$, onde $X_i \in \mathbb{R}^n$ é uma série temporal e $y_i \in C$ é seu rótulo verdadeiro. Tal conjunto, denominado conjunto de treinamento, é uma amostra de séries temporais rotuladas. O objetivo é aprender

uma função $f : \mathbb{R}^n \rightarrow C$, a partir dos conjuntos conhecidos de dados D, que permita classificar uma série temporal de acordo com seu rótulo correto (Xing et al., 2010; de Freitas Júnior, 2021).

2.2 Séries Temporais

A classificação de séries temporais pode ser dificultada por padrões ocultos que não são facilmente detectáveis no domínio temporal. Para superar essa limitação, diferentes transformações podem ser aplicadas para representar as séries em domínios alternativos. Essas transformações permitem enfatizar características relevantes, reduzir a dimensionalidade dos dados e facilitar a identificação de padrões (Bagnall et al., 2012; Giusti, 2017). Esse novo formato pode ter atributos mais representativos para determinar a classe ou ter uma dimensionalidade menor do que a série original (Silva et al., 2013; Xi et al., 2006). Por exemplo, a **transformada de Fourier** permite decompor a série em diferentes componentes de frequência, revelando as frequências predominantes presentes na série temporal. Isto é útil em diversas aplicações, como nas análises de áudio, onde diferentes frequências formam as diferentes notas musicais ou sons mais específicos.

Na Figura 2 a série temporal representa a mudança na amplitude ao longo do tempo durante um exame de pacientes que sofrem de arritmias comuns, onde os valores vão oscilando com o passar do tempo. Contudo, existem algumas informações implícitas que não são expostas nas séries temporais no domínio temporal. Utilizar uma transformação dos dados da série para um domínio de dados alternativo faz com que as informações sejam detectadas mais facilmente.

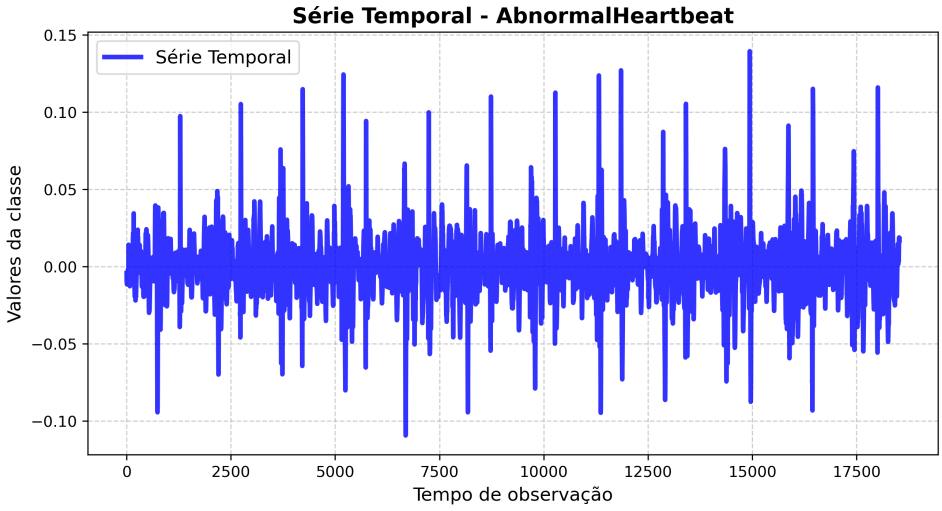


Figura 2: Exemplo de uma série temporal composta por 3.053 pontos de dados, representando medições coletadas de pacientes com arritmia. O eixo X representa o tempo, enquanto o eixo Y mostra os valores observados ao longo do período.

Em sequência, a Figura 3 demonstra uma visão de um ponto específico de tempo da Figura 2. Seu ponto de observação inicial é **0** e sua última observação no ponto **2.000**. Neste período

mais específico, já é possível identificar algumas tendências locais que podem não ser tão evidentes na visualização da série completa.

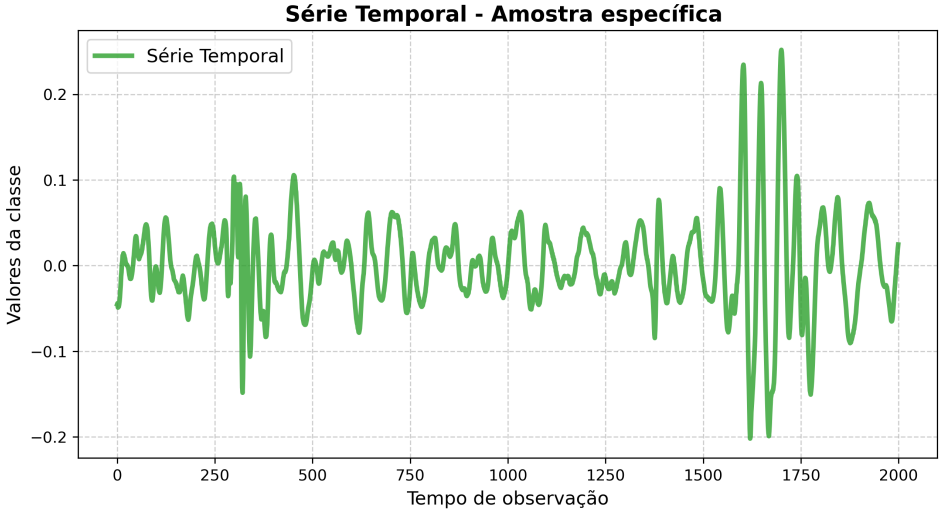


Figura 3: Está figura é uma fatia de tempo retirada da figura 2 tendo o início das observações em **0** e finalizando em **2.000**.

Por fim, a Figura 4 mostra a série em um espaço de dados alternativo. A representação de autocorrelação demonstra como a série temporal está correlacionada com ela mesma em diferentes pontos de atraso. Os picos podem significar padrões de recorrência dos dados. Isto ajuda na identificação de padrões repetitivos ou cíclicos nos dados.

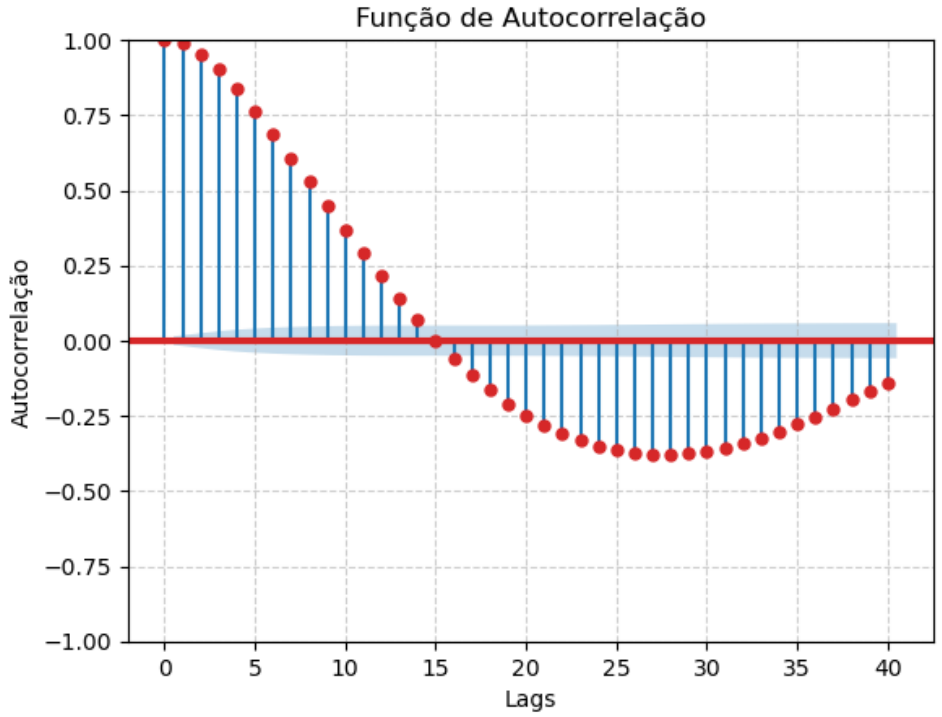
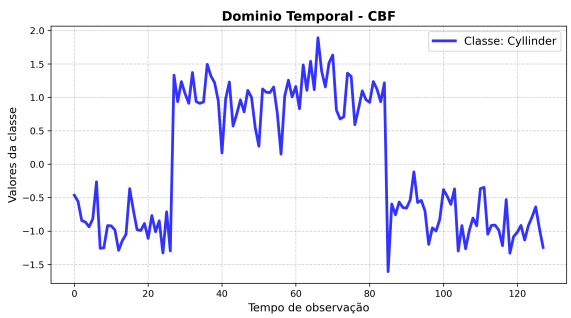


Figura 4: Representação de autocorrelação da série temporal presente na Figura 2

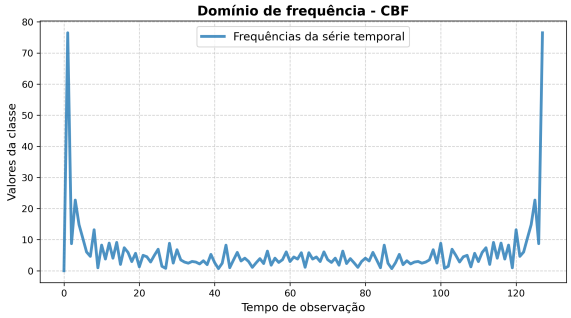
2.2.1 Representação Espectral

A representação espectral (Shumway et al., 2000) é uma técnica para análise de sinais digitais regularmente utilizada na análise e síntese de sinais de voz. Ela pode ser obtida por meio de uma transformação matemática conhecida como transformada de *Fourier*. Essa técnica é utilizada para converter um sinal no domínio do tempo em uma contraparte no domínio da frequência, na qual ficam evidenciadas as amplitudes e fases de vários componentes de frequências presentes em um sinal. A decomposição de uma série temporal em seus componentes de frequência permite a visualização e identificação dos padrões de oscilação, tendência cíclica e até mesmo a intensidade que estão presentes nos dados e quão fortemente eles contribuem para o sinal global.

Um exemplo da transformação espectral é dado na Figura 5. O sinal exibido na Figura 5-(a) foi extraído de um dos conjuntos de dados mostrados na Seção 5.1, especificamente do conjunto de dados CBF (*Cyllinder*, *Bell* e *Funnel*). O conjunto é formado por séries temporais prototípicas e suas classes são variações dessas séries, podendo ser do tipo *Cyllinder, Bell e Funnel*. Neste exemplo, utilizamos a classe do tipo *Cyllinder* para demonstrar a transformação rápida de *Fourier*. As componentes da representação são próximas de nulas em quase todo o comprimento da série, com exceção do pico de amplitude nas observações iniciais e finais próximo de 120.



(a) Demostração da série temporal em seu domínio temporal



(b) Demostração da série temporal em seu domínio de frequencia

Figura 5: Ambas as imagens representam a classe *Cyllinder*. Na figura (a) está no domínio temporal enquanto a figura (b) está apresentando apenas as frequências presentes na série.

2.2.2 Transformada Discreta de Wavelet

A Transformada Discreta de Wavelet (DWT *Discrete Wavelet Transform*)(Chan e Fu, 1999) é uma técnica matemática que permite analisar sinais e imagens em diferentes escalas de frequência e resolução. Uma *wavelet* é uma função base utilizada na representação de dados ou outras funções. Diferentemente da Transformada de Fourier, que decompõe um sinal em uma soma de senóides de diferentes frequências, a DWT permite analisar simultaneamente a frequência e o tempo. Isso a torna uma técnica mais eficaz para capturar variações temporais

nos dados, sendo amplamente utilizada em aplicações como compressão de sinais e remoção de ruído. As funções *wavelet*, que são como pequenas ondas, se ajustam melhor a diferentes partes do sinal, sendo deslocadas e escaladas para diferentes posições e tamanhos, permitindo a análise em diferentes escalas de frequência e resolução.

A DWT divide o sinal original em diferentes níveis de detalhe e aproximação, cada um correspondendo a uma escala diferente, permitindo a análise de diferentes aspectos do sinal. Na Figura 6 temos a representação de um sinal criado por um *Chirp* (*Compressed High Intensity Radar Pulse*) ou Pulso de Radar de Alta Intensidade Comprimido, também conhecido como sinal de varredura. É um sinal cuja frequência varia logaritmicamente com o tempo, podendo aumentar ou diminuir. É usado em sistemas de sonar, radar, laser e em comunicações de espectro espalhado, usados para produzir imagens de sonar mais precisas e detalhadas (Lee et al., 2016). Na Figura 7 é ilustrado como a DWT pode processar uma série em diferentes níveis de decomposição. A transformada pode ser utilizada para compressão de dados, remoção de ruído, detecção de bordas, entre outras aplicações em processamento de sinais e imagens. Em (Fu, 2011) esta técnica é utilizada para o problema de recuperação de séries temporais.

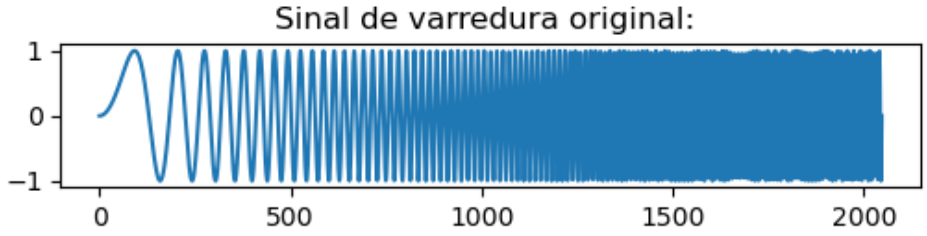


Figura 6: Sinal de varredura comumente utilizado por radares (Ataspinar, 2023)

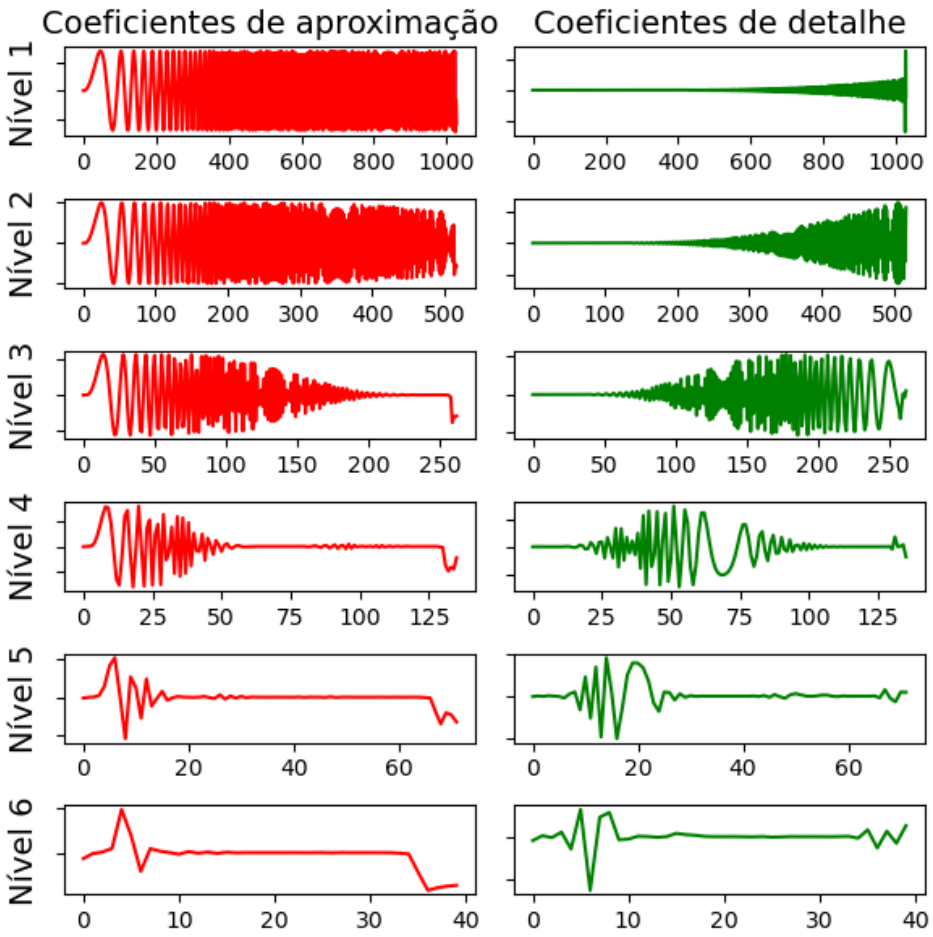


Figura 7: Representação DWT da série mostrada na Figura 6. Este gráfico demonstra como os coeficientes da DWT estão em relação à posição relativa nos níveis de decomposição. Isto permite visualizar os coeficientes DWT comparativamente e identificar padrões ou características específicas em cada nível de decomposição (Ataspinar, 2023)

2.2.3 Representação PAA (*Piecewise Aggregate Approximation*)

A representação segmentativa conhecida como aproximação agregada por partes (PAA, do inglês *Piecewise Aggregate Approximation*), proposta por (Keogh et al., 2001), visa reduzir a dimensionalidade e facilitar a indexação de dados temporais. Essa representação temporal é obtida por meio de médias de observações da série ao longo de segmentos de tamanhos iguais definidos sobre a série original. De acordo com , um dos benefícios dessa representação é a capacidade de indexar séries usando distâncias não euclidianas, o que não é possível com métodos convencionais como FFT e DWT. Na Figura 8 podemos ver a utilização da transformada PAA. A série temporal original é mostrada em azul, enquanto a **representação PAA** é mostrada em vermelho. Esta representação consiste em 19 observações representadas pelos "degraus" em vermelho.

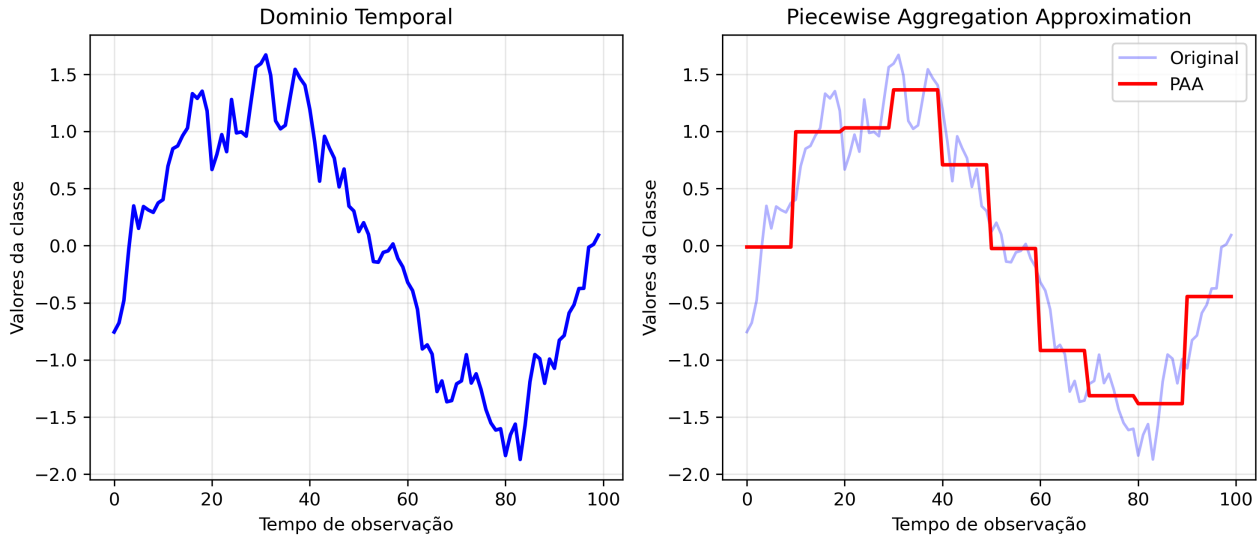


Figura 8: Representação PAA foi utilizada a fim de reduzir o número de pontos, mantendo a tendência ao longo do tempo e diminuindo o ruído da série.

2.2.4 Representação SAX (*Symbolic Aggregate ApproXimation*)

A representação *Symbolic Aggregate approXimation* (SAX) transforma séries temporais em sequências de símbolos, permitindo a aplicação de técnicas de processamento de *strings* na análise de sinais temporais (Lin et al., 2007). A transformação SAX opera sobre um alfabeto $\Sigma = w_1, w_2, \ldots, w_a$, onde a representa o tamanho do alfabeto $\Sigma = w_1, w_2, \ldots, w_N$ (efetivamente é uma função $f : \mathbb{R}^n \to \Sigma$). O processo de transformação ocorre em duas etapas principais: primeiro, a série temporal é convertida para o domínio PAA e, em seguida, cada segmento PAA é discretizado em símbolos. A discretização é projetada para cada símbolo ter aproximadamente a mesma probabilidade de ocorrência, assumindo que os dados seguem uma distribuição normal padrão (Giusti, 2017). Para realizar a discretização, são definidos pontos de corte que dividem a área sob a curva normal em regiões de probabilidade igual. A transformação SAX é controlada por dois parâmetros:

- w: Número de coeficientes na representação PAA;
- a: Tamanho do alfabeto (tipicamente limitado a 12 símbolos).

Na prática, a representação SAX de uma série temporal $S = s_1, s_2, \ldots, s_n$ é obtida através do mapeamento de seus coeficientes PAA para índices do alfabeto. Cada valor x_i da representação PAA é convertido para um símbolo baseado nos pontos de corte, resultando em uma sequência de números inteiros entre 1 e a. Por exemplo, na Figura 9 uma série temporal transformada para PAA com $w = 8$ e alfabeto de tamanho $a = 4$ poderia resultar na sequência $(1, 2, 1, 3, 1)$, onde cada número representa um símbolo do alfabeto SAX.

Para comparar séries no domínio SAX, pode-se utilizar a função **MINDIST** (Keogh et al., 2001), que preserva a propriedade de *lower bounding* em relação à distância euclidiana. No

entanto, para aplicações de classificação, é possível utilizar as funções de distância tradicionais diretamente sobre a representação numérica SAX.

A representação SAX tem se mostrado particularmente eficaz em tarefas como detecção de anomalias, onde séries temporais anômalas são identificadas por apresentarem padrões simbólicos menos frequentes no conjunto de dados.

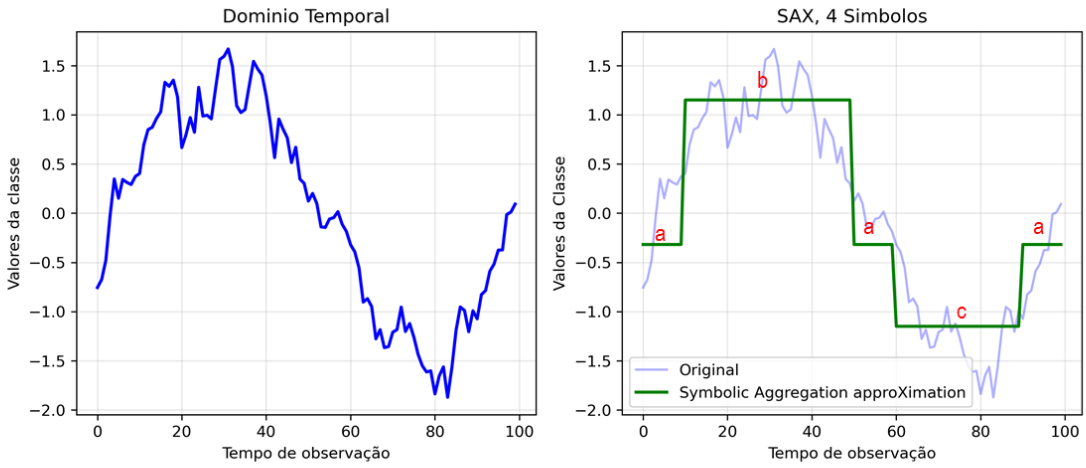


Figura 9: A representação SAX agrupa séries temporais contínuas em intervalos, transformando independentemente cada série temporal em uma sequência de símbolos, neste caso, letras.

2.3 Classificadores em Aprendizado de Máquina

Classificadores são modelos computacionais capazes de aprender a partir de características (atributos) de dados e fazer previsões sobre novos dados (Das e Behera, 2017). Sua capacidade de aprendizagem se dá por treinamento, onde são apresentados exemplos já rotulados (que possuem a classe ou categoria correta). Após a etapa de treinamento, o classificador aprendeu os atributos necessários para fazer previsões de novos dados não rotulados, fornecendo dessa forma uma classificação ou probabilidade de pertencer a uma dada categoria. O classificador pode ser avaliado de acordo com seu desempenho, utilizando métricas como precisão, *F1-Score*, *Recall* entre outras.

2.3.1 K-Vizinhos Mais Próximos (k-NN)

O k-NN (Cover, 1968), um classificador baseado em instâncias e similaridade (Giusti e Batista, 2013), é uma boa opção para a classificação de séries temporais. Ao contrário de

outros modelos de classificação, o k-NN não exige uma etapa de treinamento e utiliza a regra dos k-vizinhos mais próximos para classificar um exemplo. No caso do k sendo 1, o classificador é chamado de 1-NN e pode ser adaptado para usar funções de dissimilaridade ou de similaridade. Para estabelecer parâmetros comparativos consistentes, adotamos o algoritmo K-NN (Cover, 1968) como *baseline*, aplicando-o em diferentes representações de dados. Esta estratégia não apenas serve como referência para os experimentos individuais, mas também auxilia na identificação da representação mais adequada para cada conjunto de dados.

2.3.2 Florestas Aleatórias

Um classificador que vem se destacando em TSC são as árvores de decisão (Tu e Chung, 1992). Uma árvore de decisão é um classificador que utiliza uma estrutura hierárquica como processo de tomada de decisão, sendo composta por raiz, nós internos e folhas. A raiz representa o ponto inicial de decisão, os nós internos correspondem a testes intermediários com possíveis ramificações e as folhas determinam a classificação final. O caminho da raiz até um nó folha, determinado pela aplicação das condições dos nós internos aos valores dos atributos de um exemplo qualquer, determina como esse exemplo será classificado pela árvore.

A partir deste conceito, surgem as florestas aleatórias (Breiman, 1984). São chamadas florestas por combinarem múltiplas árvores de decisão. O modelo aprende a construir e otimizar um conjunto de árvores de decisão a partir dos dados de entrada, resultando num classificador mais robusto e adaptativo. Na Figura 10, cada árvore recebe a mesma instância de entrada e realiza a sua própria classificação. A decisão final é alcançada por meio de um sistema de votação majoritária, onde a classe com mais votos entre as árvores é selecionada como resultado. Neste trabalho, serão utilizados conjuntos de florestas aleatórias, usados em TSC para categorizar dados através da aprendizagem de regras de decisão com base nos seus atributos. Estes atributos podem ser tendências, ciclos, sazonalidade etc.

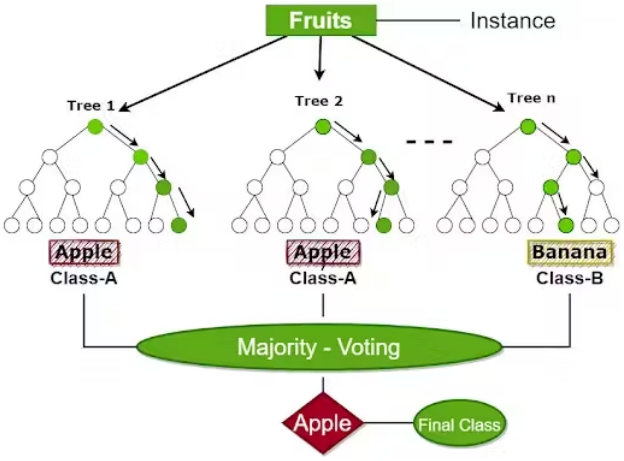


Figura 10: Exemplo da estrutura que compõe um classificador Random Forest (Mohit Chaudhary, 2023).

2.3.3 Máquina de Vetores de Suporte (SVM)

A Máquina de Vetores de Suporte ou *Support Vector Machine* (SVM) (Cortes e Vapnik, 1995) é um algoritmo de classificação que cria um hiperplano ou conjunto de hiperplanos em um espaço de alta dimensionalidade para distinguir entre pares de classes. Na Figura 11 é ilustrada uma divisão linear do hiperplano separando os dados da classe de bolinhas azuis das vermelhas. O objetivo é encontrar um hiperplano que aumente a diferença entre as classes e ajude a generalizar dados que não são conhecidos.

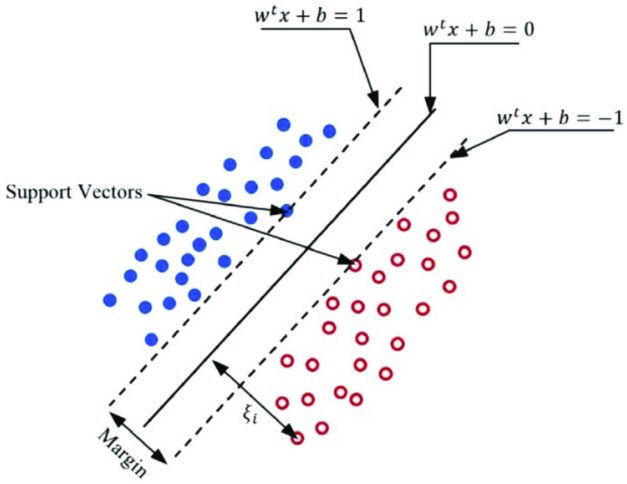


Figura 11: Ilustração do uso do hiperplano para separação de classes utilizada pelo SVM (Oyebode e Ighravwe, 2019).

Neste trabalho, o SVM é usado para analisar diferentes funções de kernel e representações eficientes de séries temporais para distinguir características de classes em um determinado domínio de dados.

2.3.4 Redes Neurais Convolucionais (CNN)

Muitos pesquisadores na área de processamento e reconhecimento de imagens (Chauhan et al., 2018) usam redes neurais convolucionais. A Figura 12 ilustra uma rede de convolução de uma dimensão. Em cada camada, um filtro (*kernel*) "desliza" sobre uma série temporal de entrada e calcula o produto escalar entre o filtro e cada subconjunto da série temporal para produzir uma nova matriz. O resultado dessa operação é um mapa de características que representa a resposta do filtro à entrada original. A convolução pode ser vista como um método de aplicação de filtros espacialmente invariantes a imagens ou séries temporais, permitindo que a rede extraia características relevantes independentemente da posição na imagem ou série temporal.

Na Figura 13, é possível visualizar o processo de convolução em redes neurais aplicadas a imagens, uma técnica amplamente utilizada em processamento de imagens. No entanto, essas redes não possuem uma aplicação tão ampla no campo de estudo de séries temporais (Nielsen,

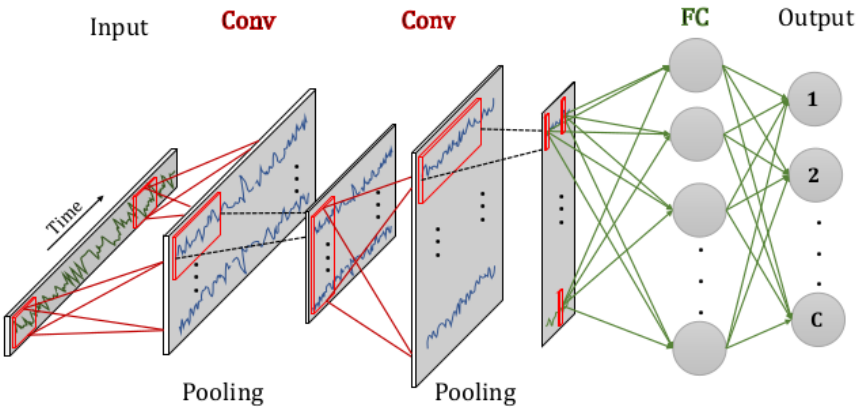


Figura 12: Exemplo de uma arquitetura de uma rede neural convolucional (Foumani et al., 2023).

2021), pois tratam todas as regiões do espaço igualmente, o que nem sempre é adequado para dados temporais, onde a proximidade entre os pontos pode variar significativamente. Devido a essa característica, é necessário estruturar os dados de forma específica para os modelos CNN. Para isso, foi aplicada a técnica de análise de imagens conhecida como gráfico de recorrência.

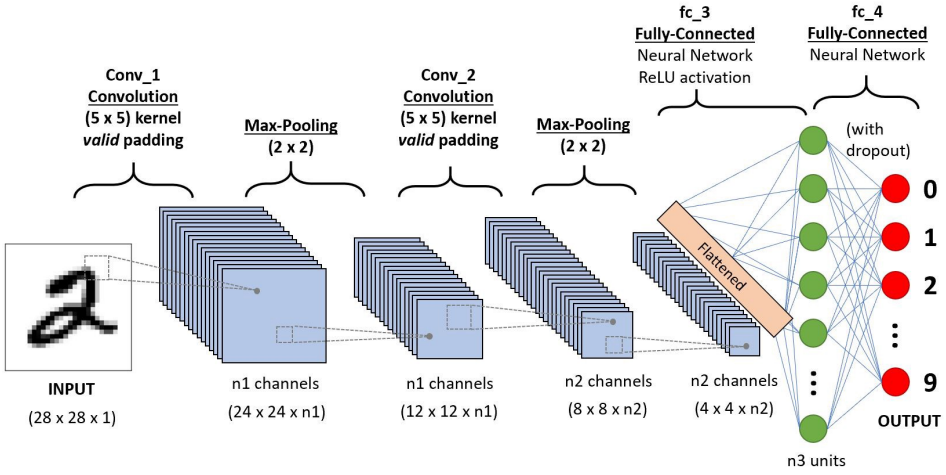


Figura 13: Exemplo de uma arquitetura de uma rede neural convolucional de 2 dimensões (Foumani et al., 2023).

Apresentado na Figura 14, o gráfico de recorrência (Marwan et al., 2002) é uma ferramenta para mapear a trajetória de um espaço de fase bidimensional para um espaço m-dimensional, destacando suas recorrências. O gráfico é obtido a partir de uma matriz que tem valor $d(i, j) = 1$ se as observações $Z[i]$ e $Z[j]$ da série temporal são próximas (isto é, se $|Z[i] - Z[j]| < \epsilon$, no qual ϵ é um limiar pré-definido). Dessa forma, o gráfico de recorrência pode representar padrões de tendência e sazonalidade da série temporal.

A utilização de gráficos de recorrência nos permite criar um novo conjunto de dados multivariado, onde cada "canal" da imagem representa uma característica diferente dos dados. Deste jeito, ao utilizar diferentes representações da mesma série temporal como canais na

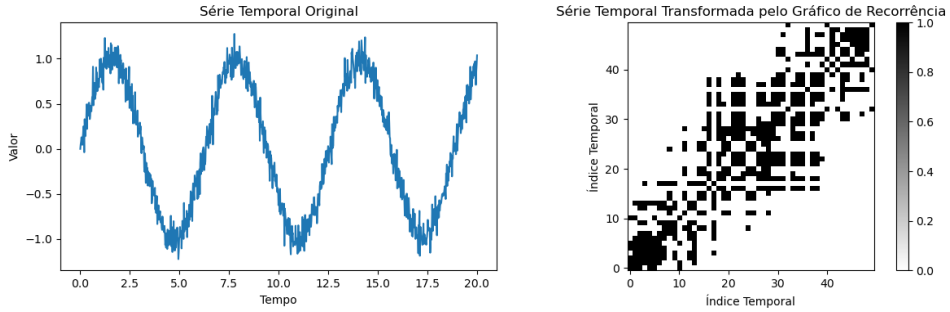


Figura 14: Uma série temporal e sua representação utilizando um gráfico de recorrência. - Elaborado pelo autor.

imagem, podemos aumentar a diversidade das características disponíveis para o classificador. Isso pode levar a um ganho no poder de predição do classificador.

A criação desse conjunto foi feita para validar que a utilização de diferentes representações poderia melhorar o desempenho do modelo. Utilizando a técnica de análise de dados conhecida como *Feature Importance*, onde ela é usada para atribuir um *score* para cada característica do modelo, indicando quão útil ou valiosa cada uma é na previsão da classe alvo. Essas pontuações ajudam a entender o impacto de cada característica no modelo (Molnar et al., 2024). Desta maneira, temos uma forma mais intuitiva de ver a presença de diferentes características afetando a previsão de saída do modelo.

2.4 Considerações Finais

Uma série temporal pode ser vista como um conjunto de observações ao longo do tempo. No entanto, no domínio temporal, algumas características não são facilmente descritas. As séries temporais podem ser usadas para representar modelos muito detalhados de coisas ou ocorrências que ocorreram durante um período. Às vezes, os dados de séries temporais contêm informações que as medidas de comparação não conseguem captar. Nesses casos, os dados podem ser transformados em outro domínio, para que mais informações possam ser analisadas. Algumas representações de séries temporais são criadas para tornar os dados compatíveis com outras estratégias de análise de dados (não temporais). Alguns exemplos desses tipos de representações de séries temporais são *SAX* e *PAA*. Ambas reduzem o número de dimensões na série temporal, tornando-a mais gerenciável. Existem outras representações de séries temporais para mostrar aspectos específicos dos dados; por exemplo, a representação *SAX* destina-se a permitir que cientistas que desenvolveram muitas teorias sobre manipulação de *strings* possam usar seus conhecimentos em algum ponto do processo de mineração de dados temporal. Sempre há alguma maneira de mostrar os dados da série temporal que dá uma impressão diferente dos dados. Mesmo que não haja exposição identificada, a série temporal ainda é mostrada de forma diferente.

Capítulo 3

Classificação de Séries Temporais

O TSC é uma tarefa de prever um rótulo discreto para uma série temporal não rotulada. O objetivo principal da tarefa TSC pode ser abordado treinando qualquer tipo de modelo (raso ou profundo) em um determinado conjunto de dados para o modelo aprender a distribuição de probabilidade do conjunto de dados fornecido. Este trabalho se concentra na utilização de séries temporais univariadas de comprimentos fixos. Ao longo deste capítulo serão descritos os modelos que mais se assemelham a este trabalho e que são muito utilizados para comparação de métricas de eficiência na tarefa de classificação de séries temporais, utilizando os dados disponíveis no repositório de pesquisa da *University of California, Riverside* (UCR) (Dau et al., 2019). O repositório UCR é o maior repositório de dados temporais utilizados por diversos pesquisadores da área de processamento e classificação de séries temporais. Por fim, será apresentado um teste de *benchmarking* para avaliar o tempo de execução dos classificadores apresentados. Este teste visa demonstrar a relação entre a acurácia obtida e o custo computacional de cada método, permitindo uma análise comparativa de seu desempenho.

3.1 Métodos de classificação de séries temporais

A classificação de séries temporais tem sido abordada por meio de diversos métodos ao longo dos anos. Na literatura, encontram-se desde abordagens tradicionais, como os modelos ARIMA (Newbold, 1983) e lineares (Jolliffe, 2002), que estabeleceram as bases fundamentais para análise temporal, até técnicas mais sofisticadas baseadas em algoritmos de aprendizado de máquina (Guo et al., 2003; Nanopoulos et al., 2001). Estes métodos evoluíram para atender diferentes necessidades, cada um apresentando características distintas em termos de precisão, complexidade computacional e interpretabilidade dos resultados, que os tornam mais ou menos adequados dependendo do contexto específico e da natureza dos dados temporais. A seguir são apresentados alguns métodos de classificação propostos na literatura:

- **Métodos baseados em distância** utilizam vários tipos de métricas de distância para classificar os dados corretamente. Uma função de distância mais utilizada é a distân-

cia euclidiana, que tem sua definição matemática como a distância entre dois pontos no espaço cartesiano e pode ser provada pela aplicação repetida do teorema de Pitágoras (Allman, 1889). A distância euclidiana possui suas variações como a Distância de *Chebyshev*, que mede a distância assumindo que apenas a dimensão mais significativa é relevante; a distância de *Manhattan*, calculada pela soma absoluta do deslocamento em cada eixo; e a distância de *Minkowski*, uma generalização que unifica a distância euclidiana, a distância de *Manhattan* e a distância de *Chebyshev* (Klamroth, 2006).

- **Métodos baseados em dicionário** constituem outra categoria de classificadores usados em séries temporais. Esta abordagem consiste em converter séries temporais numa sequência de símbolos, das quais "palavras" são extraídas por meio de uma janela deslizante. A intuição por trás dessa abordagem é que as séries temporais são semelhantes, o que significa que provavelmente pertencem à mesma classe, se contiverem palavras semelhantes. O principal processo de um classificador baseado em dicionário consiste em executar uma janela deslizante de comprimento específico sobre as séries temporais. Transformando cada *substring* numa palavra (com um comprimento específico e um conjunto fixo de letras). Por fim, é criado um histograma das palavras. A classificação pode ser feita, por exemplo, por meio de histogramas ou por técnicas de Processamento de Linguagem Natural (PLN), como *Bag-of-Words* (BOW) (Zhang et al., 2010).

- **Métodos baseados em características** geralmente podem consistir na maioria dos algoritmos que usam algum tipo de extração de características em uma determinada série temporal. Com relação às características, quase tudo é possível, desde características estatísticas simples até características mais complexas baseadas em Fourier. Em (Fulcher e Jones, 2017) é explorada uma vasta lista de características que podem ser utilizadas. No entanto, a busca por características ótimas é considerada uma tarefa computacionalmente intensiva, exigindo diversos recursos significativos de *hardware*.

- **Os conjuntos de modelos** não constituem um tipo de método distinto, mas sim uma técnica para combinar vários tipos de classificadores, visando criar uma previsão combinada mais precisa. Há vários trabalhos na literatura (Lines et al., 2018; Middlehurst et al., 2021b; Frank et al., 2021; Deng et al., 2013; Shifaz et al., 2020) que utilizam esse tipo de técnica. A utilização de conjuntos pode ser justificada pela busca do equilíbrio entre viés e variância. A redução do viés pode ser alcançada através da combinação de vários modelos individuais e o aumento da variância se dá pelo uso de diversos modelos diferentes. Isto resulta em um conjunto mais amplo e diversificado de possibilidades de aprendizado, podendo, por sua vez, melhorar o poder de discriminação das classes.

- **Métodos baseados em aprendizado profundo** (LeCun et al., 2015), apesar de serem extensivamente estudados em áreas como visão computacional e PLN, ainda não foram amplamente explorados em tarefas de TSC. No entanto, em um estudo proposto por

Fawaz et al. (2019a), mais de 60 modelos de redes neurais (NN) com seis tipos diferentes de arquitetura foram examinados: perceptrons multicamadas (Peng, 2017), redes neurais totalmente convolucionais (CNNs) (Gamboa, 2017), redes de estado de eco (baseadas em redes neurais recorrentes) (Ma et al., 2016), Codificadores (Serrà et al., 2018), CNNs profundas multiescala (Cui et al., 2016) e CNNs temporais (Zhao et al., 2017)

Existem diversos modelos de classificação de séries temporais, que se distinguem pela abordagem utilizada para a atividade de TSC. A seguir, são apresentados alguns dos principais classificadores propostos, divididos por sua abordagem.

3.1.1 Baseado em distância

Time-Box

O trabalho proposto por (Giusti, 2017) tem como ideia principal o estudo de representações e distâncias que normalmente não são o foco na literatura de classificação de séries temporais. Todas as representações e funções de distância são tratadas como espaços de busca. O objetivo central da pesquisa é desenvolver um método de conjuntos que combine diferentes versões do k-NN com variadas métricas (Faloutsos et al., 1994; Cha, 2007; Berndt e Clifford, 1994; Pranti e Batista, 2012; Assent et al., 2009; Rakthanmanon et al., 2012) para melhorar a acurácia do resultado dos classificadores. Os experimentos realizados nessa pesquisa foram feitos comparando diferentes representações de dados e funções de distância em termos de acurácia e eficiência, avaliando também se o conjunto de treinamento utilizado influenciaria o número de classificadores no desempenho do conjunto proposto. Foram utilizados os conjuntos de dados do repositório UCR (Bagnall et al., 2018) para verificar a eficácia da proposta comparada a outros métodos do estado da arte. O resultado foi um classificador capaz de lidar com diversas representações de dados e com uma acurácia competitiva, superando até outros métodos do estado da arte, como o classificador 1-NN aliado à distância **DTW** (*Dynamic Time Warping*), em vários conjuntos de dados daquela época.

3.1.2 Baseado em algoritmos de aprendizagem profunda

Inception Time

Inception Time, proposto por (Fawaz et al., 2020), é um conjunto de aprendizagem profunda que consiste em cinco modelos utilizados em TSC. Cada modelo tem a mesma arquitetura, mas com diferentes valores de peso inicializados aleatoriamente. No núcleo de cada classificador, o módulo *Inception* aplica vários filtros simultaneamente a uma série temporal de entrada e inclui filtros de comprimentos variáveis. Isto permite à rede extrair características relevantes de dados de séries temporais longas e curtas.

A rede *Inception* para classificação de séries temporais tem dois blocos residuais, cada um contendo três módulos *Inception* em vez das tradicionais camadas convolutivas. A entrada

para cada bloco é adicionada à entrada do bloco seguinte por uma ligação linear de atalho, ajudando a resolver o problema da fuga de gradiente, permitindo um fluxo direto de gradientes. Depois destes blocos, existe uma camada de *Pooling* Média Global que calcula a média das dimensões globais de saída e, finalmente, uma camada de *Softmax* totalmente ligada com neurônios iguais às classes do conjunto de dados.

A Figura 15 ilustra o módulo *Inception*. É um componente de uma rede de classificação de séries temporais que reduz a dimensionalidade e complexidade dos dados de entrada, utilizando filtros deslizantes em uma camada *Bottleneck*. O objetivo do *Bottleneck* é reduzir a dimensionalidade dos dados de entrada antes de aplicar as convoluções, reduzindo assim a carga computacional e evitando *overfitting*. Desta forma, a utilização de filtros mais longos, sem aumentar o número de parâmetros, resulta em um melhor desempenho em pequenos conjuntos de dados. O módulo também aplica múltiplas convoluções paralelas com comprimentos variáveis e operações de *MaxPooling* para extrair características hierárquicas em diferentes resoluções. O empilhamento de múltiplos módulos permite a extração de características ainda mais complexas por meio de treino de *backpropagation*.

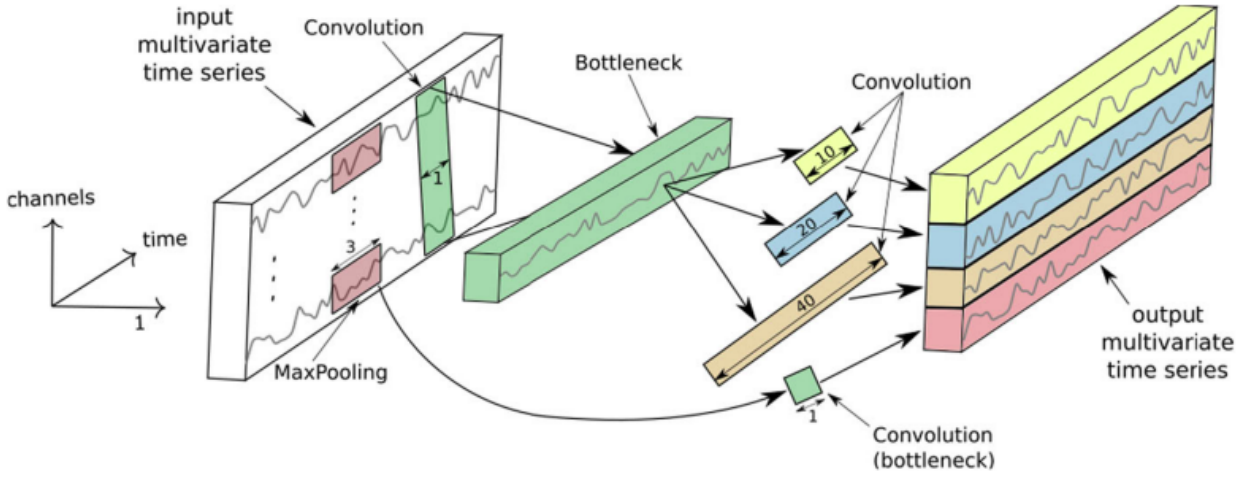


Figura 15: Representação gráfica do classificador *InceptionTime*. Figura retirada do artigo (Lines et al., 2018)

3.1.3 Baseado em algoritmos de convolução

Rocket

O algoritmo *RandOm Convolutional KErnel Transform* (ROCKET), proposto por (Dempster et al., 2022), produz estatísticas resumidas usando núcleos convolucionais inicializados aleatoriamente e as utiliza para treinar um classificador linear. O método envolve a transformação de séries temporais usando núcleos convolucionais aleatórios e o uso das características transformadas para treinar um classificador linear. São utilizados núcleos convolucionais aleatórios como uma alternativa simples e eficiente aos métodos de classificação de séries temporais mais complexos.

Quando se fala em *kernels*, referem-se a núcleos convolucionais usados para transformar dados de séries temporais e capturar características relevantes para a classificação de séries temporais. No Rocket, esses *kernels* são gerados aleatoriamente e usados em combinação para capturar várias características, diferentemente dos métodos existentes que se concentram em um único tipo de característica, como forma ou frequência. "Transformar" refere-se ao processo de usar núcleos convolucionais aleatórios para extrair características de dados de séries temporais. Na Figura 16 é possível visualizar os kernels sendo usados para treinar um classificador linear para predição de séries temporais. A transformação também é escalável para grandes conjuntos de dados, com complexidade de treinamento linear na duração da série temporal e no número de exemplos de treinamento.

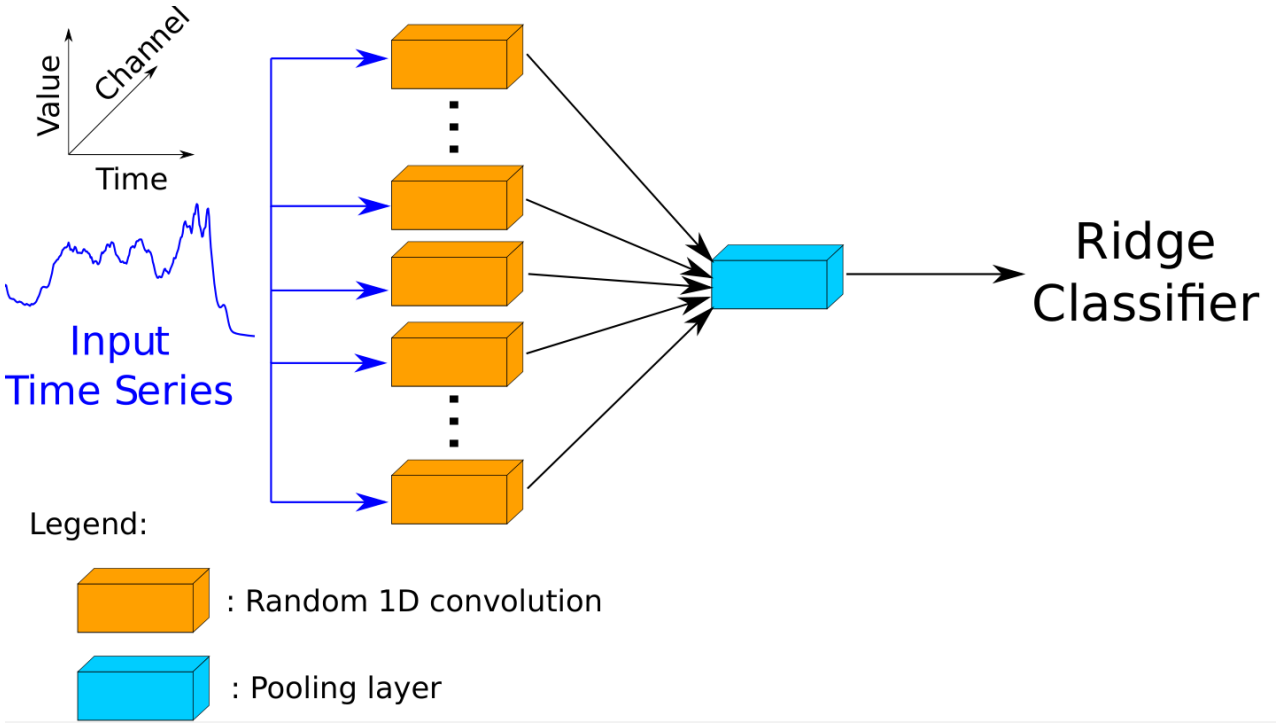


Figura 16: Representação gráfica de kernels de convolução aleatória utilizada no classificador Rocket(Dempster et al., 2022)

3.1.4 Baseado em características

Metal-Cats

Silva et al. (2022) propõem uma abordagem chamada Algoritmo de Seleção de Classificação Baseado em Meta-Aprendizado para Séries Temporais (Metal-CATS), ilustrado na Figura 17. O processo envolve a extração de medidas estatísticas descritivas simples para treinar um modelo capaz de recomendar o melhor algoritmo para um determinado conjunto de dados.

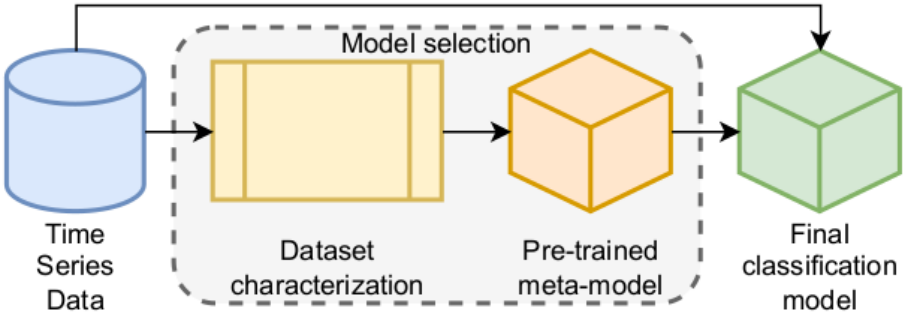


Figura 17: Representação da estrutura que compõem o classificador Metal-CATS. Figura retirada do artigo (Silva et al., 2022)

As características disponíveis no repositório UCR são usadas para avaliar vários classificadores com base em sua precisão e tempo médio de treinamento. O melhor algoritmo para lidar com um conjunto de dados é aquele que obtém os melhores resultados em seu conjunto de testes.

Os autores comparam o algoritmo proposto *Metal-CATS* com outros algoritmos de TSC existentes. Eles usam um conjunto de dez algoritmos, incluindo TS-CHIEF (Shifaz et al., 2020), Hive-COTE 1.0 (Lines et al., 2018), *Rocket* (Dempster et al., 2022), *Inception Time* (Fawaz et al., 2020), *Shapelet Transform Classifier* (STC)(Lines et al., 2012), ResNet (Wang et al., 2017), *Proximity Forest*(Lucas et al., 2019), *Word Extraction for Time Series Classification* (WEASEL)(Shafer e Leser, 2017), *Bag of Symbolic-Fourier Approximation Symbolics* (BOSS)(Schäfer, 2015) e *Spatial BOSS* (S-BOSS) (Large et al., 2019a). Para comparar o desempenho do *METAL-CATS* com outros algoritmos, é utilizada a precisão média entre todos os algoritmos concorrentes, que simula o desempenho de uma escolha aleatória de modelos TSC. O meta-alvo usado para avaliar o *Metal-CATS* é o índice do classificador que obteve a melhor precisão. Em caso de empate, o rótulo atribuído se refere ao algoritmo com menor tempo de execução médio. O meta-modelo visa prever o algoritmo com melhor precisão no conjunto de treino e teste, beneficiando os mais eficientes.

Catch22

Lubba et al. (2019) propõem uma nova técnica de geração de características, chamada *Canonical Time-Series Characteristics* (Catch22). É um conjunto de características canônicas que visam fornecer uma representação mais clara e eficiente das séries temporais. O termo canônico é empregado pelos autores para indicar que as 22 características resultantes do processo de geração representam um conjunto suficiente para aumentar consideravelmente o desempenho dos classificadores. A Figura 18 apresenta como o processo de criação das características usadas no Catch22 envolve uma análise aprofundada para identificar características eficazes e representativas para TSC. A biblioteca HCTSA (Fulcher e Jones, 2017) é usada para criar aproximadamente 7500 características para séries temporais, como, por

exemplo, autocorrelação linear e não linear, distribuições de valores, *outliers*, e outras propriedades. Dessa maneira, os dados são submetidos a diversas alterações, como a remoção de redundâncias, a análise de agrupamentos de dados, dentre outras técnicas para evitar que características que requerem um processamento complexo e oferecem melhorias mínimas de desempenho sejam consideradas. Ao final do processo, são selecionadas 22 características, os quais são consideradas robustas, representativas e eficazes em TSC.

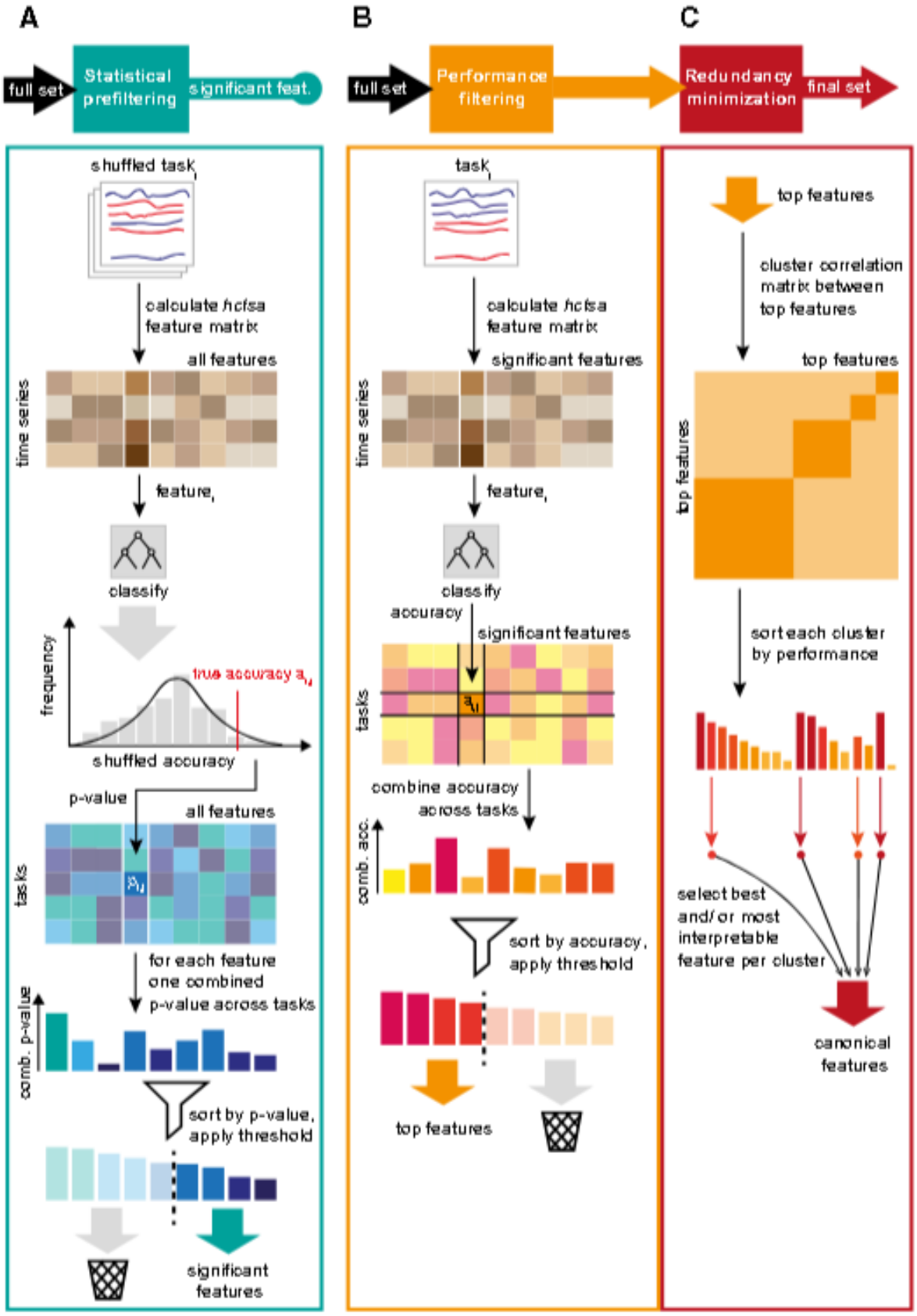


Figura 18: Fluxo das etapas de processamento e refinamento para a geração das características utilizadas no Catch22 (Lubba et al., 2019).

TSFresh e FreshPRINCE

A biblioteca *Time Series Feature Extraction of Scalable Hypothesis Tests* (TSFresh) (Christ et al., 2018) é usada para pré-processamento de dados, com foco na extração de características de séries temporais. São empregados 63 métodos para caracterizar séries temporais, resultando em uma grande variedade de informações, como características de tempo, autocorrelação, frequência etc. O *TSFresh* também pode ser usado como classificador, realizando a extração e seleção de características, utilizando uma árvore de decisão para realizar as predições. O problema de gerar as características é o alto custo de processamento em algumas operações e muitos dos resultados podem resultar em características que não serão usadas na classificação. Baseado na técnica de pré-processamento criada pelo *TSFresh*, foi proposto o classificador *Fresh Pipeline With Rotation Forest Classifier* (FreshPRINCE) (Middlehurst e Bagnall, 2022). Tem como premissa a ideia de criar novas características de dados por meio da inclusão de um ensemble chamado *Rotation Forest* (RotF) (Rodriguez et al., 2006). Esse classificador tem uma abordagem de divisão e rotação aleatória de características geradas pelas árvores aleatórias utilizadas como classificadores-base. Dessa forma, surgem novas perspectivas dos dados para os classificadores presentes na base, aumentando o poder de generalização. O resultado é alcançado por meio do voto majoritário. Na Figura 19 é apresentado o fluxo de geração e predição do classificador FreshPRINCE.

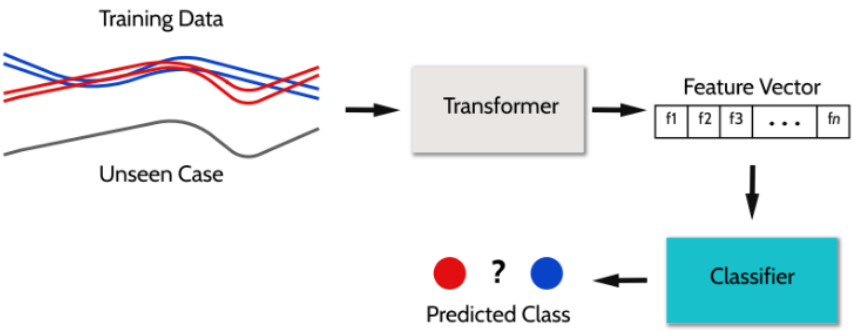


Figura 19: O fluxo de dados empregado por TSFresh e FreshPRINCE para transformação, geração e previsão de dados (Middlehurst e Bagnall, 2022).

3.1.5 Baseado em conjuntos híbridos de classificadores

TS-Chief

Proposto por (Shifaz et al., 2020), o *Time Series Combination of Heterogeneous and Integrated Embeddings Forest* (TS-CHIEF), ilustrado na Figura 20 é um conjunto de árvores que incorpora características de distância, dicionário e base espectral, com diferentes tipos de critérios de divisão. O TS-CHIEF rivaliza diretamente com o Hive-COTE (Lines et al., 2018) em precisão, mas requer apenas uma fração do tempo de execução. O Hive-COTE reconhece

que os dados de séries temporais são um tipo de dados específico para o qual a representação tradicional do valor do atributo, usada predominantemente no aprendizado de máquina, falha em fornecer uma representação relevante. O TS-CHIEF constrói um classificador de conjunto que integra as incorporações mais eficazes de séries temporais como similaridade, dicionarização, intervalos de janela, etc.

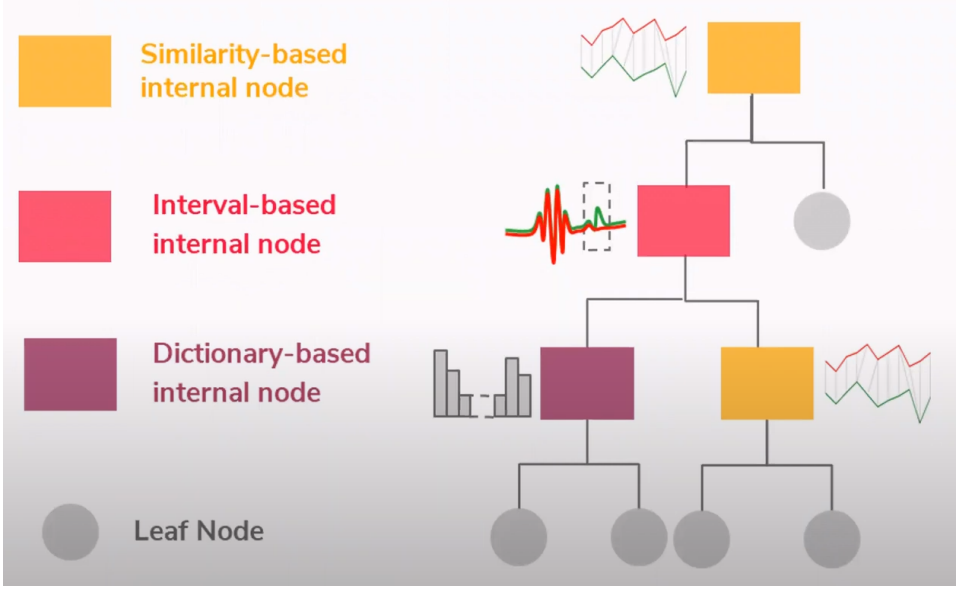


Figura 20: Representação gráfica da árvore de decisão aleatória utilizada no classificador TS-Chief (Shifaz et al., 2020; Foumani et al., 2023)

Ele usa classificadores estruturados em árvore para fazer isso eficientemente. Cada um extrai informações sobre um aspecto específico de uma série temporal, seja no domínio do tempo, domínio da frequência ou resumo de intervalos na série. Os experimentos realizados em 85 conjuntos de dados (Bagnall et al., 2018) mostram que o TS-CHIEF atinge uma precisão de ponta que rivaliza com o Hive-COTE, mas com escalabilidade e eficiência muito melhores. O método Hive-COTE será apresentado no decorrer desta seção. Os autores explicam que o TS-CHIEF pode ser treinado em 130.000 séries temporais em 2 dias, enquanto o Hive-COTE leva 8 dias para aprender com apenas 1500 séries temporais. O TS-CHIEF oferece uma estrutura para classificação de séries temporais que os pesquisadores podem facilmente integrar com novas transformações e medidas de similaridade e aplicá-las em formato de árvore. Destacando possíveis melhorias, como melhorar a compensação entre o tempo de computação e o espaço ocupado pela memória, incorporar informações de diferentes tipos de divisores potenciais e encontrar uma maneira automática de equilibrar o número de divisores candidatos considerados para cada tipo.

Hive-COTE v1.0

Proposto por (Lines et al., 2018) *Collective Transformation Ensembles* (COTE) é um conjunto hierárquico que combina diferentes estratégias de classificação utilizando quatro repre-

sentações de séries temporais. O ensemble contém classificadores construídos nos domínios de tempo, frequência, janela e transformação de formas. Os autores também apresentam um modelo denominado *flat-COTE*, o qual é uma versão não hierárquica dos mesmos classificadores-base empregados no COTE.

O *Hive-COTE* (HC) foi projetado para ser mais flexível e menos tendencioso do que o *Flat-COTE*, tornando-o um classificador mais preciso e confiável para problemas de TSC. Lines et al. (2018) propuseram uma comparação de duas abordagens diferentes, a primeira sendo o *Flat-COTE* e a outra utilizando aprendizado profundo. Dessa forma, foram criadas duas redes neurais convolucionais (CNN), uma usada para referência e outra desenvolvida específica para problemas de TSC e as compararam com *Flat-COTE*. A análise mostrou que o *Flat-COTE* superou ambas as CNN em termos de precisão em 85 conjuntos de dados do repositório UCR e 44 conjuntos de dados relatados em (Cui et al., 2016). Assim, com intuito de melhorar o *Flat-COTE* e criar um conjunto mais preciso, foi introduzido um novo meta-ensemble chamado *Hive-COTE*. Dessa forma, o novo ensemble é treinado em um conjunto de dados contendo várias tarefas de aprendizado e, em seguida, pode ser aplicado a novos conjuntos de dados, permitindo que aprenda mais rapidamente e com melhores resultados do que se fosse treinado a partir do zero para cada tarefa individual.

Nesse novo meta-*ensemble* a estrutura hierárquica é atualizada com classificadores constituintes que podem detectar cinco tipos de características discriminatórias - séries inteiras, formato, dicionário, intervalo e espectral. Os pesquisadores também definiram um novo classificador de séries temporais, o *Hierarchical Ensemble of Shapelet-based Classifiers* (HESCA) (Hills et al., 2014; Bostrom e Bagnall, 2015). Este classificador utiliza transformações de *shapelet* para obter características discriminatórias de subséries independentes de fase, ou seja, não é considerada a posição inicial exata da série temporal durante a extração.

Foi criado outro classificador semelhante ao HESCA, o*Random Interval Spectral Ensemble*(RISE) (Caiado et al., 2006; Deng et al., 2013; Lines et al., 2018; Middlehurst et al., 2021b). Este classificador utiliza a mesma técnica de extração de características descrita no HESCA, porém aplicando transformações espectrais aleatórias. É considerado um dos classificadores mais precisos, utilizando técnicas espectrais. Por fim, essas novas características geradas são combinadas e utilizadas no HC. A Figura 21 demonstra a estrutura do conjunto Hive-COTE.

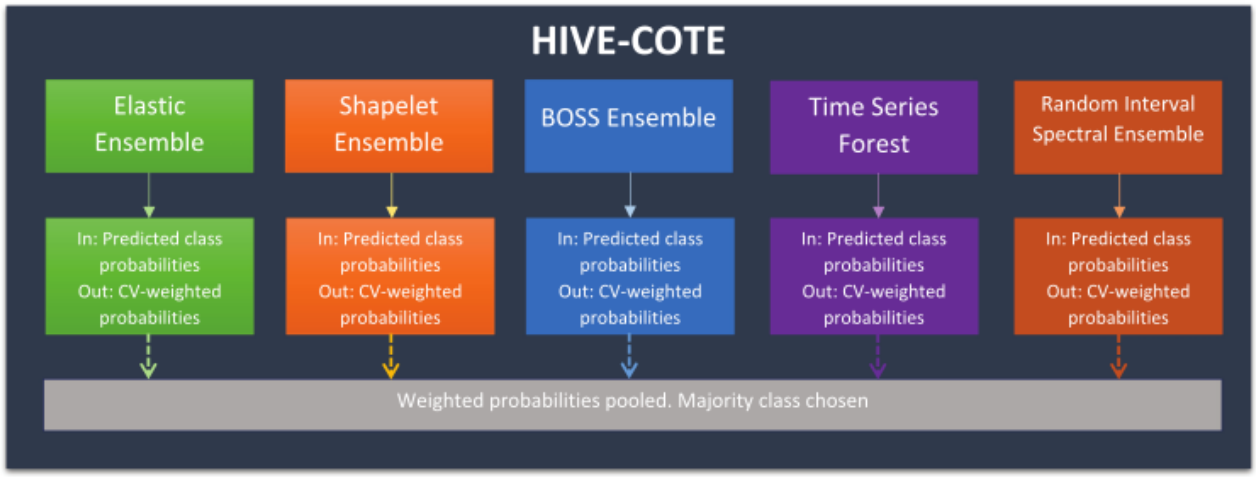


Figura 21: Representação gráfica da estrutura dos classificadores que compõe o classificador Hive-COTE (Lines et al., 2018)

Os classificadores no HC são encapsulados em módulos e combinados por meio de uma estrutura hierárquica de votação probabilística. Os pesquisadores experimentaram extensivamente 100 re-amostras de 85 conjuntos de dados públicos, 5 tipos de dados simulados e 3 novos estudos de caso para demonstrar que o HC é significativamente mais preciso do que todas as alternativas para TSC, incluindo *Flat-COTE*.

O classificador HC é uma combinação de cinco módulos diferentes que usam várias técnicas para classificar séries temporais. O algoritmo resultante contém módulos que capturam a semelhança com medidas de séries inteiras (*Elastic Ensemble* (EE) (Lines e Bagnall, 2015), subséries independentes de fase (*Shapelet Transform* (ST-HESCA), um conjunto baseado em intervalos (*Time Series Forest* (TSF))(Deng et al., 2013), um conjunto de dicionário (*Bag-of-SFA-Symbols*) (Schäfer, 2015) e o novo conjunto espectral, RISE, que usa uma combinação de características de função de autocorrelação (ACF) e espectro de potência (PS). No geral, foram comparados esses diferentes métodos e descoberto que o método proposto é significativamente mais preciso que ambas as CNN e o *Flat-COTE*.

Hive-COTE v2.0

O *Hierarchical Vote Collective of Transformation-based Ensembles Version 2.0* (Hive-COTE v2.0 ou HC2) apresentado na Figura 22, proposto por, (Middlehurst et al., 2021b) é uma versão atualizada do algoritmo Hive-COTE (Lines et al., 2018). Foram incluídos dois novos classificadores e *ensemble* de classificadores ROCKET (Dempster et al., 2022), resultando em uma precisão significativamente aprimorada em comparação com os algoritmos atuais de última geração em conjuntos de dados de séries temporais univariadas e multivariadas. Cada módulo do HC2 é treinado de forma independente, buscando otimizar sua capacidade de aprender diferentes características dos dados e produzir uma estimativa de probabilidade para cada classe. É utilizado o método *Cross-validation Accuracy Weighted Probabilistic Ensem-*

ble (CAWPE). Os classificadores-base no HC2 são escolhidos conforme a precisão estimada de cada classificador nos dados de treino. O CAWPE considera a probabilidade de cada classificador-base para criar uma distribuição ajustada com um parâmetro α que diminui as diferenças entre os classificadores. Dado que os classificadores mais precisos auxiliam nas decisões de classificação, o conjunto final torna-se mais eficiente (Middlehurst et al., 2021b). HC2 melhora significativamente sua precisão e usabilidade para classificação de séries temporais. Os autores apresentam dois novos classificadores e um conjunto de classificadores *Rocket* para substituir os membros existentes do conjunto anterior utilizado na primeira versão do Hive-COTE. Eles demonstram que o modelo atual supera o estado da arte em vários conjuntos de dados.

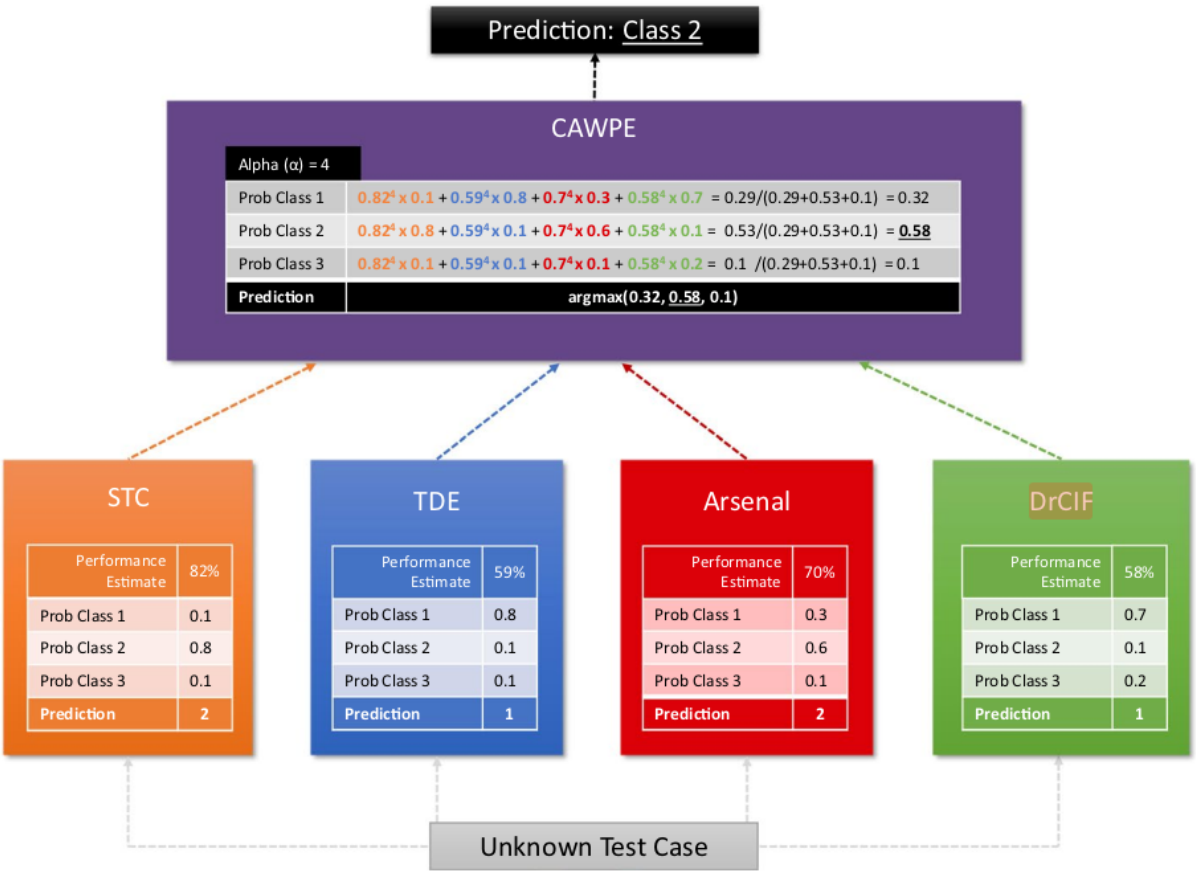


Figura 22: Representação gráfica da estrutura dos módulos que compõem o HC2. Figura retirada do artigo (Middlehurst et al., 2021b)

Os novos módulos componentes no Hive-COTE v2.0 são:

- *Shapelet Transform Classifier* (STC): é um classificador baseado em *Shapelet* que foi introduzido por (Yuan et al., 2022; Rodriguez et al., 2006).
- Arsenal: é um conjunto baseado na convolução de classificadores *Rocket* utilizados para extração de características.

- *Temporal Dictionary Ensemble* (TDE) (Schäfer, 2015; Middlehurst et al., 2021b,a; Lazebnik et al., 2006; Large et al., 2019b): Esta é uma representação baseada em dicionário utilizada para a classificação de séries cronológicas.

- *Diverse Representation Canonical Interval Forest* (DrCIF) (Middlehurst et al., 2020; Deng et al., 2013; Cabello et al., 2020; Lubba et al., 2019): Este é um classificador baseado em intervalos utilizado para a classificação de séries temporais.

HC2 se tornou um meta-conjunto heterogêneo para classificação de séries temporais. Os autores também mencionam que o HC tem sido o estado da arte em termos de precisão no repositório UCR desde sua proposta em 2016, mas outros algoritmos foram propostos que correspondem à sua precisão.

3.2 Tempo de Execução dos classificadores

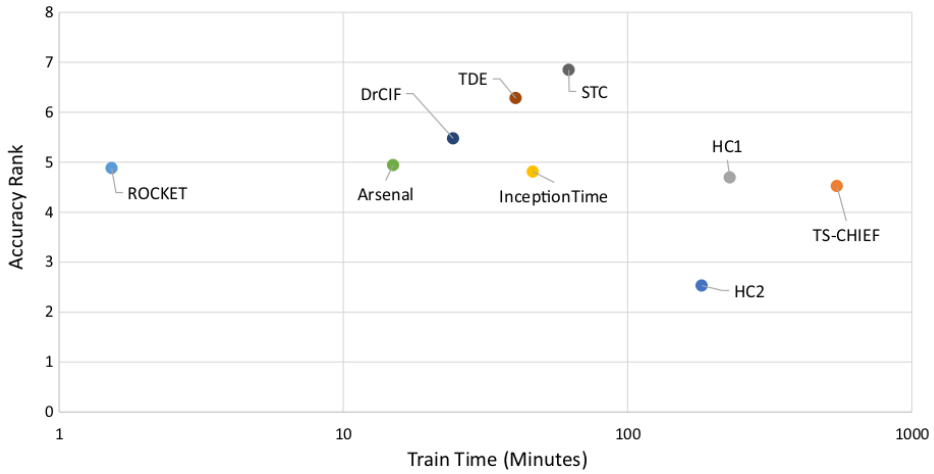


Figura 23: Comparação dos classificadores usando acurácia e tempo de treinamento. O tempo de treinamento é relativo à média de 112 conjuntos de dados multivariados, apresentados em escala logarítmica base 10 (Middlehurst et al., 2021b)

No teste de *benchmarking* proposto por (Middlehurst et al., 2021b), vários algoritmos foram avaliados com base em seu tempo de execução e acurácia, conforme apresentado na Figura 23. Foram comparados vários modelos, como *Rocket*, conjunto de modelos *Rocket* chamado de Arsenal, baseados em intervalos como o *DrCIF*, baseados em dicionário, sendo escolhido o *TDE*, *Inception Time*, baseados em *shapelets* como o STC, e por fim, híbridos como HC1 e HC2.

Embora o *Rocket* tenha pontuado como o mais rápido, sua precisão não foi eficientemente superior aos outros modelos. Por outro lado, modelos como *TDE, DrCIF e STC* tiveram tempos médios de execução, mas sua precisão foi relativamente baixa.

Apesar de *Arsenal* ser um conjunto de classificadores *Rocket*, ele é mais lento e não teve nenhuma melhoria significativa de precisão em comparação com o *Rocket* de referência. HC1,

HC2 e o TS-Chief são todos altamente acurados, mas possuem um alto custo de execução e necessitam de um *hardware* eficiente para que sua execução seja viável.

3.3 Considerações Finais

Vale enfatizar que todos os algoritmos discutidos neste capítulo enfrentaram suas respectivas limitações. Um fator que pode ter afetado seu desempenho são os dados disponíveis. Conforme detalhado em (Giusti, 2017), os conjuntos de dados usados no repositório UCR podem não ser totalmente representativos de outros contextos de aplicação.

O modelo Hive-COTE teve seus próprios desafios destacados em (Lines et al., 2018). Estes incluíram altos custos computacionais e a necessidade de ajustar parâmetros para cada conjunto de dados, bem como dificuldades na interpretação dos resultados e sensibilidade à qualidade dos dados de entrada e escolha dos algoritmos de base.

Outro modelo que apresentou um problema de escalabilidade foi o TS-Chief. Seu custo de treinamento é excepcionalmente alto devido às suas inúmeras abordagens. Sua falta de escalabilidade é uma grande desvantagem quando comparada aos demais. Enquanto isso, (Dempster et al., 2022) estabelece a necessidade de um número significativo de *Kernels* para obter uma classificação precisa, aumentando assim as características computacionais necessárias para alcançar uma boa acurácia. Em contrapartida, usar *Kernels* fixos pode resultar em *Overfitting* do classificador e, à medida que o conjunto de dados cresce, o custo computacional de treinamento também aumenta.

Em relação ao algoritmo (Fawaz et al., 2020), observa-se um problema de eficiência quando utilizado em conjuntos de dados com poucas amostras. Em sentido oposto, um conjunto de dados com muitas amostras acaba o deixando lento devido à quantidade de processamento necessário. A otimização dos hiper-parâmetros para cada conjunto de dados é crucial para superar esses problemas.

Vários trabalhos foram publicados nos últimos anos na tentativa de aprimorar a fronteira do "estado da arte". No entanto, ótimas características de todos os modelos apresentados podem se tornar algoritmos custosos dependendo do contexto em que são utilizados. As abordagens atuais, apesar de robustas, possuem grandes desafios na classificação de pequenos conjuntos de dados ou custo computacional elevado e necessidade de otimização constante. No Capítulo 4, um método diferente para classificação de dados será avaliado.

Capítulo 4

Meta-Classificador Dinâmico

Este capítulo discute a proposta geral deste trabalho. A Seção 4.1 apresenta-se a definição do escopo. Em sequência, a Seção 4.2 é apresentada a metodologia de pesquisa utilizada. Por fim, a Seção 4.3 apresenta os modelos desenvolvidos.

4.1 Definição do Escopo

Este trabalho investiga a classificação de séries temporais, concentrando-se especificamente nas representações temporais como ponto de melhoria na eficácia das classificações. A hipótese central é que a seleção da representação mais adequada para cada série temporal no conjunto de dados pode melhorar significativamente a precisão dos classificadores.

Cada conjunto de dados possui características e padrões únicos, que podem não ser adequadamente representados por uma única forma de representação. A abordagem proposta visa minimizar os efeitos limitantes de se aplicar uma única representação a todo o conjunto de dados, o que pode resultar em degradação da eficácia preditiva. Ao priorizar uma seleção dinâmica de representações para cada instância, buscamos uma modelagem mais flexível e adequada, capaz de lidar com uma ampla variedade de dados temporais.

4.2 Método de Pesquisa

Este estudo teve como objetivo coletar evidências empíricas de que, ao selecionar a melhor representação de dados para cada série temporal, pode-se melhorar a precisão de classificadores na análise de dados temporais em determinados domínios de aplicação. Para avaliar o desempenho dos classificadores propostos, utilizaremos a acurácia como métrica principal. Os resultados obtidos serão sistematicamente comparados com classificadores já estabelecidos na literatura, permitindo assim validar a hipótese central desta pesquisa.

4.2.1 Seleção e Análise Comparativa das Representações Temporais

Foi conduzido um amplo levantamento das técnicas de representação de séries temporais disponíveis na literatura. O processo envolveu a seleção de diversos conjuntos de representações, abrangendo análises no domínio do tempo, no domínio da frequência através da transformada de Fourier, além de decomposições *wavelets* e técnicas de redução de dimensionalidade. A seleção dos conjuntos de dados foi realizada com o intuito de garantir a variabilidade necessária para validar nossa hipótese inicial, abrangendo diferentes domínios de aplicação, como dados financeiros, sinais biomédicos e séries temporais de sensores industriais.

Em seguida, os conjuntos de dados originais foram transformados em diferentes representações, permitindo uma análise comparativa detalhada. A premissa adotada era que diferentes representações poderiam potencialmente destacar aspectos das séries temporais, promovendo uma visão mais detalhada para a classificação.

Cada representação foi processada e avaliada individualmente, com foco em compreender como distintos domínios de representação influenciam a classificação. Os experimentos preliminares, detalhados na Seção 5.3, demonstraram que a transformação para representações alternativas resultou em melhorias significativas no desempenho dos algoritmos.

Baseado nesses resultados, foi possível compreender as vantagens e limitações de cada domínio, sugerindo a viabilidade de uma escolha dinâmica de representações, de modo a maximizar a eficiência dos classificadores para cada série temporal específica.

4.2.2 Extração de características

A extração de características é uma etapa essencial no processo de treinamento e predição dos modelos propostos neste trabalho. As séries temporais são submetidas a um pré-processamento no qual são extraídas estatísticas descritivas, tais como soma, média, mediana, desvio-padrão, variância, mínimo, máximo, percentis, comprimento total da série, primeira e segunda derivadas da série. Para que o processo de extração de características seja desenvolvido, foi utilizada a biblioteca *TSFresh* (Christ et al., 2018), que otimizou o processo de extração das novas características. Essas funções estatísticas são aplicadas tanto em seu domínio temporal quanto nas diferentes representações utilizadas, como a *Fast Fourier Transform* (Shumway et al., 2000), *Discrete Wavelet Transform* (Chan e Fu, 1999), *Piecewise Aggregate Approximation* (Keogh et al., 2001) e *Symbolic Aggregate Approximation* (Lin et al., 2007). Neste trabalho, foram analisadas duas abordagens distintas para a extração de características, conforme descrito a seguir:

1. Empilhamento das Representações: esta abordagem, demonstrada na Tabela 2, não utiliza dados estatísticos, mas realiza o empilhamento das diferentes representações para cada instância do conjunto de dados, é garantido que exista pelo menos uma série extraída de cada representação, resultando em um vetor combinado que preserva a diversidade das informações entre as representações;

2. Estatísticas Combinadas: a seguinte abordagem, demonstrada na Tabela 3, utiliza dados estatísticos extraídos de todas as representações, concatenados em um único vetor de características. Diferentemente da primeira abordagem, as características aqui são compostas somente por estatísticas descritivas calculadas para cada representação, sem a separação explícita de qual representação gerou determinada característica. Essa abordagem busca generalizar as informações das diferentes representações em um único formato, o qual visa representar de forma genérica as informações extraídas dos dados.

id	**att_1**	**att_2**	**att_3**	**att_4**	**att_5**	**att_6**	**att_7**	**att_8**	TAG	**Target**
0	1.2	0.9	1.3	1.4	10.5	9.8	0.1	0.2	TS	Classe A
1	2.3	2.1	2.5	2.7	12.3	11.7	0.3	0.4	FFT	Classe B
2	1.5	1.2	1.6	1.8	11.1	10.4	0.2	0.3	DWT	Classe A
3	2.7	2.5	2.9	3.0	13.5	12.8	0.4	0.5	PAA	Classe A
4	3.7	1.5	5.9	1.0	9.5	8.8	1.4	7.5	SAX	Classe B

Tabela 2: Abordagem 1: Concatenação das Representações

id	**Soma**	**Média**	**Mediana**	**Desvio-Padrão**	**Variância**	**Mínimo**	**Máximo**	**Target**
0	15.2	5.2	5.0	2.1	4.41	1.0	10.5	Classe A
1	20.4	6.3	6.2	2.5	6.25	1.5	12.3	Classe B
2	18.0	5.9	5.7	2.3	5.29	1.3	11.1	Classe A
3	22.1	6.8	6.5	2.7	7.29	1.8	13.5	Classe B

Tabela 3: Abordagem 2: Estatísticas Combinadas

Ambas as abordagens apresentam diferentes formas de lidar com a identificação dos dados e suas características. O objetivo é avaliar se essas abordagens aprimoram o desempenho dos modelos, permitindo a integração eficiente de diferentes perspectivas dos dados e, assim, promovendo uma classificação mais precisa e robusta.

4.3 Meta-Classificador

Neste trabalho, desenvolveu-se um meta-classificador que analisa o desempenho de diversos algoritmos em diferentes tarefas, buscando compreender qual representação se mostra mais eficiente para cada instância no conjunto de dados. A meta-aprendizagem permite aprender a partir das experiências de diferentes técnicas de aprendizado, otimizando assim o processo de classificação (Vilalta e Drissi, 2002).

O meta-classificador fundamenta-se na premissa de que distintas representações de dados podem contribuir de maneiras diferentes para a classificação de séries temporais. Sua implementação estrutura-se em três etapas principais: pré-processamento dos dados, treinamento e inferência. Após testes comparativos, disponíveis na Seção 18, o *Support Vector Machine*

(SVM) (Cortes e Vapnik, 1995) foi selecionado como classificador-base e o *RidgeClassifierCV* como meta-classificador, devido à sua capacidade de lidar de forma eficiente com os meta-atributos gerados pelos classificadores-base.

Pré-processamento e Treinamento

O processo de criação das meta-características, ilustrado na Figura 24, fundamenta-se na análise das saídas dos classificadores-base. Durante o pré-processamento, usa-se abordagem de concatenação das representações descrita na Seção 4.2.2, onde cada classificador-base é treinado individualmente, usando diferentes representações dos dados. Esse processo permite a extração e armazenamento das características relevantes dos dados de treinamento, com suas respectivas predições. Os hiperparâmetros dos classificadores-base são ajustados por validação cruzada, utilizando a técnica *GridSearchCV*. No contexto do SVM, embora o algoritmo possua diversos hiperparâmetros ajustáveis, focamos nossa otimização nos parâmetros específicos apresentados na Tabela 4, que demonstraram maior impacto no desempenho do modelo.

Hiper-parâmetros	**Variáveis**
Kernel	Linear, RBF, Poly e Sigmoid
Regularizadores C	0.01, 0.1, 1, 10, 100, 1000

Tabela 4: Parâmetros utilizados no GridSearchCV.

Após o treinamento dos classificadores-base, considerando que estamos tratando com SVM em ambientes multiclasses, adaptamos a saída para utilizar as probabilidades dos rótulos de cada classe, em vez do rótulo da classe predita. Em seguida, a saída dos classificadores é organizada para formar um novo conjunto de dados. Esse novo conjunto de dados denominado meta-características será utilizado como base de treinamento do meta-classificador.

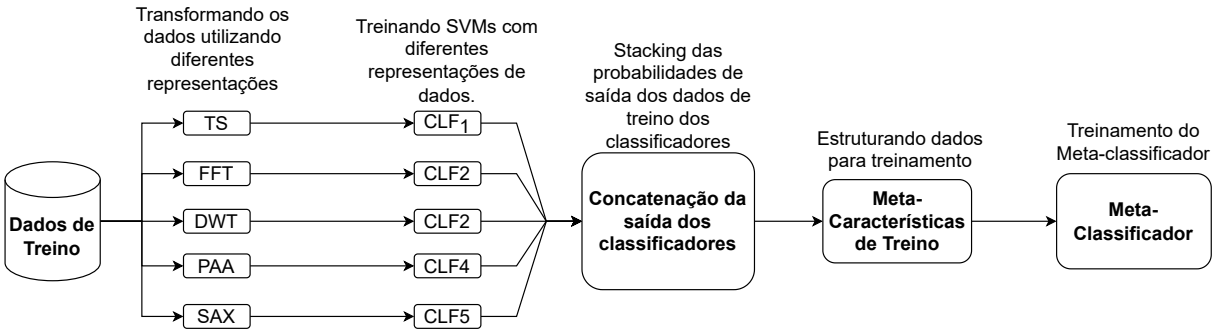


Figura 24: Fluxo de geração de meta-características utilizadas na etapa de pré-processamento e treino do Meta-classificador.

Inferência

A Figura 25 exemplifica a fase de inferência, na qual se utilizam os classificadores-base previamente treinados para realizar predições sobre novos dados. Cada classificador processa

os dados de entrada de forma independente, gerando probabilidades para cada classe. Essas predições são então agregadas em uma matriz de meta-características. É importante destacar que o processo de preparação dos dados realizado durante o treinamento é replicado nesta fase, com a diferença de que os rótulos das classes não são utilizados. A matriz de meta-características resultante é submetida ao meta-classificador, previamente treinado. O meta-classificador processa essas meta-características e, como saída, produz o rótulo da classe. Esta abordagem permite que o sistema aproveite as diferentes perspectivas oferecidas por cada representação de dados, similar à metodologia apresentada anteriormente para a classificação de séries temporais. O meta-classificador, assim, atua como um mecanismo de integração que combina estas diferentes visões para produzir uma classificação mais robusta e precisa.

Figura 25: Fluxo de extração de meta-características utilizadas na etapa validação do Meta-classificador.

4.4 Seleção Dinâmica

Neste trabalho foi implementado um modelo de seleção dinâmica utilizando classificadores baseados em florestas aleatórias (Pal, 2005), aplicando uma variação do CAWPE (Cross-validation Accuracy Weighted Probabilistic Ensemble) (Large et al., 2019b) como método de combinação. O CAWPE é uma técnica que pondera as previsões de múltiplos classificadores com base em suas precisões obtidas por validação cruzada. Este método trabalha com as probabilidades de classe geradas por cada classificador, em vez de apenas suas previsões finais. Utilizando a validação cruzada, é possível estimar a precisão de cada classificador, evitando *overfitting*. As previsões ponderadas são então combinadas para produzir uma decisão final. O uso do CAWPE se dá por sua robustez a *outliers*, melhor desempenho em comparação com métodos de votação simples (Middlehurst et al., 2024). Esta abordagem combina múltiplas representações e classificadores, visando aumentar a robustez e a precisão da classificação, ponderando os desempenhos dos classificadores individuais através da diversidade de métodos e representações.

A utilização de um conjunto de classificadores é justificada pelo aprimoramento do desempenho preditivo através da combinação de múltiplos modelos de Aprendizado de Máquina (Dong et al., 2020). Dentre as diversas metodologias disponíveis para a construção do modelo, após os experimentos disponíveis na Tabela 19, selecionou-se a metodologia da seleção dinâmica em conjunto com CAWPE. Elas foram escolhidas devido à sua eficiência na integração de modelos independentes treinados em diferentes representações de dados, apresentando-se como a abordagem mais adequada para a resolução do problema de pesquisa proposto.

Treinamento

Para o treinamento do modelo de seleção dinâmica, ilustrado na Figura 26, fundamenta-se a utilização de quatro classificadores distintos baseados em florestas aleatórias, sendo estes treinados de maneira independente em um conjunto de dados usando a abordagem de estatísticas combinadas 3 . O resultado do treinamento é a geração de probabilidades preditivas para cada classificador, constituindo uma matriz de dimensão $\boldsymbol{N} \times \boldsymbol{M}$, onde $\boldsymbol{N}$ denota o número de classificadores e $\boldsymbol{M}$ representa o número de classes. A representação desta matriz pode ser vista na Tabela 5. A avaliação dos classificadores é baseada em dois critérios fundamentais utilizados na abordagem CAWPE: O primeiro é a precisão individual de cada classificador, e o segundo é a sua contribuição relativa ao desempenho esperado no conjunto de dados de treinamento. Com esse propósito, o modelo emprega dois vetores essenciais: um contendo a média das precisões de cada classificador, e o outro com pesos normalizados, obtidos pela divisão de cada precisão pelo somatório total das precisões. Essa abordagem de ponderação permite atribuir pesos conforme o desempenho individual de cada classificador no conjunto de treinamento.

Classificador (N)	**Classe 1**	**Classe 2**	...	**Classe M**	**Precisão Média** (P_i)	**Peso Normalizado** (W_i)
Classificador 1	P_{11}	P_{12}	...	P_{1M}	P_1	$W_1 = \frac{P_1}{\sum_{i=1}^{N} P_i}$
Classificador 2	P_{21}	P_{22}	...	P_{2M}	P_2	$W_2 = \frac{P_2}{\sum_{i=1}^{N} P_i}$
...	...	...	...	...	...	...
Classificador N	P_{N1}	P_{N2}	...	P_{NM}	P_N	$W_N = \frac{P_N}{\sum_{i=1}^{N} P_i}$

Tabela 5: Matriz de probabilidades preditivas, vetores de precisão e pesos normalizados de cada classificador.

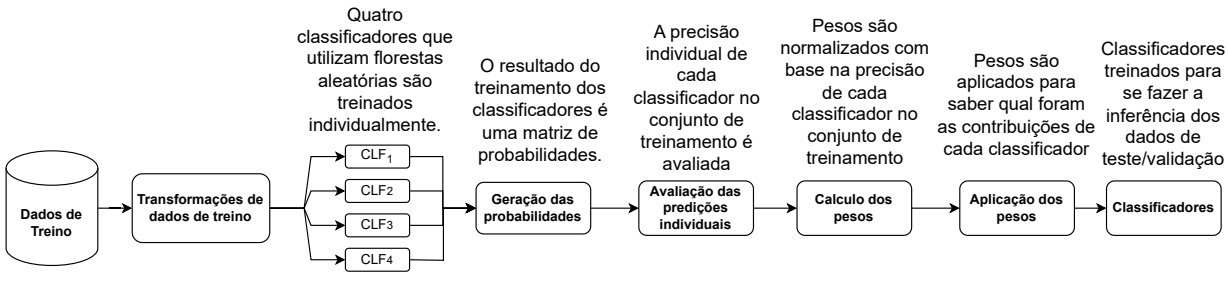


Figura 26: Fluxo de treinamento utilizado no modelo de seleção dinâmica.

Inferência

O processo de inferência, demonstrado na Figura 27, apresenta um procedimento ordenado no qual os classificadores-base realizam predições probabilísticas para novos conjuntos de da-

dos. Durante esta etapa, cada classificador-base indica a probabilidade de pertinência da série temporal a cada classe possível. Os pesos atribuídos baseiam-se no desempenho global dos classificadores-base, sendo as predições posteriormente combinadas por meio de uma média ponderada. A classificação final de cada instância é determinada pela classe que apresenta a maior probabilidade após a agregação das predições dos classificadores.

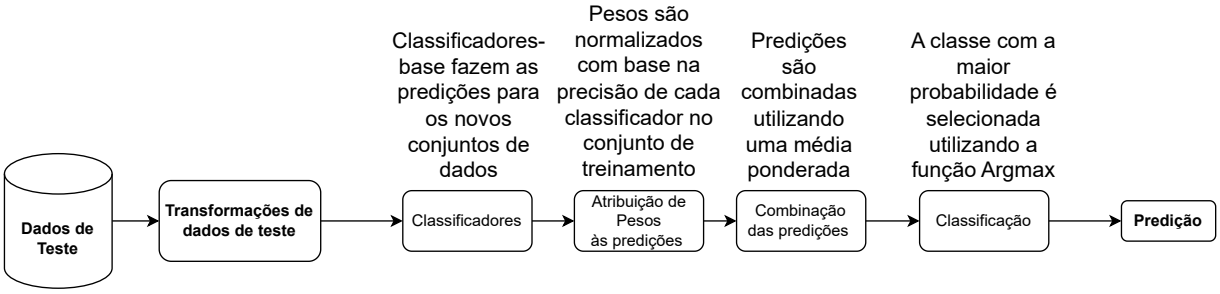


Figura 27: Fluxo de validação e predição utilizado no modelo de seleção dinâmica.

A Tabela 6 apresenta a estrutura matricial das probabilidades, onde cada coluna representa as probabilidades calculadas por um classificador-base e cada linha corresponde a uma série temporal distinta. Para um conjunto de dados com C classes, são calculadas as probabilidades para cada representação de dados (R(i)) e classe (C(j)), expressas formalmente como P(C(j)|R(i)). Desta forma, para cada classificador, obtém-se o seguinte conjunto de probabilidades:

P(classeCj |FFT), P(classeCj |DWT), P(classeCj |PAA), P(classeCj |SAX), P(classeCj |TS)

Classe 1					Classe 2					Melhor Representação
TS	FFT	DWT	PAA	SAX	TS	FFT	DWT	PAA	SAX	
0.10	0.15	0.05	0.20	0.12	0.08	0.03	0.09	0.06	0.12	PAA - Classe 1
0.23	0.49	0.05	0.02	0.01	0.08	0.03	0.04	0.03	0.02	FFT - Classe 1
0.02	0.18	0.07	0.05	0.13	0.09	0.06	0.11	0.04	0.25	SAX - Classe 2
0.05	0.02	0.30	0.03	0.35	0.02	0.01	0.12	0.03	0.02	SAX - Classe 1
0.02	0.37	0.01	0.05	0.01	0.02	0.40	0.02	0.02	0.03	FFT - Classe 2
0.07	0.20	0.06	0.05	0.04	0.04	0.20	0.22	0.07	0.05	DWT - Classe 2

Tabela 6: Probabilidades das classes para diferentes representações normalizadas com a melhor representação selecionada

A determinação da classe final é realizada através da função $\hat{C} = \text{Argmax}_{C_j}\left(P(C_j \mid R_i)\right)$, que seleciona a classe com maior probabilidade, considerando a representação de dados mais apropriada para cada instância. Esta abordagem metodológica permite que cada linha da

matriz represente uma série temporal distinta, onde a representação e a classe mais adequada sejam selecionadas para a instância alvo, proporcionando um incremento na capacidade preditiva do modelo de seleção dinâmica.

4.5 Considerações Finais

Neste capítulo, foram apresentadas a definição do escopo, assim como a metodologia empregada e as principais propostas desenvolvidas neste trabalho. Foi explicada a estrutura do desenvolvimento de um modelo de seleção dinâmica que utiliza diversos classificadores baseados em florestas aleatórias de decisão e um meta-classificador que possui uma configuração flexível e adaptável a diversos tipos de classificadores-base encontrados na literatura. Os classificadores desenvolvidos serão avaliados na Seção 5, junto aos demais trabalhos encontrados na literatura.

Capítulo 5

Avaliação Experimental

Durante a realização deste trabalho, foram analisadas diversas representações de dados temporais. O presente capítulo apresenta os experimentos realizados. A Seção 5.1 apresenta os conjuntos de dados utilizados neste trabalho para validação e comparação dos classificadores propostos aos demais encontrados na literatura. Na Seção 5.3 os experimentos tiveram como foco o uso de diferentes representações de dados temporais para validar e estabelecer uma base de dados para a criação de um novo modelo. A Seção 5.3.3 tem em vista validar a hipótese de que a seleção dinâmica de diferentes representações ao nível de instância de dados temporais aumenta o poder de predição de um classificador e, por fim, na Seção 5.4 é atestado os resultados alcançados.

5.1 Conjuntos de dados

O repositório contendo os conjuntos de dados utilizados nesta pesquisa vem do repositório UCR. Começou na Universidade da Califórnia em *Riverside* com apenas 16 conjuntos de dados. Ao longo do tempo, o repositório cresceu gradualmente, atingindo 45 conjuntos de dados em 2015 e mais de 128 em 2018 (de Freitas Júnior, 2021). Atualmente, em 2025, o repositório contém mais de 180 conjuntos de dados com séries temporais univariadas e multivariadas de diversas categorias, como áudio, movimento, dispositivos, imagens, sensores, simulados, espectros e ECG (Bagnall et al., 2017; Ruiz et al., 2021). Os conjuntos listados podem ser distribuídos nas seguintes categorias:

Conjunto de dados	**Tipo**	**Classes**	**Exemplos de Treino**	**Exemplos de Teste**	**Observações**
ACSF1	Sensor	10	100	100	1.460
Adiac	Serialização	37	390	391	176
ArrowHead	Serialização	3	36	175	251
Beef	Espectrograma	5	30	30	470
BeetleFly	Serialização	2	20	20	512
BirdChicken	Serialização	2	20	20	512

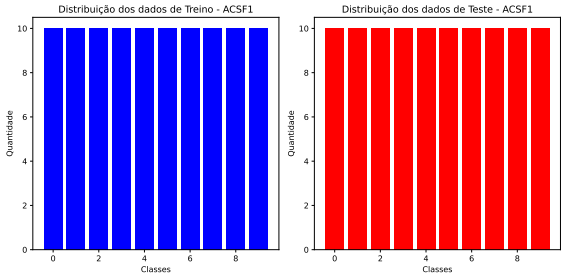
BME	Sintético	3	30	150	128
Car	Sensor	4	60	60	577
CBF	Sintético	3	30	900	128
Chinatown	Trafico	2	20	345	24
ChlorineConcentration	Sintético	3	467	3.840	166
CinCECGtorso	ECG	4	40	1.380	1.639
Coffee	Espectrograma	2	28	28	286
Computers	Sensor	2	250	250	720
CricketX	Movimento	12	390	390	300
CricketY	Movimento	12	390	390	300
CricketZ	Movimento	12	390	390	300
Crop	Serialização	24	7.200	16.800	46
DiatomSizeReduction	Serialização	4	16	306	345
DistalPhalanxOutlineAgeGroup	Serialização	3	139	400	80
DistalPhalanxOutlineCorrect	Serialização	2	276	600	80
DistalPhalanxTW	Serialização	6	139	400	80
Earthquakes	Sensor	2	139	322	512
ECG200	ECG	2	100	100	96
ECG5000	ECG	5	500	4.500	140
ECGFiveDays	ECG	2	23	861	136
ElectricDevices	Sensor	7	8.926	7.711	96
EOGHorizontalSignal	EOG	12	362	362	1.250
EOGVerticalSignal	EOG	12	362	362	1.250
EthanolLevel	Espectrograma	4	504	500	1.751
FaceAll	Serialização	14	560	1.690	131
FaceFour	Serialização	4	24	88	350
FacesUCR	Serialização	14	200	2050	131
FiftyWords	Serialização	50	450	455	270
Fish	Serialização	7	175	175	463
FordA	Sensor	2	1.320	3.601	500
FordB	Sensor	2	810	3.636	500
FreezerRegularTrain	Sensor	2	150	2.850	301
FreezerSmallTrain	Sensor	2	28	2.850	301
GunPoint	Movimento	2	50	150	150
GunPointAgeSpan	Movimento	2	135	316	150
GunPointMaleVersusFemale	Movimento	2	135	316	150
GunPointOldVersusYoung	Movimento	2	135	316	150
Ham	Espectrograma	2	109	105	431
HandOutlines	Serialização	2	370	1.000	2.709
Haptics	Movimento	5	155	308	1.092
Herring	Serialização	2	64	64	512
HouseTwenty	Sensor	2	34	101	3.000
InlineSkate	Movimento	7	100	550	1.882
InsectEPGRegularTrain	EPG	3	62	249	601
InsectEPGSmallTrain	EPG	3	17	249	601
InsectWingbeatSound	Onda	11	220	1.980	256
ItalyPowerDemand	Sensor	2	67	1.029	24

LargeKitchenAppliances	Sensor	3	375	375	720
Lightning2	Sensor	2	60	61	637
Lightning7	Sensor	7	70	73	319
Mallat	Sintético	8	55	2.345	1.024
Meat	Espectrograma	3	60	60	448
MedicalImages	Serialização	10	381	760	99
MiddlePhalanxOutlineAgeGroup	Serialização	3	154	400	80
MiddlePhalanxOutlineCorrect	Serialização	2	291	600	80
MiddlePhalanxTW	Serialização	6	154	399	80
MixedShapesRegularTrain	Serialização	5	500	2.425	1.024
MixedShapesSmallTrain	Serialização	5	100	2.425	1.024
MoteStrain	Sensor	2	20	1.252	84
NonInvasiveFetalECG1	ECG	42	1.800	1.965	750
NonInvasiveFetalECG2	ECG	42	1.800	1.965	750
OliveOil	Espectrograma	4	30	30	570
OSULeaf	Serialização	6	200	242	427
PhalangesOutlinesCorrect	Serialização	2	1.800	858	80
Phoneme	Onda	39	214	1.896	1.024
PigAirwayPressure	Serialização	52	104	208	2.000
PigArtPressure	Serialização	52	104	208	2.000
PigCVP	Serialização	52	104	208	2.000
Plane	Sensor	7	105	105	144
ProximalPhalanxOutlineAgeGroup	Serialização	3	400	205	80
ProximalPhalanxOutlineCorrect	Serialização	2	600	291	80
ProximalPhalanxTW	Serialização	6	205	400	80
RefrigerationDevices	Sensor	3	375	375	720
Rock	Espectrograma	4	20	50	2.844
ScreenType	Sensor	3	375	375	720
ShapeletSim	Sintético	2	20	180	500
ShapesAll	Sintético	60	600	600	512
SmallKitchenAppliances	Sensor	3	375	375	720
SmoothSubsapce	Sintético	3	150	150	15
SonyAIBORobotSurface	Sensor	2	20	601	70
SonyAIBORobotSurfaceII	Sensor	2	27	953	65
StarLightCurves	Onda	3	1.000	8.236	1.024
Strawberry	Espectrograma	2	370	613	235
SwedishLeaf	Serialização	15	500	625	128
Symbols	Serialização	6	25	995	398
SyntheticControl	Sintético	6	300	300	60
ToeSegmentation1	Serialização	2	40	228	277
ToeSegmentation2	Serialização	2	36	130	343
Trace	Sensor	4	100	100	275
TwoLeadECG	ECG	2	23	1139	82
TwoPatterns	Sintético	4	1.000	4.000	128
uWaveGestureLibraryAll	Movimento	8	896	3.582	945
uWaveGestureLibraryX	Movimento	8	896	3.582	315
uWaveGestureLibraryY	Movimento	8	896	3.582	315

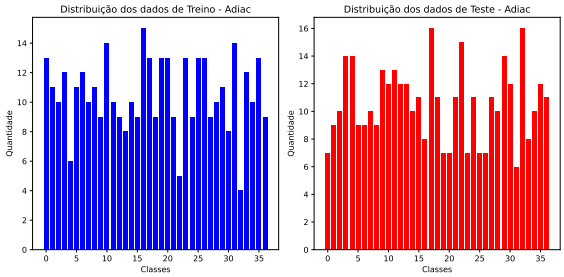
uWaveGestureLibraryZ	Movimento	8	896	3.582	315
Wafer	Sensor	2	1.000	6174	152
Wine	Espectrograma	2	57	54	234
WordsSynonyms	Serialização	25	267	638	270
Worms	Movimento	5	77	181	900
WormsTwoClass	Movimento	2	77	181	900
Yoga	Serialização	2	300	3.000	426

Tabela 7: Descrição dos conjuntos de dados utilizados neste trabalho, retirados do repositório UCR.

Com a quantidade de conjuntos de dados disponíveis, foi conduzido um experimento sobre balanceamento dos dados do repositório UCR. Na Figura 28-b é possível notar um desequilíbrio significativo na distribuição de exemplos entre classes. Este desbalanceamento pode comprometer a capacidade de generalização dos modelos, prejudicando especialmente a classificação das classes minoritárias. Os demais resultados podem ser vistos na Seção do Apêndice A.1.



(a) Conjunto de dados ACSF1 perfeitamente balanceado entre os dados de treino e teste.



(b) Conjunto de dados Adiac com desbalanceamento entre os dados de treino e teste.

Figura 28: Na figura (a) temos um conjunto de dados **ACSF1** perfeitamente balanceado, enquanto na figura (b) está o conjunto de dados **Adiac** completamente desbalanceado

5.2 Ambiente de Testes

Os experimentos foram realizados em um computador pessoal equipado com 8Gb de RAM, processador Intel Core I5-11Gen de 2.40Ghz e sistema operacional Linux Ubuntu versão 22.04.2 LTS. A implementação dos códigos foi realizada utilizando a linguagem *Python*, sendo utilizadas diversas bibliotecas que auxiliaram na condução deste trabalho, como *Pandas, NumPy, SciPy, TSLearn* (Tavenard et al., 2020), *Sktime* (Löning et al., 2019), *Pyts* (Faouzi e Janati, 2020), *Scikit-Learn* (Hao e Ho, 2019), *TensorFlow e Keras* (Developers, 2022; Géron, 2021). Além da utilização do ambiente virtual equipado com uma GPU fornecida pelo *Google Colaboratory* (Bisong, 2019) utilizada para os experimentos que necessitavam de uma placa gráfica. A utilização dos pacotes acima mencionados foi essencial para a implementação dos experimentos, ao proporcionarem a automatização, classificação e armazenamento dos resultados

obtidos.

5.3 Experimentos

O método de experimentação empregado nesta pesquisa foi o uso de testes empíricos. Os experimentos são descritos em quatro grupos:

1. Experimentos com Oráculo: Esta etapa estabeleceu um limite superior teórico de desempenho para cada representação nos conjuntos de dados. O oráculo foi implementado usando um classificador 1-NN que tinha acesso às classificações ótimas, permitindo avaliar o potencial máximo de cada representação isoladamente;

2. Experimentos com Redes Convolucionais: Investigou-se a capacidade das CNNs em identificar automaticamente a representação mais adequada para cada instância dos conjuntos de dados. Foram comparadas arquiteturas 1D e 2D, explorando sua capacidade de abstração quando expostas a um grande volume de características de uma mesma classe;

3. Experimentos com Seleção Dinâmica: Esta fase testou a hipótese de que a seleção da representação mais adequada para cada instância poderia melhorar a acurácia global do modelo. Foram conduzidos experimentos para determinar o modelo de seleção dinâmica mais eficaz, seguidos de comparações com abordagens da literatura;

4. Experimentos com Meta-aprendizado: Nesta etapa, desenvolveu-se uma abordagem baseada em meta-aprendizado para seleção de representações. O processo envolveu duas fases principais: (1) seleção do classificador-base mais eficiente para extração de características dos dados de treino e (2) identificação do meta-classificador mais adequado para classificação dos meta-dados gerados. Os experimentos foram validados em 112 conjuntos do repositório UCR, com resultados comparados às abordagens estado-da-arte da literatura.

Visando identificar a abordagem mais eficaz para a tarefa de classificação, foram conduzidos experimentos iniciais utilizando uma amostra de 20 conjuntos de dados do repositório UCR, presentes na Tabela 7, que foram selecionados de maneira aleatória. A avaliação de desempenho foi conduzida utilizando a métrica de acurácia, amplamente empregada em testes de *benchmarking* relacionados à TSC. Esta escolha se justifica por possibilitar comparações mais consistentes entre diferentes métodos e modelos, facilitando a replicabilidade e validação dos experimentos e resultados. Além disso, os conjuntos de dados utilizados apresentam-se relativamente balanceados. Por fim, foi empregada a validação cruzada para assegurar a robustez e a generalização dos resultados obtidos.

As representações estudadas neste trabalho possuem alguns parâmetros que precisam ser ajustados. A representação espectral não possui parâmetros para ajuste. A representação da

wavelet possui um parâmetro denominado *Wavelet* Mãe, para definir o filtro que define os coeficientes de aproximação. A representação PAA tem o tamanho da janela que precisa ser ajustada e a representação SAX tem o tamanho do alfabeto utilizado.

Representação	**Parâmetro**	**Descrição**	**Espaço de Busca**
Espectral	-	-	-
DWT	*Wavelet*	*Wavelet*	db1, db2,. . . ,db9
PAA	*Window Size*	Tamanho da janela	*
SAX	*N-Bins/Símbolos*	Tamanho do Alfabeto	2, 3, 4, 5,. . . ,26

Tabela 8: Parâmetros de ajuste das representações.

Para cada conjunto de dados que necessitava ser utilizado um ajuste de parâmetro, foi utilizado o método de *GridSearch*. Este método utiliza uma pesquisa exaustiva sobre valores de parâmetros especificados para um estimador. Os parâmetros do estimador usados para aplicar esses métodos são otimizados por uma pesquisa de grade de validação cruzada na grade de parâmetros. Dessa forma, é possível testar todos os parâmetros das representações e utilizar aquela que for mais bem adequada para a série temporal.

5.3.1 Oráculo

Para estabelecer um referencial de desempenho nesta pesquisa, implementamos um oráculo baseado no classificador 1-NN. Este oráculo foi construído através do treinamento do 1-NN em diferentes representações de dados, seguido da criação de um conjunto que incorpora as representações ótimas de cada instância do conjunto de treinamento. Embora este oráculo represente o 1-NN Bayes-ótimo para os modelos propostos, é importante ressaltar que classificadores que utilizam outras estratégias podem potencialmente superar seu desempenho.

O oráculo possui conhecimento completo sobre a seleção das diferentes representações das séries temporais, permitindo-lhe classificar cada instância utilizando sua representação mais adequada. Consequentemente, seu desempenho estabelece um limite superior teórico para as métricas de avaliação em cada conjunto de dados. Os resultados destes experimentos, apresentados na Tabela 9, comparam o desempenho do oráculo com o classificador 1-NN utilizando representações alternativas. Esta análise demonstra que a seleção individualizada de representações para cada série temporal pode superar a estratégia de utilizar uma única representação para todo o conjunto de dados. Estes resultados servirão como referência para avaliar o desempenho dos modelos propostos na tarefa de classificação de séries temporais.

Conjunto	TS	FFT	DWT	PAA	SAX	Oráculo
Adiac	0.611	0.731	0.618	0.606	0.427	0.831
Beef	0.666	0.700	0.667	0.666	0.833	0.833
Car	0.733	0.716	0.733	0.733	0.700	0.850
CBF	0.852	0.615	0.851	0.933	0.724	0.990
Coffee	1.0	1.0	1.0	1.0	0.821	1.0
DiatomSizeReduction	0.934	0.934	0.937	0.934	0.905	0.947
ECG200	0.880	0.880	0.800	0.890	0.850	0.960
ECGFiveDays	0.796	0.994	0.796	0.833	0.811	0.998
FaceFour	0.784	0.761	0.784	0.784	0.886	0.909
GunPoint	0.913	0.967	0.913	0.913	0.960	0.993
Lightning2	0.754	0.754	0.754	0.803	0.803	0.967
Lightning7	0.575	0.561	0.604	0.589	0.684	0.917
MedicalImages	0.684	0.630	0.700	0.659	0.619	0.856
MoteStrain	0.878	0.676	0.882	0.825	0.600	0.984
OliveOil	0.866	0.800	0.866	0.866	0.816	0.933
SonyAIBORobotSurface1	0.695	0.712	0.690	0.722	0.658	0.836
SonyAIBORobotSurface2	0.856	0.863	0.835	0.878	0.846	0.973
SyntheticControl	0.880	0.623	0.835	0.983	0.806	0.996
Trace	0,760	0,840	0.740	0.760	0.740	0.890
TwoPatterns	0.906	0.507	0.896	0.951	0.903	0.989

Tabela 9: Resultados obtidos pelo 1-NN utilizando diferentes representações de dados em comparação com o Oráculo que será utilizado como *Topline* para os demais experimentos.

5.3.2 Experimentos com Redes de Convolução

A rede neural convolucional unidimensional (1D-CNN) proposta é especializada no processamento de sequências temporais, extraindo características relevantes por meio de camadas específicas. A arquitetura inicia com uma camada de convolução 1D (CONV1D), seguida de uma camada de *MaxPooling1D* para redução de dimensionalidade, especificamente para dados sequenciais. A arquitetura presente na Figura 29 emprega uma segunda combinação de CONV1D e *MaxPooling1D*, com filtros distintos da primeira camada, seguida por uma camada *Flatten* que transforma as características em um vetor unidimensional. Uma camada densa com ativação *ReLU* estabelece relações entre as características extraídas e as classes de saída. A camada final, utilizando ativação *Softmax*, gera a distribuição de probabilidade das classes possíveis. O modelo utiliza o otimizador **Adam** e a função de perda por entropia cruzada categórica esparsa, adequada para classificação multiclasse. A avaliação do desempenho é realizada através da métrica de precisão.

A arquitetura da rede neural convolucional (CNN) adotada é amplamente utilizada na classificação de imagens, mantém uma estrutura similar à 1D-CNN, com adaptações específicas. A rede ilustrada na Figura 30 emprega camadas de convolução 2D (CONV2D) e *MaxPooling2D*, complementadas por camadas densas com ativações *Relu* e *Softmax*. Como diferencial, implementa-se a técnica de *Dropout*, que desativa aleatoriamente neurônios durante

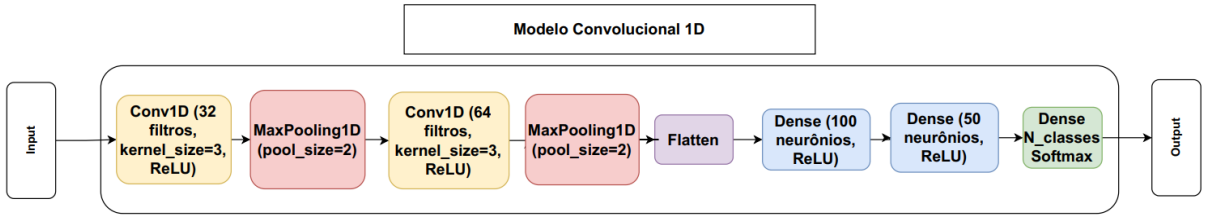


Figura 29: Arquitetura da rede neural convolucional de 1 dimensão utilizada nos experimentos deste trabalho.

o treinamento, prevenindo o *Overfitting*. Uma camada *Flatten* transforma as características extraídas em um vetor unidimensional, enquanto o otimizador *AdaW* ajusta dinamicamente a taxa de aprendizado para otimizar a convergência do modelo.

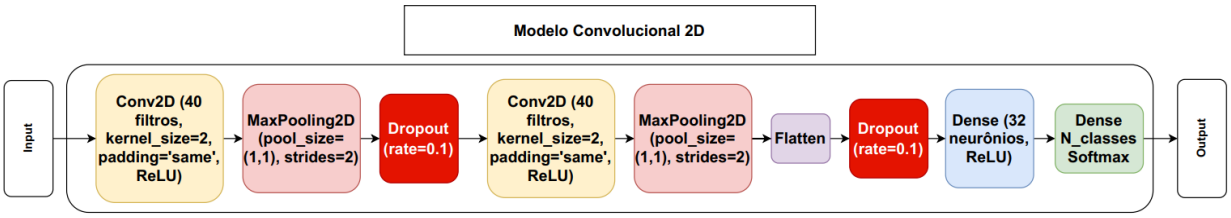


Figura 30: Arquitetura da rede neural convolucional de 2 dimensões utilizada nos experimentos deste trabalho.

Avaliamos os conjuntos de dados utilizando uma 1D-CNN, cujos resultados estão apresentados na Tabela 10. O desempenho de classificação foi melhor no domínio temporal. Algumas mudanças de representação, como, por exemplo, utilizando SAX no conjunto de dados *Adiac*, mostraram-se inviáveis devido à baixa acurácia. Porém, alguns resultados interessantes podem ser destacados. Nos conjuntos de dados como *Beef, Coffee, ECG200, GunPoint*, que possuem poucas classes para serem classificadas ou tem seu conjunto de amostras de treino e teste bem balanceadas, demonstra uma acurácia superior independente da representação de dados utilizada em comparação com conjuntos *Adiac, MedicalImages* que possuem 37 e 10 classes para serem classificadas respectivamente. Conjuntos de dados como *SyntheticControl, Trace e TwoPatterns* em comparação com os demais conjuntos de dados já mencionados, possuem mais amostras de treino e teste possibilitando dessa forma que o classificador possa aprender mais características e assim obter um desempenho superior as demais representações.

Conjunto	TS	FFT	DWT	PAA	SAX
Adiac	0.652	0.621	0.304	0.613	0.207
Beef	0.766	0.766	0.500	0.733	0.333
Car	0.801	0.500	0.700	0.816	0.583
CBF	0.945	0.614	0.846	0.982	0.876
Coffee	0.964	1.0	1.0	1.0	0.892
DiatomSizeReduction	0.915	0.669	0.888	0.888	0.872
ECG200	0.879	0.829	0.889	0.800	0.790
ECGFiveDays	0.895	0.842	0.872	0.858	0.852
FaceFour	0.750	0.465	0.852	0.715	0.590
GunPoint	0.913	0.882	0.839	0.866	0.646
Lightning2	0.754	0.713	0.704	0.723	0.737
Lightning7	0.712	0.452	0.684	0.698	0.698
MedicalImages	0.678	0.639	0.686	0.673	0.577
MoteStrain	0.853	0.662	0.824	0.839	0.849
OliveOil	0.866	0.669	0.400	0.800	0.400
SonyAIBORobotSurface1	0.645	0.838	0.748	0.622	0.722
SonyAIBORobotSurface2	0.869	0.930	0.868	0.852	0.864
SyntheticControl	0.996	0.589	0.949	0.983	0.893
Trace	0.839	0.990	0.990	0.889	0.610
TwoPatterns	0.981	0.432	0.922	0.966	0.973

Tabela 10: Resultados obtidos utilizando um 1D-CNN com diferentes representações de dados.

Avaliamos os conjuntos de dados utilizando uma 2D-CNN, cujos resultados estão apresentados na Tabela 11. O modelo 2D foi especialmente projetada para informações bidimensionais, produzindo resultados consideravelmente mais precisos, especialmente quando se trata de representações específicas. O conjunto de dados *Beef* utilizando a **representação DWT**, por exemplo, produziu uma precisão de 0.966 com a rede 2D-CNN, mas apenas 0.500 ao usar a rede 1D-CNN. Embora haja uma mudança para a **representação PAA** que teve um efeito oposto na precisão da rede 2D-CNN, caindo em 0.400 quando comparado com 0.733 da rede 1D-CNN. De modo geral, podemos concluir que uma rede 2D-CNN pode oferecer grandes aumentos de precisão em determinados cenários. Porém, é necessário avaliar cuidadosamente a estrutura da rede convolucional.

Conjunto	TS	FFT	DWT	PAA	SAX
Adiac	0.652	0.610	0.256	0.610	0.470
Beef	0.766	0.766	0.966	0.400	0.333
Car	0.801	0.583	0.150	0.783	0.766
CBF	0.945	0.583	0.662	0.941	0.894
Coffee	0.996	0.964	1.0	1.0	0.931
DiatomSizeReduction	0.915	0.663	0.303	0.925	0.898
ECG200	0.879	0.849	0.550	0.789	0.800
ECGFiveDays	0.895	0.994	0.513	0.876	0.833
FaceFour	0.832	0.316	0.193	0.681	0.657
GunPoint	0.913	0.887	0.589	0.889	0.853
Lightning2	0.754	0.591	0.491	0.754	0.754
Lightning7	0.712	0.383	0.082	0.712	0.684
MedicalImages	0.715	0.615	0.084	0.530	0.549
MoteStrain	0.853	0.531	0.549	0.848	0.851
OliveOil	0.866	0.669	0.933	0.833	0.800
SonyAIBORobotSurface1	0.783	0.828	0.527	0.570	0.541
SonyAIBORobotSurface2	0.869	0.901	0.511	0.821	0.840
SyntheticControl	0.996	0.621	0.403	0.996	0.973
Trace	0.839	0.831	0.198	0.879	0.781
TwoPatterns	0.981	0.452	0.724	0.988	0.925

Tabela 11: Resultados obtidos utilizando um 2D-CNN com diferentes representações de dados.

5.3.3 Experimentos com Seleção Dinâmica

Para selecionar os classificadores-base da seleção dinâmica, realizamos experimentos comparativos entre florestas aleatórias e outros classificadores amplamente utilizados na literatura, como *Naive Bayes*, SVM e k-NN, detalhados na Tabela 12.

Por muito tempo, o k-NN foi considerado um dos melhores classificadores rasos para séries temporais. No entanto, ao utilizar diferentes representações de dados, o classificador de florestas aleatórias se destacou em relação aos demais. Podemos concluir que utilizar seleção dinâmica unida às diferentes representações de dados pode oferecer grandes aumentos de precisão em determinados cenários. Porém, é necessário avaliar cuidadosamente os dados e os classificadores que serão utilizados. Para os demais testes comparativos desta Seção, o modelo de seleção dinâmica será chamado de modelo protótipo (MP).

Conjunto	1-NN	5-NN	SVM	NB	RF
Adiac	0.611	0.728	0.401	0.606	0.757
Beef	0.666	0.600	0.600	0.700	0.800
Car	0.733	0.683	0.666	0.550	0.716
CBF	0.902	0.940	0.915	0.897	0.995
Coffee	1.0	0.964	0.535	0.964	1.0
DiatomSizeReduction	0.934	0.882	0.584	0.875	0.945
ECG200	0.880	0.930	0.890	0.780	0.880
ECGFiveDays	0.816	0.879	0.910	0.921	0.994
FaceFour	0.784	0.750	0.829	0.806	0.852
GunPoint	0.913	0.993	0.853	0.813	0.996
Lightning2	0.770	0.786	0.737	0.639	0.786
Lightning7	0.602	0.712	0.726	0.589	0.776
MedicalImages	0.685	0.689	0.703	0.622	0.765
MoteStrain	0.877	0.869	0.861	0.858	0.898
OliveOil	0.866	0.900	0.933	0.866	0.900
SonyAIBORobotSurface1	0.886	0.812	0.916	0.956	0.838
SonyAIBORobotSurface2	0.865	0.815	0.847	0.851	0.832
SyntheticControl	0.963	1.0	0.990	0.980	0.996
Trace	0.760	0.810	0.750	0.970	0.990
TwoPatterns	0.929	0.989	0.919	0.638	0.997

Tabela 12: Resultados obtidos utilizando diferentes classificadores encontrados na literatura para validar a escolha do modelo base utilizado na seleção dinâmica.

A avaliação comparativa entre os modelos 1-NN, 1D-CNN, 2D-CNN e o MP utilizou como métrica a acurácia máxima obtida, independentemente da transformação aplicada. Os resultados, apresentados na Tabela 13, incluem um Oráculo que estabelece um *Topline* como referência de desempenho ideal. Desta forma, o MP em sua maioria alcançou as maiores acurácias dentro de todos os classificadores.

Conjunto	Oráculo	1-NN	1D-CNN	2D-CNN	MP
Adiac	0.831	0.731	0.652	0.652	0.826
Beef	0.833	0.833	0.766	0.966	0.866
Car	0.850	0.733	0.816	0.801	0.733
CBF	0.990	0.933	0.982	0.945	0.995
Coffee	1.0	1.0	1.0	1.0	1.0
DiatomSizeReduction	0.947	0.937	0.915	0.925	0.945
ECG200	0.960	0.890	0.889	0.879	0.930
ECGFiveDays	0,998	0,994	0,895	0,994	0.994
FaceFour	0.909	0.886	0.852	0.832	0.852
GunPoint	0.993	0.967	0.913	0.913	1.0
Lightning2	0.967	0.803	0.754	0.712	0.836
Lightning7	0.917	0.684	0.712	0.712	0.776
MedicalImages	0.856	0.700	0.686	0.715	0.765
MoteStrain	0.984	0.882	0.853	0.853	0.898
OliveOil	0.933	0.866	0.930	0.933	0.933
SonyAIBORobotSurface1	0.836	0.722	0.838	0.956	0.956
SonyAIBORobotSurface2	0.973	0.878	0.930	0.828	0.865
SyntheticControl	0.996	0.983	0.996	0.996	1.0
Trace	0.890	0.840	0.990	0.879	0.990
TwoPatterns	0.989	0.951	0.981	0.988	0.997

Tabela 13: Resultados dos testes da acurácia dos modelos 1NN, 1D-CNN, 2D-CNN e MP.

Foram realizadas comparações entre os modelos mencionados na Seção 3.1 e o MP, com exceção do modelo Hive-COTE v2.0, cujo código não estava disponível na biblioteca *Sktime* e os resultados não foram disponibilizados até o momento da escrita desta pesquisa. Na Tabela 14 os resultados dos algoritmos foram apresentados, demonstrando um bom desempenho para a proposta da pesquisa. Considerando os diversos conjuntos de dados e suas respectivas acurácias, o MP mostrou-se competitivo, superando frequentemente os demais modelos analisados.

Conjunto	TimeBox	Hive-COTE	TS-Chief	Rocket	ITime*	MP
Adiac	0.739	0.850	0.779	0.771	0.822	0.826
Beef	0.700	0.735	0.632	0.760	0.682	0.866
Car	0.766	0.868	0.878	0.911	0.901	0.733
CBF	0.992	0.998	0.998	0.995	0.996	0.995
Coffee	0.964	0.992	0.990	1.0	0.998	1.0
DiatomSizeReduction	0.947	0.914	0.945	0.957	0.905	0.945
ECG200	0.900	0.858	0.855	0.899	0.896	0.930
ECGFiveDays	0.958	0.993	0.994	0.995	0.995	0.998
FaceFour	0.863	0.973	0.999	0.931	0.938	0.852
GunPoint	0.966	0.998	1.0	0.992	0.995	1.0
Lightning2	0.852	0.773	0.768	0.776	0.816	0.836
Lightning7	0.726	0.757	0.793	0.797	0.820	0.776
MedicalImages	0.752	0.740	0.799	0.805	0.796	0.765
MoteStrain	0.903	0.936	0.930	0.905	0.880	0.898
OliveOil	0.833	0.883	0.916	0.902	0.874	0.933
SonyAIBORobotSurface1	0.710	0.826	0.889	0.958	0.954	0.956
SonyAIBORobotSurface2	0.872	0.936	0.901	0.935	0.951	0.865
SyntheticControl	0.990	0.994	0.999	0.997	0.995	1.0
Trace	0.990	1.0	1.0	1.0	1.0	0.990
TwoPatterns	0.998	0.999	0.999	1.0	1.0	0.997

*InceptionTime

Tabela 14: Resultados dos testes dos modelos encontrados na literatura em comparação com o MP.

Foram feitas análises comparativas do desempenho dos classificadores utilizando o teste de hipótese de Friedman, seguido pelo teste post-hoc de Nemenyi. O ranking médio foi adotado como métrica, com valores mais próximos de 1 indicando melhor desempenho relativo. Para todos os grupos de classificadores analisados, rejeitamos a hipótese nula do teste de Friedman ($\alpha < 0.05$), evidenciando diferenças estatisticamente significativas entre os valores medianos das populações. Neste contexto, a distância crítica (CD) representa o limiar mínimo de diferença entre os rankings médios de dois classificadores para serem considerados estatisticamente diferentes. O teste de Nemenyi, baseado nesta distância crítica, apontou diferenças significativas entre diversos classificadores, permitindo uma avaliação mais precisa do desempenho relativo de cada modelo.

Para se fazer uma comparação com os demais classificadores utilizados neste trabalho, foi escolhido o classificador que utiliza florestas aleatórias, como demonstrado nos experimentos da Tabela 13 ao apresentar um resultado mais satisfatório que os demais. Como pode ser visto na Figura 31 a distância crítica neste experimento foi de 1.686 no teste de Friedman.

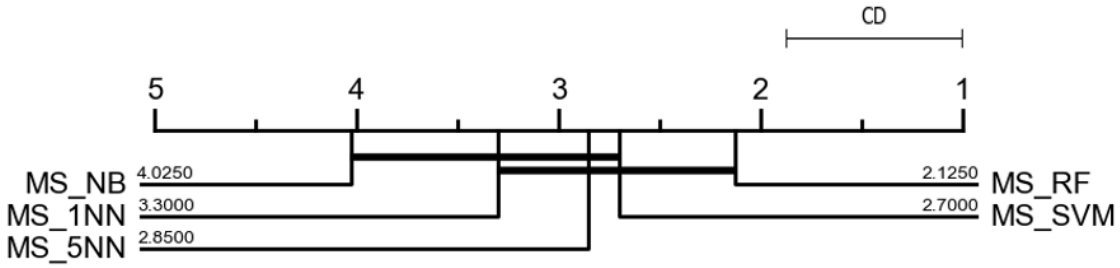


Figura 31: Gráfico de distância crítica entre os classificadores testados para compor o MP.

A Figura 32 expõe o desempenho superior do classificador que utiliza florestas aleatórias em comparação às variações de CNN e o classificador k-NN. A distância crítica neste experimento foi de 1.049.

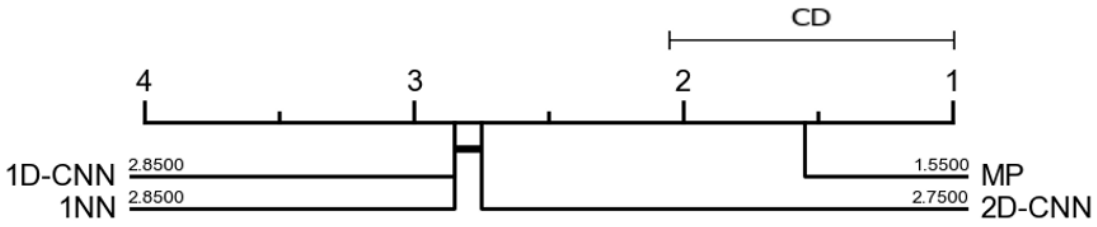


Figura 32: Gráfico de distância crítica entre os classificadores 1-NN, CNN e MP.

Os resultados vistos na Figura 33 exibem que o MP tem desempenho inferior ao classificador *Rocket* porém, superior ao modelo *Inception Time*, apresentando benefícios na seleção de dinâmica de representações para a pesquisa.

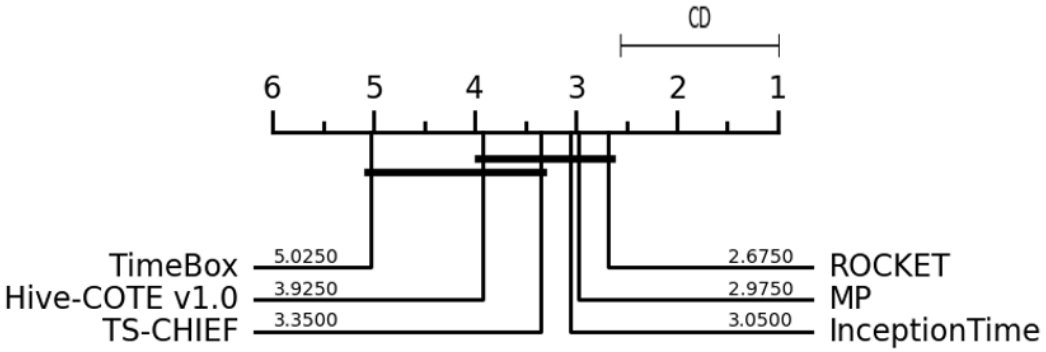


Figura 33: Gráfico de distância crítica comparando os classificadores avaliados na primeira etapa deste trabalho.

5.3.4 Experimentos com Meta-Classificador

Novos testes foram realizados com diferentes variações do meta-classificador e do modelo de seleção dinâmica, comparados com os classificadores presentes na literatura que adotam a mesma abordagem de extração de características dos dados vistos na Seção 3.1. Os resultados dos testes podem ser vistos na Seção Apêndice A. Os testes realizados com os classificadores k-NN e CNN foram utilizados para confirmar a evidência de que diferentes representações de dados têm um impacto significativo na predição dos classificadores. Nesta segunda fase do trabalho, os testes focam apenas na verificação da hipótese de uso de meta-classificação e seleção dinâmica. Dessa forma, para esta etapa do estudo, não foram replicados os testes realizados com o k-NN e CNN.

Foram desenvolvidas variações do meta-classificador para testar as diferentes abordagens de extração de dados descritas na Seção 4.2.2.

Os experimentos empregados nesta etapa são descritos em quatro grupos:

1. Concatenação das representações sem distinção de representação: é utilizado um meta-classificador representado pelo nome meta-classificador_NL, que utiliza a extração de dados sem distinguir as representações. Nesta abordagem, as características não possuem rótulos que indiquem sua origem, tendo sido removida a coluna de *TAG* que anteriormente destacava cada representação.

2. Concatenação das representações sem distinção de representação utilizando *Tunning*: O modelo meta-classificador_TG adota a mesma abordagem citada acima, mas com a adição de uma nova etapa de ajuste *(Tunning)* durante a fase de treinamento, permitindo que os classificadores apresentem os melhores parâmetros para cada conjunto de dados;

3. Seleção Dinâmica usando Estatísticas Combinadas: experimentos com o modelo de seleção dinâmica e comparações com modelos da literatura.

4. Concatenação das representações com meta-aprendizado: o modelo meta-classificador_WL utiliza a abordagem da concatenação das diferentes representações para cada instância do conjunto de dados.

A fim de validar os experimentos realizados e obter uma melhor compreensão do desempenho dos classificadores, a Figura 34 representa a distribuição e a simetria das predições alcançadas. As caixas indicam os quartis, os bigodes indicam os limites máximos e mínimos e, por fim, os *outliers* são representados pelos símbolos bola logo abaixo.

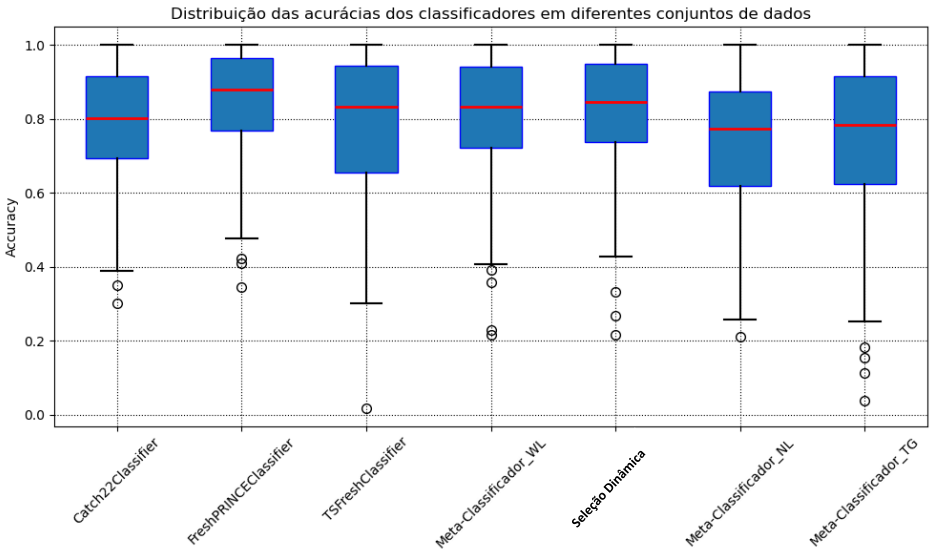


Figura 34: *Boxplot* comparando os classificadores que utilizam extração de características e os modelos propostos neste trabalho.

Para ilustrar de forma mais clara o teste de média de acurácia, o teste de vitórias, derrotas e empates apresentado na Tabela 15 proposto por (Silva et al., 2022) foi utilizado. Este teste é empregado para avaliar o classificador mais preciso em relação aos demais. Neste caso, foi escolhido o classificador FreshPRINCE (Middlehurst e Bagnall, 2022). Além disso, foi desenvolvida uma métrica que calcula a média entre os mínimos e máximos dos resultados obtidos por todos os classificadores, referida como *baseline* e *topline*. O *baseline* foi obtido utilizando o classificador 1NN, enquanto o *topline* corresponde ao desempenho ideal obtido através do oráculo visto na Seção 5.3.1. Os resultados alcançados pelos classificadores meta-classifcador_NL e meta-classifcador_TG apresentaram um desempenho inferior às suas variações, superando apenas o algoritmo Catch22 (Lubba et al., 2019). Os classificadores*Ensemble* e meta-classificador_WL obtiveram bons resultados, demonstrando que, apesar dos resultados satisfatórios, é possível aprimorar tanto a construção dos classificadores quanto as etapas de extração de características.

Classificador	Acurácia Média	Desvio-padrão	Vitórias/Derrotas/Empates contra o FreshPRINCE
Baseline	0.647019	0.208606	0/106/6
Catch22Classifier	0.781368	0.162459	16/90/6
FreshPRINCEClassifier	0.841600	0.148842	-
TSFreshClassifier	0.782757	0.197377	23/75/14
meta-classificador_WL	0.801030	0.170501	31/71/10
Seleção Dinâmica	0.812766	0.170571	36/65/11
meta-classificador_NL	0.741028	0.177113	12/91/9
meta-classificador_TG	0.746674	0.207303	20/88/4
Topline	0.874120	0.128711	62/0/50

Tabela 15: Comparação entre Vitórias, Derrotas e Empates dos classificadores contra o classificador FreshPrince

Ilustrado na Figura 35 o modelo de seleção dinâmica se destaca sobre as outras propostas apresentadas neste trabalho, sendo superado apenas pelo classificador FreshPRINCE (Middlehurst e Bagnall, 2022). A aplicação da abordagem desenvolvida neste estudo mostrou, através dos resultados, vantagens para a classificação de séries temporais. Por fim, os resultados são considerados satisfatórios em comparação com outras técnicas encontradas na literatura.

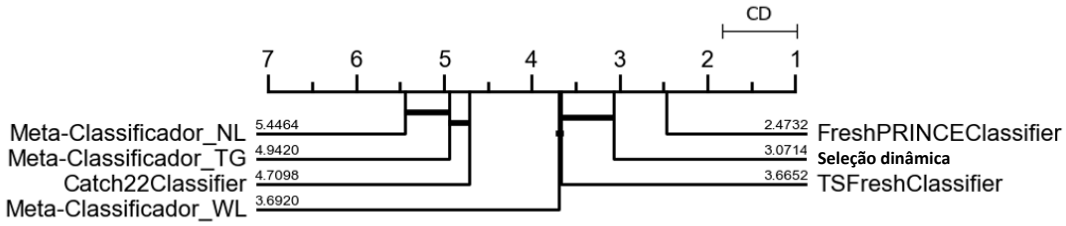


Figura 35: Gráfico de distância crítica comparando os classificadores que utilizam extração de características e os modelos propostos.

5.4 Considerações Finais

Esta pesquisa focou no uso da seleção dinâmica das representações de séries temporais, analisando várias representações encontradas na literatura. Os experimentos preliminares foram realizados com os modelos 1-NN, 1D-CNN, 2D-CNN e uma versão–protótipo do modelo de seleção dinâmica, cujos resultados preliminares indicaram que a transformação dos dados para representações alternativas proporcionou uma melhoria significativa no desempenho dos algoritmos avaliados. Baseado nesses resultados, foi possível compreender as vantagens e limitações de cada domínio, sugerindo a viabilidade de uma escolha dinâmica de representações, de modo a maximizar a eficiência dos classificadores para cada série temporal específica.

Na segunda etapa dos experimentos, foram analisadas diferentes versões do meta-classificador em conjunto com a criação de um ensemble. Foram empregados dois métodos para extrair as características dos dados. A utilização destas abordagens revelou um aumento significativo no poder de generalização dos classificadores. A utilização de rótulos para a extração de características contribui significativamente para a melhoria dos resultados das predições. Conforme percebido nas comparações dos classificadores meta-classifcador_NL e meta-classifcador_WL presentes na Figura 35. A utilização da abordagem CAWPE junto ao modelo de seleção dinâmica de florestas aleatórias demonstrou resultados satisfatórios, ainda que não superiores a todos os algoritmos disponíveis na literatura. No entanto, a utilização de extração de características evidenciou ser uma excelente técnica para aumentar a generalização dos classificadores. Assim, o trabalho confirmou a eficácia da seleção dinâmica de representações para classificar séries temporais, aprimorando o poder de predição dos modelos. Esta seção conclui as discussões sobre as atividades realizadas durante a execução deste projeto.

Capítulo 6

Conclusão

Este trabalho explorou diversas representações de dados para aprimorar a classificação de séries temporais, desenvolvendo abordagens de extração de características com dois classificadores distintos: um utilizando meta-aprendizado e outro com seleção dinâmica. A escolha baseou-se na capacidade de aprender com as experiências de diferentes classificadores. Foram analisadas diversas representações de séries, como *Fourier e wavelets* (detalhadas na Seção 2.2), testadas em diferentes conjuntos de dados para avaliar sua eficácia. O estudo revelou que técnicas de representação específicas podem ser mais eficientes em domínios de aplicação distintos, contribuindo significativamente para o aprimoramento da precisão dos modelos de classificação.

No capítulo 4, foram exploradas diferentes estratégias de extração de características, assim como o detalhamento do desenvolvimento dos classificadores que utilizam meta-aprendizagem e seleção dinâmica em sua composição.

No capítulo 5, os resultados iniciais apresentaram uma perspectiva promissora em relação ao uso de diferentes representações de dados. Embora os experimentos tenham sido bem-sucedidos, a arquitetura dos algoritmos e a estrutura de dados foram alteradas adequadamente para a segunda etapa do trabalho (detalhadas na Seção 5.3.4). Posteriormente, os experimentos realizados com as variações das abordagens de extração de características unidos aos modelos propostos mostraram que seu uso tem uma influência positiva na predição final dos classificadores. A utilização da abordagem de estatísticas combinadas em cada classificador-base mostrou-se uma estratégia mais eficiente em comparação com a concatenação das representações. Os resultados de ambas as abordagens, juntamente com os modelos desenvolvidos, contribuem positivamente para o avanço do campo de pesquisa de classificação de séries temporais, fornecendo métodos significativos para o desenvolvimento eficaz de novas estratégias. Os dados utilizados neste trabalho estão disponíveis para consulta e download no repositório público, através dos links Código e Datasets. A disponibilização dos dados visa garantir a transparência, reprodutibilidade e validação dos resultados apresentados nesta dissertação.

6.1 Limitações

Algumas das limitações encontradas são claramente percebidas no meta-classificador. Apesar dos resultados promissores na seleção dinâmica das representações, há potencial para aprimoramento através da automatização na extração e seleção das melhores características. O modelo de seleção dinâmica, que utiliza Florestas Aleatórias, requer validação cruzada para garantir o treinamento adequado das árvores. Como são treinados quatro classificadores simultaneamente, o custo computacional torna-se significativo quando se busca maior eficiência.

A otimização dos parâmetros nas representações também demanda validação cruzada, sendo que a quantidade de representações impacta diretamente no tempo e custo computacional. Além disso, o repositório UCR tem limitações significativas em seus conjuntos de dados, como apontado por (Giusti, 2017). Todos os conjuntos de dados utilizados na Tabela 7 são normalizados e pré-processados para garantir que todas as séries possuam a mesma quantidade de pontos de observações. Este pré-processamento acaba limitando a eficácia das abordagens de extração de características desenvolvidas neste trabalho.

6.2 Trabalhos Futuros

Para trabalhos futuros, sugere-se explorar representações alternativas que não foram exploradas em sua totalidade neste trabalho e que ofereçam menor custo computacional, bem como outros métodos de classificação, como o uso de *ensemble* com técnicas de *Ranking* e *Voting*. A otimização do processo de extração de características também se mostra relevante, visando reduzir o custo computacional no ajuste de parâmetros. Uma extensão interessante desta pesquisa seria implementar Redes Neurais Convolucionais para extração de características utilizando meta-classificadores rasos, oferecendo um equilíbrio entre eficiência computacional e precisão na classificação.

Referências Bibliográficas

Allman, G. J. (1889). *Greek geometry from Thales to Euclid*. Hodges, Figgis, & Company. Citado na página 18.

Applequist, S., Durre, I., e Vose, R. (2024). The global historical climatology network monthly precipitation dataset, version 4. *Scientific Data*, 11(1):633. Citado na página 1.

Assent, I., Wichterich, M., Krieger, R., Kremer, H., e Seidl, T. (2009). Anticipatory dtw for efficient similarity search in time series databases. *Proceedings of the VLDB Endowment*, 2(1):826–837. Citado na página 19.

Ataspinar (2023). A guide for using the wavelet transform in machine learning. Disponível em: https://ataspinar.com/2018/12/21/a-guide-for-using-the-wavelet-transform-in-machine-learning/. Acesso em: 30 de Julho de 2023. Citado nas páginas 9 and 10.

Bagnall, A., Dau, H. A., Lines, J., Flynn, M., Large, J., Bostrom, A., Southam, P., e Keogh, E. (2018). The uea multivariate time series classification archive, 2018. *arXiv preprint arXiv:1811.00075*. Citado nas páginas 19 and 26.

Bagnall, A., Davis, L., Hills, J., e Lines, J. (2012). Transformation based ensembles for time series classification. Em *Proceedings of the 2012 SIAM international conference on data mining*, páginas 307–318. SIAM. Citado na página 6.

Bagnall, A., Lines, J., Bostrom, A., Large, J., e Keogh, E. (2017). The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, 31:606–660. Citado na página 40.

Batista, G. E., Keogh, E. J., Tataw, O. M., e De Souza, V. M. (2014). Cid: an efficient complexity-invariant distance for time series. *Data Mining and Knowledge Discovery*, 28:634–669. Citado na página 5.

Berndt, D. J. e Clifford, J. (1994). Using dynamic time warping to find patterns in time series. Em *KDD workshop*, volume 10, páginas 359–370. Seattle, WA, USA:. Citado na página 19.

Bisong, E. (2019). Google colaboratory. *Building machine learning and deep learning models on google cloud platform: a comprehensive guide for beginners*, páginas 59–64. Citado na página 43.

Bostrom, A. e Bagnall, A. (2015). Binary shapelet transform for multiclass time series classification. Em *Big Data Analytics and Knowledge Discovery: 17th International Conference, DaWaK 2015, Valencia, Spain, September 1-4, 2015, Proceedings 17*, páginas 257–269. Springer. Citado na página 27.

Breiman, L. (1984). *Classification and regression trees*. Routledge. Citado na página 13.

Cabello, N., Naghizade, E., Qi, J., e Kulik, L. (2020). Fast and accurate time series classification through supervised interval search. Em *2020 IEEE International Conference on Data Mining (ICDM)*, páginas 948–953. IEEE. Citado na página 30.

Caiado, J., Crato, N., e Peña, D. (2006). A periodogram-based metric for time series classification. *Computational Statistics & Data Analysis*, 50(10):2668–2684. Citado na página 27.

Casali, Y., Yonca, N. A., Comes, T., e Casali, Y. (2022). Machine learning for spatial analyses in urban areas: a scoping review. *Sustainable Cities and Society*, página 104050. Citado na página 1.

Catalano, R. (1981). Contending with rival hypotheses in correlation of aggregate time-series (cats): An overview for community psychologists. *American journal of community psychology*, 9(6):667–679. Citado na página 1.

Cha, S.-H. (2007). Comprehensive survey on distance/similarity measures between probability density functions. *City*, 1(2):1. Citado na página 19.

Chan, K.-P. e Fu, A. W.-C. (1999). Efficient time series matching by wavelets. Em *Proceedings 15th International Conference on Data Engineering (Cat. No. 99CB36337)*, páginas 126–133. IEEE. Citado nas páginas 8 and 33.

Chauhan, R., Ghanshala, K. K., e Joshi, R. (2018). Convolutional neural network (cnn) for image detection and recognition. Em *2018 first international conference on secure cyber computing and communication (ICSCCC)*, páginas 278–282. IEEE. Citado na página 14.

Chen, C. W., Liu, F.-C., e So, M. K. (2011). A review of threshold time series models in finance. *Statistics and its Interface*, 4(2):167–181. Citado na página 1.

Chiu, B., Keogh, E., e Lonardi, S. (2003). Probabilistic discovery of time series motifs. Em *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, páginas 493–498. Citado na página 5.

Christ, M., Braun, N., Neuffer, J., e Kempa-Liehr, A. W. (2018). Time series feature extraction on basis of scalable hypothesis tests (tsfresh–a python package). *Neurocomputing*, 307:72–77. Citado nas páginas 25 and 33.

Cortes, C. e Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20:273–297. Citado nas páginas 14 and 35.

Cover, T. (1968). Estimation by the nearest neighbor rule. *IEEE Transactions on Information Theory*, 14(1):50–55. Citado nas páginas 12 and 13.

Cui, Z., Chen, W., e Chen, Y. (2016). Multi-scale convolutional neural networks for time series classification. *arXiv preprint arXiv:1603.06995*. Citado nas páginas 19 and 27.

Das, K. e Behera, R. N. (2017). A survey on machine learning: concept, algorithms and applications. *International Journal of Innovative Research in Computer and Communication Engineering*, 5(2):1301–1309. Citado na página 12.

Dau, H. A., Bagnall, A., Kamgar, K., Yeh, C.-C. M., Zhu, Y., Gharghabi, S., Ratanamahatana, C. A., e Keogh, E. (2019). The ucr time series archive. *IEEE/CAA Journal of Automatica Sinica*, 6(6):1293–1305. Citado na página 17.

Davis, H. T. e Nelson, W. (1935). *The analysis of time series.* Principia press. Citado na página 5.

de Freitas Júnior, M. A. (2021). Extração e seleção de características para a classificação eficiente de séries temporais. *Programa de Pós-Graduação em Ciência da Computação - Universidade Federal de Uberlândia*, 1(1):1–72. Citado nas páginas 1, 6, and 40.

Dempster, A., Petitjean, F., e Webb, G. I. (2022). Rocket: exceptionally fast and accurate time series classification using random convolutional kernels. *Data Mining and Knowledge Discovery*, 34(5):1454–1495. Citado nas páginas 2, 20, 21, 22, 28, and 31.

Deng, H., Runger, G., Tuv, E., e Vladimir, M. (2013). A time series forest for classification and feature extraction. *Information Sciences*, 239(1):142–153. Citado nas páginas 18, 27, 28, and 30.

Developers, T. (2022). Tensorflow. *Zenodo*. Citado na página 43.

Dong, X., Yu, Z., Cao, W., Shi, Y., e Ma, Q. (2020). A survey on ensemble learning. *Frontiers of Computer Science*, 14:241–258. Citado na página 36.

Faloutsos, C., Ranganathan, M., e Manolopoulos, Y. (1994). Fast subsequence matching in time-series databases. *ACM SIGMOD*, 23(2):419–429. Citado na página 19.

Faouzi, J. e Janati, H. (2020). pyts: A python package for time series classification. *The Journal of Machine Learning Research*, 21(1):1720–1725. Citado na página 43.

Fawaz, H. I., Forestier, G., Weber, J., Idoumghar, L., e Muller, P.-A. (2019a). Deep learning for time series classification: A review. *Data Mining and Knowledge Discovery*, 33(4):917–963. Citado na página 19.

Fawaz, H. I., Lucas, B., Forestier, G., Pelletier, C., Schmidt, D. F., Weber, J., Webb, G. L., Idoumghar, L., Muller, P.-A., e Petitjean, F. (2020). Inceptiontime: Finding alexnet for time series classification. *Data Mining and Knowledge Discovery*, 34(6):1936–1962. Citado nas páginas 2, 19, 22, and 31.

Fawaz, I., Forestier, H., e Weber, G. (2019b). Deep learning for time series classification: A review. *Data Mining Knowlegde Discovery*, 33(1):917–963. Citado na página 5.

Fida, K., Abbasi, U., Adnan, M., Iqbal, S., e Mohamed, S. E. G. (2024). A comprehensive survey on load forecasting hybrid models: Navigating the futuristic demand response patterns through experts and intelligent systems. *Results in Engineering*, página 102773. Citado na página 1.

Foumani, N. M., Miller, L., Tan, C. W., Webb, G. I., Forestier, G., e Salehi, M. (2023). Deep learning for time series classification and extrinsic regression: A current survey. *Arxiv*, 1(1):1–37. Citado nas páginas 15 and 26.

Frank, H., Kersting, K., Lijffit, J., Valera, I., Middlehurst, M., Large, J., Cawley, G., e Bagnall, A. (2021). The temporal dictionary ensemble (tde) classifier for time series classification. Em *Machine Learning and Knowledge Discovery in Databases*, páginas 660–676. Citado nas páginas 2 and 18.

Fu, T.-c. (2011). A review on time series data mining. *Engineering Applications of Artificial Intelligence*, 24(1):164–181. Citado na página 9.

Fulcher, B. D. e Jones, N. S. (2017). hctsa: A computational framework for automated time-series phenotyping using massive feature extraction. *Cell systems*, 5(5):527–531. Citado nas páginas 18 and 22.

Gamboa, J. C. B. (2017). Deep learning for time-series analysis. *arXiv preprint arXiv:1701.01887*. Citado na página 19.

Giusti, R. (2017). Classificação de séries temporais utilizando diferentes representações de dados e ensembles. *Biblioteca Digital de Teses e Dissertações da USP*. Citado nas páginas 6, 11, 19, 31, and 59.

Giusti, R. e Batista, G. E. (2013). An empirical comparison of dissimilarity measures for time series classification. Em *2013 Brazilian Conference on Intelligent Systems*, páginas 82–88. IEEE. Citado na página 12.

Granger, C. W. e Newbold, P. (1974). Spurious regressions in econometrics. *Journal of econometrics*, 2(2):111–120. Citado na página 5.

Guo, G., Wang, H., Bell, D., Bi, Y., e Greer, K. (2003). Knn model-based approach in classification. Em *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE: OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2003, Catania, Sicily, Italy, November 3-7, 2003. Proceedings*, páginas 986–996. Springer. Citado nas páginas 2 and 17.

Géron, A. (2021). *Mãos à Obra: Aprendizado de Máquina com Scikit-Learn, Keras & TensorFlow: Conceitos, Ferramentas, e Técnicas para a Construção de Sistemas Inteligentes*. Alta Books Editora & O'Reilly. Citado na página 43.

Hammond, D. C. (2011). What is neurofeedback: An update. *Journal of neurotherapy*, 15(4):305–336. Citado na página 1.

Hao, J. e Ho, T. K. (2019). Machine learning made easy: a review of scikit-learn package in python programming language. *Journal of Educational and Behavioral Statistics*, 44(3):348–361. Citado na página 43.

Hills, J., Lines, J., Baranauskas, E., Mapp, J., e Bagnall, A. (2014). Classification of time series by shapelet transformation. *Data mining and knowledge discovery*, 28:851–881. Citado na página 27.

Hyndman, R. J. e Killick, R. (2023). Cran task view: Time series analysis. *Data Mining and Knowledge Discovery*. Citado na página 5.

Jin, M., Koh, H. Y., Wen, Q., Zambon, D., Alippi, C., Webb, G. I., King, I., e Pan, S. (2024). A survey on graph neural networks for time series: Forecasting, classification, imputation, and anomaly detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. Citado na página 1.

Jolliffe, I. (2002). Pca for time series and other non-independent data. *Principal component analysis (2nd Ed.). New York: Springer-Verlag*. Citado na página 17.

Keogh, E., Chakrabarti, K., Pazzani, M., e Mehrotra, S. (2001). Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and information Systems*, 3:263–286. Citado nas páginas 10, 11, and 33.

Klamroth, K. (2006). *Single-facility location problems with barriers*. Springer Science & Business Media. Citado na página 18.

Large, J., Bagnall, A., Malinowski, S., e Tavenard, R. (2019a). On time series classification with dictionary-based classifiers. *Intelligent Data Analysis*, 23(5):1073–1089. Citado na página 22.

Large, J., Lines, J., e Bagnall, A. (2019b). A probabilistic classifier ensemble weighting scheme based on cross-validated accuracy estimates. *Data mining and knowledge discovery*, 33(6):1674–1709. Citado nas páginas 30 and 36.

Lazebnik, S., Schmid, C., e Ponce, J. (2006). Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. Em *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*, volume 2, páginas 2169–2178. IEEE. Citado na página 30.

LeCun, Y., Bengio, Y., e Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444. Citado na página 18.

Lee, T.-Y., Jeon, S.-Y., Han, J., Skvortsov, V., Nikitin, K., e Ka, M.-H. (2016). A simplified technique for distance and velocity measurements of multiple moving objects using a linear frequency modulated signal. *IEEE Sensors Journal*, 16(15):5912–5920. Citado na página 9.

Lin, J., Keogh, E. J., Wei, L., e Lonardi, S. (2007). Experiencing sax: a novel symbolic representation of time series. *Data mining and Knowlegde Discovery*, 15(1):107–144. Citado nas páginas 11 and 33.

Lines, J. e Bagnall, A. (2015). Time series classification with ensembles of elastic distance measures. *Data Mining and Knowledge Discovery*, 29:565–592. Citado na página 28.

Lines, J., Davis, L. M., Hills, J., e Bagnall, A. (2012). A shapelet transform for time series classification. Em *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, páginas 289–297. Citado nas páginas 2 and 22.

Lines, J., Taylor, S., e Bagnall, A. (2018). Time series classification with hive-cote: The hierarchical vote collective of transformation-based ensembles. *ACM Transactions on Knowledge Discovery from Data*, 12(5):1–35. Citado nas páginas 18, 20, 22, 25, 26, 27, 28, and 31.

Löning, M., Bagnall, A., Ganesh, S., Kazakov, V., Lines, J., e Király, F. J. (2019). sktime: A unified interface for machine learning with time series. *arXiv preprint arXiv:1909.07872*. Citado na página 43.

Lubba, C. H., Sethi, S. S., Knaute, P., Schultz, S. R., Fulcher, B. D., e Jones, N. S. (2019). catch22: Canonical time-series characteristics: Selected through highly comparative time-series analysis. *Data Mining and Knowledge Discovery*, 33(6):1821–1852. Citado nas páginas 2, 22, 24, 30, and 55.

Lucas, B., Shifaz, A., Pelletier, C., O'Neill, L., Zaidi, N., Goethals, B., Petitjean, F., e Webb, G. I. (2019). Proximity forest: an effective and scalable distance-based classifier for time series. *Data Mining and Knowledge Discovery*, 33(3):607–635. Citado nas páginas 2 and 22.

Ma, Q., Shen, L., Chen, W., Wang, J., Wei, J., e Yu, Z. (2016). Functional echo state network for time series classification. *Information Sciences*, 373:1–20. Citado na página 19.

Marwan, N., Thiel, M., e Nowaczyk, N. R. (2002). Cross recurrence plot based synchronization of time series. *Nonlinear processes in Geophysics*, 9(3/4):325–331. Citado na página 15.

Mazzonetto, S. e Pigato, P. (2024). Estimation of parameters and local times in a discretely observed threshold diffusion model. *arXiv preprint arXiv:2403.06858*. Citado na página 1.

Middlehurst, M. e Bagnall, A. (2022). The freshprince: A simple transformation based pipeline time series classifier. Em *International Conference on Pattern Recognition and Artificial Intelligence*, páginas 150–161. Springer. Citado nas páginas 25, 55, and 56.

Middlehurst, M., Large, J., e Bagnall, A. (2020). The canonical interval forest (cif) classifier for time series classification. *2020 IEEE International Conference on Big Data (Big Data)*, 161(1):188–195. Citado na página 30.

Middlehurst, M., Large, J., Cawley, G., e Bagnall, A. (2021a). The temporal dictionary ensemble (tde) classifier for time series classification. Em *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2020, Ghent, Belgium, September 14–18, 2020, Proceedings, Part I*, páginas 660–676. Springer. Citado na página 30.

Middlehurst, M., Large, J., Flynn, M., Lines, J., Bostrom, A., e Bagnall, A. (2021b). Hive-cote 2.0: a new meta ensemble for time series classification. *Machine Learning*, 110(11-12):3211–3243. Citado nas páginas 18, 27, 28, 29, and 30.

Middlehurst, M., Schäfer, P., e Bagnall, A. (2024). Bake off redux: a review and experimental evaluation of recent time series classification algorithms. *Data Mining and Knowledge Discovery*, páginas 1–74. Citado nas páginas 1 and 36.

Mohammadi Foumani, N., Miller, L., Tan, C. W., Webb, G. I., Forestier, G., e Salehi, M. (2024). Deep learning for time series classification and extrinsic regression: A current survey. *ACM Computing Surveys*, 56(9):1–45. Citado na página 5.

Mohit Chaudhary (2023). Random forest algorithm - how it works and why it's so effective. Disponível em: https://www.turing.com/kb/random-forest-algorithm. Acesso em: 30 de Setembro de 2024. Citado na página 13.

Molnar, C., König, G., Bischl, B., e Casalicchio, G. (2024). Model-agnostic feature importance and effects with dependent features: a conditional subgroup approach. *Data Mining and Knowledge Discovery*, 38(5):2903–2941. Citado na página 16.

Nanopoulos, A., Alcock, R., e Manolopoulos, Y. (2001). Feature-based classification of time-series data. *International Journal of Computer Research*, 10(3):49–61. Citado na página 17.

Newbold, P. (1983). Arima model building and the time series analysis approach to forecasting. *Journal of forecasting*, 2(1):23–35. Citado na página 17.

Nielsen, A. (2021). *Análise Prática de Séries Temporais - Predição com Estatística e Aprendizado de Máquina*. Alta Books Editora & O'Reilly. Citado na página 14.

Oyebode, O. e Ighravwe, D. (2019). Urban water demand forecasting: A comparative evaluation of conventional and soft computing techniques. *Resources*, 8:156. Citado na página 14.

Pal, M. (2005). Random forest classifier for remote sensing classification. *International journal of remote sensing*, 26(1):217–222. Citado na página 36.

Peng, Z. (2017). Multilayer perceptron algebra. *arXiv preprint arXiv:1701.04968*. Citado na página 19.

Pranti, R. C. e Batista, G. E. d. A. P. A. (2012). Distância invariante à complexidade baseada em dimensão fractal para classificação de séries temporais. Em *Proceedings*. Citado na página 19.

Rakthanmanon, T., Campana, B., Mueen, A., Batista, G., Westover, B., Zhu, Q., Zakaria, J., e Keogh, E. (2012). Searching and mining trillions of time series subsequences under dynamic time warping. Em *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, páginas 262–270. Citado na página 19.

Rodriguez, J. J., Kuncheva, L. I., e Alonso, C. J. (2006). Rotation forest: A new classifier ensemble method. *IEEE transactions on pattern analysis and machine intelligence*, 28(10):1619–1630. Citado nas páginas 25 and 29.

Ruiz, A. P., Flynn, M., Large, J., Middlehurst, M., e Bagnall, A. (2021). The great multivariate time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, 35(2):401–449. Citado na página 40.

Schäfer, P. (2015). The boss is concerned with time series classification in the presence of noise. *Data Mining and Knowledge Discovery*, 29:1505–1530. Citado nas páginas 2, 22, 28, and 30.

Serrà, J., Pascual, S., e Karatzoglou, A. (2018). Towards a universal neural network encoder for time series. Em *CCIA*, páginas 120–129. Citado na página 19.

Shafer, P. e Leser, U. (2017). Fast and accurate time series classification with weasel. *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, 1(1):637–646. Citado na página 22.

Shifaz, A., Pelletier, C., Petitjean, F., e Webb, G. I. (2020). Ts-chief: a scalable and accurate forest algorithm for time series classification. *Data Mining and Knowledge Discovery*, 34(3):742–775. Citado nas páginas 18, 22, 25, and 26.

Shumway, R. H., Stoffer, D. S., e Stoffer, D. S. (2000). *Time series analysis and its applications*, volume 3. Springer. Citado nas páginas 8 and 33.

Silva, D. F., De Souza, V. M., e Batista, G. E. (2013). Time series classification using compression distance of recurrence plots. Em *2013 IEEE 13th International Conference on Data Mining*, páginas 687–696. IEEE. Citado na página 6.

Silva, D. F., Giacomini, A. H., e Ueno, C. L. (2022). Meta-learning-based selection of classification algorithm for time series. Data Mining and Knowledge Discovery. Citado nas páginas 21, 22, and 55.

Tavenard, R., Faouzi, J., Vandewiele, G., Divo, F., Androz, G., Holtz, C., Payne, M., Yurchak, R., Rußwurm, M., Kolar, K., et al. (2020). Tslearn, a machine learning toolkit for time series data. *The Journal of Machine Learning Research*, 21(1):4686–4691. Citado na página 43.

Tu, P.-L. e Chung, J.-Y. (1992). A new decision-tree classification algorithm for machine learning. Em *TAI'92-proceedings fourth international conference on tools with artificial intelligence*, páginas 370–371. IEEE Computer Society. Citado na página 13.

Vilalta, R. e Drissi, Y. (2002). A perspective view and survey of meta-learning. *Artificial intelligence review*, 18:77–95. Citado na página 34.

Voigt, J. D., Mosier, M., e Tendler, A. (2024). Systematic review and meta-analysis of neurofeedback and its effect on posttraumatic stress disorder. *Frontiers in Psychiatry*, 15:1323485. Citado na página 1.

Wang, Z., Yan, W., e Oates, T. (2017). Time series classification from scratch with deep neural networks: A strong baseline. Em *2017 International joint conference on neural networks (IJCNN)*, páginas 1578–1585. IEEE. Citado na página 22.

Wooldridge, J. M. (2015). *Introductory econometrics: A modern approach*. Cengage learning. Citado na página 5.

Xi, X., Keogh, E., Shelton, C., Wei, L., e Ratanamahatana, C. A. (2006). Fast time series classification using numerosity reduction. Em *Proceedings of the 23rd international conference on Machine learning*, páginas 1033–1040. Citado na página 6.

Xing, Z., Pei, J., e Keogh, E. (2010). A brief survey on sequence classification. *ACM Sigkdd Explorations Newsletter*, 12(1):40–48. Citado na página 6.

Yankov, D., Keogh, E., Medina, J., Chiu, B., e Zordan, V. (2007). Detecting time series motifs under uniform scaling. Em *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, páginas 844–853. Citado na página 5.

Yuan, J., Shi, M., Wang, Z., Liu, H., e Li, J. (2022). Random pairwise shapelets forest: an effective classifier for time series. *Knowledge and Information Systems*, 64(1):143–174. Citado na página 29.

Zhang, Y., Jin, R., e Zhou, Z.-H. (2010). Understanding bag-of-words model: a statistical framework. *International journal of machine learning and cybernetics*, 1:43–52. Citado na página 18.

Zhao, B., Lu, H., Chen, S., Liu, J., e Wu, D. (2017). Convolutional neural networks for time series classification. *Journal of Systems Engineering and Electronics*, 28(1):162–169. Citado na página 19.

Zhu, Q., Batista, G., Rakthanmanon, T., e Keogh, E. (2012). A novel approximation to dynamic time warping allows anytime clustering of massive time series datasets. *Proceedings of the 2012 SIAM International Conference on Data Mining (SDM)*, 1(1):999–1010. Citado na página 1.

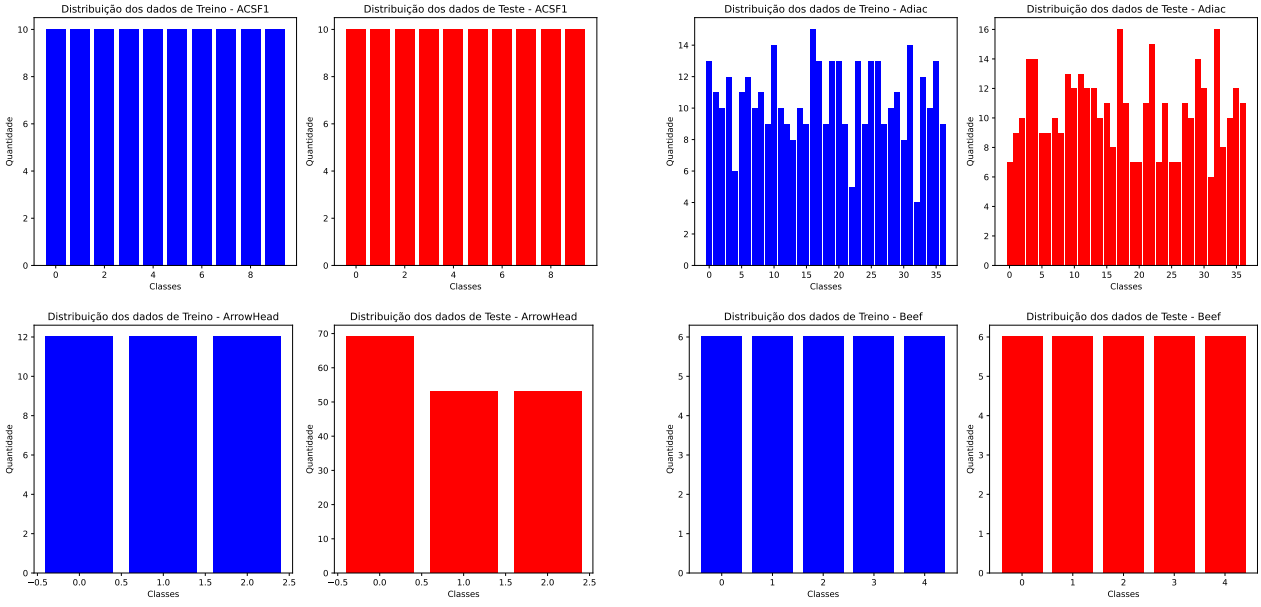
Apêndice A

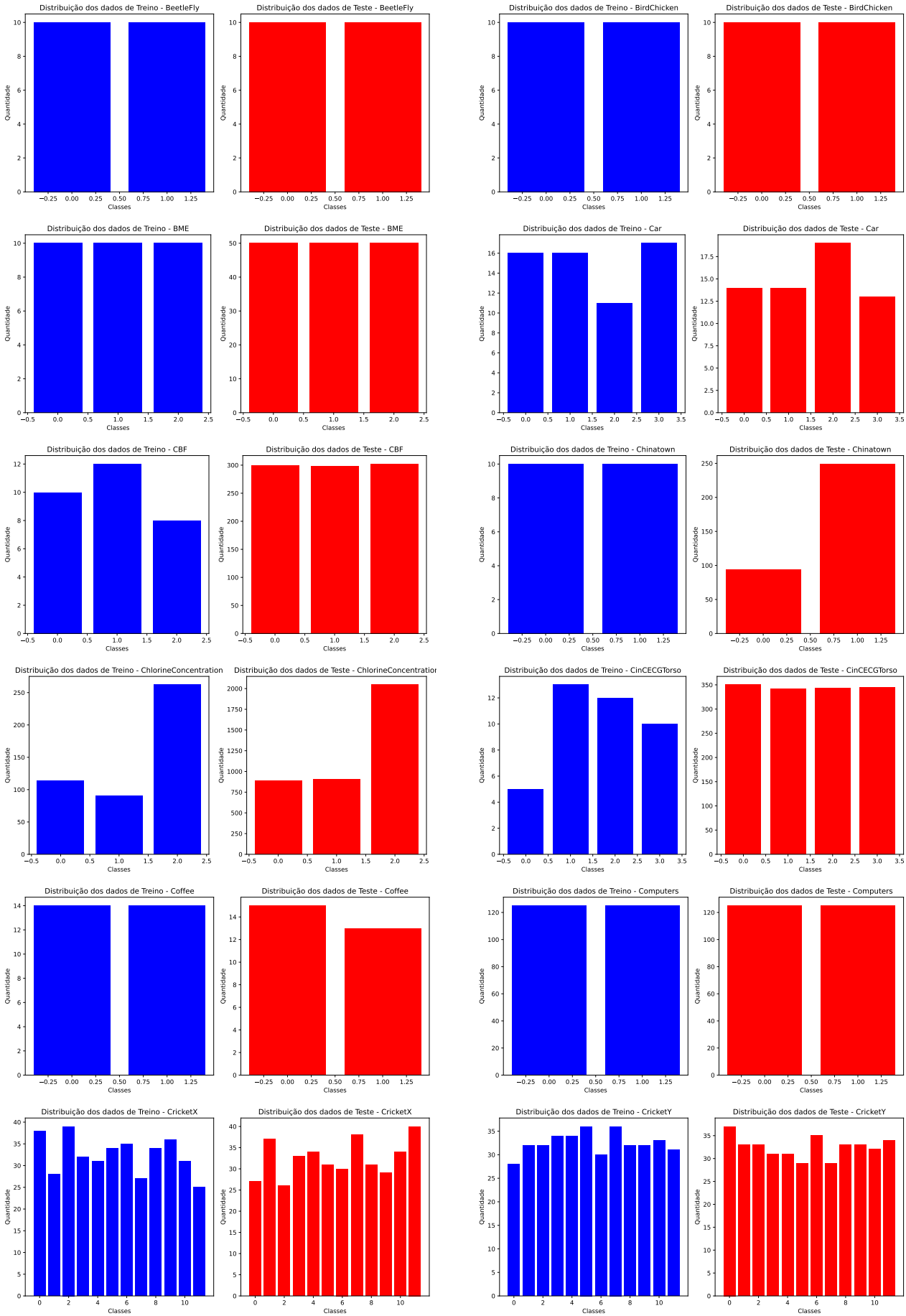
Resultados Estendidos

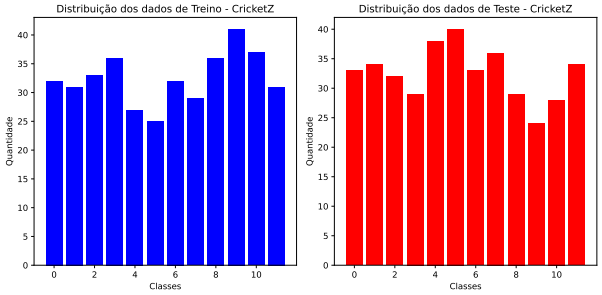
Esta Seção apresenta os resultados ampliados dos experimentos realizados neste trabalho. Os resultados, mesmo os menos relevantes, foram considerados necessários para a realização do estudo e alcance dos objetivos esperados.

A.1 Balanceamento dos conjuntos de dados do repositório UCR

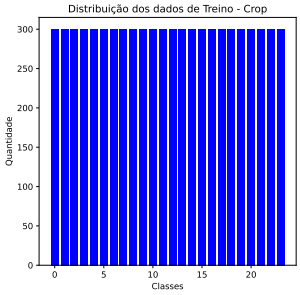
Os gráficos apresentados a seguir demonstram os resultados de um experimento que avaliou o balanceamento dos conjuntos de dados disponíveis no repositório UCR. A análise revelou que diversos conjuntos apresentavam dados desbalanceados, com algumas classes contendo significativamente mais exemplos que outras. Este desequilíbrio pode resultar em modelos que não generalizam bem, que, embora apresentem bom desempenho nas classes com mais exemplos, falham em classificar adequadamente as demais classes.

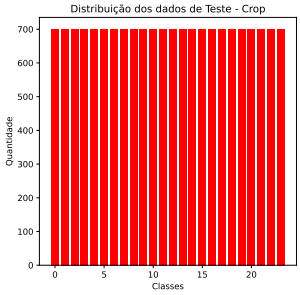


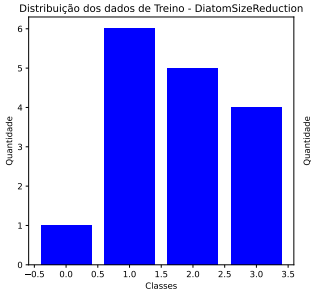

Distribuição dos dados de Treino - BeetleFly
Distribuição dos dados de Teste - BeetleFly
Distribuição dos dados de Treino - BirdChicken
Distribuição dos dados de Teste - BirdChicken
Distribuição dos dados de Treino - BME
Distribuição dos dados de Teste - BME
Distribuição dos dados de Treino - Car
Distribuição dos dados de Teste - Car
Distribuição dos dados de Treino - CBF
Distribuição dos dados de Teste - CBF
Distribuição dos dados de Treino - Chinatown
Distribuição dos dados de Teste - Chinatown
Distribuição dos dados de Treino - ChlorineConcentration
Distribuição dos dados de Teste - ChlorineConcentration
Distribuição dos dados de Treino - CinCECGTorso
Distribuição dos dados de Teste - CinCECGTorso
Distribuição dos dados de Treino - Coffee
Distribuição dos dados de Teste - Coffee
Distribuição dos dados de Treino - Computers
Distribuição dos dados de Teste - Computers
Distribuição dos dados de Treino - CricketX
Distribuição dos dados de Teste - CricketX
Distribuição dos dados de Treino - CricketY
Distribuição dos dados de Teste - CricketY
Quantidade
Classes

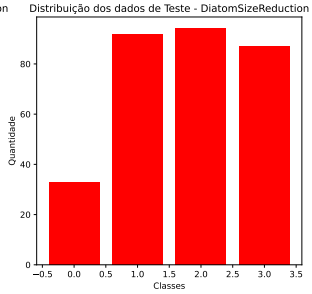

Distribuição dos dados de Treino - CricketZ
Quantidade
Classes

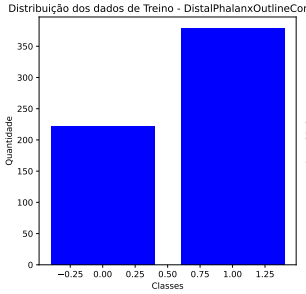
Distribuição dos dados de Teste - CricketZ
Quantidade
Classes

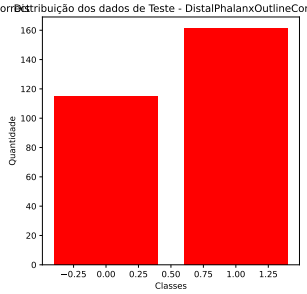

Distribuição dos dados de Treino - Crop
Quantidade
Classes

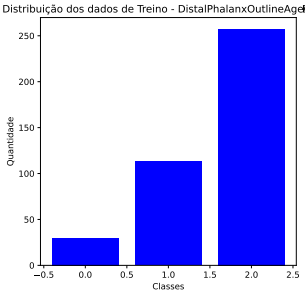

Distribuição dos dados de Teste - Crop
Quantidade
Classes

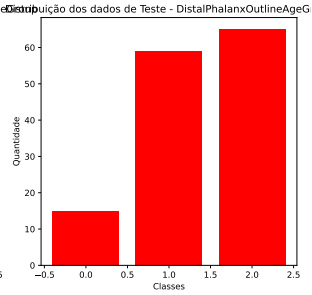

Distribuição dos dados de Treino - DiatomSizeReduction
Quantidade
Classes

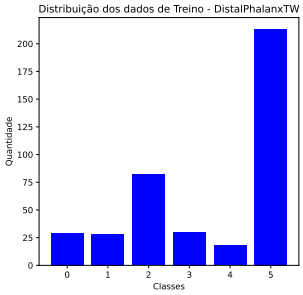

Distribuição dos dados de Teste - DiatomSizeReduction
Quantidade
Classes

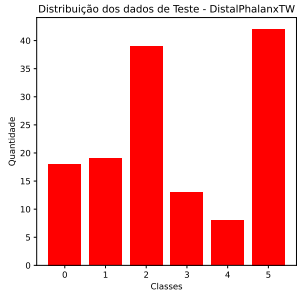

Distribuição dos dados de Treino - DistalPhalanxOutlineCor
Quantidade
Classes

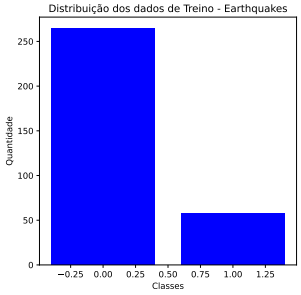

Distribuição dos dados de Teste - DistalPhalanxOutlineCor
Quantidade
Classes

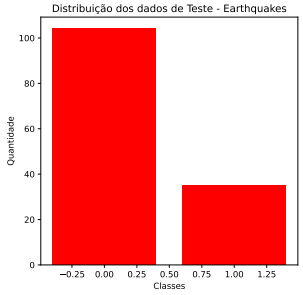

Distribuição dos dados de Treino - DistalPhalanxOutlineAgeG
Quantidade
Classes

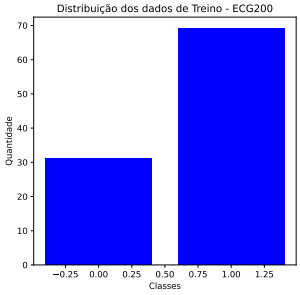

Distribuição dos dados de Teste - DistalPhalanxOutlineAgeG
Quantidade
Classes

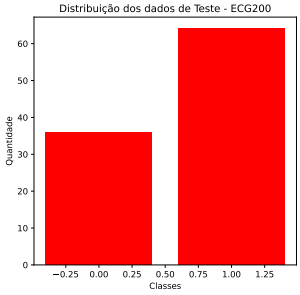

Distribuição dos dados de Treino - DistalPhalanxTW
Quantidade
Classes

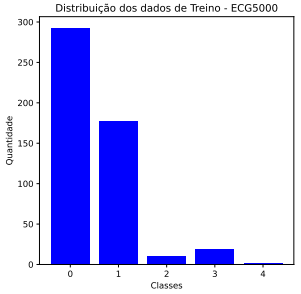

Distribuição dos dados de Teste - DistalPhalanxTW
Quantidade
Classes

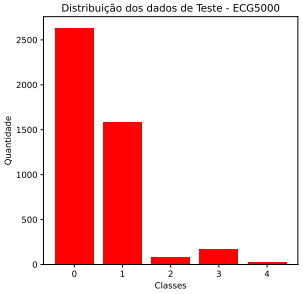

Distribuição dos dados de Treino - Earthquakes
Quantidade
Classes

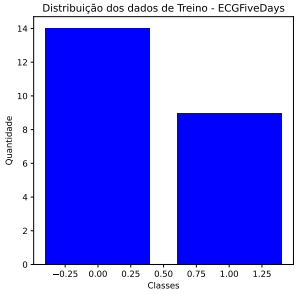

Distribuição dos dados de Teste - Earthquakes
Quantidade
Classes

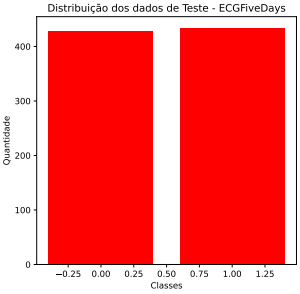

Distribuição dos dados de Treino - ECG200
Quantidade
Classes

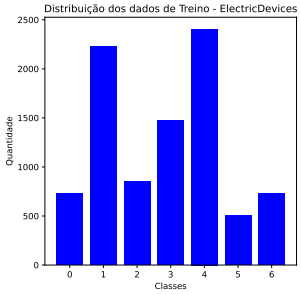

Distribuição dos dados de Teste - ECG200
Quantidade
Classes

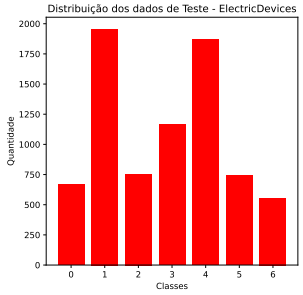

Distribuição dos dados de Treino - ECG5000
Quantidade
Classes

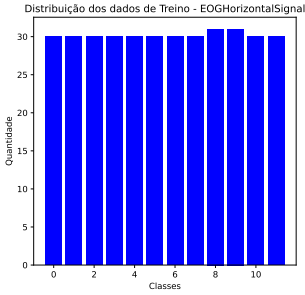

Distribuição dos dados de Teste - ECG5000
Quantidade
Classes

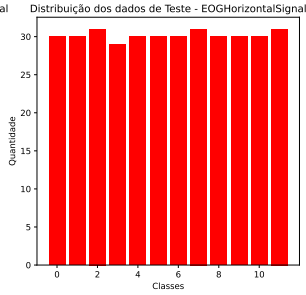

Distribuição dos dados de Treino - ECGFiveDays
Quantidade
Classes

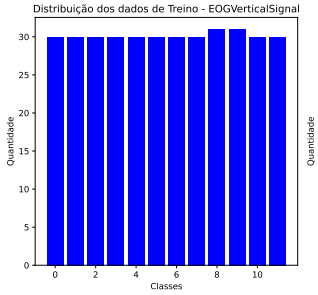

Distribuição dos dados de Teste - ECGFiveDays
Quantidade
Classes

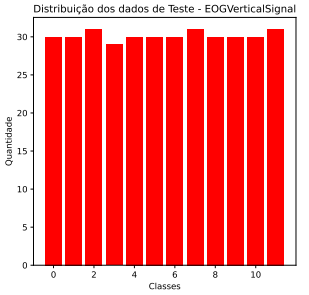

Distribuição dos dados de Treino - ElectricDevices
Quantidade
Classes

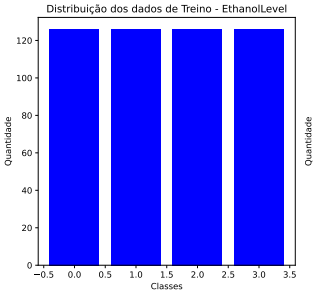

Distribuição dos dados de Teste - ElectricDevices
Quantidade
Classes

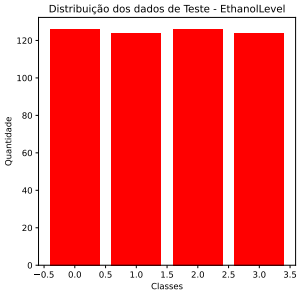

Distribuição dos dados de Treino - EOGHorizontalSignal
Quantidade
Classes

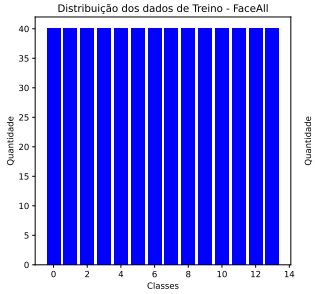

Distribuição dos dados de Teste - EOGHorizontalSignal
Quantidade
Classes

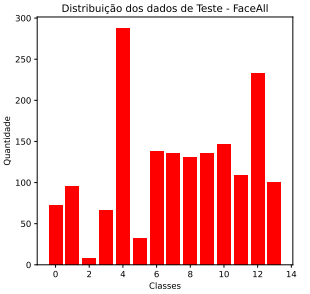

Distribuição dos dados de Treino - EOGVerticalSignal
Quantidade
Classes

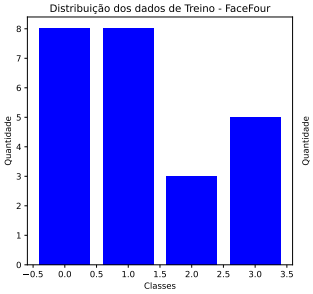

Distribuição dos dados de Teste - EOGVerticalSignal
Quantidade
Classes

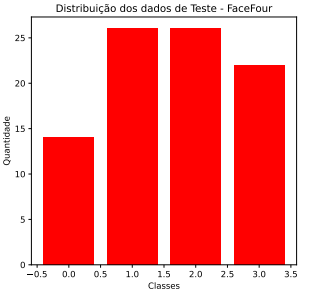

Distribuição dos dados de Treino - EthanolLevel
Quantidade
Classes

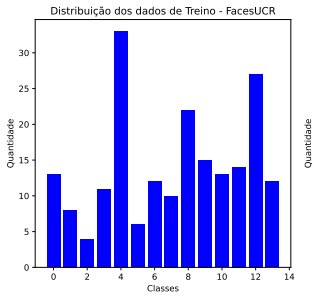

Distribuição dos dados de Teste - EthanolLevel
Quantidade
Classes

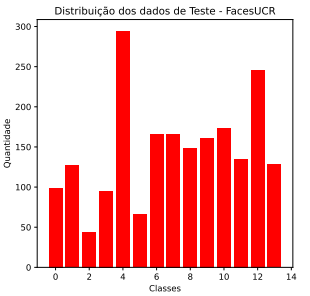

Distribuição dos dados de Treino - FaceAll
Quantidade
Classes

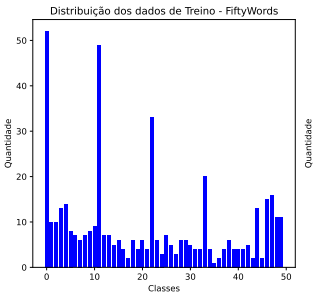

Distribuição dos dados de Teste - FaceAll
Quantidade
Classes

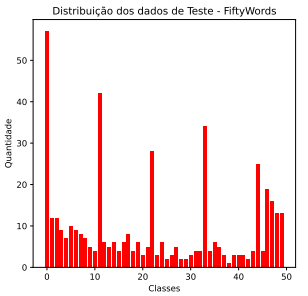

Distribuição dos dados de Treino - FaceFour
Quantidade
Classes

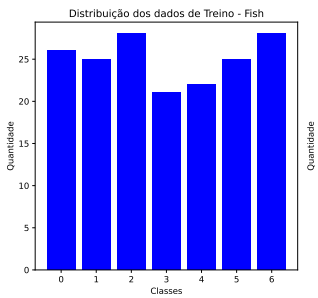

Distribuição dos dados de Teste - FaceFour
Quantidade
Classes

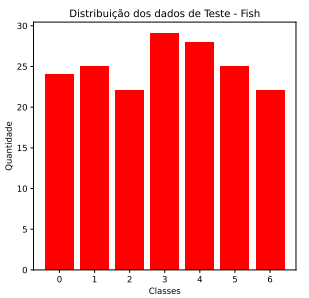

Distribuição dos dados de Treino - FacesUCR
Quantidade
Classes

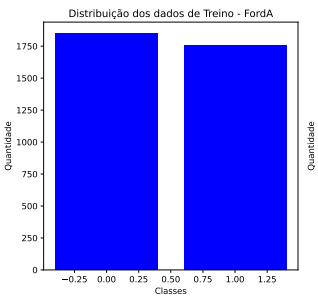

Distribuição dos dados de Teste - FacesUCR
Quantidade
Classes

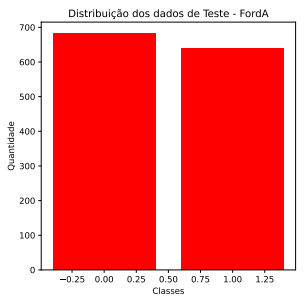

Distribuição dos dados de Treino - FiftyWords
Quantidade
Classes

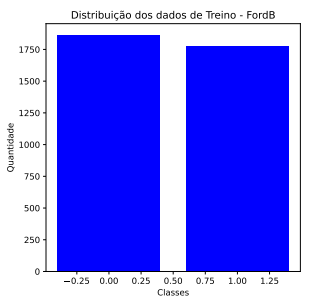

Distribuição dos dados de Teste - FiftyWords
Quantidade
Classes

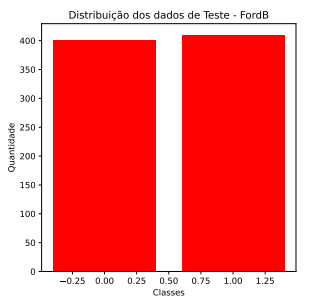

Distribuição dos dados de Treino - Fish
Quantidade
Classes

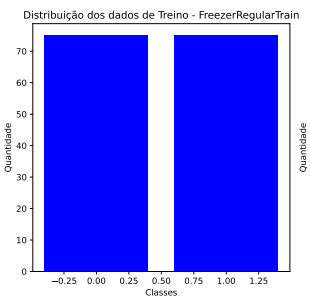

Distribuição dos dados de Teste - Fish
Quantidade
Classes

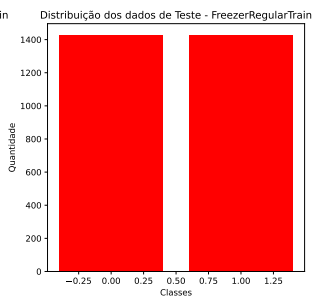

Distribuição dos dados de Treino - FordA
Quantidade
Classes

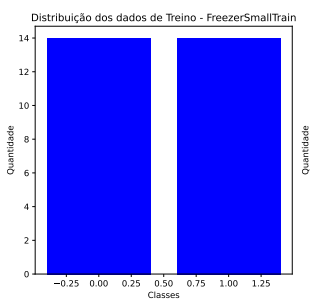

Distribuição dos dados de Teste - FordA
Quantidade
Classes

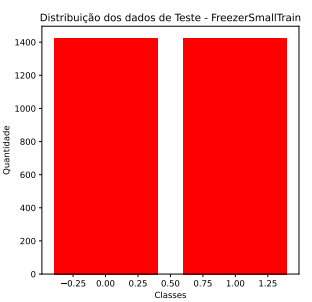

Distribuição dos dados de Treino - FordB
Quantidade
Classes

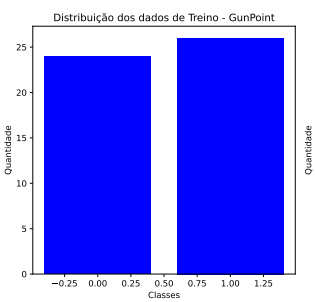

Distribuição dos dados de Teste - FordB
Quantidade
Classes

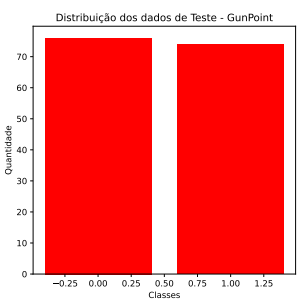

Distribuição dos dados de Treino - FreezerRegularTrain
Quantidade
Classes

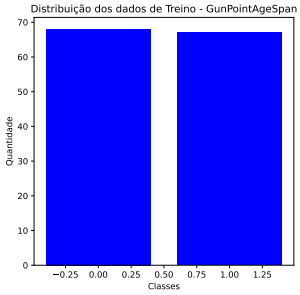

Distribuição dos dados de Teste - FreezerRegularTrain
Quantidade
Classes

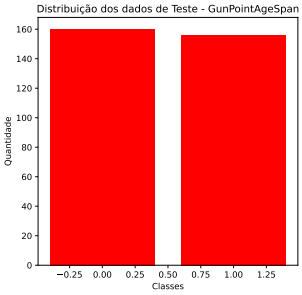

Distribuição dos dados de Treino - FreezerSmallTrain
Quantidade
Classes

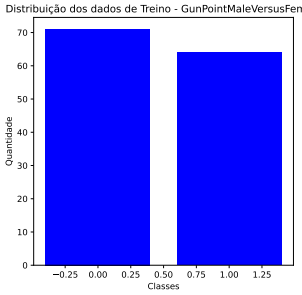

Distribuição dos dados de Teste - FreezerSmallTrain
Quantidade
Classes

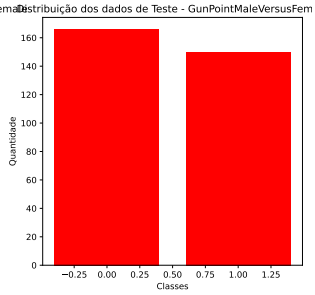

Distribuição dos dados de Treino - GunPoint
Quantidade
Classes

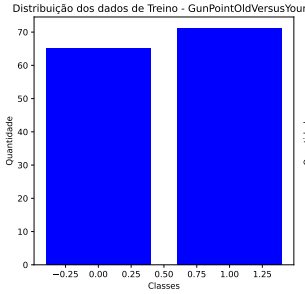

Distribuição dos dados de Teste - GunPoint
Quantidade
Classes

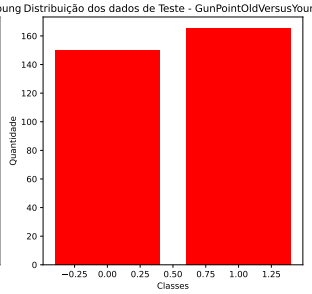

Distribuição dos dados de Treino - GunPointAgeSpan
Quantidade
Classes

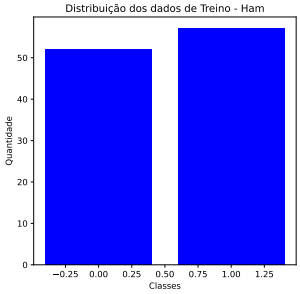

Distribuição dos dados de Teste - GunPointAgeSpan
Quantidade
Classes

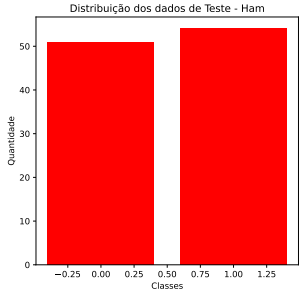

Distribuição dos dados de Treino - GunPointMaleVersusFemale
Quantidade
Classes

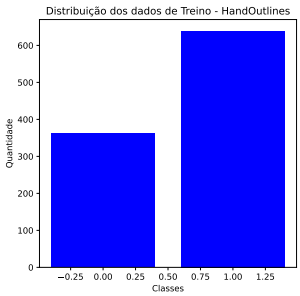

Distribuição dos dados de Teste - GunPointMaleVersusFem
Quantidade
Classes

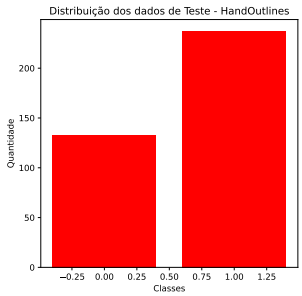

Distribuição dos dados de Treino - GunPointOldVersusYoung
Quantidade
Classes

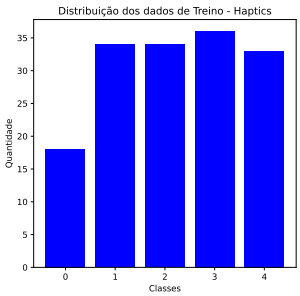

Distribuição dos dados de Teste - GunPointOldVersusYour
Quantidade
Classes

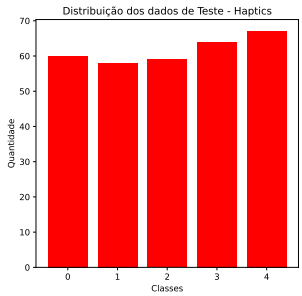

Distribuição dos dados de Treino - Ham
Quantidade
Classes

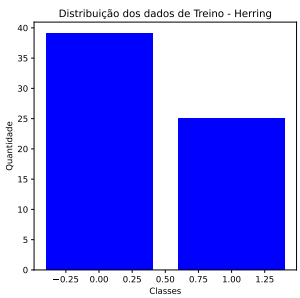

Distribuição dos dados de Teste - Ham
Quantidade
Classes

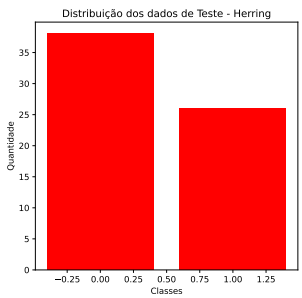

Distribuição dos dados de Treino - HandOutlines
Quantidade
Classes

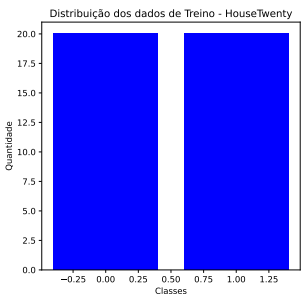

Distribuição dos dados de Teste - HandOutlines
Quantidade
Classes

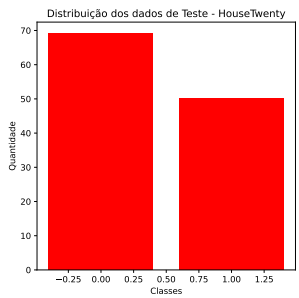

Distribuição dos dados de Treino - Haptics
Quantidade
Classes

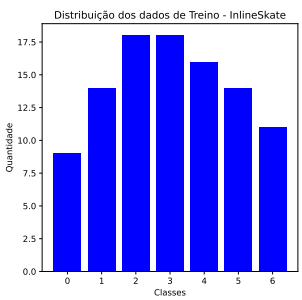

Distribuição dos dados de Teste - Haptics
Quantidade
Classes

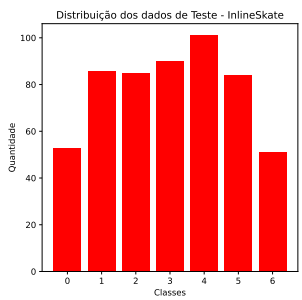

Distribuição dos dados de Treino - Herring
Quantidade
Classes

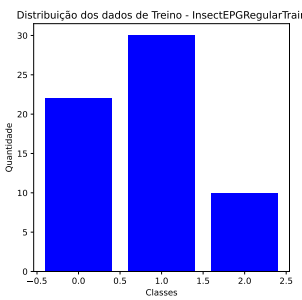

Distribuição dos dados de Teste - Herring
Quantidade
Classes

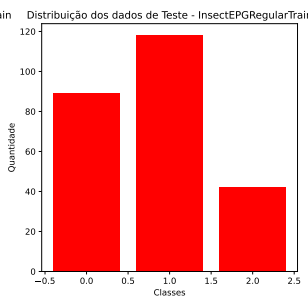

Distribuição dos dados de Treino - HouseTwenty
Quantidade
Classes

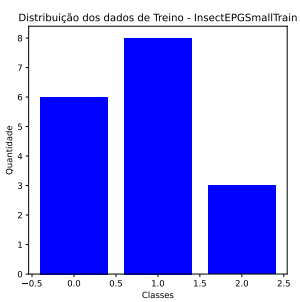

Distribuição dos dados de Teste - HouseTwenty
Quantidade
Classes

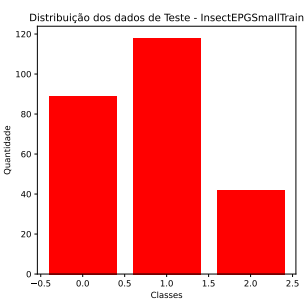

Distribuição dos dados de Treino - InlineSkate
Quantidade
Classes

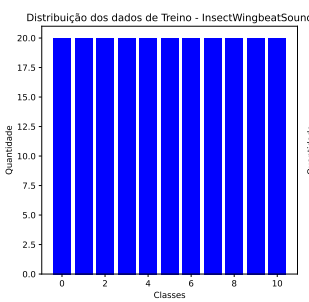

Distribuição dos dados de Teste - InlineSkate
Quantidade
Classes

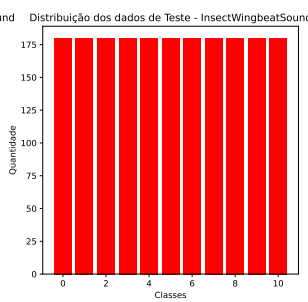

Distribuição dos dados de Treino - InsectEPGRegularTrain
Quantidade
Classes

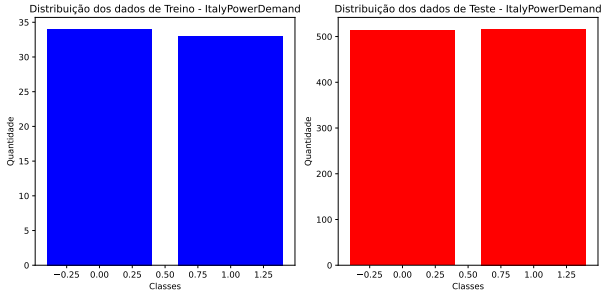

Distribuição dos dados de Teste - InsectEPGRegularTrain
Quantidade
Classes


Distribuição dos dados de Treino - InsectEPGSmallTrain
Quantidade
Classes

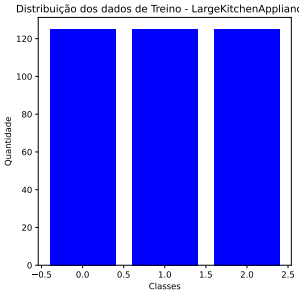

Distribuição dos dados de Teste - InsectEPGSmallTrain
Quantidade
Classes

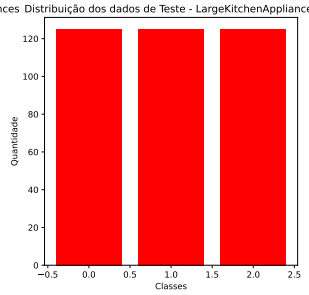

Distribuição dos dados de Treino - InsectWingbeatSound
Quantidade
Classes

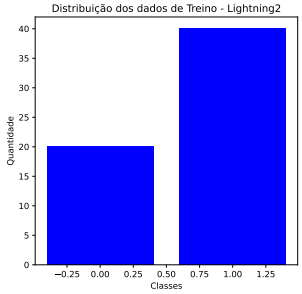

Distribuição dos dados de Teste - InsectWingbeatSound
Quantidade
Classes

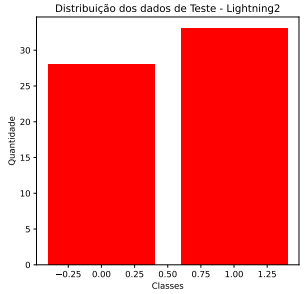

Distribuição dos dados de Treino - ItalyPowerDemand
Quantidade
Classes

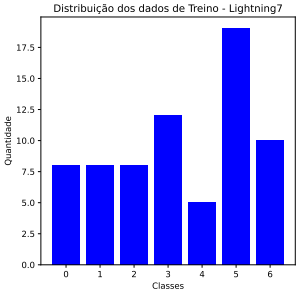
Distribuição dos dados de Teste - ItalyPowerDemand
Quantidade
Classes

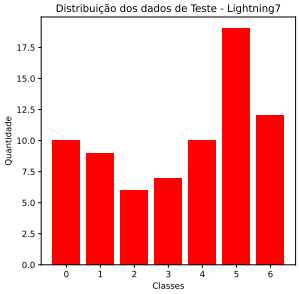

Distribuição dos dados de Treino - LargeKitchenAppliances
Quantidade
Classes

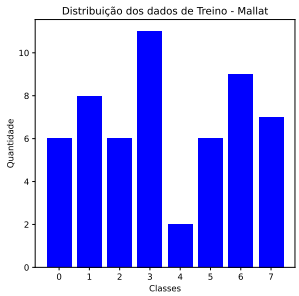

Distribuição dos dados de Teste - LargeKitchenAppliance
Quantidade
Classes

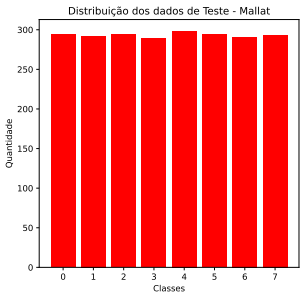

Distribuição dos dados de Treino - Lightning2
Quantidade
Classes

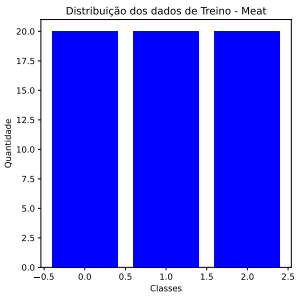

Distribuição dos dados de Teste - Lightning2
Quantidade
Classes

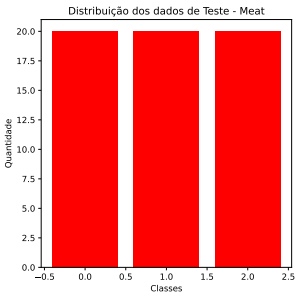

Distribuição dos dados de Treino - Lightning7
Quantidade
Classes

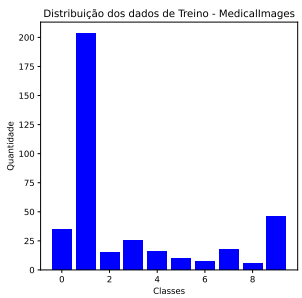

Distribuição dos dados de Teste - Lightning7
Quantidade
Classes

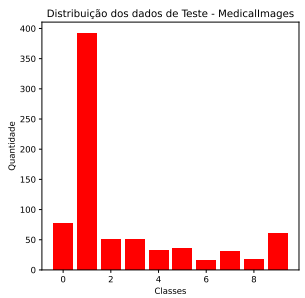

Distribuição dos dados de Treino - Mallat
Quantidade
Classes

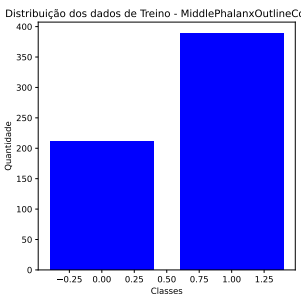

Distribuição dos dados de Teste - Mallat
Quantidade
Classes

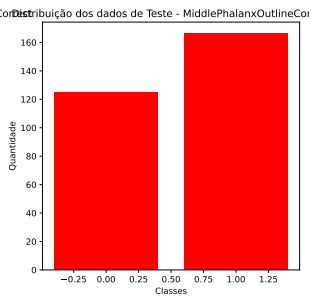

Distribuição dos dados de Treino - Meat
Quantidade
Classes

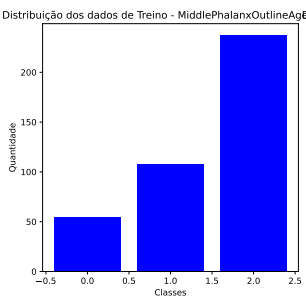

Distribuição dos dados de Teste - Meat
Quantidade
Classes

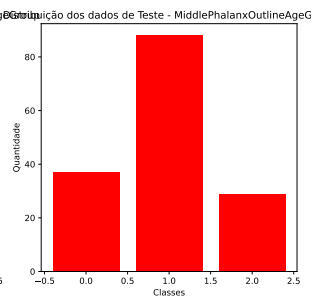

Distribuição dos dados de Treino - MedicalImages
Quantidade
Classes

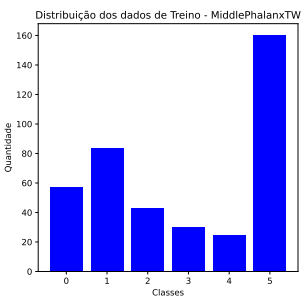

Distribuição dos dados de Teste - MedicalImages
Quantidade
Classes

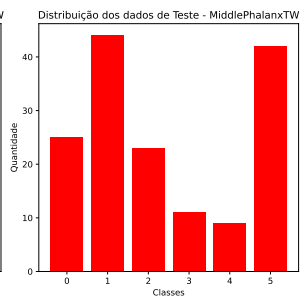

Distribuição dos dados de Treino - MiddlePhalanxOutlineCor
Quantidade
Classes

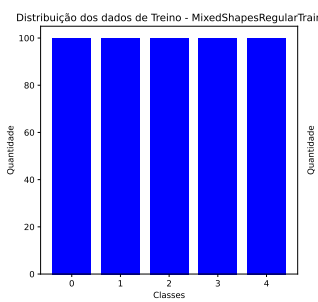

Distribuição dos dados de Teste - MiddlePhalanxOutlineCo
Quantidade
Classes

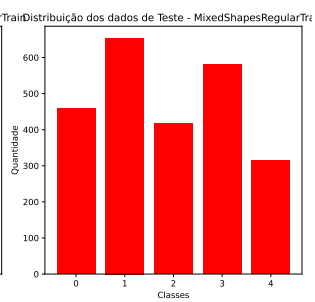

Distribuição dos dados de Treino - MiddlePhalanxOutlineAgeG
Quantidade
Classes

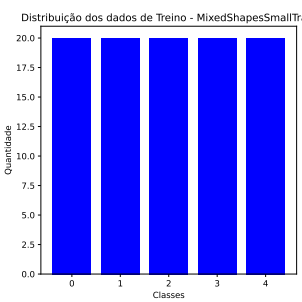

Distribuição dos dados de Teste - MiddlePhalanxOutlineAgeG
Quantidade
Classes

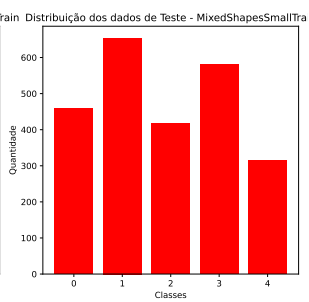

Distribuição dos dados de Treino - MiddlePhalanxTW
Quantidade
Classes

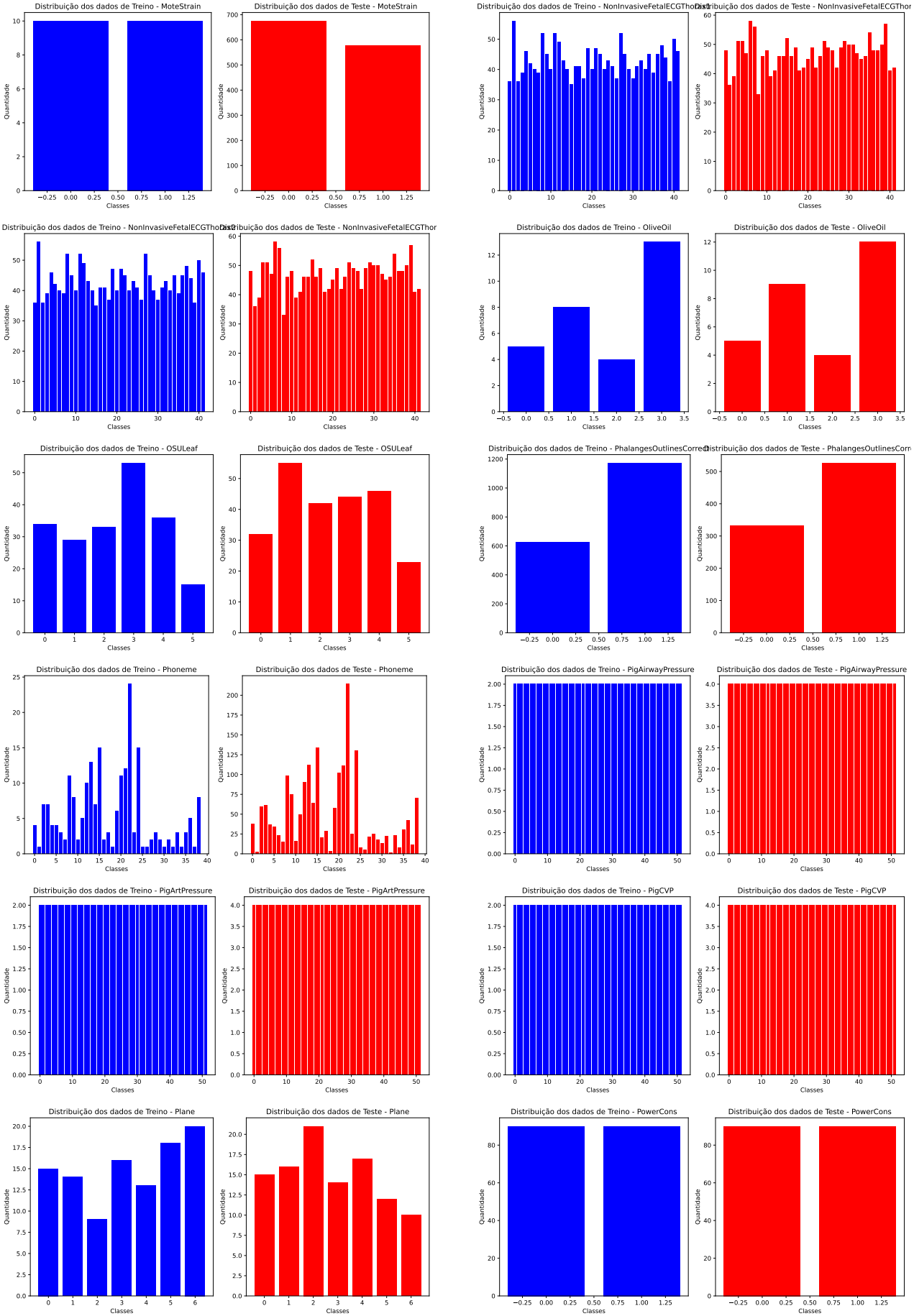

Distribuição dos dados de Teste - MiddlePhalanxTW
Quantidade
Classes

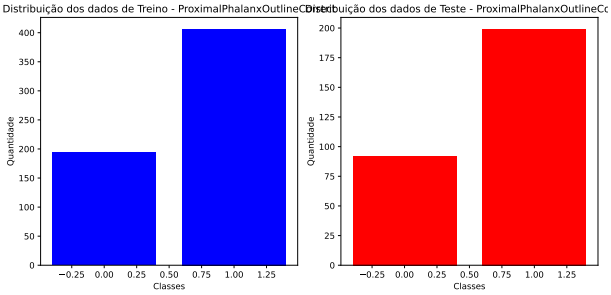

Distribuição dos dados de Treino - MixedShapesRegularTrain
Quantidade
Classes

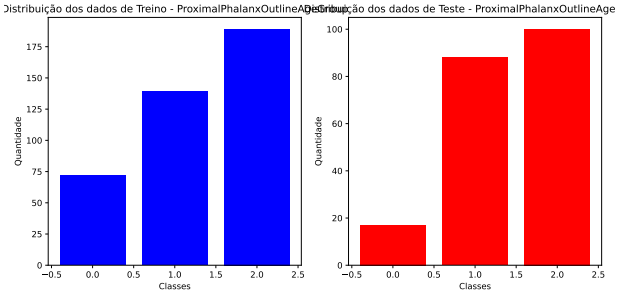

Distribuição dos dados de Teste - MixedShapesRegularTra
Quantidade
Classes

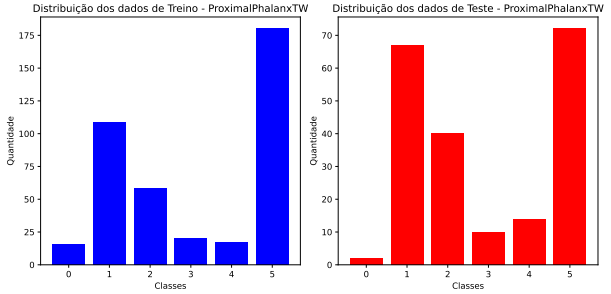

Distribuição dos dados de Treino - MixedShapesSmallTrain
Quantidade
Classes

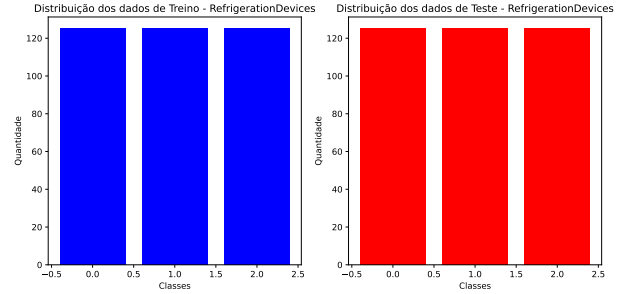

Distribuição dos dados de Teste - MixedShapesSmallTra
Quantidade
Classes

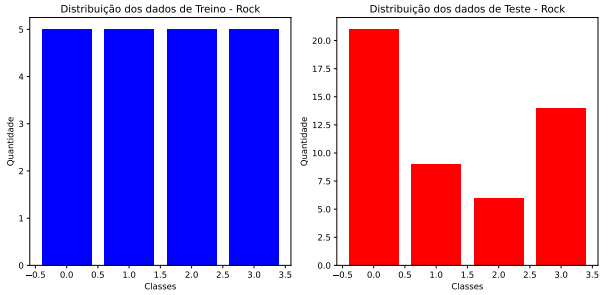

Distribuição dos dados de Treino - MoteStrain
Distribuição dos dados de Teste - MoteStrain
Distribuição dos dados de Treino - NonInvasiveFetalECGThorax1
Distribuição dos dados de Teste - NonInvasiveFetalECGThor
Distribuição dos dados de Treino - NonInvasiveFetalECGThorax2
Distribuição dos dados de Teste - NonInvasiveFetalECGThor
Distribuição dos dados de Treino - OliveOil
Distribuição dos dados de Teste - OliveOil
Distribuição dos dados de Treino - OSULeaf
Distribuição dos dados de Teste - OSULeaf
Distribuição dos dados de Treino - PhalangesOutlinesCorrect
Distribuição dos dados de Teste - PhalangesOutlinesCorr
Distribuição dos dados de Treino - Phoneme
Distribuição dos dados de Teste - Phoneme
Distribuição dos dados de Treino - PigAirwayPressure
Distribuição dos dados de Teste - PigAirwayPressure
Distribuição dos dados de Treino - PigArtPressure
Distribuição dos dados de Teste - PigArtPressure
Distribuição dos dados de Treino - PigCVP
Distribuição dos dados de Teste - PigCVP
Distribuição dos dados de Treino - Plane
Distribuição dos dados de Teste - Plane
Distribuição dos dados de Treino - PowerCons
Distribuição dos dados de Teste - PowerCons
Quantidade
Classes

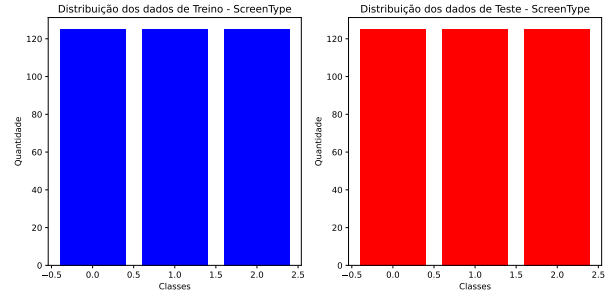

Distribuição dos dados de Treino - ProximalPhalanxOutlineCorrect
Distribuição dos dados de Teste - ProximalPhalanxOutlineCo
Quantidade
Classes

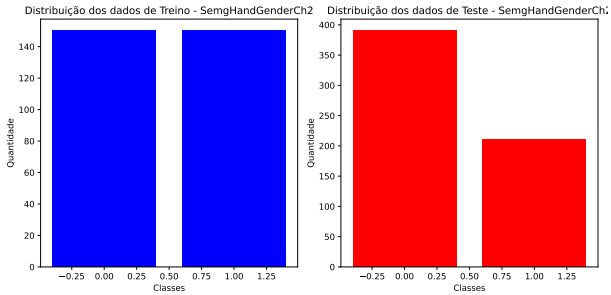

Distribuição dos dados de Treino - ProximalPhalanxOutlineAg
Distribuição dos dados de Teste - ProximalPhalanxOutlineAge
Quantidade
Classes

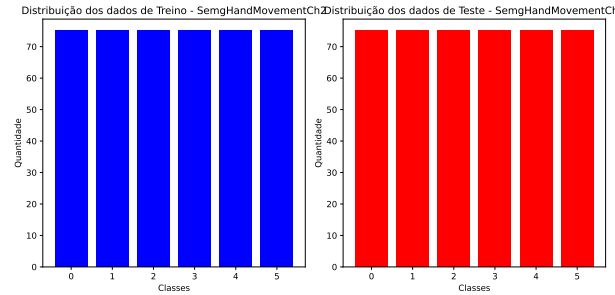

Distribuição dos dados de Treino - ProximalPhalanxTW
Distribuição dos dados de Teste - ProximalPhalanxTW
Quantidade
Classes

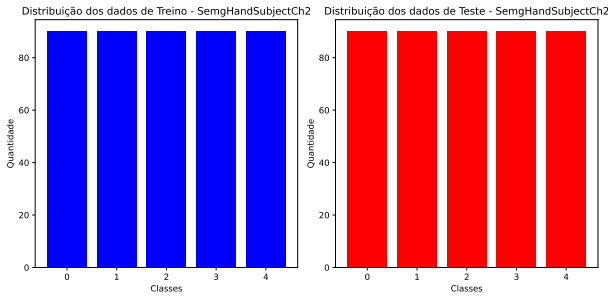

Distribuição dos dados de Treino - RefrigerationDevices
Distribuição dos dados de Teste - RefrigerationDevices
Quantidade
Classes

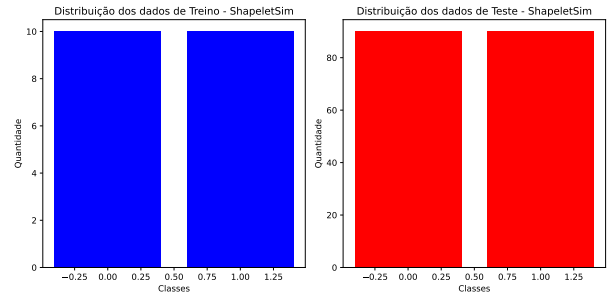

Distribuição dos dados de Treino - Rock
Distribuição dos dados de Teste - Rock
Quantidade
Classes

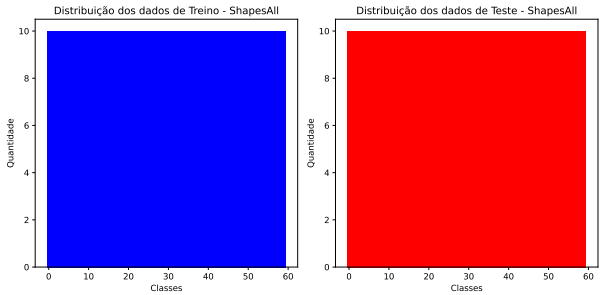

Distribuição dos dados de Treino - ScreenType
Distribuição dos dados de Teste - ScreenType
Quantidade
Classes

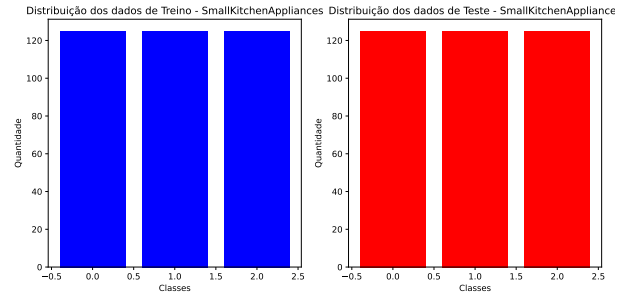

Distribuição dos dados de Treino - SemgHandGenderCh2
Distribuição dos dados de Teste - SemgHandGenderCh2
Quantidade
Classes

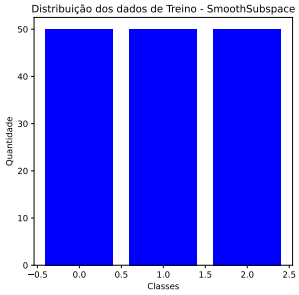

Distribuição dos dados de Treino - SemgHandMovementCh2
Distribuição dos dados de Teste - SemgHandMovementCh
Quantidade
Classes

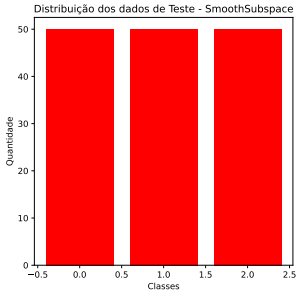

Distribuição dos dados de Treino - SemgHandSubjectCh2
Distribuição dos dados de Teste - SemgHandSubjectCh2
Quantidade
Classes

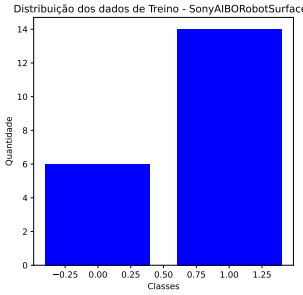

Distribuição dos dados de Treino - ShapeletSim
Distribuição dos dados de Teste - ShapeletSim
Quantidade
Classes

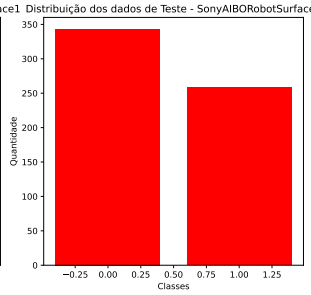

Distribuição dos dados de Treino - ShapesAll
Distribuição dos dados de Teste - ShapesAll
Quantidade
Classes

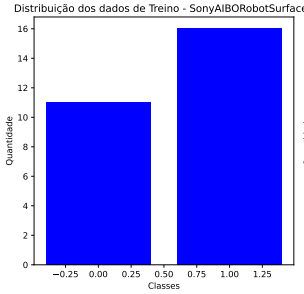

Distribuição dos dados de Treino - SmallKitchenAppliances
Distribuição dos dados de Teste - SmallKitchenAppliance
Quantidade
Classes

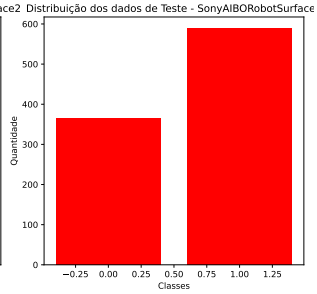

Distribuição dos dados de Treino - SmoothSubspace
Quantidade
Classes

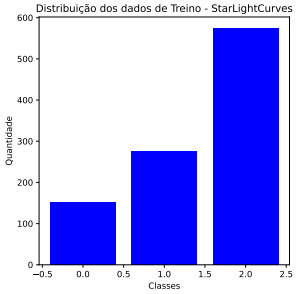

Distribuição dos dados de Teste - SmoothSubspace
Quantidade
Classes

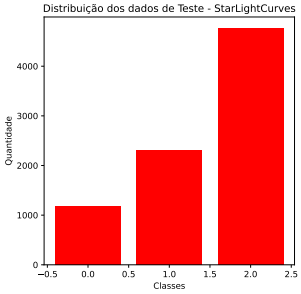

Distribuição dos dados de Treino - SonyAIBORobotSurface1
Quantidade
Classes

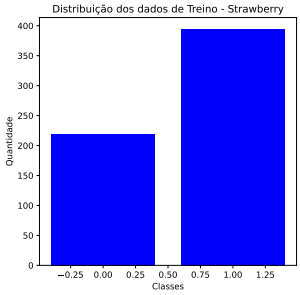

Distribuição dos dados de Teste - SonyAIBORobotSurface
Quantidade
Classes

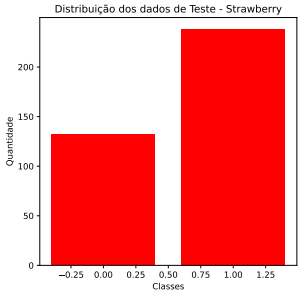

Distribuição dos dados de Treino - SonyAIBORobotSurface2
Quantidade
Classes

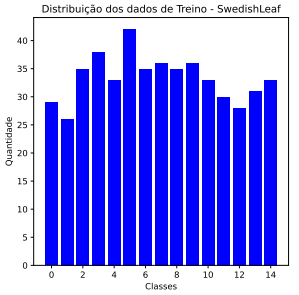

Distribuição dos dados de Teste - SonyAIBORobotSurface
Quantidade
Classes

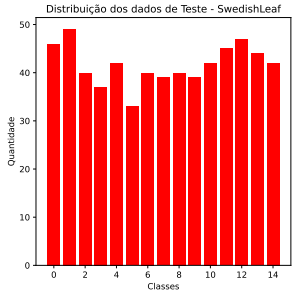

Distribuição dos dados de Treino - StarLightCurves
Quantidade
Classes

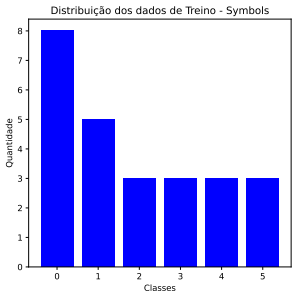

Distribuição dos dados de Teste - StarLightCurves
Quantidade
Classes

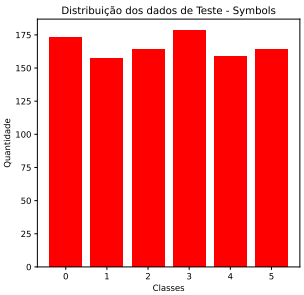

Distribuição dos dados de Treino - Strawberry
Quantidade
Classes

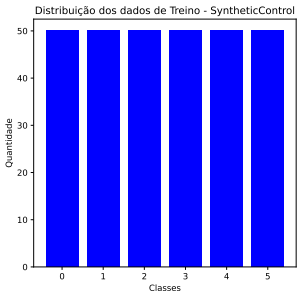

Distribuição dos dados de Teste - Strawberry
Quantidade
Classes

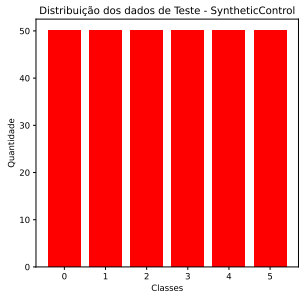

Distribuição dos dados de Treino - SwedishLeaf
Quantidade
Classes

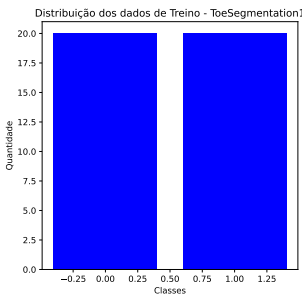

Distribuição dos dados de Teste - SwedishLeaf
Quantidade
Classes

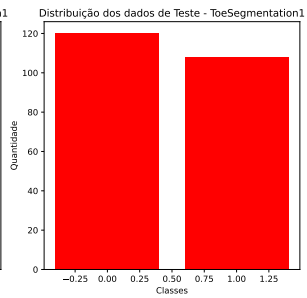

Distribuição dos dados de Treino - Symbols
Quantidade
Classes

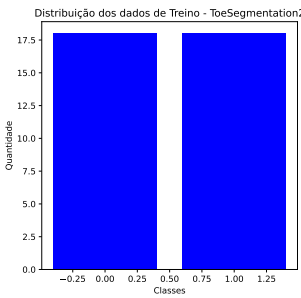

Distribuição dos dados de Teste - Symbols
Quantidade
Classes

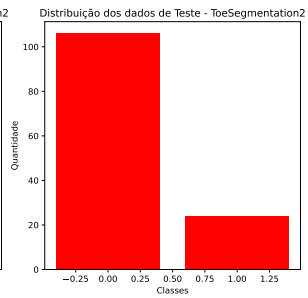

Distribuição dos dados de Treino - SyntheticControl
Quantidade
Classes

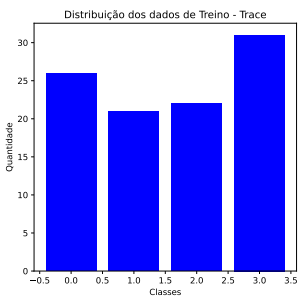

Distribuição dos dados de Teste - SyntheticControl
Quantidade
Classes

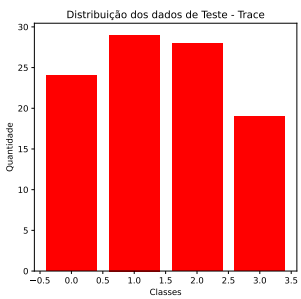

Distribuição dos dados de Treino - ToeSegmentation1
Quantidade
Classes

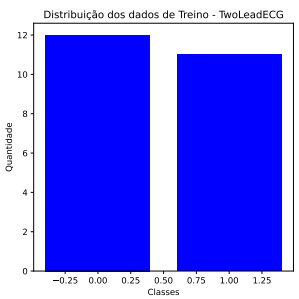

Distribuição dos dados de Teste - ToeSegmentation1
Quantidade
Classes

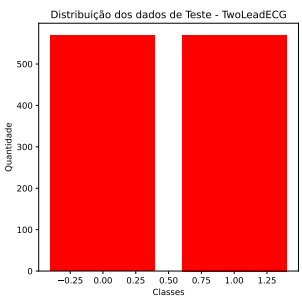

Distribuição dos dados de Treino - ToeSegmentation2
Quantidade
Classes

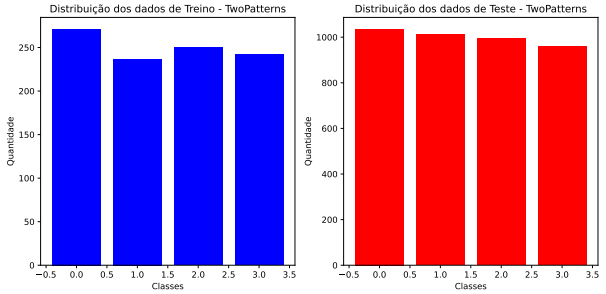

Distribuição dos dados de Teste - ToeSegmentation2
Quantidade
Classes


Distribuição dos dados de Treino - Trace
Quantidade
Classes

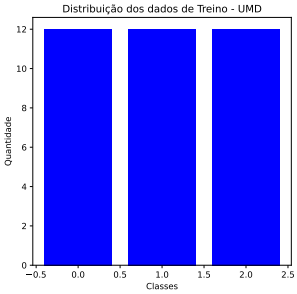

Distribuição dos dados de Teste - Trace
Quantidade
Classes

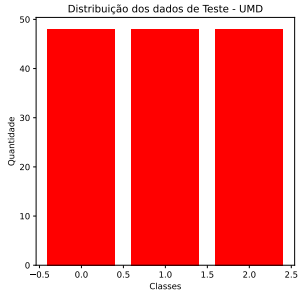

Distribuição dos dados de Treino - TwoLeadECG
Quantidade
Classes

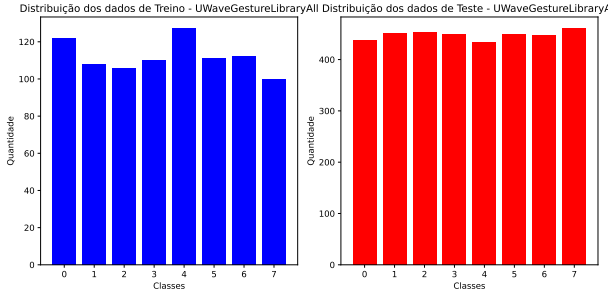

Distribuição dos dados de Teste - TwoLeadECG
Quantidade
Classes


Distribuição dos dados de Treino - TwoPatterns
Quantidade
Classes

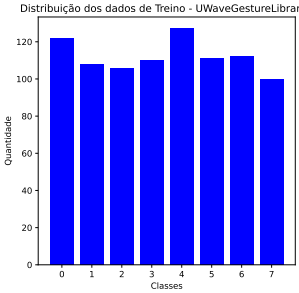
Distribuição dos dados de Teste - TwoPatterns
Quantidade
Classes

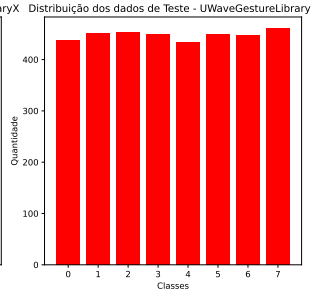

Distribuição dos dados de Treino - UMD
Quantidade
Classes

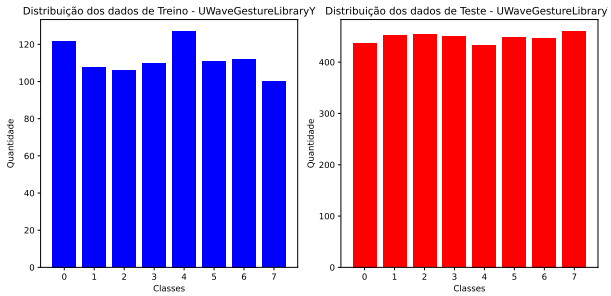

Distribuição dos dados de Teste - UMD
Quantidade
Classes


Distribuição dos dados de Treino - UWaveGestureLibraryAll
Quantidade
Classes

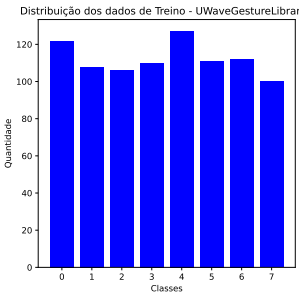
Distribuição dos dados de Teste - UWaveGestureLibraryA
Quantidade
Classes

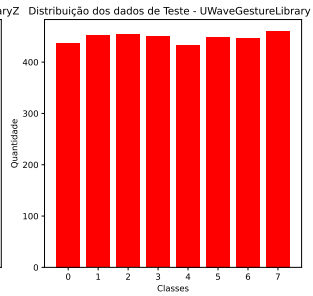

Distribuição dos dados de Treino - UWaveGestureLibraryX
Quantidade
Classes

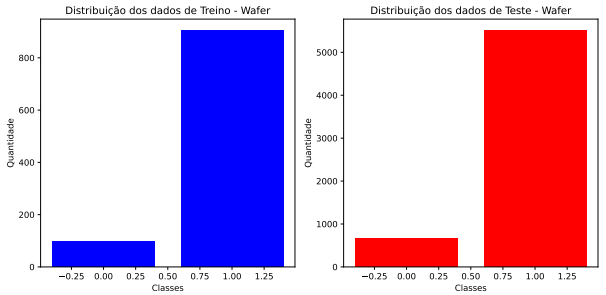

Distribuição dos dados de Teste - UWaveGestureLibrary
Quantidade
Classes


Distribuição dos dados de Treino - UWaveGestureLibraryY
Quantidade
Classes

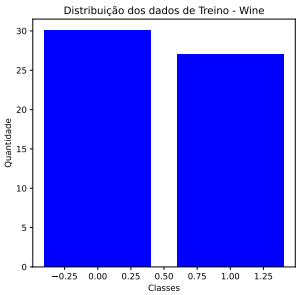
Distribuição dos dados de Teste - UWaveGestureLibrary
Quantidade
Classes

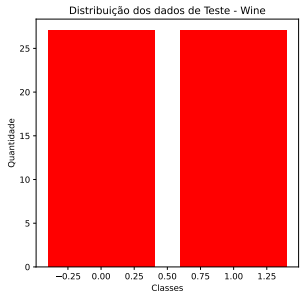

Distribuição dos dados de Treino - UWaveGestureLibraryZ
Quantidade
Classes

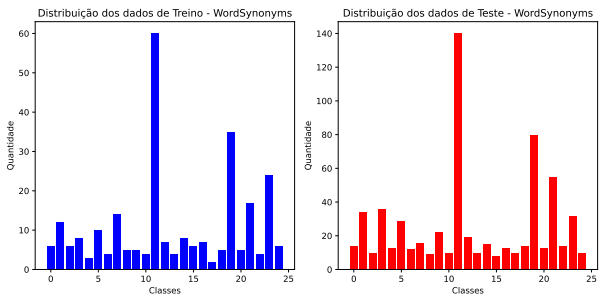

Distribuição dos dados de Teste - UWaveGestureLibrary
Quantidade
Classes


Distribuição dos dados de Treino - Wafer
Quantidade
Classes

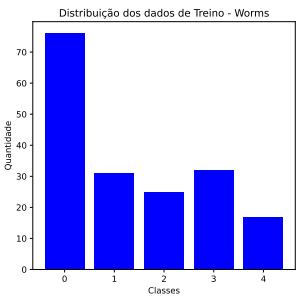
Distribuição dos dados de Teste - Wafer
Quantidade
Classes

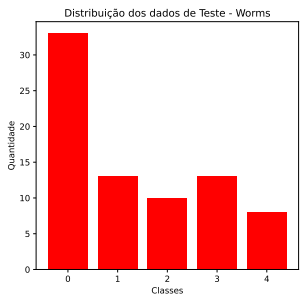

Distribuição dos dados de Treino - Wine
Quantidade
Classes

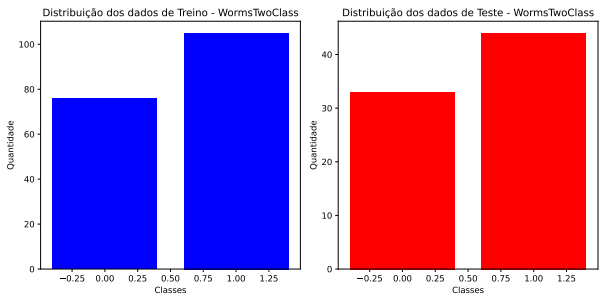

Distribuição dos dados de Teste - Wine
Quantidade
Classes


Distribuição dos dados de Treino - WordSynonyms
Quantidade
Classes

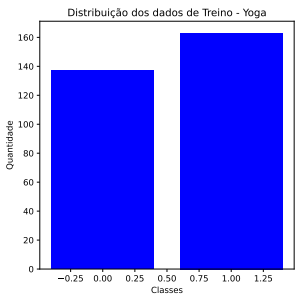
Distribuição dos dados de Teste - WordSynonyms
Quantidade
Classes

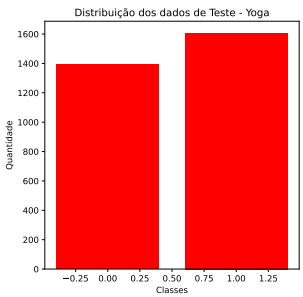

Distribuição dos dados de Treino - Worms
Quantidade
Classes


Distribuição dos dados de Teste - Worms
Quantidade
Classes


Distribuição dos dados de Treino - WormsTwoClass
Quantidade
Classes

Distribuição dos dados de Teste - WormsTwoClass
Quantidade
Classes


Distribuição dos dados de Treino - Yoga
Quantidade
Classes


Distribuição dos dados de Teste - Yoga
Quantidade
Classes

A.2 Comparação dos classificadores desenvolvidos em relação aos encontrados na literatura

A Tabela 16 apresenta os resultados de acurácia dos classificadores encontrados na literatura, que usam a geração de características como método para aumentar a precisão do classificador em relação aos classificadores desenvolvidos neste trabalho.

Conjuntos	Catch22	Fresh PRINCE	TSFresh	MC WL	Ensemble	MC NL	MC TG
ACSF1	0.850000	0.890000	0.870000	0.830000	0.840000	0.880000	0.670000
Adiac	0.716113	0.841432	0.810742	0.813299	0.762148	0.726343	0.734015
ArrowHead	0.725714	0.628571	0.691429	0.742857	0.708571	0.720000	0.822857
BME	0.913333	1.000000	1.000000	0.960000	0.733333	0.733333	0.833333
Beef	0.600000	0.800000	0.800000	0.866667	0.950000	0.950000	0.750000
BeetleFly	0.750000	0.900000	0.500000	0.950000	0.900000	0.900000	0.600000
BirdChicken	0.900000	1.000000	0.900000	0.750000	1.000000	0.680000	0.933333
CBF	0.966667	0.994444	0.995556	0.994444	0.975556	0.700000	0.800000
Car	0.766667	0.766667	0.833333	0.816667	0.800000	0.650000	0.912222
Chinatown	0.915452	0.982507	0.976676	0.982507	0.979592	0.862974	0.962099
ChlorineConcentration	0.601302	0.774740	0.459375	0.740365	0.741406	0.636458	0.890885
CinCECGTorso	0.817391	0.966667	0.984783	0.844203	0.950000	0.729710	0.734783
Coffee	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
Computers	0.716000	0.768000	0.760000	0.612000	0.764000	0.808000	0.544000
CricketX	0.582051	0.710256	0.697436	0.674359	0.705128	0.546154	0.641026
CricketY	0.515385	0.689744	0.687179	0.687179	0.741026	0.497436	0.625641
CricketZ	0.643590	0.753846	0.730769	0.702564	0.738462	0.556410	0.589744
Crop	0.691607	0.770417	0.756607	0.754881	0.772798	0.672440	0.754345
DiatomSizeReduction	0.944444	0.852941	0.300654	0.960784	0.957516	0.921569	0.960784
DistalPhalanxOutlineAgeGroup	0.705036	0.755396	0.733813	0.762590	0.755396	0.733813	0.712230
DistalPhalanxOutlineCorrect	0.797101	0.789855	0.764493	0.786232	0.775362	0.742754	0.681159
DistalPhalanxTW	0.661871	0.690647	0.640288	0.676259	0.690647	0.604317	0.640288
ECG200	0.840000	0.880000	0.850000	0.900000	0.748201	0.741007	0.748201
ECG5000	0.939333	0.944222	0.942000	0.940444	0.860000	0.790000	0.890000
ECGFiveDays	0.772358	1.000000	0.497096	1.000000	0.940889	0.925778	0.939333
EOGHorizontalSignal	0.552486	0.591160	0.566298	0.516575	0.979094	0.759582	0.986063
EOGVerticalSignal	0.505525	0.477901	0.491713	0.480663	0.734405	0.696667	0.617819
Earthquakes	0.741007	0.769784	0.755396	0.748201	0.530387	0.447514	0.411602
ElectricDevices	0.727402	0.769161	0.766308	0.671768	0.513812	0.372928	0.400552
EthanolLevel	0.350000	0.410000	0.382000	0.492000	0.478000	0.298000	0.780000
FaceAll	0.763905	0.781657	0.878698	0.794083	0.798817	0.627811	0.859763
FaceFour	0.636364	0.965909	0.579545	0.829545	0.977273	0.659091	0.863636
FacesUCR	0.707317	0.897561	0.871220	0.855122	0.895610	0.620488	0.819512
FiftyWords	0.595604	0.738462	0.685714	0.740659	0.756044	0.465934	0.674725
Fish	0.794286	0.914286	0.920000	0.868571	0.862857	0.697143	0.868571
FordA	0.915909	1.000000	1.000000	0.904545	0.941667	0.793939	0.834848
FordB	0.733333	0.795062	0.830864	0.781481	0.787654	0.629630	0.690123
FreezerRegularTrain	0.998596	0.989474	0.983158	0.954035	0.989825	0.970877	0.985965
FreezerSmallTrain	0.958596	0.983860	0.923158	0.801404	0.864561	0.847368	0.700702
GunPoint	0.966667	0.940000	0.953333	0.946667	0.960000	0.933333	0.913333
GunPointAgeSpan	0.974684	0.981013	0.984177	0.996835	0.990506	0.984177	0.876582
GunPointMaleVersusFemale	0.993671	0.984177	0.996835	0.996835	1.000000	0.996835	0.993671
GunPointOldVersusYoung	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
Ham	0.600000	0.742857	0.647619	0.695238	0.695238	0.552381	0.657143
HandOutlines	0.859459	0.905405	0.902703	0.905405	0.913514	0.781081	0.916216
Haptics	0.480519	0.503247	0.470779	0.454545	0.480519	0.438312	0.461039
Herring	0.531250	0.656250	0.593750	0.656250	0.640625	0.578125	0.593750
HouseTwenty	0.957983	0.932773	0.924370	0.865546	0.899160	0.873950	0.756303

InlineSkate	0.425455	0.512727	0.356364	0.358182	0.429091	0.396364	0.252727
InsectEPGRegularTrain	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
InsectEPGSmallTrain	0.987952	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
InsectWingbeatSound	0.567172	0.657071	0.658586	0.663636	0.665657	0.258081	0.636364
ItalyPowerDemand	0.896016	0.897959	0.962099	0.952381	0.968902	0.866861	0.970845
LargeKitchenAppliances	0.845333	0.880000	0.816000	0.613333	0.624000	0.773333	0.496000
Lightning2	0.688525	0.754098	0.540984	0.836066	0.786885	0.754098	0.721311
Lightning7	0.698630	0.712329	0.657534	0.821918	0.753425	0.561644	0.616438
Mallat	0.926652	0.963753	0.927505	0.964179	0.959062	0.793603	0.955224
Meat	0.933333	0.916667	0.950000	0.966667	0.933333	0.900000	0.966667
MedicalImages	0.748684	0.814474	0.780263	0.734211	0.809211	0.707895	0.753947
MiddlePhalanxOutlineAgeGroup	0.603896	0.584416	0.584416	0.610390	0.564935	0.603896	0.564935
MiddlePhalanxOutlineCorrect	0.766323	0.869416	0.807560	0.824742	0.821306	0.783505	0.601375
MiddlePhalanxTW	0.525974	0.558442	0.597403	0.564935	0.571429	0.558442	0.571429
MixedShapesRegularTrain	0.915464	0.950515	0.941856	0.920825	0.936082	0.849485	0.914227
MixedShapesSmallTrain	0.859381	0.898557	0.905155	0.852784	0.876701	0.805361	0.854021
MoteStrain	0.865815	0.911342	0.539137	0.885783	0.909744	0.798722	0.539137
NonInvasiveFetalECGThorax1	0.850891	0.929771	0.911450	0.886514	0.913995	0.786260	0.947583
NonInvasiveFetalECGThorax2	0.872774	0.936387	0.935369	0.919593	0.929771	0.838677	0.950636
OSULeaf	0.694215	0.867769	0.867769	0.574380	0.900000	0.833333	0.833333
OliveOil	0.733333	0.866667	0.866667	0.900000	0.628099	0.714876	0.561983
PhalangesOutlinesCorrect	0.785548	0.837995	0.812354	0.818182	0.835664	0.776224	0.667832
Phoneme	0.300633	0.345464	0.366561	0.217300	0.268987	0.212553	0.112342
PigAirwayPressure	0.533654	0.423077	0.408654	0.408654	0.216346	0.653846	0.038462
PigArtPressure	0.961538	0.913462	0.966346	0.931346	0.432692	0.855769	0.182692
PigCVP	0.692308	0.798077	0.019231	0.230769	0.331731	0.615385	0.153846
Plane	0.990476	1.000000	1.000000	0.990476	1.000000	1.000000	0.971429
PowerCons	0.933333	0.972222	0.966667	1.000000	1.000000	0.938889	0.988889
ProximalPhalanxOutlineAgeGroup	0.843902	0.834146	0.843902	0.858537	0.858537	0.829268	0.839024
ProximalPhalanxOutlineCorrect	0.835052	0.896907	0.869416	0.879725	0.879725	0.852234	0.824742
ProximalPhalanxTW	0.760976	0.775610	0.780488	0.804878	0.814634	0.785366	0.800000
RefrigerationDevices	0.522667	0.600000	0.560000	0.560000	0.570667	0.536000	0.464000
Rock	0.760000	0.760000	0.700000	0.840000	0.800000	0.500000	0.820000
ScreenType	0.509333	0.584000	0.605333	0.392000	0.480000	0.514667	0.368000
SemgHandGenderCh2	0.905000	0.958333	0.933333	0.895000	0.966667	0.905000	0.928333
SemgHandMovementCh2	0.780000	0.893333	0.835556	0.680000	0.857778	0.704444	0.664444
SemgHandSubjectCh2	0.860000	0.946667	0.920000	0.877778	0.924444	0.804444	0.844444
ShapeletSim	0.988889	1.000000	0.500000	0.683333	0.927778	0.916667	0.494444
ShapesAll	0.808333	0.861667	0.858333	0.815000	0.838333	0.756667	0.788333
SmallKitchenAppliances	0.808000	0.816000	0.813333	0.749333	0.824000	0.797333	0.437333
SmoothSubspace	0.840000	0.973333	0.966667	0.980000	0.993333	0.946667	0.866667
SonyAIBORobotSurface1	0.856905	0.918469	0.916805	0.797005	0.863561	0.891847	0.806988
SonyAIBORobotSurface2	0.926548	0.916055	0.902413	0.881427	0.838405	0.887723	0.810073
StarLightCurves	0.969524	0.979723	0.979480	0.979480	0.973409	0.963696	0.947790
Strawberry	0.927027	0.959459	0.948649	0.970270	0.970270	0.918919	0.972973
SwedishLeaf	0.902400	0.939200	0.945600	0.913600	0.939200	0.892800	0.918400
Symbols	0.960804	0.972864	0.936683	0.878392	0.935678	0.833166	0.761809
SyntheticControl	0.963333	0.996667	1.000000	0.996667	0.993333	0.883333	0.973333
ToeSegmentation1	0.864035	0.815789	0.526316	0.723684	0.793860	0.767544	0.618421
ToeSegmentation2	0.792308	0.861538	0.838462	0.853846	0.884615	0.707692	0.776923
Trace	1.000000	1.000000	1.000000	0.980000	0.980000	1.000000	0.860000
TwoLeadECG	0.832309	0.991220	0.499561	0.751536	0.850746	0.817384	0.938543
TwoPatterns	0.848000	0.997000	0.995500	0.944500	0.997750	0.550750	0.972000
UMD	0.895833	0.986111	0.951389	0.979167	0.986111	0.972222	0.881944
UWaveGestureLibraryAll	0.828029	0.963707	0.951145	0.953936	0.814070	0.584310	0.959520
UWaveGestureLibraryX	0.754606	0.840313	0.803462	0.794249	0.731993	0.578727	0.782245
UWaveGestureLibraryY	0.695422	0.768286	0.725293	0.716639	0.757398	0.571468	0.689559
UWaveGestureLibraryZ	0.702401	0.785874	0.746790	0.784874	0.998053	0.593523	0.713568
Wafer	0.997404	1.000000	1.000000	0.999133	0.722222	0.990266	0.995782

Wine	0.388889	0.796296	0.500000	0.851852	0.658307	0.537037	0.629630
WordSynonyms	0.537618	0.653605	0.587774	0.620690	0.649351	0.402821	0.601881
Worms	0.727273	0.818182	0.766234	0.584416	0.714286	0.779221	0.571429
WormsTwoClass	0.857143	0.831169	0.610390	0.753247	0.847333	0.870130	0.610390
Yoga	0.784000	0.914667	0.914667	0.839667	0.847333	0.781000	0.838667

Tabela 16: Comparação dos algoritmos encontrados na literatura com os modelos propostos.

A.3 Teste das comparações individuais dos classificadores que compõem o modelo Seleção Dinâmica

A Tabela 17 apresenta os resultados de acurácia dos classificadores utilizados para compor o modelo de seleção dinâmica proposto neste trabalho. Os testes foram realizados para verificar se a proposta do modelo de seleção dinâmica seria eficiente com diferentes algoritmos baseados em árvores de decisão.

Conjunto	RandomForest	ExtraForest	TimeSeriesForest	Supervised TSF
ACSF1	0.780000	0.730000	0.82	0.810000
Adiac	0.703325	0.713555	0.7647058823529411	0.800512
ArrowHead	0.742857	0.754286	0.6971428571428572	0.674286
Beef	0.980000	0.993333	0.8	0.866667
BeetleFly	0.766667	0.766667	0.95	0.900000
BirdChicken	0.850000	0.900000	0.85	0.950000
BME	0.700000	0.700000	1.0	1.000000
Car	0.984444	0.975556	0.7666666666666667	0.833333
CBF	0.733333	0.733333	0.9755555555555555	0.987778
Chinatown	0.979592	0.985423	0.9766763848396501	0.979592
ChlorineConcentration	0.747396	0.760677	0.7427083333333333	0.766406
CinCECGTorso	0.854348	0.843478	0.9485507246376812	0.993478
Coffee	1.000000	1.000000	1.0	1.000000
Computers	0.628000	0.628000	0.748	0.736000
CricketX	0.643590	0.635897	0.7025641025641025	0.674359
CricketY	0.617949	0.638462	0.7384615384615385	0.689744
CricketZ	0.676923	0.658974	0.7256410256410256	0.697436
Crop	0.751726	0.754881	0.7724404761904762	0.757560
DiatomSizeReduction	0.762590	0.748201	0.9477124183006536	0.748201
DistalPhalanxOutlineAgeGroup	0.782609	0.786232	0.762589928057554	0.793478
DistalPhalanxOutlineCorrect	0.683453	0.697842	0.7789855072463768	0.669065
DistalPhalanxTW	0.850000	0.870000	0.697841726618705	0.748201
Earthquakes	0.940667	0.939333	0.7482014388489209	0.900000
ECG200	0.981417	0.987224	0.86	0.943778
ECG5000	0.458564	0.472376	0.9415555555555556	1.000000
ECGFiveDays	0.453039	0.425414	0.9779326364692218	0.697834
ElectricDevices	0.748201	0.748201	0.7316820127091168	0.563536
EOGHorizontalSignal	0.679549	0.671768	0.5303867403314917	0.535912
EOGVerticalSignal	0.486000	0.510000	0.5193370165745856	0.650000
EthanolLevel	0.875148	0.881065	0.49	0.834911
FaceAll	0.931818	0.920455	0.7964497041420119	1.000000
FaceFour	0.808780	0.813659	1.0	0.819512
FacesUCR	0.703297	0.716484	0.8985365853658537	0.727473
FiftyWords	0.788571	0.788571	0.7538461538461538	0.908571
Fish	0.881818	0.887879	0.8685714285714285	0.969697
FordA	0.724691	0.720988	0.9477272727272728	0.832099
FordB	0.951579	0.945263	0.7913580246913581	0.997544

FreezerRegularTrain	0.748070	0.737895	0.9901754385964913	0.996140
FreezerSmallTrain	0.986667	0.973333	0.8856140350877193	0.946667
GunPoint	0.946203	0.939873	0.9666666666666667	0.977848
GunPointAgeSpan	0.993671	0.987342	0.990506329113924	0.993671
GunPointMaleVersusFemale	0.961905	0.980952	1.0	0.990476
GunPointOldVersusYoung	0.723810	0.714286	1.0	0.714286
Ham	0.910811	0.905405	0.6952380952380952	0.918919
HandOutlines	0.464286	0.477273	0.9108108108108108	0.535714
Haptics	0.625000	0.656250	0.4805194805194805	0.625000
Herring	0.907563	0.865546	0.59375	0.941176
HouseTwenty	0.401818	0.407273	0.9243697478991597	0.587273
InlineSkate	1.000000	1.000000	0.4109090909090909	0.600000
InsectEPGRegularTrain	1.000000	1.000000	1.0	1.000000
InsectEPGSmallTrain	0.648990	0.645960	1.0	1.000000
InsectWingbeatSound	0.958212	0.961127	0.6671717171717172	0.663636
ItalyPowerDemand	0.656000	0.656000	0.9689018464528668	0.969874
LargeKitchenAppliances	0.754098	0.803279	0.616	0.840000
Lightning2	0.643836	0.616438	0.7868852459016393	0.786885
Lightning7	0.961194	0.960341	0.7534246575342466	0.712329
Mallat	0.933333	0.933333	0.95863539445629	0.955650
Meat	0.747368	0.756579	0.9333333333333333	0.966667
MedicalImages	0.577922	0.558442	0.8	0.817105
MiddlePhalanxOutlineAgeGroup	0.821306	0.804124	0.577922077922078	0.577922
MiddlePhalanxOutlineCorrect	0.551948	0.564935	0.8316151202749141	0.845361
MiddlePhalanxTW	0.919175	0.914639	0.5714285714285714	0.512987
MixedShapesRegularTrain	0.877526	0.872577	0.9327835051546391	0.939381
MixedShapesSmallTrain	0.877796	0.876198	0.8775257731958763	0.889072
MoteStrain	0.887532	0.889567	0.9137380191693291	0.878594
NonInvasiveFetalECGThorax1	0.914504	0.917048	0.9124681933842239	0.927226
NonInvasiveFetalECGThorax2	0.628099	0.615702	0.9323155216284987	0.935878
OliveOil	0.900000	0.900000	0.9333333333333333	0.933333
OSULeaf	0.822844	0.821678	0.6446280991735537	0.776860
PhalangesOutlinesCorrect	0.212553	0.217300	0.8298368298368298	0.846154
Phoneme	0.129808	0.134615	0.27531645569620256	0.256857
PigAirwayPressure	0.187500	0.163462	0.2403846153846154	0.341346
PigArtPressure	0.235577	0.230769	0.41346153846153844	0.889423
PigCVP	1.000000	1.000000	0.3605769230769231	0.697115
Plane	0.983333	0.983333	0.9904761904761905	1.000000
PowerCons	0.848780	0.853659	1.0	1.000000
ProximalPhalanxOutlineAgeGroup	0.869416	0.876289	0.8585365853658536	0.853659
ProximalPhalanxOutlineCorrect	0.800000	0.800000	0.8797250859106529	0.893471
ProximalPhalanxTW	0.581333	0.584000	0.8146341463414634	0.809756
RefrigerationDevices	0.780000	0.720000	0.5573333333333333	0.565333
Rock	0.453333	0.477333	0.8	0.840000
ScreenType	0.843333	0.810000	0.496	0.525333
SemgHandGenderCh2	0.555556	0.557778	0.9766666666666667	0.973333
SemgHandMovementCh2	0.724444	0.728889	0.8644444444444445	0.826667
SemgHandSubjectCh2	0.711111	0.577778	0.9311111111111111	0.944444
ShapeletSim	0.795000	0.785000	0.95	0.977778
ShapesAll	0.645333	0.661333	0.8433333333333334	0.871667
SmallKitchenAppliances	0.953333	0.953333	0.816	0.824000
SmoothSubspace	0.871880	0.838602	0.9866666666666667	0.973333
SonyAIBORobotSurface1	0.885624	0.858342	0.8685524126455907	0.898502
SonyAIBORobotSurface2	0.965396	0.965881	0.8300104931794333	0.884575
StarLightCurves	0.962162	0.962162	0.9720738222438077	0.979359
Strawberry	0.910400	0.913600	0.972972972972973	0.967568
SwedishLeaf	0.843216	0.848241	0.944	0.945600
Symbols	0.983333	0.986667	0.9437185929648241	0.975879
SyntheticControl	0.710526	0.701754	0.99	0.986667

ToeSegmentation1	0.730769	0.746154	0.7982456140350878	0.890351
ToeSegmentation2	0.950000	0.910000	0.8769230769230769	0.853846
Trace	0.783143	0.775241	0.99	1.000000
TwoLeadECG	0.921500	0.925000	0.8481123792800702	0.978929
TwoPatterns	0.923611	0.895833	0.99775	0.997250
UMD	0.942769	0.943886	0.9930555555555556	0.979167
UWaveGestureLibraryAll	0.772194	0.776382	0.9609156895589056	0.954495
UWaveGestureLibraryX	0.707147	0.707984	0.8115577889447236	0.795924
UWaveGestureLibraryY	0.721385	0.727527	0.7272473478503629	0.741485
UWaveGestureLibraryZ	0.995620	0.994971	0.7537688442211056	0.761027
Wafer	0.555556	0.685185	0.9975665152498377	0.997242
Wine	0.628527	0.637931	0.7222222222222222	0.648148
WordSynonyms	0.727273	0.727273	0.658307210031348	0.576803
Worms	0.779221	0.805195	0.6363636363636364	0.818182
WormsTwoClass	0.825000	0.829667	0.7142857142857143	0.844156
Yoga	0.847667	0.846667	0.8486666666666667	0.855333

Tabela 17: Comparação dos algoritmos baseados em árvore utilizados para compor o modelo de seleção dinâmica.

A.4 Teste das comparações dos classificadores base utilizando *RidgeClassifierCV* como meta-classificador

A Tabela 18 apresenta os resultados da acurácia dos classificadores testados em 20 diferentes conjuntos de dados. Foram aplicados classificadores que utilizam métodos baseados em similaridade, vetores de suporte, métodos bayesianos e florestas aleatórias. Nos testes, foram avaliados o k-NN com k igual a 1 e 5, SVM, *Naive Bayes* (NB), florestas aleatórias (RF) e uma versão extra aleatória de florestas (ExRF). Majoritariamente, o classificador baseado em florestas extra aleatórias apresentou os melhores resultados, demonstrando uma predominância significativa em diversos datasets, como *SyntheticControl* e *CBF*. Os resultados serviram de base para a organização do treinamento do meta-classificador.

Dataset	MS-1NN	MS-5NN	MS-SVM	MS-NB	MS-RF	MS-ExRF
Adiac	0.616368	0.728	0.780051	0.624041	0.731458	0.813299
Beef	0.666667	0.600	0.800000	0.600000	0.766667	0.866667
Car	0.733333	0.683	0.833333	0.600000	0.683333	0.816667
CBF	0.882222	0.940	0.870000	0.891111	0.938889	0.994444
Coffee	1.000000	0.964	1.000000	0.892857	1.000000	1.000000
DiatomSizeReduction	0.934641	0.882	0.977124	0.869281	0.928105	0.960784
ECG200	0.870000	0.930	0.820000	0.780000	0.820000	0.900000
ECGFiveDays	0.853659	0.879	0.995354	0.816492	0.975610	1.000000
FaceFour	0.795455	0.750	0.840909	0.829545	0.875000	0.829545
GunPoint	0.913333	0.993	0.913333	0.806667	0.960000	0.946667
Lightning2	0.819672	0.786	0.672131	0.704918	0.737705	0.836066
Lightning7	0.602740	0.712	0.712329	0.712329	0.753425	0.821918
MedicalImages	0.698684	0.689	0.644737	0.603947	0.757895	0.734211
MotesStrain	0.849042	0.869	0.864217	0.835463	0.879393	0.885783
OliveOil	0.866667	0.900	0.833333	0.900000	0.933333	0.900000
SonyAIBORobotSurface1	0.710483	0.812	0.730449	0.925125	0.818636	0.797005
SonyAIBORobotSurface2	0.873033	0.815	0.873033	0.792235	0.825813	0.881427
SyntheticControl	0.963333	1.000	0.970000	0.983333	0.986667	0.983333
Trace	0.790000	0.810	0.970000	0.870000	0.930000	0.980000
TwoPatterns	0.931500	0.989	0.919000	0.478250	0.900500	0.944500

Tabela 18: Resultados das acurácias do meta-classificador RidgeClassifierCV.

A.5 Teste das comparações dos métodos de construção do modelo de seleção dinâmica

A Tabela 19 apresenta os resultados de acurácia das técnicas empregadas na criação do modelo de seleção dinâmica. Foram testadas técnicas de *Voting* e *Ranking* em comparação com a técnica desenvolvida. Os resultados demostraram que a técnica de escolha dinâmica obteve resultados satisfatórios.

Conjunto	Escolha dinâmica	Voting	Ranking
Adiac	0.7416879795396419	0.7216879795396419	0.7442455242966752
Beef	0.7	0.6666666666666666	0.7
Car	0.7666666666666667	0.7333333333333333	0.7666666666666667
CBF	0.8633333333333333	0.8822222222222222	0.8644444444444445
Coffee	1.0	1.0	1.0
DiatomSizeReduction	0.9313725490196079	0.934640522875817	0.9313725490196079
ECG200	0.88	0.87	0.87
ECGFiveDays	0.9721254355400697	0.8536585365853658	0.9721254355400697
FaceFour	0.7727272727272727	0.7954545454545454	0.7840909090909091
GunPoint	0.98	0.9133333333333333	0.98
Lightning2	0.7704918032786885	0.6885245901639344	0.7704918032786885
Lightning7	0.6438356164383562	0.6027397260273972	0.6438356164383562
MedicalImages	0.6881578947368421	0.6986842105263158	0.6855263157894737
MoteStrain	0.7292332268370607	0.853035143769968	0.7308306709265175
OliveOil	0.8	0.8666666666666667	0.8
SonyAIBORobotSurface1	0.7088186356073212	0.7104825291181365	0.7088186356073212
SonyAIBORobotSurface2	0.8478488982161595	0.8730325288562435	0.8499475341028332
SyntheticControl	0.92	0.9633333333333334	0.92
Trace	0.84	0.79	0.84
TwoPatterns	0.942	0.932	0.6585

Tabela 19: Comparação entre os resultados das estratégias empregadas na construção do modelo de seleção dinâmica.

A.6 Gráficos Boxplot

As figuras a seguir apresentam a distribuição e a simetria das predições obtidas nos testes utilizando 112 conjuntos de dados com diferentes classificadores e abordagens. As caixas dos gráficos representam os quartis, os bigodes indicam os limites máximos e mínimos, e os *outliers* são assinalados por símbolos em formato de bola.

A Figura 36 exibe o desempenho dos classificadores individuais, enquanto a Figura 37 apresenta as três técnicas de seleção utilizadas na composição do modelo de seleção dinâmica. Na Figura 38 é demonstrada a distribuição das acurácias dos classificadores extratores. Nota-se que o MS-ExRF tem o melhor desempenho médio, enquanto o MS-NB apresenta maior variabilidade e o pior desempenho médio. A próxima Figura 39 é possível constatar que o classificador SVM apresenta o melhor desempenho médio e a menor variabilidade, enquanto o classificador NB tem o pior desempenho médio e a maior variabilidade. No gráfico seguinte, é comparada a distribuição das acurácias dos classificadores testados neste trabalho contra um modelo protótipo. A Figura 40 demostra que o Modelo Protótipo (MP) destaca-se com o melhor desempenho médio, mas também com a maior variabilidade, enquanto o 1NN tem o pior desempenho médio. Por fim, a Figura 41 compara classificadores da literatura e o modelo protótipo utilizando diferentes algoritmos. Nessa comparação, ROCKET e Inception Time

apresentam os melhores desempenhos médios, enquanto o TimeBox exibe a maior variabilidade.

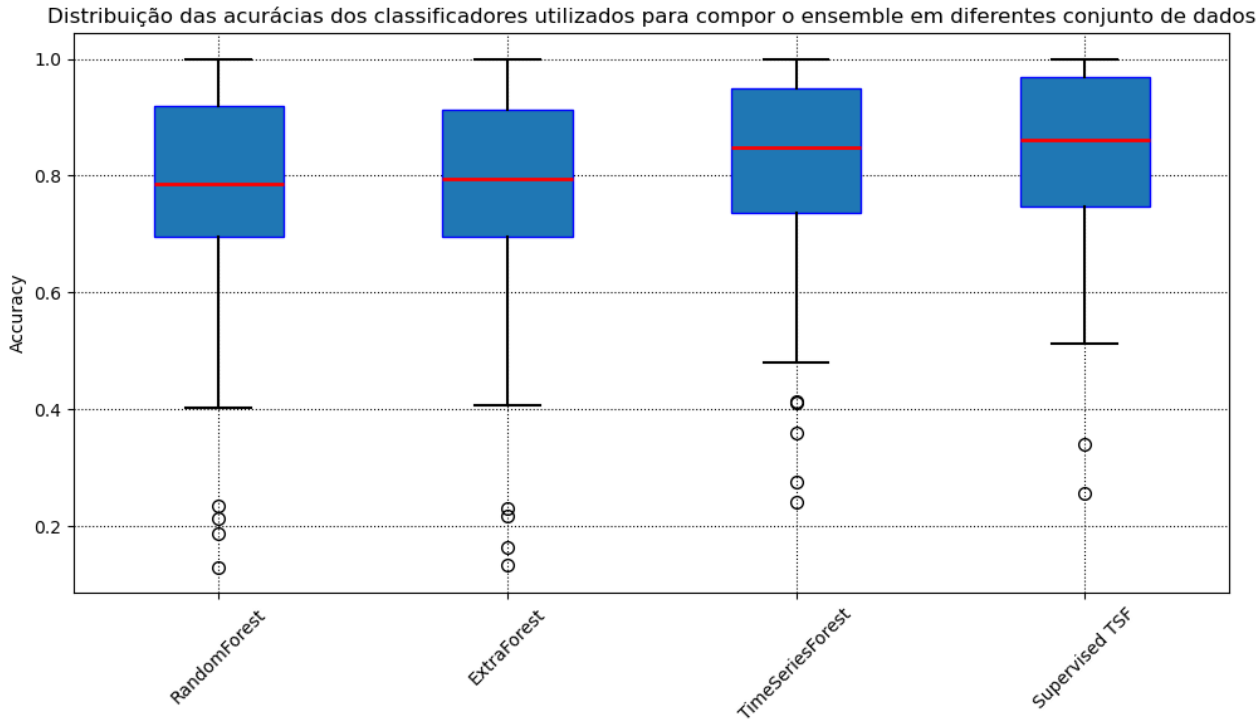


Figura 36: Gráfico Boxplot dos classificadores que compõem o modelo de seleção dinâmica individualmente

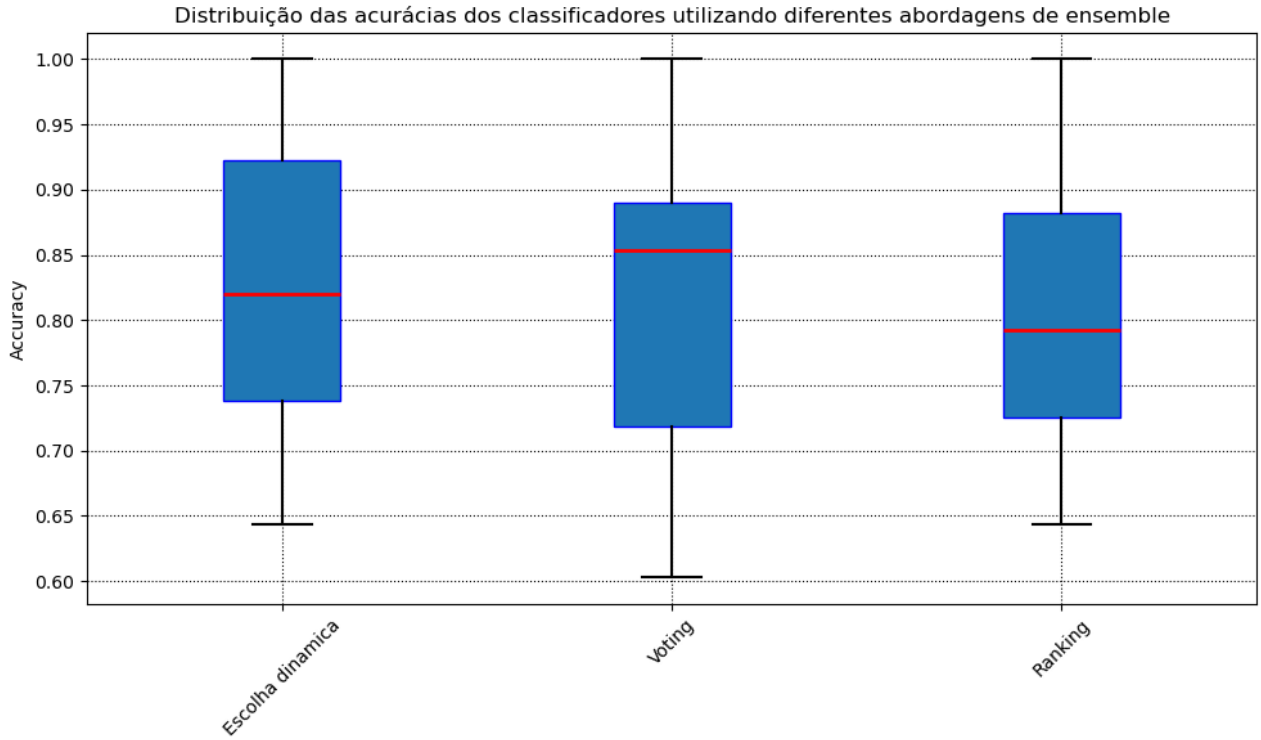


Figura 37: Gráfico Boxplot das técnicas de escolha utilizadas no modelo de seleção dinâmica

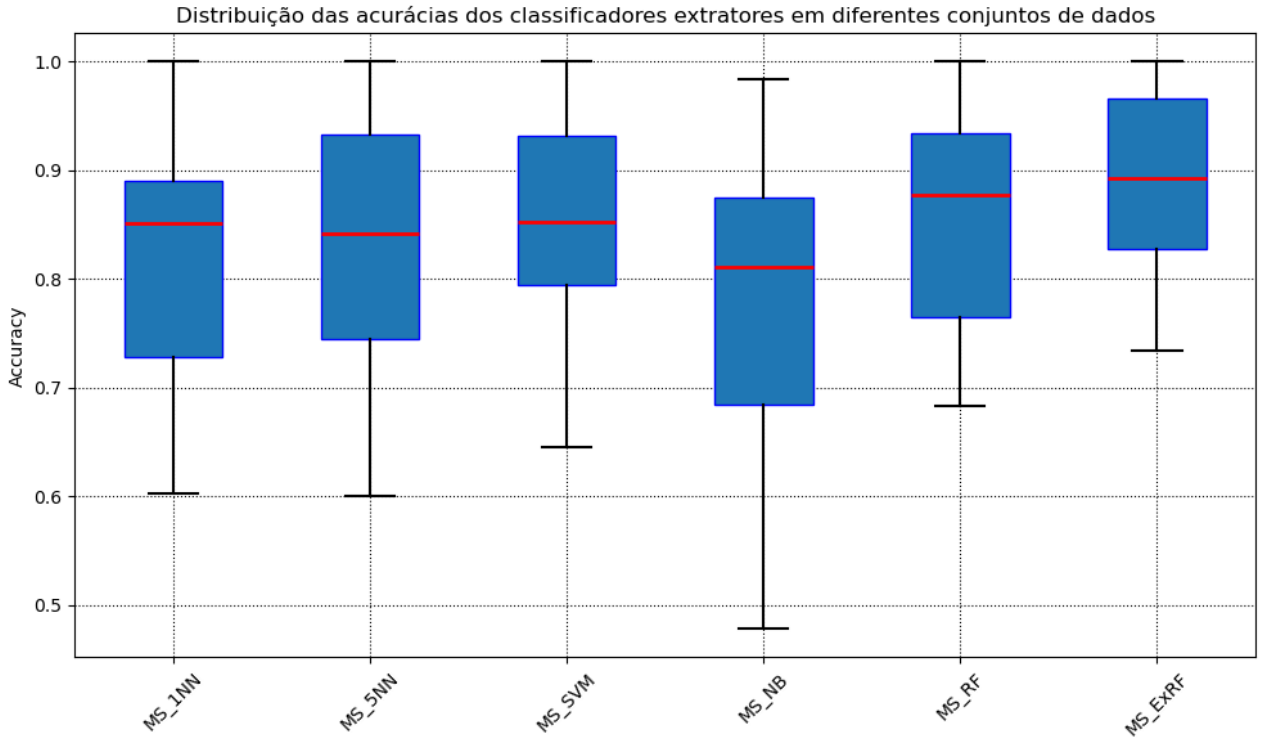


Figura 38: Gráfico Boxplot dos classificadores utilizados como extratores para o meta-classificador

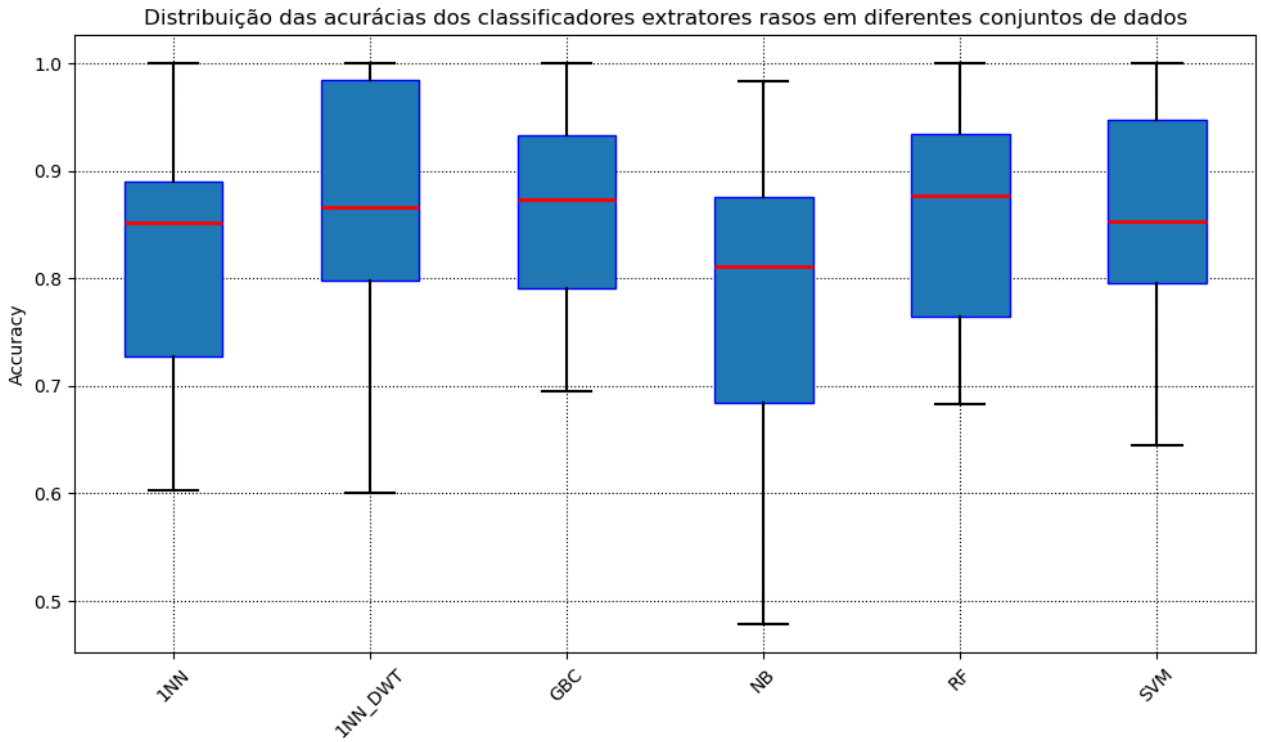


Figura 39: Gráfico Boxplot dos classificadores utilizados como extratores rasos para o meta-classificador

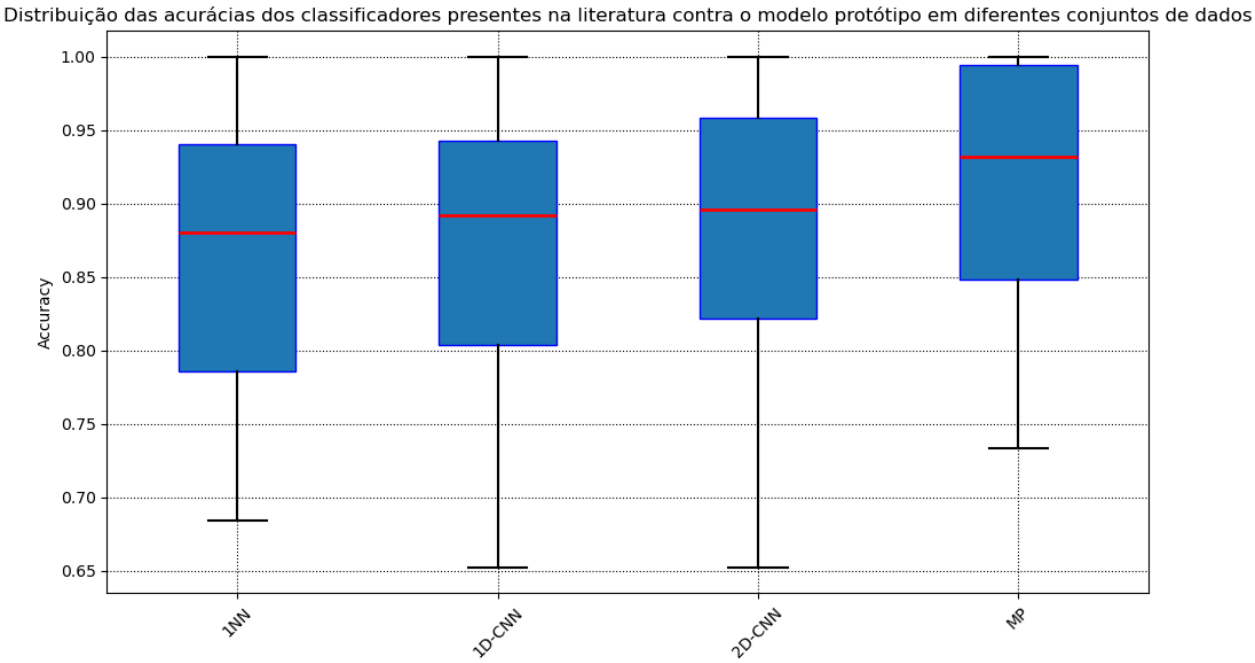


Figura 40: Gráfico Boxplot dos classificadores MP1NNCNN

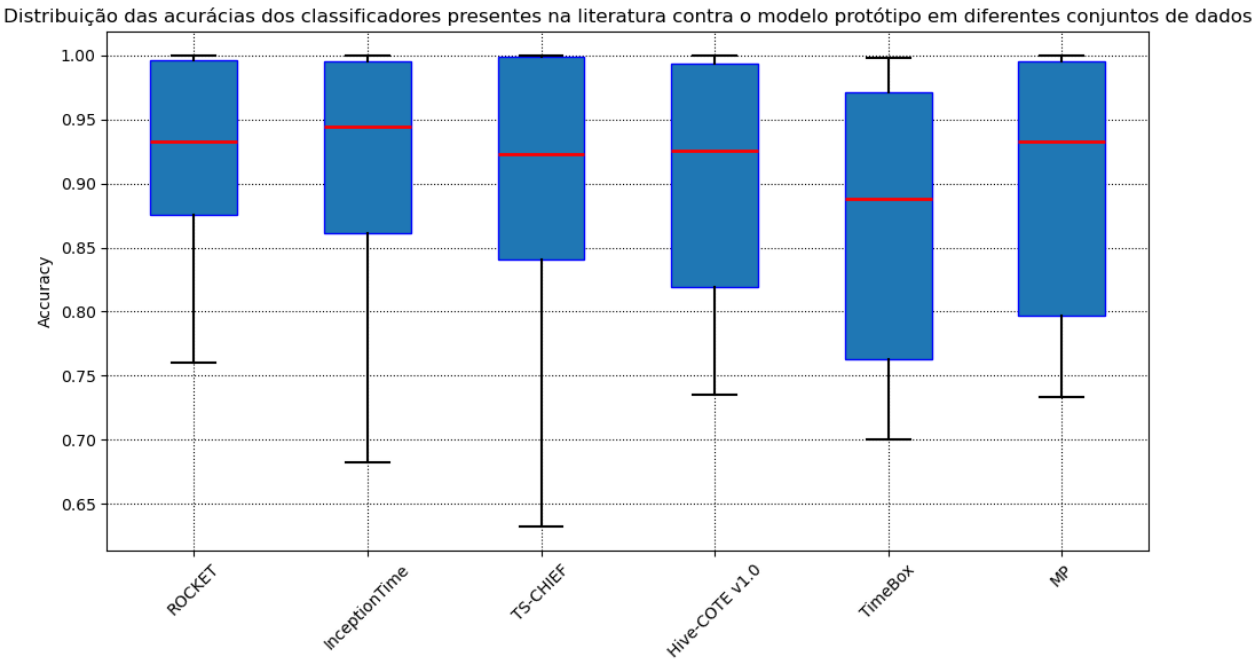


Figura 41: Gráfico Boxplot dos classificadores encontrados na literatura contra o modelo protótipo