Extração e Seleção de Características para a Classificação Eficiente de Séries Temporais

Márcio Antônio de Freitas Júnior

Universidade Federal de Uberlândia
Faculdade de Computação
Programa de Pós-Graduação em Ciência da Computação

Uberlândia
2021

Márcio Antônio de Freitas Júnior

Extração e Seleção de Características para a Classificação Eficiente de Séries Temporais

Dissertação de mestrado apresentada ao Programa de Pós-graduação da Faculdade de Computação da Universidade Federal de Uberlândia como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Área de concentração: Ciência da Computação

Orientador: Prof. Dr. Marcelo Keese Albertini

Uberlândia
2021

Ficha Catalográfica Online do Sistema de Bibliotecas da UFU
com dados informados pelo(a) próprio(a) autor(a).

F866 2022

Freitas Junior, Marcio Antonio de, 1997-
Extração e Seleção de Características para a Classificação Eficiente de Séries Temporais [recurso eletrônico] / Marcio Antonio de Freitas Junior. - 2022.

Orientador: Marcelo Keese Albertini.
Dissertação (Mestrado) - Universidade Federal de Uberlândia, Pós-graduação em Ciência da Computação.
Modo de acesso: Internet.
Disponível em: http://doi.org/10.14393/ufu.di.2022.53
Inclui bibliografia.
Inclui ilustrações.

1. Computação. I. Albertini, Marcelo Keese,1984-, (Orient.). II. Universidade Federal de Uberlândia. Pós-graduação em Ciência da Computação. III. Título.

CDU: 681.3

Bibliotecários responsáveis pela estrutura de acordo com o AACR2:

Gizele Cristine Nunes do Couto - CRB6/2091

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
Coordenação do Programa de Pós-Graduação em Ciência da Computação
Av. João Naves de Ávila, nº 2121, Bloco 1A, Sala 243 - Bairro Santa Mônica, Uberlândia-MG, CEP 38400-902
Telefone: (34) 3239-4470 - www.ppgco.facom.ufu.br - cpgfacom@ufu.br

ATA DE DEFESA - PÓS-GRADUAÇÃO

Programa de Pós-Graduação em:	Ciência da Computação				
Defesa de:	Mestrado Acadêmico, 2/2022, PPGCO				
Data:	24 de janeiro de 2022	Hora de início:	14:00	Hora de encerramento:	16:36
Matrícula do Discente:	11922CCP010				
Nome do Discente	Márcio Antônio de Freitas Júnior				
Título do Trabalho:	Extração de Características baseadas em Dicionário para Classificação Eficiente de Séries Temporais				
Área de concentração:	Ciência da Computação				
Linha de pesquisa:	Ciência de Dados				
Projeto de Pesquisa de vinculação:	-				

Reuniu-se, por videoconferência, a Banca Examinadora, designada pelo Colegiado do Programa de Pós-graduação em Ciência da Computação, assim composta: Professores Doutores: André Ricardo Backes - FACOM/UFU; Ricardo Araújo Rios - DCC/UFBA e Marcelo Keese Albertini - FACOM/UFU, orientador da candidato.

Os examinadores participaram desde as seguintes localidades: Ricardo Araújo Rios - Salvador/BA; André Ricardo Backes e Marcelo Keese Albertini - Uberlândia/MG. O discente participou da cidade de Uberlândia/MG.

Iniciando os trabalhos o presidente da mesa, Prof. Dr. Marcelo Keese Albertini, apresentou a Comissão Examinadora e o candidato, agradeceu a presença do público, e concedeu ao Discente a palavra para a exposição do seu trabalho. A duração da apresentação do Discente e o tempo de arguição e resposta foram conforme as normas do Programa.

A seguir o senhor presidente concedeu a palavra, pela ordem sucessivamente, aos examinadores, que passaram a arguir o candidato. Ultimada a arguição, que se desenvolveu dentro dos termos regimentais, a Banca, em sessão secreta, atribuiu o resultado final, considerando o candidato:

Aprovado.

Esta defesa faz parte dos requisitos necessários à obtenção do título de Mestre.

O competente diploma será expedido após cumprimento dos demais requisitos, conforme as normas do Programa, a legislação pertinente e a regulamentação interna da UFU.

Nada mais havendo a tratar foram encerrados os trabalhos. Foi lavrada a presente ata que após lida e achada conforme foi assinada pela Banca Examinadora.

Este trabalho é dedicado a quem quer que acredite que
o conhecimento tem o poder para transformar toda a nossa sociedade.

Agradecimentos

Primeiramente quero agradecer a minha família, meu pai Márcio Freitas, minha mãe Vânia Florêncio e minha irmã Lorrayne Florêncio, por terem sempre motivado o estudo como uma fonte de amadurecimento e me apoiado a fazê-lo. Apoiaram novamente de todas as formas para que eu pudesse me dedicar ao máximo. Agradeço a Daniela Farina por entender a importância deste trabalho e por me motivar durante o longo processo de desenvolvimento de toda a pesquisa. Ela que foi a pessoa de quem eu mais me aproximei durante esta pandemia e aquela com quem eu tive maior proximidade durante toda a vida. Agradeço os meus amigos Douglas Cavalcanti e Tiago Faria por se manterem juntos durante todo o programa. Nós três fizemos aulas juntos (presenciais e virtuais), compartilhamos ideias, discutimos muito mas sempre apoiando uns aos outros. Agradeço o Marcelo Albertini por ter me orientado de forma cirúrgica e por ter confiado em meu trabalho apesar do meu pequeno problema com as datas de entrega. Ele normalmente usa poucas palavras, mas que são precisas e proveitosas, e foi responsável por grande parte da qualidade deste trabalho. Agradeço também ao programa de bolsas CAPES por proporcionarem o apoio financeiro que foi essencial. Isso permitiu que eu tivesse uma dedicação muito maior do que eu teria em outras situações. Por fim, agradeço a entidade ou força criadora, nomeada de várias formas. Ela que concebeu e uniu todos os elementos aqui presentes permitindo que construíssem esse momento chamado de coincidência, mas que também pode ser chamado de destino.

"Eu não quero ser melhor que as outras pessoas,
eu não quero é não ser o melhor de mim mesmo naquilo que eu possa ser."
(Prof. Mario Sergio Cortella)

Resumo

Com o aumento da produção de séries temporais, houve também o aumento da necessidade de minerá-las. Uma das tarefas de mineração que mais ganhou destaque nos últimos anos foi a classificação de séries temporais. Essa tarefa recebeu muitas publicações e soluções que focaram principalmente na acurácia das classificações. Isso levou a um estado da arte especializado em resultados de alta acurácia, mas também com um alto tempo de processamento. Essa particularidade inviabiliza o uso das soluções em problemas reais de maior escala. Com o objetivo de obter resultados tão acurados quanto rápidos, este trabalho propõe o 4T que é um algoritmo de extração e seleção de características baseado em dicionário com foco na eficiência da classificação de séries temporais. A eficiência, proposta nesta dissertação como uma métrica de avaliação, foi definida como a razão entre o *score* e o tempo de treinamento de uma classificação. Os resultados obtidos pelo 4T mostram uma eficiência média maior que a eficiência dos resultados disponíveis de todo o estado da arte. Esses resultados incluem os *scores* de acurácia e AUROC e o tempo de treinamento na classificação de 71 *datasets* do repositório UEA & UCR.

Palavras-chave: 4T, baseado em dicionário, classificação eficiente, eficiência, séries temporais, TSC.

Abstract

As the production of time series increases, so does the need to mine them. Currently one of the most prominent mining tasks has been the time series classification. This task received many publications and solutions mainly focused on classification accuracy. This led to a state of the art specialized in high accuracy results, but also with a high processing time. This characteristic makes the solution usability infeasible for large scale problems. Aiming to obtain both accurate and fast results, this work proposes 4T. It is a dictionary-based algorithm of feature extraction and selection focused on the efficiency of time series classification. The efficiency was proposed in this dissertation as an evaluation metric and was defined as the fraction between score and fitting time of a classification. The results obtained by 4T show an average efficiency higher than the efficiency of the available state-of-the-art results. These results include two scores: accuracy and AUROC. Along with fitting time the scores were calculated by classifying 71 *datasets* of the UEA & UCR archive.

Keywords: 4T, dictionary-based, efficient classification, efficiency, time series, TSC.

Lista de ilustrações

Lista de tabelas

Lista de algoritmos

Lista de siglas

BoB *bag of bags*

BoP *Bag of Pattern*

BOSS *Bag-of-SFA-Symbols*

CNN *Convolutional Neural Network*

COTE *Collection Of Transformation Ensembles*

DTW *Dynamic Time Warping.*

DFT *Discrete Fourier Transform*

HIVE-COTE *Hierarchical Vote* COTE

k-NN k-*Nearest Neighbor*

MCB *multiple coefficient binning*

PAA *piecewise aggregate approximation*

RISE *Random Interval Spectral Ensemble*

ROCKET *random convolutional kernel transform*

SAX *Symbolic Aggregate approXimation*

SFA *Symbolic Fourier Approximation*

SVM *Suport Vector Machine*

TSC *Time Series Classification*

WEASEL *Word Extraction for Time Series Classification*

V0 Variante zero

V1 Variante um

V2 Variante dois

V3 Variante três

V4 Variante quatro

Sumário

Capítulo 1

Introdução

Considerada um dos problemas mais desafiadores da mineração de dados (YANG; WU, 2006; ESLING; AGON, 2012), a classificação de séries temporais (*Time Series Classification* - TSC) é um tema que ganhou destaque nos últimos anos (GEURTS, 2001; BAGNALL et al., 2017; RUIZ et al., 2021). Uma das razões para isso foi o aumento da produção de séries temporais (SILVA et al., 2018), encontradas em diversas áreas como na medicina (GOLDBERGER et al., 2000; WEI; KEOGH, 2006), na astronomia (JR et al., 2018) e na econometria (SONG; LI, 2008). Uma segunda razão foi a centralização da base de dados compartilhada pela comunidade científica elaborada pelo maior repositório de série temporais UEA & UCR (BAGNALL; LINES; KEOGH, 2021)).

As séries temporais, caracterizadas por sua ordenação natural, são sequências de observações que representam objetos que variam de acordo com o tempo (ESLING; AGON, 2012). Seja uma série formada de L observações e x_i o valor da observação no tempo i, um objeto o é representado pela série S_o onde $S_o = [x_1, x_2, ...x_L]$. Sendo assim, um conjunto de N objetos pode ser representado por um conjunto D de N séries da forma $D = S_1, S_2, ..., S_o, ..., S_N$.

No contexto da classificação, cada objeto é associado a uma classe y pertencente a um conjunto Y com C classes predeterminadas $Y = \{y_1, y_2, ..., y_C\}$. Dentro da classificação, o conjunto D será então formado por pares de séries com suas respectivas classes, onde $D = (S_1, y_1), (S_2, y_2), ..., (S_o, y_o), ..., (S_N, y_N)$. Utilizando as séries de classes conhecidas é possível estimar uma função de classificação por meio de um processo de treinamento. Essa função recebe como entrada uma série e retorna sua respectiva classe. Dessa forma, é possível executar a classificação de séries temporais por meio de uma função estimada a partir de uma base de dados conhecida.

Diferentemente da classificação convencional, as características de cada série (x_i) possuem uma ordenação natural ao objeto proveniente da forma como é feita a sua amostragem. Essa ordenação aumenta a dificuldade da classificação e pode levar algoritmos tradicionais a perderem informações importantes dos dados, uma vez que muitos desses algoritmos se baseiam na suposição que as características dos objetos são independentes

e igualmente distribuídas (i.i.d) (ATLURI; KARPATNE; KUMAR, 2018).

Para contornar as dificuldades da classificação de séries temporais foram desenvolvidos centenas de soluções e algoritmos (BAGNALL et al., 2017) que buscam por características discriminatórias para diferenciar as séries uma das outras, chamados de modelos discriminativos (FAWAZ et al., 2019). A fim de facilitar a comparação e a organização, esses algoritmos foram agrupados de acordo com a característica que cada um deles busca. Podendo ser divididos entre os tipos de: similaridade, intervalo, *shapelets*, baseado em dicionário, combinação e *deep learning*.

Dentre os algoritmos propostos, um de grande sucesso foi o 1-NN DTW (RATANAMAHATANA; KEOGH, 2005; DING et al., 2008). Reconhecido como o *benchmark* para a tarefa de TSC (BAGNALL et al., 2017; RUIZ et al., 2021), o 1-NN DTW é basicamente formado por um 1-*nearest neighbor* com a função de distância *dynamic time warping*. Um classificador simples que busca por características de similaridade e obtém bons resultados, fazendo com que outros algoritmos tenham dificuldade em superar sua acurácia de forma significativa (BAGNALL et al., 2017; RUIZ et al., 2021).

A evolução dos algoritmos que se deu a partir do *benchmark* foi significativa, proporcionando principalmente uma melhora da acurácia nos resultados de classificação (KEOGH; KASETTY, 2003). Apesar do resultado acurado dos melhores classificadores como o COTE, BOSS, WEASEL, MUSE, HIVE-COTE, CIF, TDE, HIVE-COTE 2.0, ser necessário em alguns problemas, em outros ele pode não ser satisfatório devido ao seu alto custo computacional (JI et al., 2019). Seja pelo alto consumo de memória ou pelo alto consumo de tempo para o treinamento, como foi no caso dos algoritmos MUSE e HIVE-COTE testados em (RUIZ et al., 2021).

Por este motivo, pesquisas estão surgindo e propondo algoritmos com foco tanto no *score* quanto no tempo da classificação de séries temporais. O estado da arte desse tipo de algoritmo é composto pelos algoritmos Proximity Forest, TS-CHIEF, Inception-Time, ROCKET, cBOSS e o MrSQM. Os *scores* da classificação desses algoritmos são competitivas mesmo em comparação com algoritmos que tem apenas esse foco. O ROCKET, por exemplo, reduziu de forma significativa a complexidade e o tempo de processamento e obteve um dos melhores *scores* na classificação de séries temporais (RUIZ et al., 2021).

Contudo, como dito em (MIDDLEHURST et al., 2021), a escalabilidade alcançada pelos algoritmos do estado da arte ainda não é o suficiente. Quando um conjunto de dados muito grande é executado por eles, o tempo de processamento começa a se tornar inviável. Além disso, há uma relação entre o tempo de processamento e o *score* do classificador. Reduzir o tempo de treinamento pode reduzir também a seu *score*. Essa relação pode ser medida através da eficiência onde quanto maior o seu valor maior a razão entre o *score* e o tempo de classificação.

Aprimorar a eficiência na classificação de séries temporais, tornando a classificação mais rápida e mais precisa, é uma necessidade que surge à medida que a quantidade e

a complexidade dos dados aumentam. Conforme mencionado anteriormente, problemas eminentes de análise e classificação de séries temporais estão surgindo com situações que podem demandar uma classificação em tempo real para tomadas de decisões (JI et al., 2019), possuir limitações de processamento para experimentos (RUIZ et al., 2021) ou possuir bases de dados muito grandes que se renovam e precisam ser processadas todos os dias para evitar um acúmulo de dados (JR et al., 2018).

Dito isto, a motivação deste trabalho são os problemas que consideram tão importante o tempo quanto a acurácia da classificação de séries temporais. Tais problemas podem ser atendidos por soluções cujo tempo de processamento é pequeno, abdicando-se do melhor *score* possível. Para avaliar o balanço entre a pontuação e o tempo de processamento e definir qual solução é mais eficiente, uma função de eficiência foi proposta. Essa função é uma métrica que calcula um valor número para cada classificação possibilitando a sua comparação e seu ranqueamento.

Sendo assim, o objetivo principal deste trabalho é melhorar a eficiência na classificação para séries temporais. Para tanto foi proposto um algoritmo de extração e seleção de características baseadas em dicionário para classificação de séries temporais construído maximizando a sua eficiência. O algoritmo proposto, denominado 4T, extrai características baseadas em dicionário e faz uma seleção no vasto espaço de busca extraído para treinar um classificador convencional. Uma estrutura comumente utilizada na literatura presente em algoritmos como o WEASEL, MUSE, MrSEQL, MrSQM e o ROCKET.

Neste trabalho acredita-se que os algoritmos baseados em dicionário possuem um grande potencial em eficiência que ainda não foi alcançado. Em (RUIZ et al., 2021), apesar de sua limitação de alto consumo de memória alguns dos algoritmos baseados em dicionário alcançaram uma boa acurácia, destacando um potencial na pontuação da classificação. Em (NGUYEN et al., 2019) e (NGUYEN; IFRIM, 2021) os respectivos algoritmos propostos MrSEQL e MrSQM alcançaram tempos de processamentos competitivos em relação ao estado da arte, destacando assim um potencial em relação ao tempo.

1.1 Contribuições

A principal contribuição deste trabalho foi o desenvolvimento de um novo algoritmo de extração e seleção de características baseadas em dicionário para a classificação eficiente de séries temporais denominado 4T. As contribuições secundárias, porém tão importantes quanto, foram o desenvolvimento de ferramentas para a realização dos experimentos e seus resultados. Duas das ferramentas são a função de eficiência e as variantes. A primeira é uma métrica desenvolvida para avaliar e comparar diferentes classificações, enquanto a segunda ferramenta é uma conjunto de algoritmos projetados para avaliar as interações das técnicas de extração selecionadas e facilitar a composição de todas elas na implementação do algoritmo 4T. Os resultados obtidos por meio das ferramentas e experimentos

contribuíram para a identificação de interações não desejadas e para a identificação de melhorias para a solução proposta. Além de ilustrarem uma maior eficiência dos algoritmos 4T e suas variantes em relação ao estado da arte na classificação de séries temporais univariadas.

1.2 Organização da Dissertação

Dividido entre seus capítulos, este trabalho apresenta os conceitos básicos (Capítulo 2) que explicam o que são as séries temporais, como elas são classificadas, o que são algoritmos baseados em dicionário e algumas de suas técnicas de extração e seleção; os trabalhos correlatos (Capítulo 3) que descrevem dois algoritmos recentes que mais se assemelham com o algoritmo 4T e o estado da arte da classificação de séries temporais; a proposta da pesquisa (Capítulo 4) que descreve detalhadamente o algoritmo 4T e a implementação de cada técnica que o compõe; os experimentos (Capítulo 5) que descrevem os métodos e as avaliações propostas para construir e comparar o 4T; os resultados dos experimentos (Capítulo 5) que exibem os valores das comparações e destacam as classificações mais eficientes; e, por fim, a conclusão da pesquisa (Capítulo 6) que descreve os resultados desta pesquisa de forma geral, destaca suas contribuições e finaliza o texto sugerindo possíveis continuidades.

Capítulo 2

Conceitos Básicos

Os principais conceitos utilizados neste trabalho estão divididos entre as seções deste capítulo que começam descrevendo o que são as séries temporais, suas distinções por meio das características discriminantes, sua definição formal, e como é feita a sua classificação; como funciona os algoritmos baseados em dicionários de forma geral e quais são suas quatro principais etapas; o que é o processo de transformação das séries temporais com valores reais para sequências simbólicas discretas chamado de discretização e como executam os seus dois principais algoritmos, o SAX e o SFA; e, por fim, a última seção encerra o capítulo descrevendo as técnicas utilizadas na implementação da proposta, começando pela estrutura base de busca e classificação e pelo algoritmo de seleção *chi-square* seguidos pelas quatro técnicas de extração e seleção de características responsáveis empregadas na construção algoritmo 4T.

2.1 Séries Temporais

Em alguns fenômenos, um conjunto de dados pode conter diferentes observações de um mesmo objeto, obtidas variando o tempo da observação. Tal sequência de observações é chamada de série temporal (*time series*). Uma série temporal é uma sequência ordenada de observações de tamanho finito que geralmente são obtidas por mudanças temporais (ABANDA; MORI; LOZANO, 2019). Frequentemente, o histórico econômico é observado na forma de série temporal, como exemplos temos o índice de preço ao consumidor, a taxa de desemprego e a produção de um país. Ciências naturais fornecem também exemplos de séries temporais como o nível de uma represa, a temperatura do ar, as observações astronômicas e o tamanho de populações naturais (FULLER, 2009).

Existem diferentes campos de estudos em torno de séries temporais, como a clusterização, detecção de anomalias e a predição (ESLING; AGON, 2012). Entretanto, como mencionado anteriormente, o foco deste trabalho é na classificação de séries temporais, a tarefa de TSC. Uma área importante que vem crescendo rapidamente nas últimas décadas com notável aumento na quantidade de dados e no aumento da necessidade de

processá-los, como mostrado em (SHIFAZ et al., 2020).

2.1.1 Classificação de Séries Temporais

Antes de definir o que é TSC, primeiro as séries temporais são definidas formalmente espelhada nos trabalhos (FAWAZ et al., 2019; FAWAZ et al., 2020). Deste modo, temos:

Definição 1: Uma série temporal univariada $X = [x_1, x_2, ..., x_L]$ é um conjunto ordenado com L valores reais.

Definição 2: Uma série temporal multivariada $\mathbf{X} = [X^1, X^2, ..., X^M]$, M-dimensional, é um conjunto não ordenado com M de séries temporais univariadas, onde cada série $X^i \in \mathbb{R}^L$.

Definição 3: Seja um conjunto de c classes distintas $Y = \{y_1, y_2, ..., y_C\}$, um conjunto de dados $D = \{(X_1, y_1), (X_2, y_2), ..., (X_N, y_N)\}$ é uma coleção de pares (X_i, y_i) sendo X_i uma série temporal, univariada ou multivariada, e y_i a sua classe correspondente.

Considerando o conjunto de dados D onde cada série temporal possui uma e apenas uma classe associada a ela, temos que a tarefa de classificação é um mapeamento de uma série para a sua classe (XING; PEI; KEOGH, 2010). Um classificador, portanto, é uma função de mapeamento $f(série) \rightarrow classe$ estimada a partir de uma base de dados conhecida. Formalmente, podemos definir a classificação como:

Definição 4: A tarefa de classificação de séries temporais é estimar uma função $f : X \rightarrow y$, a partir dos conjuntos conhecidos de dados D e de classes Y, com $X \in D$ e $y \in Y$.

2.1.2 Características Discriminantes

Para organizar e facilitar a comparação de pesquisas, os algoritmos de classificação de séries temporais são categorizados agrupando os algoritmos semelhantes. Dentre as diferentes maneiras de agrupá-los, a mais usual é categorizar os algoritmos de TSC de acordo com o tipo de característica discriminantes que eles pretendem alcançar (BAGNALL et al., 2017; SHIFAZ et al., 2020). As principais características são: Similaridade, Intervalos, Shapelets, Baseadas em Dicionário, Baseadas em Modelo, Combinações e Aprendizado Profundo. Dentre estas categorias, a baseada em modelo é a que possui menos desenvolvimentos. Segundo (BAGNALL et al., 2017), tanto em experimentos práticos quanto nos resultados da literatura, abordagens de modelos gerativos não atingem resultados competitivos para a tarefa de classificação.

No artigo (FAWAZ et al., 2019), as abordagens para aprendizado profundo também podem ser subdivididas em modelos discriminativos e modelos gerativos. O primeiro consiste em um classificador/regressor que aprende diretamente o mapeamento entre os dados de entrada de uma série temporal, ou suas características, e a saída com a distribuição de probabilidade sobre as classes em um conjunto de dados. O segundo geralmente possui

uma fase de treinamento que precede a fase de aprendizado do classificador. O objetivo deste treinamento é encontrar uma boa representação da série temporal que posteriormente será passada para o classificador de fato.

A seguir há uma breve descrição de cada uma das características começando por uma das primeiras a serem utilizadas que é a similaridade. A partir dela foram surgindo formas mais complexas ou criativas de representar as informações extraídas de séries temporais. Note que cada uma delas adéqua-se melhor a um certo tipo de padrão ou informação que possa estar presente na série.

1. **Similaridade** Este tipo de algoritmo compara a similaridade de duas séries através de uma medida de distância com ou sem uma compensação por distorções locais. É apropriado quando há características discriminatórias ao longo da série inteira, podendo ter deslocamentos nessas características no eixo do tempo. Uma referência clássica para TSC tem sido o 1-NN utilizando o DTW (NN-DTW). A janela de distorção (*warping window*) é um parâmetro que controla a elasticidade da medida. Uma janela de distorção de tamanho 0 equivale à distância euclidiana, enquanto uma janela grande permite corresponder pontos de uma série e outra com longos espaços de tempo.

   Muitas das pesquisas de séries temporais, com algoritmos baseados na similaridade, se concentraram em alternativas para as medidas de distâncias elásticas. Para TSC, estas medidas distâncias têm sido avaliadas quase que exclusivamente utilizando o classificador 1-NN. Algumas das alternativas propostas são: *weighted* DTW, *Time Warp Edit*, *Move-split-merge*, *derivative* DTW e *weighted derivative* DTW. Detalhes sobre essas medidas podem ser encontrados em (BAGNALL et al., 2015; BAGNALL et al., 2017).

   Uma melhora nessa família de algoritmos foi a formação de *ensembles*. Um conjunto de múltiplos classificadores 1-NN com diversas medidas de similaridade que provaram ser significantemente mais acurado do que 1-NN com uma única distância (BAGNALL et al., 2015). Como exemplo tem-se o *Elastic Ensemble* (BAGNALL et al., 2015), que combina 11 1-NN algoritmos, cada um com uma das 11 distâncias elásticas (BAGNALL et al., 2015), e a *Proximity Forest* (LUCAS et al., 2019), um conjunto de árvores de decisão, onde cada nó da árvore se divide baseado na similaridade com uma série temporal que representa cada classe no conjunto de dados. *Proximity Forest* supera o *Elastic Ensemble* em escalabilidade e acurácia.

2. **Intervalos** Esse conjunto de algoritmos extraem características de intervalos de cada série temporal. Os intervalos são selecionados a partir das séries inteiras e uma transformação é aplicada a fim de gerar um novo vetor de características. O número de intervalos possíveis para uma série de tamanho m é de $m(m-1)/2$ intervalos contínuos. O vetor de características gerado é então usado para treinar

um classificador tradicional, geralmente uma floresta de Árvores Aleatórias, similar a uma floresta aleatória, porém sem *bagging*.

Os problemas que se beneficiam da seleção de intervalos, ao invés de analisar a série completa, são aqueles que provavelmente possuem longas séries temporais com subséries discriminatórias dependentes de fase (ocorrem aproximadamente na mesma região) e regiões de ruído que podem confundir um classificador. Dito isto, a dificuldade está em encontrar o melhor intervalo. Nos algoritmos de TSC, ao invés de buscar os intervalos nos dados de treino, os algoritmos geram muitos intervalos aleatórios diferentes e classifica em através de cada um deles, utilizando *ensembles* nas predições resultantes.

Para solucionar o problema da grande quantidade de intervalos, o *time series forest* seleciona $\sqrt{m}$ intervalos aleatórios e aplica em cada um três transformações, a média, o desvio padrão e a inclinação, todas de domínio temporal. Com esta nova representação dos dados, o *time series forest* treina uma árvore de decisão e repete o processo até treinar todo o conjunto de árvores em diferentes conjuntos de intervalos aleatórios. Além deste, outros algoritmos como o *time series bag of features*, *learned pattern similarity* e introduzido mais recentemente o *random interval spectral ensemble* (RISE) são descritos no (BAGNALL et al., 2017; SHIFAZ et al., 2020).

3. **Shapelets** Ao contrário de buscar intervalos, essa família de algoritmos buscam subséries que permitem discriminar as classes independentemente de onde a subsérie seja encontrada. Idealmente, um bom *shapelet* deve ser similar aos elementos de uma mesma classe e dissimilar aos elementos de outras classes. Geralmente a medida de similaridade é calculada por meio da menor distância Euclidiana de um dado *shapelet* para todas as subséries de mesmo tamanho de outras séries temporais.

   O algoritmo originalmente proposto em (YE; KEOGH, 2011) encontra os *shapelets* buscando dentre todos os possíveis candidatos e utiliza o melhor como critério de divisão em cada nó de uma árvore de decisão. Para isto ele usa o critério de Ganho de Informação para avaliar quão bem um candidato pode dividir os dados. Assim o melhor candidato encontrado e uma distância limite são definidos para um nó da árvore. A busca continua recursivamente até que encontre folhas puras.

   Como é lento avaliar todos os possíveis *shapelets*, a complexidade de treino do algoritmo é de $O(n^2l^4)$. Isto levou a maioria das pesquisas sobre *shapelets* a focar em aumentar a velocidade do algoritmo. Em vez de buscar pelo melhor candidato dentre todos os possíveis, alguns algoritmos buscam encontrar rapidamente apenas bons candidatos. São eles: *Fast Shapelets*, *Learned Shapelets*, *Shapelet Transform* e *Generalized Random Shapelet Forest*, todos descritos em (BAGNALL et al., 2017; SHIFAZ et al., 2020).

4. **Baseada em dicionário** Enquanto os *shapelets* conseguem encontrar um único padrão de fase independente que discrimina as classes, os algoritmos baseados em dicionários distinguem as classes por meio da frequência relativa em que os padrões são encontrados. Assim, uma subsérie encontrada em todas as classes pode discriminá-las pelo número de repetições da subsérie em cada classe. Esse tipo de algoritmo é bom em lidar com dados ruidosos e encontrar características discriminantes em padrões recorrentes.

   Para isto, os algoritmos baseados em dicionários aproximam as séries em séries menores, reduzem a dimensionalidade transformando-as em palavras representativas e utiliza a similaridade da distribuição das palavras para alimentar o classificador. A aproximação é feita através de uma janela deslizante de tamanho l que produz w valores, reduzindo o comprimento da série, e então um método de quantização é aplicado para gerar as palavras. Cada série é então representada por um histograma que conta a frequência de cada palavra. 1-NN com alguma medida de similaridade de histogramas pode ser treinado como um modelo classificador. Alguns exemplos citados em (BAGNALL et al., 2017; SHIFAZ et al., 2020) são: *Bag of Pattern* (BoP), *Symbolic Aggregate Approximation-Vector Space Model*, *Bag-of-SFA-Symbols* (BOSS), BOSS com *Vector Space* e *Word Extraction for Time Series Classification* (WEASEL) (SCHÄFER; LESER, 2017).

5. **Baseada em modelo** Algoritmos baseados em modelos estimam um modelo gerativo para cada série temporal para então medir a similaridade das séries através de seus modelos. Algumas abordagens incluem modelos autorregressivos, modelos ocultos de Markov e modelos de *kernels*. O artigo (BAGNALL et al., 2017) destaca três dificuldades para a utilização de tais modelos na tarefa de TSC. A primeira dificuldade é proveniente do objetivo principal desses algoritmos. Eles geralmente são projetados para solucionarem tarefas como regressão ou *forecasting* e não para a classificação de dados. A segunda é a dificuldade de acesso às implementações desse tipo de algoritmo e, por último, baseado em experimentos e resultados publicados, os autores do artigo afirmam que muitos destes algoritmos não são competitivos para a classificação.

6. **Combinações** Nas pesquisas recentes de TSC surgiram muitos *ensembles* altamente competitivos. Os algoritmos *Time Series Forest*, *Time Series Bag of Features* e BOSS são *ensembles* que replicam um mesmo classificador. O *Elastic Ensemble* e o *Shapelet Transform* utilizam classificadores diferentes para formar seus *ensembles*. Todos eles, porém, construídos sobre a mesma representação dos dados. Dois algoritmos iniciaram a proposta, com resultados promissores, de um conjunto de classificadores e representações diferentes. São eles o DTW *features* (KATE, 2016) e o *Collection Of Transformation Ensembles* (COTE). O primeiro combina técni-

cas baseadas em similaridade e baseadas em dicionários em um único classificador, enquanto o segundo utiliza características de similaridade, intervalos, *shapelets* e baseadas em modelo. Uma variação mais recente e melhorada do COTE é o *Hierarchical Vote* COTE ou HIVE-COTE (LINES; TAYLOR; BAGNALL, 2018).

7. **Aprendizado Profundo** O grupo mais recente, aplicado em TSC, é o de algoritmos de aprendizado profundo. Esses algoritmos possuem uma estrutura bem formada, oferecida pelas pesquisas passadas que tem bons resultados em processamento de imagem e vídeo, e uma escalabilidade linear com relação à quantidade de dados de treino. O sucesso obtido pelas CNN em dados com duas dimensões (imagens) sugere que elas também sejam efetivas para dados com uma dimensão a menos (séries temporais) (BENGIO; COURVILLE; VINCENT, 2013). CNN, ou *Convolutional Neural Network*, representa uma abordagem diferente para TSC em relação aos outros métodos. Ao invés de atuar sobre representações dos dados predefinidas, estas redes utilizam *kernels* convolucionais para buscar sua própria representação dos dados que melhor representam os padrões dos dados de entrada.

   A maioria dos algoritmos de aprendizado profundo tem focado em desenvolver arquiteturas específicas baseadas em CNN junto a técnicas de aumento de dados. Dois algoritmos recentemente propostos mostraram atingir o estado da arte em acurácia, HIVE-COTE, comparados em 85 datasets de referência presentes no repositório UEA & UCR (BAGNALL; LINES; KEOGH, 2021). Um deles é o InceptionTime (FAWAZ et al., 2020), um conjunto de cinco redes neurais convolucionais profundas inspiradas na arquitetura da Inception-v4. O segundo algoritmo é o *random convolutional kernel transform* (ROCKET) (DEMPSTER; PETITJEAN; WEBB, 2020), que utiliza diversos *kernels* convolucionais aleatórios para a extração de características e um classificador linear que utiliza as características extraídas para estimar a função de mapeamento, possibilitando alterar entre o algoritmo *logistic regression* ou *Ridge regression*.

2.2 Algoritmos de TSC baseados em dicionário

Os classificadores baseados em dicionários são algoritmos que buscam identificar uma classe de objetos por meio da frequência de padrões que se repetem ao longo das séries temporais. Para isso, esse algoritmos aproximam e reduzem a dimensionalidade das séries transformando-as em sequências simbólicas, chamadas de palavras. As distribuições dessas palavras, então, são utilizadas para calcular a similaridade entre diferentes séries.

No geral, os algoritmos baseados em dicionários utilizam uma janela deslizante que varre cada série produzindo uma subsequência. Tanto a janela quanto as subsequências de tamanho l. Em seguida, cada subsequência passa por uma aproximação de valores

normalmente reduzindo sua dimensionalidade de l para w. As subsequências aproximadas são discretizadas, transformando seus valores reais em valores simbólicos, formando de fato as palavras de tamanho w.

Seguindo todo esse processo, cada série, transformada em um conjunto de palavras, é representada por um histograma que contém a frequência das palavras daquela série, abordagem esta chamada de *bag of words*. Uma das maneiras de classificar utilizando essa representação é utilizando um classificador baseado em distância, como o 1-NN, calculando a similaridade das séries por meio da diferença de seus histogramas, que são utilizadas para calcular a similaridade entre as séries.

As diferenças entre os classificadores baseados em dicionários podem ser identificadas em quatro etapas principais dos algoritmos (LARGE et al., 2019). São elas: aproximação, discretização, representação e classificação. Quase sempre as duas primeiras são executadas por discretizadores, como o SAX e o SFA descritos na próxima seção. Comumente, na terceira etapa, é encontrada a representação *bag of words*, como nos algoritmos BOSS, WEASEL, MUSE, ou a representação bigram, como nos algoritmos WEASEL, MUSE e MrSEQL. A última etapa utiliza um classificador convencional de dados, que não considera a ordem das características, contudo diferentes algoritmos são encontrados.

2.2.1 Algoritmos de Discretização

Os discretizadores mais conhecidos e mais utilizados são o SAX e o SFA. Ambos executam as etapas de aproximação e discretização dos algoritmos baseados em dicionários extraindo características relevantes em dois domínios diferentes. O SAX atua no domínio do tempo enquanto o SFA na frequência. Os dois algoritmos são importantes e bastante utilizados na literatura, portanto, foram selecionados para executar a discretização das séries temporais neste trabalho. Sendo assim, cada um deles tem seu funcionamento detalhado a seguir.

SAX, ou *Symbolic Aggregate approXimation*, é um algoritmo de discretização de séries temporais que transforma séries temporais em sequências simbólicas. Proposto em (LIN et al., 2007), o SAX é capaz de reduzir a dimensionalidade dos dados e de preservar o limite inferior das séries temporais nas séries transformadas. O primeiro algoritmo de discretização a possuir essas duas funcionalidades.

O SAX possibilita a transformação de uma série temporal de tamanho L em uma sequência simbólica de tamanho w qualquer menor ou igual a n por meio da redução de dimensionalidade. A sequência simbólica gerada possui uma quantidade a de símbolos distintos que formam o alfabeto, sendo a um inteiro arbitrário maior que 2. No entanto, o alfabeto do SAX na implementação usada na literatura tem seu tamanho limitado a um máximo de 4.

Entre as séries de valores reais e as sequências simbólicas, o SAX foi o primeiro algoritmo a executar um processo intermediário de aproximação. O algoritmo utilizado para

esta tarefa é o *piecewise aggregate approximation* (PAA) proposto em (KEOGH et al., 2001) reduzindo a dimensionalidade da série que é estendida para a sequência final.

O PAA é o algoritmo que efetivamente executa a redução da série temporal por meio de um processo simples mas ainda competitivo em relação a outros métodos mais sofisticados como aproximações de Fourier ou *wavelets*. Primeiro, o algoritmo PAA recebe uma série para ser processada e o tamanho resultante após a redução de dimensionalidade (w). Em seguida ele divide a série recebida em w segmentos e calcula a média de cada um deles resultando em um vetor de w valores reais. O vetor resultante é a série temporal reduzida e aproximada. Para questão de simplicidade é assumido que L seja divisível por w.

Para uma melhor comparação das séries com a representação PAA, é esperado que elas possuam uma mesma faixa de valores e uma mesma amplitude (KEOGH; KASETTY, 2003). Portanto, antes de executar a transformação PAA, o algoritmo SAX normaliza todas as séries temporais para média igual a 0 e desvio padrão igual 1.

Esse processo intermediário de aproximação demonstra duas vantagens, a primeira delas é a redução de dimensionalidade, utilizando um algoritmo bem definido como o algoritmo PAA que muitas vezes conduz para a transformação da representação real em simbólica de forma direta. A segunda vantagem é a simplificação da prova do limite inferior das distâncias entre duas séries temporais, uma tarefa não trivial, que é dividida em duas etapas: a prova do limite entre a série original e a representação aproximada e a prova entre a representação aproximada e a simbólica. Dependendo do algoritmo de aproximação utilizado a prova da primeira etapa é conhecida, poupando um certo esforço, enquanto a prova da segunda etapa geralmente é mais fácil comparada com a prova direta entre a série original e sua representação simbólica. No caso do algoritmo SAX, a prova do limite inferior foi desenvolvida entre as representações PAA e simbólica, uma vez que a primeira etapa é conhecida.

Uma vez que a série temporal foi transformada para a representação PAA, a transformação para a representação simbólica pode ser aplicada. O processo de discretização assume uma distribuição Gaussiana dos dados da série dividindo os valores reais em a segmentos equiprováveis. Cada segmento é representado por um símbolo do alfabeto, constituindo uma função de transformação de números reais para símbolos. Portanto, o algoritmo SAX calcula os segmentos e discretiza a série aproximada finalizando o processo com uma sequência simbólica como resultado.

Os autores do SAX testaram a suposição sobre a distribuição gaussiana dos dados que se mostrou verdadeira quando utilizada em um grande conjunto de séries temporais. Para um subconjunto pequeno, onde a suposição falha, a eficiência do algoritmo é reduzida. Contudo, a corretude do algoritmo é mantida, uma vez que ela se baseia na prova dos limites inferiores das distâncias entre as séries.

Seguindo a ideia do modelo BoP, todo esse processo de discretização do algoritmo SAX é aplicado às subsequências da série temporal recebido definindo uma janela deslizante de

tamanho l. Portanto, o algoritmo recebe uma série temporal de comprimento L, ou um conjunto de séries, o tamanho da janela deslizante l, o tamanho da palavra resultante w e o tamanho do alfabeto utilizado a. Como resposta, o algoritmo produz um conjunto de $n - w$ palavras de tamanho w com a símbolos distintos.

SFA é um algoritmo de discretização que introduz a representação simbólica baseada no domínio da frequência, contrapondo o domínio de tempo. Foi proposto em (SCHÄFER; HÖGQVIST, 2012) e denominado como *Symbolic Fourier Approximation*. Como o algoritmo SAX, o SFA possui uma etapa de aproximação e uma de discretização, porém se diferencia em ambas as etapas.

A aproximação, que também permite a redução de dimensionalidade, é executada pelo algoritmo *Discrete Fourier Transform* (DFT) (OPPENHEIM; SCHAFER, 1975). De maneira sucinta, a aproximação DFT transforma uma série temporal em uma soma de função ortogonais usando ondas senoidais. Cada função, ou onda, é definida por um coeficiente de Fourier onde os primeiros representam com as frequências mais baixas e os últimos as frequências mais altas. Na série temporal essas frequências são entendidas como mudanças graduais e lentas. Enquanto as frequências mais altas, correlacionadas com os coeficientes de ordem maior, retratam as mudanças mais bruscas e rápidas dos valores da série temporal.

O resultado da transformação DFT é um vetor de coeficientes de Fourier. Cada coeficiente é representado por um número complexo necessitando de dois números para descrevê-lo. Portanto, a representação DFT de uma série é um vetor de números reais e imaginários que alternam entre si. A redução de dimensionalidade é efetivada ao definir um número de coeficientes w menor do que duas vezes o tamanho da série $2n$. Tal aproximação suaviza a curva presente na série e ainda é possível deslocá-la para a média 0 removendo o primeiro coeficiente, que representa a média do sinal do somatória das curvas.

Após a transformação da série para a representação DFT, é necessário definir as faixas de valores reais para executar a discretização dos valores reais em símbolos. Para isso, o algoritmo SFA não supõe uma distribuição dos dados, ele faz um preprocessamento utilizando o algoritmo *multiple coefficient binning* (MCB), proposto pelos autores do SFA (SCHÄFER; HÖGQVIST, 2012) com o intuito de reduzir a perda de informações no processo de discretização.

Também seguindo o modelo BoP de transformação utilizando janelas deslizantes, onde cada série é transformada em um conjunto de palavras, o SFA transforma uma série em um conjunto de subsequências DFT. O MCB então utiliza todas as subsequências geradas para treinar um modelo e definir os valores limites de cada faixa para cada coeficiente. Para tanto, todas as subsequências, de uma série ou de um conjunto de séries, são arranjadas em uma matriz com w colunas onde cada coluna terá sua própria definição de faixas de valores definidas de forma que cada faixa inclua o mesmo número de subsequências, uma

abordagem *equi-depth*.

Feito o pré processamento do algoritmo MCB, a discretização das representações DFT é feita por meio do mapeamento de valores definido pelo MCB para cada coeficiente. O resultado final da transformação de uma série é um conjunto de $n - l$ palavras SFA de tamanho w com a símbolos distintos. Os autores do SFA também provaram os limites inferiores para as novas representações junto com a corretude do seu algoritmo.

2.3 Técnicas de Extração e Seleção

Uma abordagem comum para a classificação de séries temporais é executar uma seleção em um vasto espaço de características para treinar um classificador convencional. Essa estrutura, aqui chamada de estrutura base, foi adotada na proposta em conjunto com um algoritmo seleção *chi-square* executado sobre as características extraídas por meio de 4 técnicas: multirresolução, multidomínio, representação *ngram* e *ensemble*.

Estrutura-base foi o nome dado nesta dissertação para a abordagem de classificação representada na Figura 1. Essa estrutura é citada em (NGUYEN; IFRIM, 2021) como uma abordagem de extração de características e seleção em grandes espaços de busca associada a um classificador linear. Embora o classificador linear possa ter vantagens nessa situação, outros classificadores também podem ser utilizados. Esses classificadores são chamados de classificadores simples, pois é esperado que assumam uma independência das características e é esperado que sejam rápidos pelo tamanho do espaço de busca. Durante os experimentos foram testados os algoritmos de *random forest*, *suport vector machines* e o *logistic regression*. O algoritmo de *random forest* foi o que obteve os melhores resultados.

Chi-square é um algoritmo de seleção de atributos que utiliza o teste estatístico *chi-square* para avaliar a correlação entre duas variáveis. Nesse caso as variáveis comparadas são uma característica e o rótulo dos dados. Assim, a característica com a maior correlação com o rótulo dos dados, aquela que é mais discriminante, terá o maior valor do algoritmo *chi-square*. As características analisadas devem ser *booleanas* ou conter uma frequência, valores não negativos.

Com o intuito de atacar o problema de eficiência dos algoritmos baseados em dicionário, foi realizada uma análise da literatura que resultou na seleção de quatro técnicas de extração e seleção de características. Foram analisados os artigos do estado da arte considerando tanto os algoritmos baseados em dicionários quanto algoritmos que buscam por outras características discriminantes.

Esta análise foi feita através da leitura, do entendimento e da comparação de diversos artigos relacionados ao tema de TSC identificando e comparando as principais técnicas

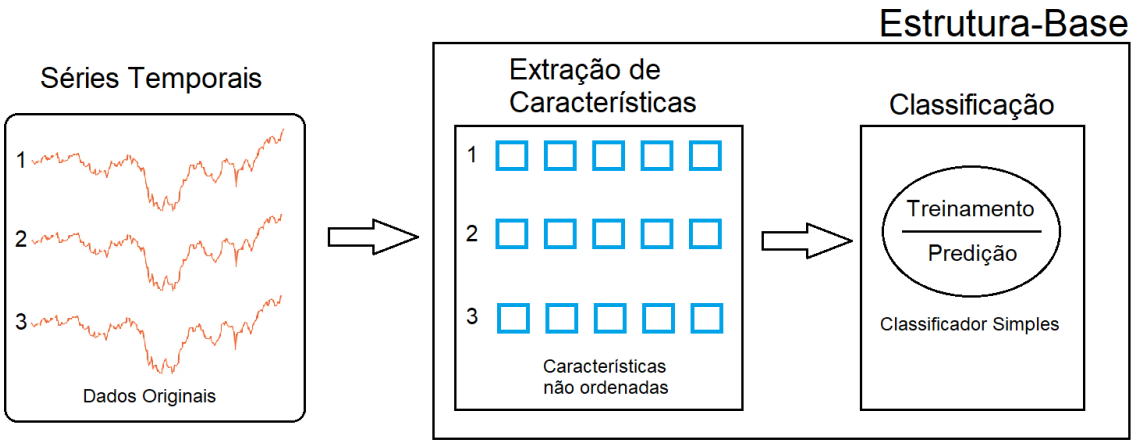

Figura 1 – *Diagrama representando a estrutura utilizada pelo algoritmo 4T com a extração removendo a dependência das características e a classificação utilizando um classificador simples*

neles empregadas. As técnicas consideradas mais promissoras foram aquelas que tiveram impacto positivo em sua implementação, de acordo com os próprios autores, e que foram utilizadas em diferentes algoritmos. De forma objetiva, cada uma delas pode ser descrita da seguinte maneira:

Multirresolução: A multirresolução é uma técnica de extração para algoritmos baseados em dicionários que utiliza palavras provenientes de discretizações configuradas com diversos parâmetros. Cada configuração do discretizador é chamada de resolução e o uso simultâneo de palavras extraídas de diferentes resoluções, para treinamento e classificação dos dados, é chamado de multirresolução. Utilizando essa técnica em conjunto com algum algoritmo de seleção de características, é possível utilizar várias resoluções aleatórias e selecionar dentre as palavras extraídas aquelas que forem discriminantes com relação às classes dos dados. Com isso, evita-se uma busca por uma resolução ideal e acelera o processo de treinamento desse tipo de classificador. Essa abordagem é chamada de *shotgun* e foi utilizada pelos algoritmos BOSS, WEASEL, MUSE, MrSEQL e MrSQM. Nesses dois últimos algoritmos, a técnica de multirresolução foi utilizada em conjunto com técnica de multidomínio permitindo que palavras de diferentes domínios, portanto de diferentes resoluções, sejam utilizadas simultaneamente no processo de treinamento e classificação. Incluindo ainda mais informações nas características extraídas.

Multidomínio: A técnica de multidomínio é uma abordagem que utiliza mais de um domínio na extração de características das séries temporais. Pode se dizer que cada tipo de algoritmo citado na Seção 2.1.2 busca características em um domínio diferente e, geralmente, os algoritmos de *ensemble* incluem dentro de si outros al-

goritmos de classificação com tipos diferentes, se tornando portanto, um algoritmo de multidomínio como é o caso do HIVE-COTE e HIVE-COTE 2.0. Essa técnica possibilita a extração características relevantes que não poderiam ser encontradas utilizando apenas um domínio. O aumento da informação extraída permite uma classificação mais acurada em troca de um consumo maior de recursos. No contexto de algoritmos baseados em dicionários é possível encontrar pelo menos dois domínios diferentes que são os domínios de tempo e de frequência, representados respectivamente pelos algoritmos de discretização SAX e SFA. Os algoritmos baseados em dicionários de multidomínio analisados foram o MrSEQL e o MrSQM.

Representação Ngram: A representação de palavras por meio de *ngrams* foi a primeira técnica selecionada pela análise, podendo ser observada nos algoritmos MrSEQL, WEASEL, MUSE, TDE e no HIVE-COTE 2.0. Ela vem de um conceito de processamento de texto onde cada *n-gram* é uma sequência de n palavras (*tokens*) de um texto e é utilizada para encontrar dependências na ordenação das palavras. No contexto das séries temporais, apesar do processo de discretização tentar retirar a dependência sequencial das amostragens de uma série, é possível que as palavras produzidas ainda possuam uma certa ordenação relevante. Isso ocorre pois o tamanho da janela definido na discretização pode ser menor do que um padrão que melhor representa uma série temporal, sendo necessário janelas sequenciais para discretizar tal padrão. Dessa forma, a possível ordenação entre as palavras discretizadas é perdida no processo de representação da série em histograma. Visto que a *bag of words* armazena apenas as frequências de cada palavra. Portanto, armazenar as frequências das *ngrams*, as sequências de palavras, é uma forma de preservar a ordenação das palavras, evitando a perda de certas informações possivelmente relevantes.

Ensemble: *Ensemble* é uma técnica bem conhecida na área de aprendizado de máquina. Na área de TSC, encontra-se algoritmos implementando o conceito para incorporar informações de diferentes domínios, como o HIVE-COTE, o TDE e o TS-CHIEF, e para evitar a busca por parâmetros utilizando vários componentes com parâmetros aleatórios, como o ROCKET e o *Proximity Forest*. Para esta pesquisa, a técnica de ensemble foi selecionada através da análise com o objetivo de otimizar a extração de características e o treinamento do classificador, distribuindo extrações e amostras diferentes dos dados entre os componentes do *ensemble*. Dessa forma é evitado que todas as séries passem por todas as extrações e torna o modelo menos suscetível à *overfitting*.

Capítulo 3

Trabalhos Correlatos

Os trabalhos que mais se assemelham a este são o (NGUYEN et al., 2019) e o (NGUYEN; IFRIM, 2021). Ambos propuseram melhorar a eficiência da classificação de séries temporais por meio de características baseadas em dicionários. Suas soluções são respectivamente os algoritmos MrSEQL e MrSQM. De maneira geral são dois algoritmos do estado da arte dos classificadores eficientes e dos classificadores baseados em dicionários que utilizam a estrutura base de classificação (Seção 2.3).

Ao longo desta seção estão descritos de forma mais detalhada os algoritmos MrSEQL e MrSQM, o estado da arte dos algoritmos eficientes, o repositório UEA & UCR que é muito utilizado para realizar pesquisas e comparar algoritmos de TSC e, por fim, o diagrama de diferença crítica que geralmente é utilizado como métrica na comparação de algoritmos fazendo uma ordenação dos resultados de classificação.

3.1 MrSEQL

O MrSEQL é um algoritmo baseado em dicionários que preza tanto pela eficiência quanto pela interpretabilidade de seus resultados. Para isso, ele baseia-se em três ideias principais: múltiplas resoluções de representações simbólicas (multirresolução), múltiplos domínios para as representações (multidomínio) e navegação eficiente em um grande espaço de palavras simbólicas.

Sua primeira ideia é a utilização da técnica de multirresolução, extraindo suas características a partir da combinação de diferentes representações das séries temporais que, por sua vez, é obtida por meio da variação do parâmetro de discretização l (o tamanho da janela). Os autores do algoritmo compararam a técnica de multirresolução utilizando um *ensemble* de classificadores treinados cada um com uma resolução e a extração de todas as resoluções simultaneamente para treinar o classificador *logistic regression*. Em ambos os casos foram utilizados os algoritmos de discretização SAX e SFA de forma separada e em conjunto. Este último constituindo a sua ideia principal do algoritmo, os múltiplos domínios.

A técnica de multidomínio foi aplicada tanto para o *ensemble* quanto para as extrações simultâneas. Foi facilmente implementada, pois os algoritmos testados possuíam a técnica de multirresolução, mantendo a mesma variação e o mesmo número de discretizadores. Apenas o tamanho da palavra (w) foi diferente, 16 para o algoritmo SAX e 8 para o algoritmo SFA.

O problema desta abordagem é o aumento do espaço de características que aumentou devido às características de diferentes resoluções. Para trabalhar eficientemente com esse grande espaço de características, após a discretização das séries, cada resolução passa por um processo de seleção definido pelo algoritmo SEQL. Constituindo terceira e última ideia do MrSEQL que é fazer uma busca em todo esse espaço de características de forma eficiente.

O algoritmo SEQL foi originalmente proposto como um classificador binário para sequências como DNA ou texto com a capacidade de selecionar características relevantes dentro de todo o espaço de subsequências. Os autores do MrSEQL fizeram duas adaptações do SEQL selecionar características. Uma utilizando o algoritmo SAX e a outra utilizando o SFA. O processo de seleção do algoritmo utiliza um algoritmo de gradiente descendente em conjunto a uma busca *branch and bound*, evitando assim testar todas as subsequências mas permitindo estimar um valor para todas elas de forma eficiente.

Todas as diferentes implementações apresentadas pelos autores foram testadas utilizando os dados do repositório UEA & UCR e comparadas entre si e entre os algoritmos do estado da arte. Os resultados apresentaram um algoritmo eficiente utilizando as características de multidomínio de forma simultânea e um algoritmo eficiente e interpretável utilizando as características de multirresolução de forma simultânea com o discretizador SAX.

3.2 MrSQM

O MrSQM é um algoritmo que estende o trabalho do MrSEQL mantendo suas três ideias principais: multirresolução, multidomínio e seleção de características em todo o espaço de subsequências. A principal diferença reside na terceira ideia. Para a execução dessa busca, o MrSQM propõe uma nova abordagem de seleção utilizando o teste *chi-square*. A abordagem limita as subsequências usando o seu prefixo. Dessa forma, caso o valor calculado de um prefixo seja baixo, todas as subsequências com aquele prefixo terão valores baixos. Isso evita calcular o valor de cada subsequência, porém fornecendo uma estimativa para todo o espaço de busca.

Em seu artigo, os autores do MrSQM apresentam diferentes estratégias para executar a seleção das subsequências, incluindo uma abordagem aleatória, e as compara entre si. A estratégia com os melhores resultados foi adotada no algoritmo e este foi então comparado com os algoritmos do estado da arte. A comparação utilizou a métricas de acurácia e o

tempo de execução. Os experimentos foram realizados utilizando a base de dados do repositório UEA & UCR com séries temporais univariadas.

A melhor estratégia observada foi uma abordagem híbrida que primeiro seleciona as subsequências aleatoriamente e em seguida executa a seleção supervisionada utilizando o algoritmo *chi-square*. A justificativa para esse resultado é que muitas muitas características correlacionadas são selecionadas. Isso ocorre devido à construção da características utilizando as subsequências.

A comparação do MrSQM revela uma melhora da acurácia na classificação, embora não estatisticamente significante, em relação aos algoritmos baseados em dicionários do estado da arte. Seus resultados ficaram próximos aos resultados do estado da arte considerando todos os tipos de algoritmos. Embora obtenha acurácias altas, o MrSQM consegue mantar um tempo de execução mais baixo do que a maioria dos algoritmos comparados.

3.3 Estado da Arte

O estado da arte dos algoritmos de TSC pode variar de acordo com a métrica de avaliação utilizada. Apesar disso, todos os algoritmos foram testados utilizando os dados do repositório UEA & UCR para a obtenção de resultados mais justos e precisos. A comparação normalmente é feita através da métrica de acurácia, mas neste trabalho também estão sendo utilizadas a eficiência e a área sobre a curva ROC (AUROC).

Ao decorrer desta seção estará descrito em mais detalhes o repositório UEA & UCR e o estado da arte dos algoritmos avaliados na eficiência das métricas de acurácia e de AUROC.

3.3.1 Repositório UEA & UCR

O repositório de classificação de séries temporais UEA & UCR foi introduzido por Keogh & Folias na Universidade de Califórnia em Riverside em 2002 e contava com apenas 16 *datasets* (KEOGH; KASETTY, 2003). Foi se expandindo gradualmente até que em 2015 recebeu uma expansão que aumentou de 45 para 85 *datasets* e em 2018 recebeu novamente uma expansão totalizando 128 *datasets* A última expansão significativa foi em outubro de 2018 adicionando 30 *datasets* com séries temporais multivariadas (BAGNALL et al., 2018) com o intuito de aprimorar os resultados da classificação da mesma forma como ocorreu para a classificação de séries temporais univariadas.

A evolução do repositório UEA & UCR não atendeu apenas aos propósitos de seus autores e contribuintes. Ela ocorreu atendendo também às críticas da comunidade, tanto as formais (HU; CHEN; KEOGH, 2016) quanto as informais. Consequentemente, o repositório se tornou cada vez mais importante e proveitoso para os pesquisadores da área de processamento de séries temporais, principalmente para a tarefa de classificação. Houve

uma convergência das pesquisas na parte dos dados mantendo toda a liberdade de escopo e objetivo. Isso permitiu comparar mais soluções para os diversos problemas a cerca de séries temporais. Pelo menos 1000 artigos publicados podem ser contados utilizando pelo menos um *dataset* do repositório (DAU et al., 2019), evoluindo rapidamente as técnicas e soluções para a tarefa de TSC.

O repositório UEA & UCR está hospedado no site *www.timeseriesclassification.com* contendo os dados de séries temporais bem como os resultados da classificação de muitos algoritmos. Os resultados disponíveis levam certo tempo para serem atualizados com as pesquisas mais recentes, como em (MIDDLEHURST et al., 2021), porém os resultados de grandes experimentos como (BAGNALL et al., 2017; RUIZ et al., 2021) estão disponíveis.

Os dados dispõem de uma separação de treino e teste para permitir a reprodutibilidade exata dos experimentos e comparar de forma mais precisa os algoritmos TSC (DAU et al., 2019). Uma abordagem para evitar variações de amostragem para separação dos dados que também poderia ser resolvida por meio da exposição dos parâmetros e do código utilizado. Contudo, este último é ainda uma prática em crescimento na comunidade científica.

3.3.2 Resultados

Os resultados disponíveis mais recentes em (BAGNALL; LINES; KEOGH, 2021) para a classificação de séries temporais estão apresentados na Figura 2. A figura considera as métricas AUROC e acurácia em relação ao tempo de treinamento de 14 classificadores de TSC. Os resultados apresentados foram obtidos na classificação de 71 *datasets* do repositório UCR & UEA. A mesma base de dados utilizado neste trabalho (Seção 4.2.3). A figura é dividida em dois gráficos *a*) e *b*) onde cada uma apresenta as pontuação AUROC e acurácia, respectivamente.

Os valores de cada classificador são as médias de cada um na classificação de todos os *datasets*. O eixo **x** apresenta a média de tempo de treinamento em segundos e o eixo **y** apresenta a média da pontuação com sua respectiva métrica em porcentagem. As pontuações estão em porcentagem para melhor visualização de outros gráficos dos experimentos (Seção 5). Seus valores originais variam entre $[0, 1]$.

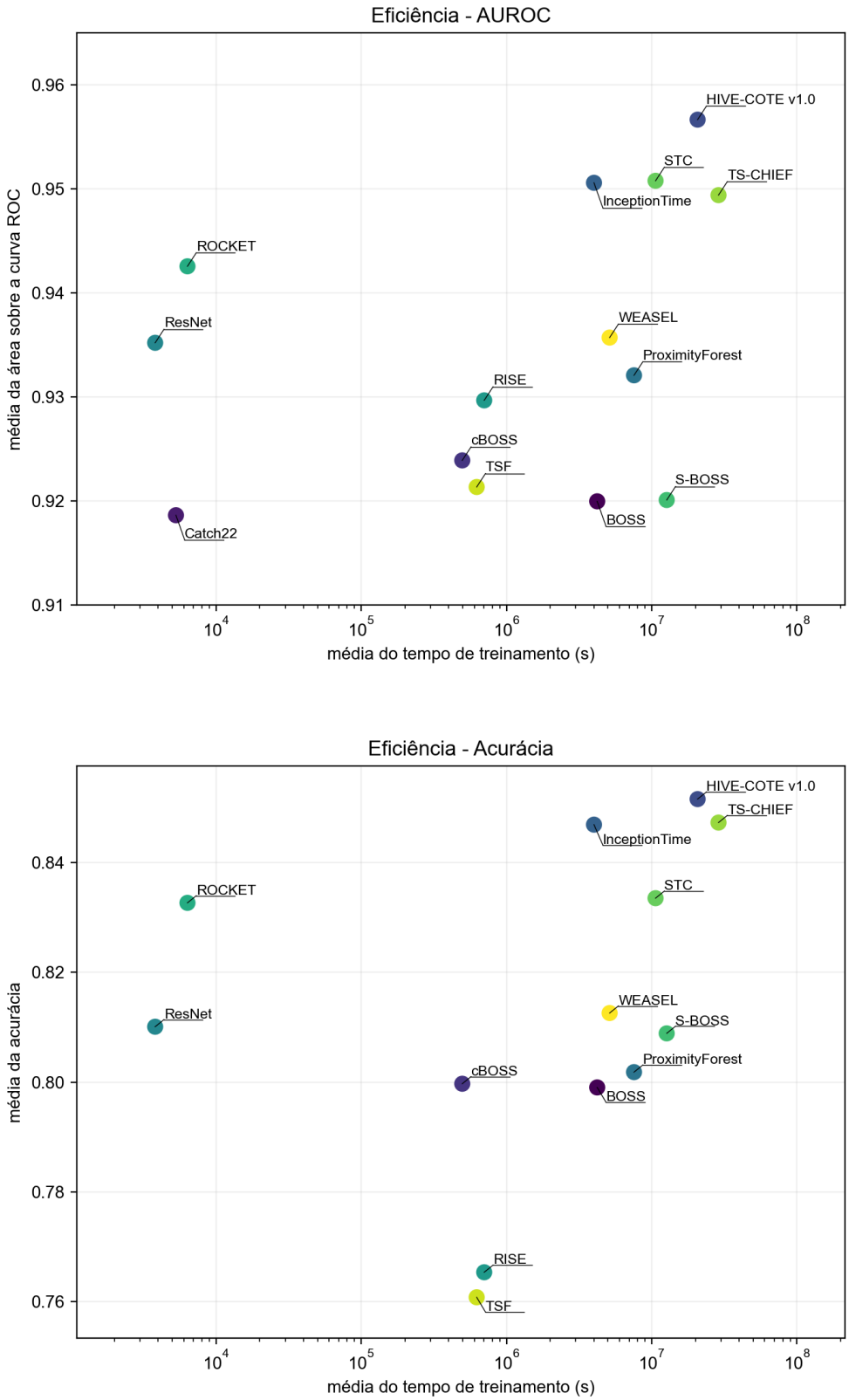

Figura 2 – Gráficos de dispersão dos valores AUROC e acurácia pelo tempo de treinamento dos algoritmos do estado da arte de 2020 na classificação de 71 conjuntos de séries temporais univariadas com tamanhos fixos em cada conjunto. Os resultados foram retirados do repositório UEA & UCR (BAGNALL; LINES; KEOGH, 2021), em 09/2021.

Capítulo 4

Proposta

Este trabalho propõe um novo algoritmo de extração e seleção de características baseadas em dicionário com foco na classificação eficiente de séries temporais denominado 4T. Seu objetivo é solucionar problemas que necessitam de uma solução que seja rápida e acurada. O algoritmo 4T foi construído a partir da composição de quatro técnicas bem sucedidas comumente encontradas na literatura. Os detalhes dessa composição são encontrados ao longo deste capítulo juntamente com três ferramentas que foram desenvolvidas para a realização dos experimentos que auxiliaram na construção e na avaliação do algoritmo 4T. As três ferramentas são: uma função para calcular a eficiência, a implementação de algoritmos chamados de variantes para a identificação de interações entre as técnicas de extração e uma divisão dos dados em duas bases para a condução dos experimentos.

4.1 Algoritmo 4T

Utilizando quatro técnicas de extração e seleção, foi proposto um novo algoritmo, denominado 4T, com o intuito de obter resultados mais eficientes conforme a Equação 6 na classificação de séries temporais por meio de características baseadas em dicionários. O 4T é um algoritmo baseado em dicionário que adota a estrutura-base (Figura 1) de maneira similar aos algoritmos ROCKET, o WEASEL, o MUSE e o MrSEQL. Uma abordagem comum na literatura que é composta por uma etapa de extração e seleção de características seguida por uma etapa de classificação.

A etapa de classificação do algoritmo 4T consiste no treinamento e na predição dos dados usando um classificador simples que seja eficiente mesmo com um número alto de características mantendo boa acurácia. Recordando que um classificador simples é aquele que desconsidera a ordenação das características dos dados. Um classificador frequentemente utilizado é o *logistic regression*, como foi o caso de todos os algoritmos com a estrutura-base analisados no Capítulo 3. Contudo o *logistic regression* foi trocado pela floresta aleatória, pois este se mostrou ser o mais eficiente dentre os classificadores comparados. Os resultados desta comparação são apresentados na Tabela 5. A etapa de

seleção de características remove as características irrelevantes dos dados para reduzir a dimensionalidade e aumentar a pontuação do modelo. Para isso o 4T faz uso do algoritmo *chi-square* que executa um processo supervisionado para avaliar e ranquear as características dos dados. A escolha do algoritmo seguiu as recomendações da literatura e uma análise mais profunda com a comparação de outros seletores foi deixada para trabalhos futuros.

A etapa de extração de características é composta pelas técnicas de extração e por um processo de discretização. O processo de discretização é realizado pelos algoritmos SAX e SFA que efetivamente transformam as séries em objetos com características baseadas em dicionários. Dessa forma, as técnicas de extração atuam de forma indireta na extração das características. Contudo elas determinam quais as resoluções serão passadas como parâmetro do discretizador, quais os algoritmos utilizados e como são representadas as características discretizadas. Foram selecionadas quatro técnicas para compor o algoritmo proposto: multirresolução, multidomínio, representação *ngram* e *ensemble*. Todas elas descritas na Seção 2.3.

A implementação do 4T seguiu o caminho mais fácil para compor todas as técnicas. Para isso foi feito um processo incremental das técnicas, incluindo uma a uma, descrito posteriormente na Seção 4.2.2. O resultado final desse processo pode ser definido como um *ensemble* de seletores. Essa definição é representada pelos diagramas das Figuras 3, 4, 5 e 6. Os diagramas estão divididos entre as etapas de treinamento e de predição bem como entre o *ensemble 4T* e o seletor. A implementação de ambos algoritmos pode ser encontrada no arquivo *search_technique_Ensemble.py* dentro das pastas *source/technique* hospedadas no GitHub. O endereço da página é <https://github.com/marcio55afr/MasterDegreeWorkspace>.

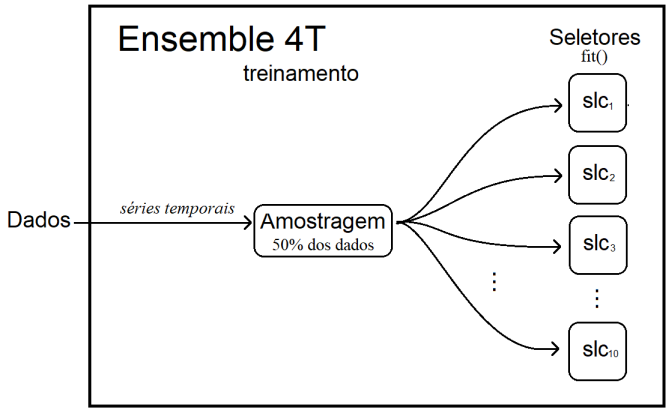

Figura 3 – Diagrama de fluxo do processo de treinamento do *ensemble* 4T.

A Figura 3 representa a etapa de treinamento do *ensemble* 4T. Ela ilustra o treinamento de 10 seletores a partir de uma amostragem de 50% dos dados de entrada que foi a amostragem com a maior eficiência dentre os valores testados (Tabela 11). O algoritmo

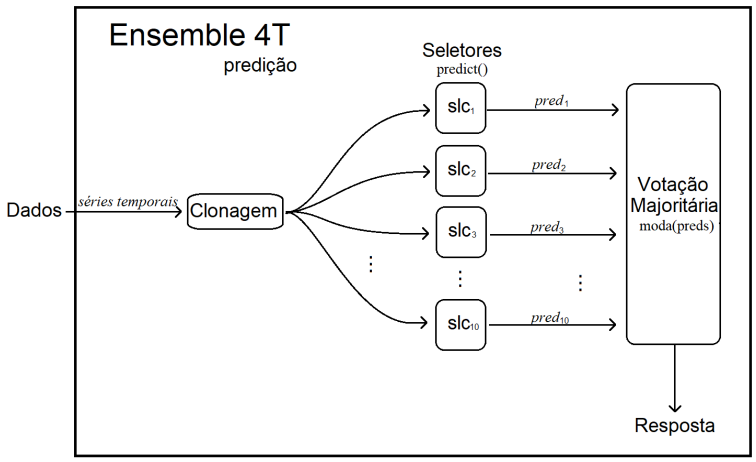

Figura 4 – Diagrama de fluxo do processo de predição do *ensemble* 4T.

salva os seletores treinados para serem utilizados posteriormente. A Figura 4 ilustra a entrada de dados na etapa de predição do *ensemble* 4T mostrando a passagem dos dados em todos os seletores para formar uma única resposta ao final. Cada seletor fornece a sua predição somando 10 predições independentes. Essas predições são então processadas em uma votação, utilizando a moda dos valores, que resulta na predição do *ensemble*.

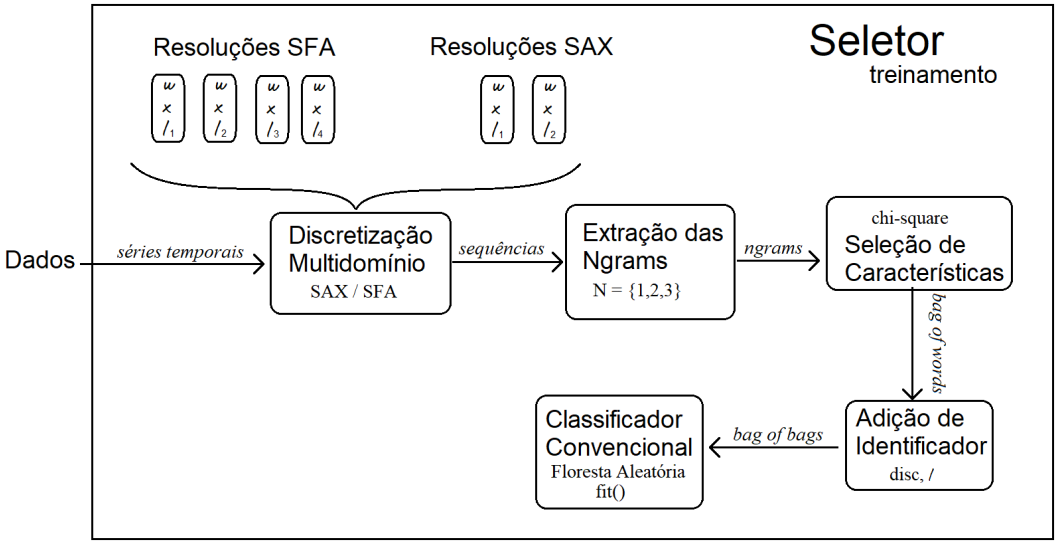

Figura 5 – Diagrama de fluxo do processo de treinamento do seletor que compõe o *ensemble 4T*.

O seletor presente nos diagramas também foi criado no processo de composição incremental, mas contendo três técnicas. Ele foi mantido na implementação do algoritmo 4T conforme a mesma ideia de seguir o caminho mais fácil para a composição das técnicas. Como o seletor possuía todas as três técnicas com exceção do *ensemble*, montar um conjunto de seletores foi a maneira mais fácil de implementar a quarta técnica e assim originar o algoritmo 4T.

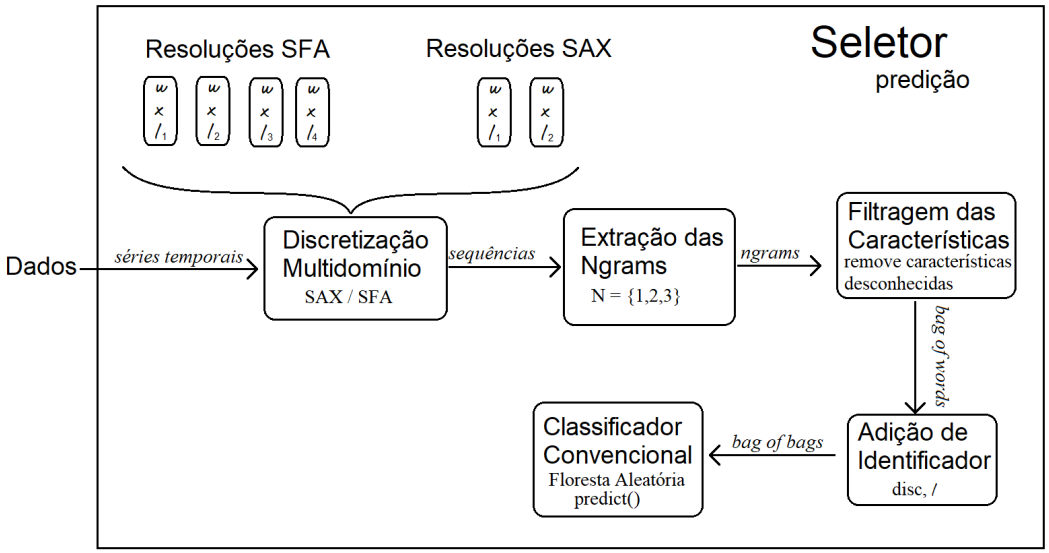

Figura 6 – Diagrama de fluxo da predição do algoritmo seletor.

A Figura 5 representa a etapa de treinamento do seletor. Nela é possível identificar as técnicas de multirresolução, multidomínio e representação *ngram* e o caminho que os dados percorrem através das técnicas. Nesse caminho há um laço de repetição implícito que faz com que os dados percorram todo o processo até a *adição de identificador* várias vezes. Cada repetição utilizando uma resolução diferente. O resultado da adição *bag of words* é então mantido em uma variável chamada *bag of bags* (BoB) que, quando completa, é enviada ao classificador simples.

A Figura 6 representa a predição do seletor ilustrando um processo similar ao treinamento. A diferença desta etapa é a filtragem, ao invés da seleção, de características. A filtragem é um processo para remover todas as características não selecionadas no treinamento e para adicionar as características faltantes definindo uma frequência igual 0. Tanto a remoção quanto a adição da filtragem são necessárias, pois o classificador simples exige que todos os objetos possuam exatamente as mesmas características, incluindo os objetos de treino e de predição.

4.2 Ferramentas

4.2.1 Função de Eficiência

A eficiência como foco desta pesquisa é métrica principal para a avaliação de resultados nos experimentos. Uma possível interpretação para a eficiência no contexto deste trabalho é o quanto de tempo é necessário para alcançar uma unidade da pontuação alcançada. Sendo assim, uma das formas mais simples para calcular a eficiência de uma classificação é por meio de uma fração entre a pontuação e o tempo gasto. Essa fração foi desenvolvida com o intuito de ordenar as classificações e definir qual foi mais eficiente. O resultado

desse desenvolvimento foi uma função denominada função de eficiência. A qual mensurou a eficiência dos algoritmos implementados neste trabalho e permitiu a sua comparação com os algoritmos do estado da arte.

Como mencionado, a função de eficiência foi definida como uma razão entre a pontuação e o tempo gasto de uma classificação. O valor da pontuação foi calculado utilizando as métricas AUROC e acurácia enquanto o tempo gasto foi medido considerando apenas o tempo de treinamento dos classificadores. O resultado dessa equação é representado pela Equação 1 onde a pontuação e o tempo são representados pelos símbolos v e t, respectivamente. O desenvolvimento de uma função de eficiência considerando o tempo de predição foi deixada para trabalhos futuros.

$$f_E(v,t) = \frac{v}{t} \tag{1}$$

Partindo desta primeira definição, a função de eficiência foi sendo ajustada para modelar o problema da eficiência de forma mais precisa. O primeiro ajuste foi considerar que o *score* de uma classificação não varia da mesma forma que o tempo de predição. Essa diferença ocorre pois quanto maior a pontuação de uma classificação, mais difícil é para aumentá-la. Consequentemente, mais tempo é gasto para alcançar tal objetivo. Exemplificando, um classificador provavelmente irá gastar mais tempo para aumentar a acurácia de 90% para 95% de uma classificação do que aumentar de 70% para 75%. Sendo assim, razão passou a ser uma função da pontuação sobre outra função do tempo definida como:

$$f_E(v,t) = \frac{f_{Ex}(v)}{f_L(t)} \tag{2}$$

Para uma variação linear da pontuação foi assumido uma variação exponencial do tempo. Assim, foi definido que a função da pontuação fosse exponencial (Equação 3) e a função do tempo fosse linear (Equação 4), mantendo a mesma proporção entre os valores da fração.

$$f_{Ex}(v) = a2^{bv} \tag{3}$$

$$f_L(t) = ct \tag{4}$$

As constantes a, b e c foram ajustadas de forma com que todos os resultados do estado da arte obtivessem eficiência menor ou igual a 1. Tornando os experimentos mais diretos e seus resultados mais fáceis de interpretar, bastando alcançar uma classificação com eficiência maior do que 1 para ser mais eficiente do que o estado da arte. As constantes a, b e c foram ajustadas para os valores 1, $\frac{1}{4}$ e 1, respectivamente, de forma empírica com o intuito de definir $f_E = 1$ à esquerda dos resultados da estado da arte. Guiando de forma mais intuitiva a construção das variante para um resultado de eficiência maior que

1. Com as constantes ajustadas e substituindo as Equações 3 e 4 na função de eficiência 2 obtém-se a função de eficiência na Equação 5.

$$f_E(v,t) = \frac{2^{v/4}}{t} \tag{5}$$

A última alteração feita na função de eficiência foi reduzir o valor v. Essa redução tem o intuito de evitar que pontuações muito baixas obtidas rapidamente sejam consideradas eficientes. Assim, soluções com resultados menores ou iguais à uma classificação aleatória são consideradas ineficientes com a eficiência $f_E = 0$. Esse ajuste foi feito utilizando o resultado médio do classificador aleatório obtidos nos mesmos *datasets* usados para a obtenção dos resultados do estado da arte e das implementações deste trabalho. O resultado médio considerou as métricas AUROC e a acurácia obtendo os respectivos valores 49.58 e 40.55. O resultado deste ajuste é mostrado pelas Equações 6 e 7.

$$f_E^{auroc}(v,t) = \frac{2^{(v-49.58)/4}}{t} \tag{6}$$

$$f_E^{acc}(v,t) = \frac{2^{(v-40.55)/4}}{t} \tag{7}$$

A função de eficiência final de cada métrica é mostrada na Figura 7. Nela estão as retas $f_E = 1$ de cada métrica, AUROC e acurácia, nos gráficos *a*) e *b*) separando os valores maiores que um, à sua esquerda, e os valores menores que 1, à sua direita. A figura também apresenta as médias das pontuações AUROC e acurácia e as médias do tempo de treinamento de cada algoritmo do estado da arte com exceção do MrSEQL e do MrSQM. Como a função de eficiência desconsidera a quantidade ou o tamanho das séries temporais de um *dataset*, ela será única em cada *dataset* ou conjunto de *datasets* utilizados. Esse problema é contornado utilizando sempre a mesma base de dados na comparação das classificações algoritmos. Dessa forma, a eficiência das classificações podem ser comparadas e ordenadas para definir o algoritmo mais eficiente para aquele dado conjunto de dados.

4.2.2 Variantes

A estratégia adotada para a composição das quatro técnicas de extração selecionadas foi seguir o caminho mais simples. Para isso, um processo incremental das técnicas foi desenvolvido partindo de um algoritmo baseado em dicionário com apenas a estrutura-base implementada e finalizando no algoritmo que inclui todas as técnicas. Embora as quatro técnicas tenham mostrado sucesso na literatura e em algoritmos como o ROCKET, WEASEL e MUSE, a nova composição que contém as quatro técnicas pode produzir resultados contrários aos esperados.

Pensando nisso, algumas das combinações das técnicas de extração foram implementadas, chamadas de variantes. A primeira delas foi a variante zero (V0), desenvolvida

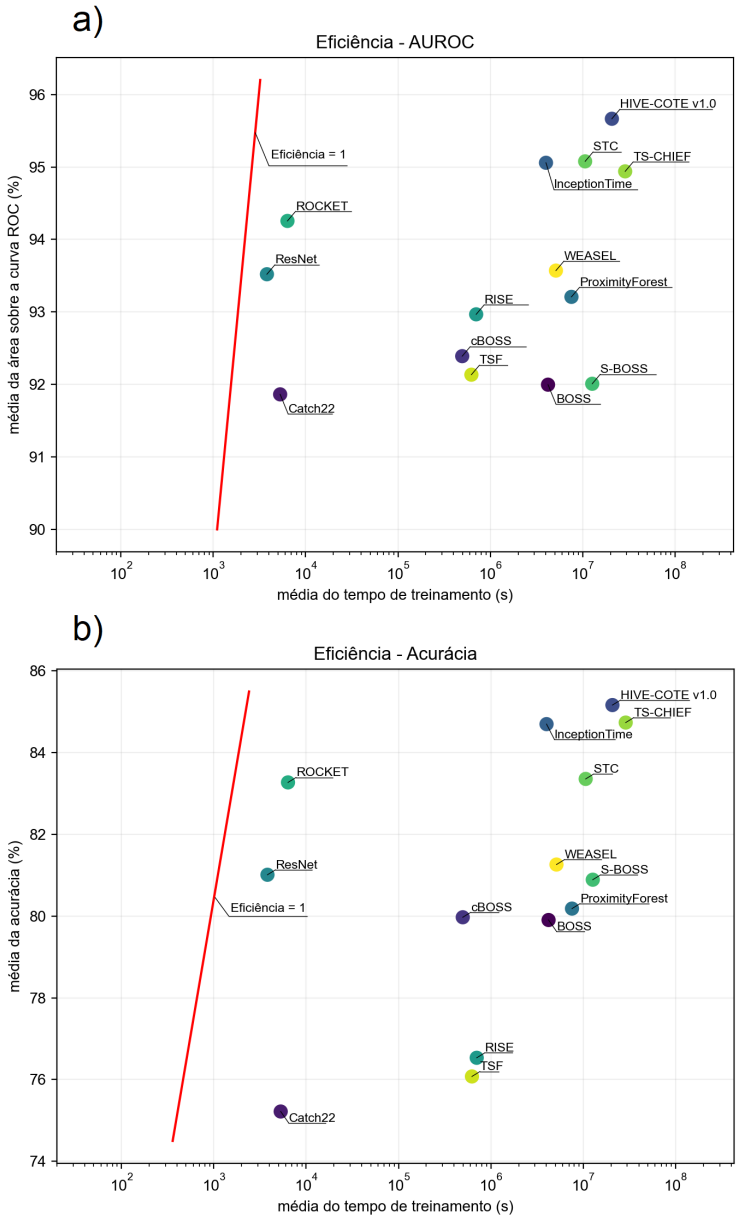

Figura 7 – Eficiência dos resultados disponíveis do estado da arte na classificação da base de dados F. As retas vermelhas mostram a eficiência igual a 1 considerando as métricas AUROC e acurácia em relação ao tempo de treinamento. As métricas AUROC e acurácia estão separadas entre os gráficos a) e b). As retas separam os resultados dos estado da arte à sua direita e os possíveis resultados mais eficientes à sua esquerda .

a partir de um modelo simples de classificação com nenhuma das técnicas de extração analisadas. Ela foi a base para as variantes subsequentes que se formaram a partir da adição de uma técnica à sua variante precedente, nomeadas de V1, V2, V3 e V4. A última delas, que se formou com a combinação de todas as quatro técnicas de extração, originou o próprio algoritmo 4T.

A adição das técnicas para a formação das variantes seguiu uma ordenação definida

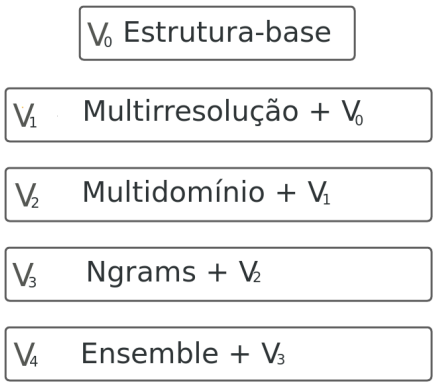

Figura 8 – Ordenação das técnica de extração para a composição e construção das variantes. Começando com a V0 com apenas a estrutura base e terminando com V4 com a composição de todas as quatro técnicas.

como mostra o diagrama da Figura 8. Começando pela técnica de multirresolução em seguida a de multidomínio, representação *ngram* e finalizando com a técnica de *ensemble*. Essa ordenação foi estabelecida de acordo com a dificuldade ou facilidade de implementação da combinação das técnicas. Exemplificando, a implementação da técnica de multidomínio em um algoritmo é facilitada quando este possui a técnica de multirresolução, assim a técnica de multirresolução é adicionada antes que a técnica de multidomínio.

A seguir cada uma das variantes, a começar pela V0, é explicada de forma mais detalhada.

V0 é a variante mais simples, não possui nenhuma das quatro técnicas de extração selecionadas na pesquisa. Porém é um algoritmo baseado em dicionário por utilizar a frequência das características extraídas pelos discretizadores SAX e SFA para classificar as séries temporais. Utilizando para isso a estrutura-base de extração e seleção.

Essa variante é representada pelo Algoritmo 1 que, de forma abrangente, executa uma busca pelo melhor tamanho de janela para a discretização das séries utilizando o SAX ou o SFA. A busca é feita por meio de uma validação cruzada que compara as janelas definidas em um conjunto inicial e determina aquela com a maior acurácia média como a melhor janela. Essa melhor janela é então utilizada novamente na discretização das séries para treinar um classificador que será utilizado nas predições futuras.

Tanto a validação cruzada quanto a classificação final utilizam o seletor de características *chi-square* e o algoritmo *logistic regression* como classificador. A seleção tem o intuito de remover características não discriminantes mantendo apenas aquelas com os maiores valores calculados pelo seletor provendo uma melhora da classificação em relação à utilização de todas as características.

De forma mais detalhada, a variante V0 primeiro define os parâmetros de discretização w, a e l (tamanho das palavras, do alfabeto e da janela). Seguindo as recomendações da literatura, os parâmetros w e a recebem valores fixos enquanto l varia entre os valores

Algoritmo 1 Variante 0 - função *fit*

```
1: Receive series, rotulos
2: Set w = 6                                          ▷ tamanho da palavra
3: Set α = 4                                          ▷ tamanho do alfabeto
4: Set n_l = 20                                       ▷ número de janelas
5: Set n_w = 200                                      ▷ Número de características selecionadas
6: clf ⇐ LogisticRegression()                         ▷ classificador simples
7: L ⇐ tamanho(series)
8: J ⇐ cria_janelas(w, L/2, n_l)                      ▷ n_l janelas equidistantes entre [w, L/2]
9: for all l in J do
10:     disc ⇐ SFA(w, α, l)
11:     seqs ⇐ disc.transform(series, rotulos)        ▷ séries discretizadas para sequências
12:     seqs_chi ⇐ χ²(n_w, seqs, rotulos)             ▷ Seleção de carac. chi-square
13:     r ⇐ validacao_cruzada(seqs_chi, rotulos, clf)
14:     if r for o maior then best_l ⇐ l
15:     end if
16: end for
17: disc ⇐ SFA(w, α, best_l)
18: seqs ⇐ disc.transform(series, rotulos)
19: seqs_chi ⇐ χ²(seqs, rotulos)
20: clf.fit(seqs_chi, rotulos)
```

$[w, ..., L/2]$, sendo L o tamanho da série temporal. A quantidade de valores para l é limitada a um máximo de 20 janelas que são distribuídas de forma equidistante entre o tamanhos w e $L/2$. Sendo o primeiro valor o tamanho da palavra, não podendo ter uma janela menor do que o tamanho da palavra produzida, e o último a metade do tamanho da série temporal.

Em seguida, cada resolução, contendo uma janela diferente, é avaliada pela acurácia média na classificação da base de treino utilizando a uma validação cruzada de 10-*folds*. Essa classificação utiliza as características extraídas pela resolução avaliada após uma etapa de seleção que mantém apenas as 200 primeiras características ordenadas de acordo com o seu valor calculado pelo algoritmo *chi-square*.

O classificador utilizado tanto para a validação cruzada quanto para o treinamento do modelo é o *logistic regression* com os parâmetros definidos com seus valores padrão com exceção de um, definido como *max_iter = 5000* para permitir a convergência do modelo o qual recebe 200 características para cada série.

V1 é a variante subsequente à variante base recebendo a primeira a técnica de extração, a multirresolução. Com essa técnica a variante passa a utilizar todas as resoluções simultaneamente fazendo com o que a validação cruzada deixe de ser necessária dispensando a busca pela melhor resolução, ou janela no caso da V0. Em decorrência disso, o tempo de treinamento de V1 é reduzido consideravelmente em relação à V0.

Para permitir a utilização simultânea de todas as resoluções e consequentemente de todas as características discretizadas, cada palavra de cada resolução recebe um identi-

ficador que indica a janela de origem daquela palavra. Assim, palavras provenientes de janelas diferentes, mesmo que idênticas, terão suas frequências contabilizadas em características diferentes. Isto é feito pois palavras de diferentes resoluções, provenientes de janelas de tamanhos diferentes, representam padrões distintos que, portanto, necessitam de características individuais. Essas características são construídas a partir da junção $l +$ palavra discretizada.

Após a identificação de todas as palavras, é feita a seleção de características removendo aquelas com os menores valores *chi-square* para cada tamanho de janela separadamente. Dessa forma a quantidade de características de cada resolução será a mesma com exceção das raras ocasiões em que a quantidade de características de uma resolução é menor do que a quantidade definida para ser selecionada. Nesse caso todas as características desta resolução serão selecionadas, finalizando o processo com uma quantidade menor de características do que com outras resoluções. Uma segunda abordagem para esse processo seria executar a seleção de forma simultânea considerando as características de todas as resoluções, porém esta se mostrou menos eficiente (Seção A.1 do Apêndice).

Com as características selecionadas a etapa de classificação é realizada. Para esta variante e as demais o algoritmo utilizado é a floresta aleatória, que foi definida após a comparação dos resultados de V1 utilizando diferentes algoritmos na classificação do conjunto S. A V1 que possui a primeira das técnicas de extração analisadas e que utiliza muitas características para a classificação comparou alguns classificadores e os resultados mostraram que a floresta aleatória foi a mais eficiente. Os resultados dessa comparação estão na Tabela 5 do Apêndice.

Algoritmo 2 Variante 1 - função *fit*

```
1: Receive series, rotulos
2: Set w = 6
3: Set α = 4
4: Set n_l = 10                          ▷ Número reduzido de janelas
5: Set n_w = 200
6: clf ⇐ RandomForest()                  ▷ Substituição do classificador convencional
7: L ⇐ tamanho(series)
8: J ⇐ cria_janelas(w, L/2, n_l)
9: bob ⇐ None
10: for all l in J do
11:     disc ⇐ SFA(w, α, l)
12:     seqs ⇐ disc.transform(series, rotulos)
13:     seqs_chi ⇐ χ²(seqs, rotulos)
14:     seqs_id ⇐ add_id(seqs_chi, l)    ▷ Adição de um identificador da janela
15:     bob ⇐ seqs_id
16: end for
17: clf.fit(bob, rotulos)
```

As três principais diferenças desta variante para a sua precedente são a inclusão da va-

riável *bag of bags* para armazenar todas as discretizações, a exclusão da validação cruzada e a adição de um identificador em cada palavra para diferenciar as palavras de diferentes resoluções.

V2 possui as técnicas de multirresolução e multidomínio. Essa variante recebeu a nova capacidade de extrair características no domínio do tempo e da frequência de forma simultânea utilizando ambos os algoritmos SAX e SFA. Mantendo a estrutura e escolha das janelas da variante V1, a variante dois se difere construindo dois conjuntos de janelas mas com os tamanhos no mesmo intervalo $[w, L/2]$ e com a mesma quantidade total de janelas.

Os dois conjuntos de janelas foram criados separadamente para cada algoritmo pois duas janelas de tamanhos iguais representam padrões diferentes nos domínios do tempo e da frequência. A quantidade total dos dois conjuntos permaneceu sendo de 10 janelas, como na variante anterior, pois dentre os valores testados foi o mais eficiente enquanto a proporção de janelas para cada domínio foi definida desigualmente. Foram definidas duas janelas para o algoritmo SAX e oito para o SFA, uma vez que as discretizações do SAX, para a base de dados utilizada, estava com um gasto maior de tempo em relação às discretizações feitas pelo SFA.

Após a criação do novo conjunto de janelas e a utilização de dois discretizadores, o restante da execução desta variante se mantém similar alterando apenas o identificador de cada palavra que passa a receber, além do tamanho da janela de origem, um valor que identifica o discretizador utilizado.

Algoritmo 3 Variante 2 - função *fit*

1: **Receive** *series, rotulos*
2: **Set** $w = 6$
3: **Set** $\alpha = 4$
4: **Set** $n_l^{sax} = 2$ ▷ Divisão das janelas entre o SAX e o SFA
5: **Set** $n_l^{sfa} = 8$ ▷ Proporção maior para o SFA
6: **Set** $n_w = 200$
7: $clf \Leftarrow RandomForest()$
8: $L \Leftarrow tamanho(series)$
9: $J_{sax} \Leftarrow cria\_janelas(w, L/2, n_l^{sax})$
10: $J_{sfa} \Leftarrow cria\_janelas(w, L/2, n_l^{sfa})$
11: $bob \Leftarrow None$
12: **for all** l in J_{sax}, J_{sfa} **do**
13: $disc \Leftarrow SAX/SFA(w, \alpha, l)$
14: $seqs \Leftarrow disc.transform(series, rotulos)$
15: $seqs_{chi} \Leftarrow \chi^2(n_w, seqs, rotulos)$
16: $seqs_{id} \Leftarrow add\_id(seqs_{chi}, l)$ ▷ Adição do identificador da janela
17: $seqs_{id} \Leftarrow add\_id(seqs_{id}, disc)$ ▷ Adição do identificador do discretizador
18: $bob \Leftarrow seqs_{id}$
19: **end for**
20: $clf.fit(bob, rotulos)$

Para simplificar o pseudocódigo, os dois tipos de janelas, SAX e SFA, foram condensados em um único laço de repetição. Na linha 12 do Algoritmo 3 l varia em J_{sax} e J_{sfa} definindo o discretizador da linha 13. Esse processo altera a adição do identificador de discretização porém mantendo a mesma *bag of bags* em ambas as execuções.

V3 é a próxima variante, ela recebeu a técnica de representação *ngram* a qual constrói novas palavras a partir da sequência de palavras provenientes da discretização da série. Uma sequência de n palavras constitui uma *n-gram* que pode ser contabilizada na discretização de uma série formando uma nova característica com a frequência dessa *n-gram*.

Sendo assim, a V3 utiliza as discretizações de ambos os algoritmos SAX e SFA e constrói as representações *n-grams* das palavras de cada série, calcula a sua frequência e compõe as novas características com os identificadores de janela, discretizador. Essa etapa é realizada imediatamente após a discretização antes mesmo da contagem da frequência de cada palavra. Para isso foi necessário alterar os algoritmos SAX e SFA para retornarem a sequência de palavras discretizadas de cada série ao invés do retorno padrão com a frequência de cada palavra em cada série.

Os valores de n foram definidos entre os valores {1,2,3} enquanto a seleção de características foi realizada para cada n individualmente. Para fazer essa separação não foi necessário adicionar um identificador de *ngram* para cada palavra, pois duas *ngrams* de n diferentes não podem ser iguais em decorrência de sua construção. Exemplificando, uma 2-*gram* "aabb bbcc" não pode ser igual a uma 3-*gram* "aabb bbcc acbc" devido aos espaços dentro da *ngram* adicionados na concatenação das palavras que a originou.

Essa abordagem de seleção foi comparada à seleção de todas as palavras de uma mesma resolução considerando todas as *ngrams*. Contudo esta última obteve resultados inferiores que podem estar relacionados com a maior frequência das palavras de 1-*gram*, que são as palavras originais sem a representação *ngram*, e com a forma com que o algoritmo *chi-square* calcula os valores de cada palavra. A comparação dos resultados está descrita no Apêndice (Seção A.1) enquanto a relação entre as abordagens de seleção foi deixada para trabalhos futuros.

Para implementar a técnica de *ngrams* foi adicionado mais um laço de repetição. Nele a variável i é incrementada percorrendo os valores de 1 até m que define a maior *ngram* extraída. Cada *ngram* passa por um processo de seleção de características separado e não é necessário adicionar um identificador pois nesse caso é impossível que duas palavras sejam iguais quando provenientes de n-grams diferentes.

V4 é a variante que representa a última combinação de técnicas e também o próprio algoritmo 4T. A V4 se diferencia da V3 com o acréscimo da técnica de *ensemble*. Essa técnica foi implementada utilizando amostragem dos dados e aleatoriedade das características no processo de treinamento de classificador que compõem o *ensemble*.

Os Algoritmos 5 e 6 apresentam o pseudocódigo do funcionamento da variante quatro.

Algoritmo 4 Variante 3 - função *fit*

Receive *series, rotulos*
Set $w = 4$ ▷ Tamanho da palavra reduzido
Set $\alpha = 4$
Set $n = 3$ ▷ n-gram máximo
Set $n_l^{sax} = 2$
Set $n_l^{sfa} = 8$
Set $n_w = 200$
$clf \Leftarrow RandomForest()$
$L \Leftarrow tamanho(series)$
$J_{sax} \Leftarrow cria\_janelas(w, L/2, n_l^{sax})$
$J_{sfa} \Leftarrow cria\_janelas(w, L/2, n_l^{sfa})$
$bob \Leftarrow []$
for all l in J_{sax}, J_{sfa} **do**
    $i = 1$
    while $i \leq n$ **do**
        $disc \Leftarrow SAX/SFA(w, \alpha, l)$
        $seqs \Leftarrow disc.transform(series, rotulos)$
        $ngrams \Leftarrow extrai\_ngrams(seqs, i)$
        $ngrams_{chi} \Leftarrow \chi^2(n_w, ngrams, rotulos)$
        $ngrams_{id} \Leftarrow add\_id(ngrams_{chi}, l)$
        $ngrams_{id} \Leftarrow add\_id(ngrams_{id}, disc)$
        $bob \Leftarrow ngrams_{id}$
        $i \Leftarrow i + 1$
    end while
end for
$clf.fit(bob, rotulos)$

De maneira sucinta, a V4 inicializa 10 seletores, executa uma amostragem de 50% dos dados, treina cada seletor com uma amostragem diferente e faz a predição com uma votação dos seletores treinados. O Algoritmo 6 apresenta a inicialização dos seletores, a amostragem dos dados e o treinamento de cada um dos seletores. O Algoritmo 5 executa todo o processo de discretização, extração e seleção das características. Ele possui uma quantidade reduzida de janelas em relação à V3 e a criação das janelas é aleatória. Ao contrário da abordagem de equidistância na definição do tamanho das janelas nas variantes precedentes.

4.2.3 Base de Dados

Os dados de séries temporais utilizados foram obtidos do repositório UEA & UCR. Ao todo são mais de 80 *datasets* de séries temporais univariadas, de tamanhos fixos e sem valores nulos. Destes foram criados dois conjuntos distintos, porém não disjuntos, para a execução de experimentos com objetivos diferentes. As duas bases de dados foram denominadas de S e F. Os *datasets* de cada base de dados podem ser encontrados

Algoritmo 5 Seletor - função *fit*

Receive *series, rotulos*
Set $w = 4$
Set $\alpha = 2$ ▷ Tamanho do alfabeto reduzido
Set $m = 3$
Set $n_l^{sax} = 1$ ▷ Número reduzido de janelas SAX
Set $n_l^{sfa} = 4$ ▷ Número reduzido de janelas SFA
Set $n_w = 200$
$clf \Leftarrow RandomForest()$
$L \Leftarrow tamanho(series)$
$J_{sax} \Leftarrow cria\_janelas\_aleatorias(w, L/2, n_l^{sax})$
$J_{sfa} \Leftarrow cria\_janelas\_aleatorias(w, L/2, n_l^{sfa})$
$bob \Leftarrow []$
for all l in J_{sax}, J_{sfa} **do**
    $i \Leftarrow 1$
    while $i \leq m$ **do**
        $disc \Leftarrow SAX/SFA(w, \alpha, l)$
        $seqs \Leftarrow disc.transform(series, rotulos)$
        $ngrams \Leftarrow extrai\_ngrams(seqs, i)$
        $ngrams_{chi} \Leftarrow \chi^2(n_w, ngrams, rotulos)$
        $ngrams_{id} \Leftarrow add\_id(ngrams_{chi}, l)$
        $ngrams_{id} \Leftarrow add\_id(ngrams_{id}, disc)$
        $bob \Leftarrow ngrams_{id}$
        $i \Leftarrow i + 1$
    end while
end for
$clf.fit(bob, rotulos)$

Algoritmo 6 Variante 4 - função *fit*

Receive *series, rotulos*
Set $n_{slc} = 10$ ▷ Número de seletores
Set $frac = 50\%$ ▷ Tamanho da amostragem
$SLCs \Leftarrow inicializa\_seletores(n_{slc})$
for slc in $SLCs$ **do**
    $amostra, rotulos_a \Leftarrow amostragem(series, rotulos, frac)$
    $slc.fit(amostra, rotulos_a)$
end for

no arquivo *config.py* dentro da pasta *source/experiments/UEA_Experiments/datasets/* hospedadas no GitHub.

A primeira base de dados S é a menor delas. Possui 19 *datasets* com o tamanho das séries limitados a 100. Tanto a quantidade quanto o tamanho das séries foram definidos dessa forma para que os experimentos executados utilizando essa base fossem mais rápidos.

A base de dados F é o maior conjunto contendo 71 *datasets*. É considerada a base que representa todos os dados com apenas alguns dos *datasets* removidos aleatoriamente. Essa base foi projetada para classificações mais precisas tendo um custo de tempo maior. Tanto a base F quanto a S estão definidas na implementação dos experimentos (Seção 5.2.1)

Capítulo 5

Experimentos e Análise dos Resultados

Os experimentos conduzidos neste trabalho se dividem em três. Cada um com um dos objetivos: maximizar a eficiência dos algoritmos implementados otimizando seus parâmetros, analisar as interações das técnicas utilizadas no algoritmo 4T e comparar os resultados do algoritmo 4T com os do estado da arte. Para isso, dois métodos de avaliação foram elaborados: a avaliação de parâmetros e a avaliação de classificadores. Tanto os métodos quanto os experimentos, bem como seus resultados e seu ambiente de teste, estão descritos detalhadamente entre as seções abaixo.

De forma resumida a otimização de parâmetros utilizou o método avaliação de parâmetros, enquanto os experimentos de análise das interações e de comparação dos resultados utilizaram o método de avaliação de classificadores. Todos os experimentos foram executados no mesmo ambiente e os resultados se mostraram promissores. Foi identificada uma interação não desejada V4, porém todas as variantes, com exceção da variante base, tiveram resultados mais eficientes que o estado da arte. A eficiência foi calculada usando tanto o score AUROC quanto a acurácia. Isso levou este trabalho a confirmar o potencial dos algoritmos baseados em dicionário apesar das várias tarefas que ainda precisam ser realizadas para a melhoria do algoritmo 4T. Na Seção 6 são citadas as tarefas para melhorias e outras sugestões para a continuidade desta pesquisa.

5.1 Métodos para a Avaliação

As três ferramentas desenvolvidas, função de eficiência, variantes e base de dados (Seção 4.2), foram utilizadas para elaborar os dois métodos: avaliação de parâmetros e avaliação de classificadores. Estes métodos permitiram a realização dos três experimentos que auxiliaram na construção do algoritmo 4T e avaliaram seus resultados. Ambos comparam os resultados de classificações com base na função de eficiência e definem o ranqueamento das classificações. Contudo, as duas principais diferenças entre eles são o objetivo e a base de dados de cada um.

A **avaliação de parâmetros** calcula a eficiência e compara os resultados provenientes

de um mesmo algoritmo mas com parâmetros diferentes. O objetivo é definir qual conjunto de parâmetros de um dado classificador é mais eficiente. A base de dados utilizada neste método é a base S, uma vez que a comparação dos parâmetros não necessita de uma precisão tão alta e a quantidade de testes e de parâmetros testados é alta. Dessa forma, o experimento de otimização se beneficia com a base de dados menor pela sua rapidez e pela seleção de parâmetros não enviesados. A eficiência calculada é a eficiência da média da pontuação e da média do tempo de treinamento para um conjunto de parâmetros em cada *dataset* da base de dados.

A **avaliação de classificadores** também calcula a eficiência, porém compara os resultados de classificadores diferentes. Os classificadores são comparados utilizando seus parâmetros *default* e é esperado que estes estejam otimizados para a classificação de séries temporais. Para uma comparação mais precisa, este método utiliza a base de dados F. A eficiência calculada é a eficiência da média das pontuações e dos tempos de treinamento de um classificador para cada *dataset*.

Na literatura o diagrama de diferença crítica é utilizado como métrica de comparação de algoritmos. Contudo, essa métrica não considera o custo de tempo e as implementações encontradas não foram consistentes umas com as outras. Portanto, a comparação de classificadores utilizando o diagrama de diferença crítica foi deixada para trabalhos futuros e substituída pela comparação direta das pontuações AUROC e acurácia. Apesar do foco ser a eficiência, a comparação das pontuações tem o intuito de manter uma comparação amplamente utilizada na literatura, facilitando o entendimento de seus valores, e de servir como um critério de decisão para a escolha de um classificador para um determinado problema.

5.2 Experimentos

De forma resumida, os experimentos foram divididos em dois processos de avaliações que se diferenciam no conjunto de dados, nas métricas de avaliação e no propósito de sua utilização, detalhados ao longo desta seção.

O intuito dessa separação é para que tanto a avaliação das relações entre as características de extração e seleção quanto a otimização dos parâmetros dos algoritmos implementados ocorressem de forma independente ao conjunto de dados utilizados.

Enquanto a avaliação de parâmetros tem como objetivo otimizar os parâmetros dos algoritmos de forma a maximizar a eficiência dos resultados na classificação de um conjunto de dados pequeno S. A avaliação de classificadores tem como objetivo avaliar e comparar os classificadores, previamente otimizados, de acordo com as métricas de eficiência, AUROC e acurácia na classificação de um conjunto de dados grande F.

5.2.1 Ambiente de testes

Os experimentos foram conduzidos em uma máquina pessoal de 8GB de RAM, processador Ryzen 5 de 3.60 GHz e sistema operacional Windows 10. As implementações foram feitas na linguagem Python utilizando alguns pacotes como o pandas, NumPy e o sktime. O pacote sktime (LÖNING et al., 2019; LONING et al., 2021) foi essencial para a implementação dos experimentos por automatizar as classificações e guardar todos os resultados. Os experimentos utilizaram uma constante como semente do gerador de número aleatórios para permitir a reprodutibilidade dos resultados. O código do algoritmo 4T, das ferramentas (função de eficiência, variantes e base de dados) e dos experimentos estão hospedados no site do GitHub acessível através do link <github.com/marcio55afr/MasterDegreeWorkspace>.

5.2.2 Otimização dos Parâmetros

A otimização dos parâmetros foi um experimento realizado para otimizar todos os algoritmos implementados durante o trabalho. Para isso, uma busca entre possíveis conjuntos de parâmetros foi elaborada utilizando o método de avaliação de parâmetros para determinar o melhor conjunto. Aproveitando a rapidez desta avaliação foi possível testar diversos conjuntos a partir da variação dos parâmetros de cada variante. Entretanto, alguns parâmetros foram fixados seguindo as recomendações da literatura a fim de reduzir a quantidade de conjuntos e acelerar o experimento. Seguindo a mesma ideia, o restante dos parâmetros variaram entre valores predefinidos. Isso ocorreu devido às limitações de tempo e de ambiente de execução.

Os parâmetros de todas as quatro variantes junto com os valores testados estão apresentados na Tabela 3 do Apêndice. Com o intuito de apresentar os resultados de forma direta e simplificada, a tabela 1 apresenta apenas o melhor conjunto de parâmetros para cada variante. Os conjuntos foram avaliados de acordo com a eficiência AUROC, porém outras métricas como a acurácia e o tempo de treinamento também são exibidas.

5.2.3 Análise das Interações

O experimento de análise das interações foi conduzido utilizando o método de avaliação de classificadores para identificar a presença de interações indesejadas. Dada que todas as variantes tiveram seus parâmetros otimizados, cada uma executou a classificação da base da dados F com o seu melhor conjunto e parâmetros para que este experimento comparasse seus resultados. Como as variantes foram construídas de forma incremental, uma variante difere-se da sua variante precedente com o acréscimo de um técnica. Sendo assim, caso a pontuação de classificação dessa variante sejam menores do que sua precedente, uma interação ruim é detectada. É considerado que essa interação ruim seja entre a nova técnica da variante testada com alguma outra que também compõe tal variante. Uma das

Tabela 1 – Melhores parâmetros para cada variante e seus resultados. Resultados obtidos na classificação do conjunto de dados menor (S).

Variante:	parâmetros	média (%) Acurácia	média (%) AUROC	média (s) treinamento	Eficiência Acurácia	Eficiência AUROC
V0:	SFA LR nw200 nl20	72,28	84,01	22,06	11,06	17,69
V1-clf:	SFA RF nw200 nl20	76,41	88,88	9,35	53,43	97,06
V1-SG:	SFA RF nw200 nl10	76,54	89,06	5,59	91,43	167,48
V1-kWS:	K10-max SFA RF nw200 nl20	76,86	89,03	9,81	55,07	94,89
V2:	SS RF nw200 nl2-8	76,05	88,65	18,72	25,07	46,52
V3-SU:	3gram SS RF nw200 nl2-8 w4 a2	79,06	89,05	17,86	44,23	52,33
V3-NR:	3gram SS RF nw200 nl2-8 w4	78,47	89,31	48,77	14,62	20,03
V4:	s10 frac.50 pc3 3gram SS RF nw200 nl1-4 a2	76,52	89,11	30,67	16,60	30,80

opções para evitar tal interação é a remoção da última técnica incluída nas variantes que é equivalente a remover a própria variante que inclui tal técnica. Remover uma variante, devido ao processo de sua construção, significa que as variantes subsequentes deixam de ter uma das técnicas que seria incluída pela variante removida.

A remoção das interações indesejadas, seja por meio da remoção de variantes ou de outra abordagem, foi deixada para trabalhos futuros. Neste trabalho é feito apenas a primeira etapa de identificação. Como mostram os resultados apresentados na Tabela 2, pelo menos uma interação não desejada foi identificada na variante 4. A interação é entre a técnica de *ensemble* e uma das outras três técnicas, podendo ser até com mais de uma das três técnicas. Recordando que as técnicas são bem sucedidas na literatura, é esperado que nenhuma delas tenham uma interação ruim com a estrutura-base ou com o seletor *chi-square*. Apesar disso, uma análise mais profunda é sugerida com a implementação individual de cada técnica analisando as interações com os algoritmos de discretização, seleção e classificação. É possível que a interação venha da forma como foram implementadas as técnicas e não apenas do seu uso em conjunto. Diferentes implementações das técnicas também é sugerida para trabalhos futuros para uma investigação mais precisa.

5.2.4 Comparação dos Classificadores

A comparação de classificadores foi um experimento realizado para comparar os resultados das variantes e, consequentemente, do algoritmo 4T em relação ao estado da arte. A comparação utilizou o método de avaliação de classificadores e comparou todos os resultados de todos os algoritmos, com exceção do MrSQM. Os resultado do MrSQM não estavam hospedados no repositório UEA & UCR e o seu código não estava disponível na biblioteca do sktime. Além dos algoritmos implementados neste trabalho, os algoritmos que foram comparados são aqueles citados na Seção 3.3.1. O algoritmo MrSEQL também

Tabela 2 – Análise de interação entre as técnicas de extração. Resultado obtido na classificação do conjunto de dados F.

Variante	média (%) Acurácia	média (%) AUROC	média (s) treinamento	Eficiência Acurácia	Eficiência AUROC
V0	62,24	73,57	117,71	0,36	0,54
V1-kWS	77,06	84,65	15,26	36,62	28,57
V1-SG	77,92	85,46	7,69	84,34	65,25
V2	81,14	91,39	100,52	11,27	13,94
V3-SU	83,43	92,52	71,75	23,51	23,76
V3-NR	85,07	94,24	179,18	12,49	12,81
V4	70,38	83,53	73,93	2,38	4,85

não tinha seus resultados no repositório, contudo ele foi o único algoritmo da literatura que teve seus resultados obtidos localmente. O algoritmo MrSEQL disponível na biblioteca sktime rodou os mesmos experimentos realizados para a obtenção dos resultados das variantes e do algoritmo 4T.

Os resultados de todos esses algoritmos são apresentados na Figura 9. Ela apresenta as pontuações e o tempo de treinamento de cada algoritmo na classificação da base de dados F. As pontuações estão divididas entre os gráficos a) e b) ilustrando as métricas AUROC e acurácia, respectivamente. Os valores da pontuação variam entre [0,100] e representam a porcentagem da pontuação em relação ao seu valor original que varia entre [0,1]. O tempo de treinamento foi marcado em segundos e está representado no eixo x de ambos os gráficos. A reta vermelha representa a função de eficiência $f_E = 1$ dividindo o plano cartesiano em dois. Do lado esquerdo os valores de classificação com eficiência maior que 1 e, do lado direito da reta, as classificações com eficiência menor que 1.

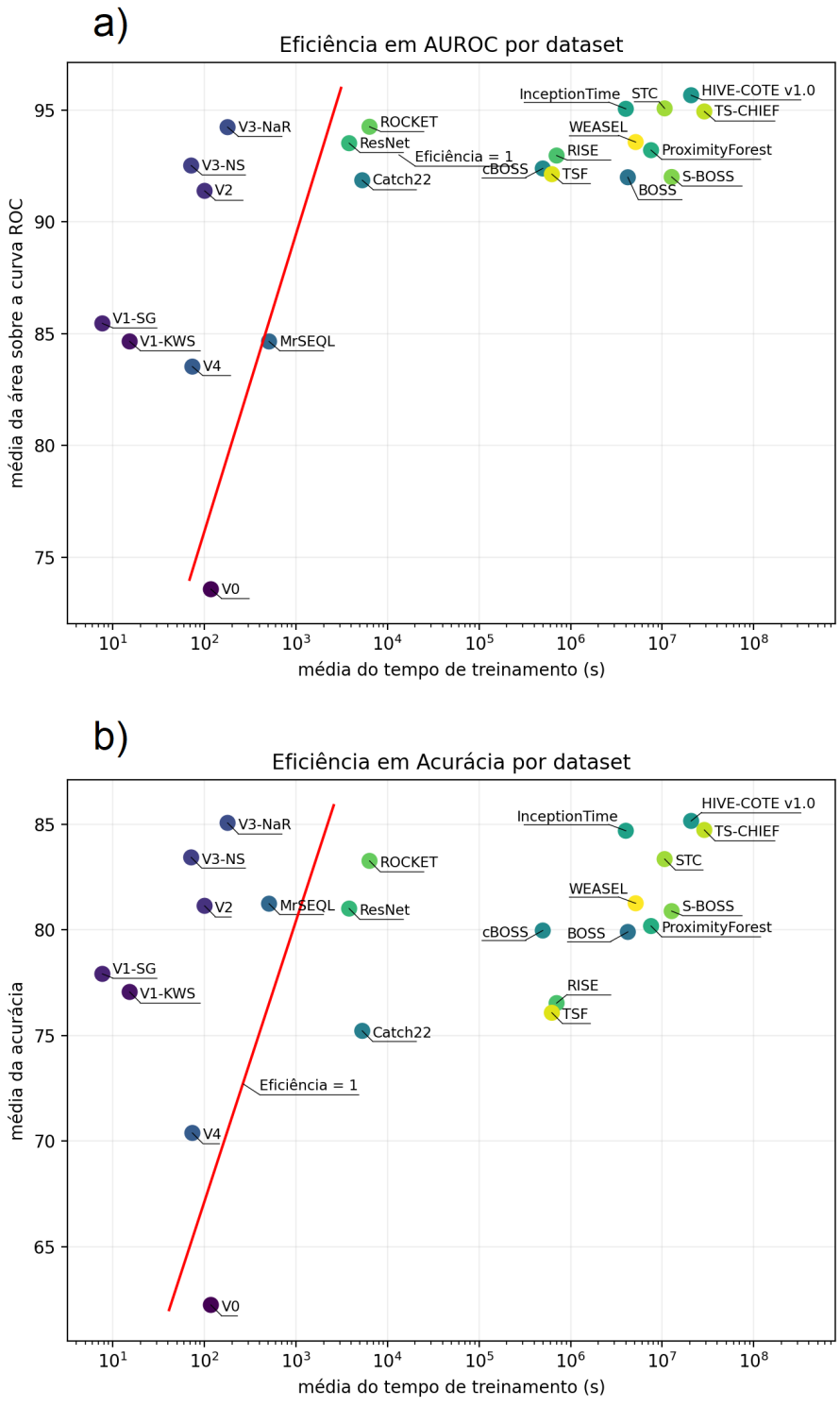

Figura 9 – Eficiência da média dos resultados de 22 algoritmos na classificação de 71 *datasets*. Os resultados estão divididos entre os gráficos a) e b) que representam as métricas AUROC e acurácia, respectivamente. A reta vermelha indica a eficiência igual a 1.

Capítulo 6

Conclusão

Esta pesquisa propôs um novo algoritmo de extração e seleção de características para a classificação eficiente de séries temporais desenvolvido a partir de quatro técnicas de extração da literatura. A proposta, chamada de algoritmo 4T, foi construída em um processo incremental criando em cada iteração o que chamamos de variante. Os resultados das variantes mostraram uma classificação com o tempo de treinamento 100 vezes mais rápido que o MrSEQL (algoritmo mais rápido do estado da arte) e com o *score* comparável até com o HIVE-COTE v1.0 (algoritmo de maior *score* do estado da arte). Neste último, a diferença do tempo de treinamento da variante mais acurada em relação do HIVE-COTE v1.0 foi de 100.000 vezes. A consequência destas classificações precisas e com baixo custo de tempo foi a obtenção de resultados mais eficientes que todo o estado da arte avaliado. Evidenciando assim o potencial dos algoritmos baseados em dicionário e a importância da estrutura do classificador para a obtenção de um resultado específico que, neste caso, foi a eficiência.

Apesar dos bons resultados, há melhorias para ser feitas e interações não desejadas para ser removidas tanto nas variantes quanto no próprio algoritmo 4T. Tendo como o próximo foco a melhoria das ferramentas e do algoritmo 4T, é possível que os resultados sejam ainda melhores. Podendo alcançar pontuações ou eficiências ainda maiores.

Mencionando as ferramentas, todas as três podem ser aprimoradas. A função de eficiência pode incluir em seu cálculo de forma gradual o tempo de predição, a quantidade de séries temporais e o tamanho das séries para uma dada classificação. Tornando o valor da eficiência independente do *dataset* que a classificação ocorreu e dependente apenas do algoritmo utilizado. As variantes podem ser construídas utilizando um algoritmo de busca para evitar as interações indesejadas ou ainda testar novas formas para a implementação e composição das técnicas de extração. As bases de dados, como a última ferramenta, podem ser aprimoradas incluindo séries temporais de tamanho variável ou multidimensionais.

Quanto às melhorias da solução proposta, duas podem ser citadas. A primeira é a discretização no domínio do tempo realizada pelo algoritmo SAX. Essa tarefa consumiu um

tempo muito maior que a discretização no domínio da frequência mesmo não possuindo nenhum processo de treinamento para a discretização. A segunda melhoria é a extensão da solução para problemas mais complexos como a classificação de séries temporais multivariadas.

Referências

ABANDA, A.; MORI, U.; LOZANO, J. A. A review on distance based time series classification. **Data Mining and Knowledge Discovery**, Springer, v. 33, n. 2, p. 378–412, 2019. Disponível em: <https://doi.org/10.1007/s10618-018-0596-4>.

ATLURI, G.; KARPATNE, A.; KUMAR, V. Spatio-temporal data mining: A survey of problems and methods. **ACM Computing Surveys (CSUR)**, ACM New York, NY, USA, v. 51, n. 4, p. 1–41, 2018. Disponível em: <https://doi.org/10.1145/3161602>.

BAGNALL, A.; DAU, H. A.; LINES, J.; FLYNN, M.; LARGE, J.; BOSTROM, A.; SOUTHAM, P.; KEOGH, E. The uea multivariate time series classification archive, 2018. **arXiv preprint arXiv:1811.00075**, 2018. Disponível em: <https://arxiv.org/abs/1811.00075>.

BAGNALL, A.; LINES, J.; BOSTROM, A.; LARGE, J.; KEOGH, E. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. **Data mining and knowledge discovery**, Springer, v. 31, n. 3, p. 606–660, 2017. Disponível em: <https://doi.org/10.1007/s10618-016-0483-9>.

BAGNALL, A.; LINES, J.; HILLS, J.; BOSTROM, A. Time-series classification with cote: the collective of transformation-based ensembles. **IEEE Transactions on Knowledge and Data Engineering**, IEEE, v. 27, n. 9, p. 2522–2535, 2015. Disponível em: <https://doi.org/10.1109/TKDE.2015.2416723>.

BAGNALL, A.; LINES, J.; KEOGH, E. **UEA & UCR Time Series Classification Repository**. 2021. Disponível em: <http://www.timeseriesclassification.com/>.

BENGIO, Y.; COURVILLE, A.; VINCENT, P. Representation learning: A review and new perspectives. **IEEE transactions on pattern analysis and machine intelligence**, IEEE, v. 35, n. 8, p. 1798–1828, 2013. Disponível em: <https://doi.org/10.1109/TPAMI.2013.50>.

DAU, H. A.; BAGNALL, A.; KAMGAR, K.; YEH, C.-C. M.; ZHU, Y.; GHARGHABI, S.; RATANAMAHATANA, C. A.; KEOGH, E. The ucr time series archive. **IEEE/CAA Journal of Automatica Sinica**, IEEE, v. 6, n. 6, p. 1293–1305, 2019. Disponível em: <https://doi.org/10.1109/JAS.2019.1911747>.

DEMPSTER, A.; PETITJEAN, F.; WEBB, G. I. Rocket: exceptionally fast and accurate time series classification using random convolutional kernels. **Data Mining**

and Knowledge Discovery, Springer, v. 34, n. 5, p. 1454–1495, 2020. Disponível em: <https://doi.org/10.1007/s10618-020-00701-z>.

DING, H.; TRAJCEVSKI, G.; SCHEUERMANN, P.; WANG, X.; KEOGH, E. Querying and mining of time series data: experimental comparison of representations and distance measures. **Proceedings of the VLDB Endowment**, VLDB Endowment, v. 1, n. 2, p. 1542–1552, 2008. Disponível em: <https://doi.org/10.14778/1454159.1454226>.

ESLING, P.; AGON, C. Time-series data mining. **ACM Computing Surveys (CSUR)**, ACM New York, NY, USA, v. 45, n. 1, p. 1–34, 2012. Disponível em: <https://doi.org/10.1145/2379776.2379788>.

FAWAZ, H. I.; FORESTIER, G.; WEBER, J.; IDOUMGHAR, L.; MULLER, P.-A. Deep learning for time series classification: a review. **Data Mining and Knowledge Discovery**, Springer, v. 33, n. 4, p. 917–963, 2019. Disponível em: <https://doi.org/10.1007/s10618-019-00619-1>.

FAWAZ, H. I.; LUCAS, B.; FORESTIER, G.; PELLETIER, C.; SCHMIDT, D. F.; WEBER, J.; WEBB, G. I.; IDOUMGHAR, L.; MULLER, P.-A.; PETITJEAN, F. Inceptiontime: Finding alexnet for time series classification. **Data Mining and Knowledge Discovery**, Springer, v. 34, n. 6, p. 1936–1962, 2020. Disponível em: <https://doi.org/10.1007/s10618-020-00710-y>.

FULLER, W. A. **Introduction to statistical time series**. John Wiley & Sons, 2009. v. 428. Disponível em: <https://doi.org/10.1002/9780470316917>.

GEURTS, P. Pattern extraction for time series classification. In: SPRINGER. **European conference on principles of data mining and knowledge discovery**. 2001. p. 115–127. Disponível em: <https://doi.org/10.1007/3-540-44794-6_10>.

GOLDBERGER, A.; AMARAL, L.; GLASS, L.; HAUSDORFF, J.; IVANOV, P. C.; MARK, R.; MIETUS, J.; MOODY, G.; PENG, C.; STANLEY, H. Components of a new research resource for complex physiologic signals. **PhysioBank, PhysioToolkit, and Physionet**, 2000. Disponível em: <https://doi.org/10.1161/01.CIR.101.23.e215>.

HU, B.; CHEN, Y.; KEOGH, E. Classification of streaming time series under more realistic assumptions. **Data mining and knowledge discovery**, Springer, v. 30, n. 2, p. 403–437, 2016. Disponível em: <https://doi.org/10.1007/s10618-015-0415-0>.

JI, C.; ZHAO, C.; PAN, L.; LIU, S.; YANG, C.; MENG, X. A just-in-time shapelet selection service for online time series classification. **Computer Networks**, Elsevier, v. 157, p. 89–98, 2019. Disponível em: <https://doi.org/10.1016/j.comnet.2019.04.020>.

JR, T. A.; BAHMANYAR, A.; BISWAS, R.; DAI, M.; GALBANY, L.; HLOŽEK, R.; ISHIDA, E. E.; JHA, S. W.; JONES, D. O.; KESSLER, R. et al. The photometric lsst astronomical time-series classification challenge (plasticc): Data set. **arXiv**, 2018. Disponível em: <https://arxiv.org/abs/1810.00001>.

KATE, R. J. Using dynamic time warping distances as features for improved time series classification. **Data Mining and Knowledge Discovery**, Springer, v. 30, n. 2, p. 283–312, 2016. Disponível em: <https://doi.org/10.1007/s10618-015-0418-x>.

KEOGH, E.; CHAKRABARTI, K.; PAZZANI, M.; MEHROTRA, S. Dimensionality reduction for fast similarity search in large time series databases. **Knowledge and information Systems**, Springer, v. 3, n. 3, p. 263–286, 2001. Disponível em: <https://doi.org/10.1007/PL00011669>.

KEOGH, E.; KASETTY, S. On the need for time series data mining benchmarks: a survey and empirical demonstration. **Data Mining and knowledge discovery**, Springer, v. 7, n. 4, p. 349–371, 2003. Disponível em: <https://doi.org/10.1023/A:1024988512476>.

LARGE, J.; BAGNALL, A.; MALINOWSKI, S.; TAVENARD, R. On time series classification with dictionary-based classifiers. **Intelligent Data Analysis**, IOS Press, v. 23, n. 5, p. 1073–1089, 2019. Disponível em: <https://doi.org/10.3233/IDA-184333>.

LIN, J.; KEOGH, E.; WEI, L.; LONARDI, S. Experiencing sax: a novel symbolic representation of time series. **Data Mining and knowledge discovery**, Springer, v. 15, n. 2, p. 107–144, 2007. Disponível em: <https://doi.org/10.1007/s10618-007-0064-z>.

LINES, J.; TAYLOR, S.; BAGNALL, A. Time series classification with hive-cote: The hierarchical vote collective of transformation-based ensembles. **ACM Transactions on Knowledge Discovery from Data**, v. 12, n. 5, 2018. Disponível em: <https://doi.org/10.1145/3182382>.

LÖNING, M.; BAGNALL, A.; GANESH, S.; KAZAKOV, V.; LINES, J.; KIRÁLY, F. J. sktime: A unified interface for machine learning with time series. **arXiv**, 2019. Disponível em: <https://arxiv.org/abs/1909.07872>.

LONING, M.; BAGNALL, T.; MIDDLEHURST, M.; GANESH, S.; OASTLER, G.; KIRALY, F.; LINES, J.; VIKTORKAZ; WALTER, M.; RNKUHNS; ROCKENSCHAUB, P.; OWOSENI, T.; JESELLIER; BULATOVA, G.; AND LOVKUSH MEYER, S. M.; AIDENRUSHBROOKE; SCHAFER, P.; OLESKIEWICZ; XUAN, Y. X.; ANSARI, A.; HONGYI; SAKSHI, A.; ARELO; LARGE, J.; ORDUZ, J.; KUMAR, B.; EDWARD, M. G.; WANG, A.; PONG, J. **alan-turing-institute/sktime: v0.8.2**. Zenodo, 2021. Disponível em: <https://doi.org/10.5281/zenodo.5610006>.

LUCAS, B.; SHIFAZ, A.; PELLETIER, C.; O'NEILL, L.; ZAIDI, N.; GOETHALS, B.; PETITJEAN, F.; WEBB, G. I. Proximity forest: an effective and scalable distance-based classifier for time series. **Data Mining and Knowledge Discovery**, Springer, v. 33, n. 3, p. 607–635, 2019. Disponível em: <https://doi.org/10.1007/s10618-019-00617-3>.

MIDDLEHURST, M.; LARGE, J.; FLYNN, M.; LINES, J.; BOSTROM, A.; BAGNALL, A. Hive-cote 2.0: a new meta ensemble for time series classification. **arXiv preprint arXiv:2104.07551**, 2021. Disponível em: <https://doi.org/10.1007/s10994-021-06057-9>.

NGUYEN, T. L.; GSPONER, S.; ILIE, I.; O'REILLY, M.; IFRIM, G. Interpretable time series classification using linear models and multi-resolution multi-domain symbolic representations. **Data mining and knowledge discovery**, Springer, v. 33, n. 4, p. 1183–1222, 2019. Disponível em: <https://doi.org/10.1007/s10618-019-00633-3>.

NGUYEN, T. L.; IFRIM, G. Mrsqm: Fast time series classification with symbolic representations. **arXiv**, 2021. Disponível em: <https://arxiv.org/abs/2109.01036>.

OPPENHEIM, A. V.; SCHAFER, R. W. Digital signal processing(book). **Research supported by the Massachusetts Institute of Technology, Bell Telephone Laboratories, and Guggenheim Foundation. Englewood Cliffs, N. J., Prentice-Hall, Inc., 1975. 598 p**, 1975. Disponível em: <https://doi.org/10.1002/piuz19760.07041>.

RATANAMAHATANA, C. A.; KEOGH, E. Three myths about dynamic time warping data mining. In: SIAM. **Proceedings of the 2005 SIAM international conference on data mining**. 2005. p. 506–510. Disponível em: <https://doi.org/10.1137/1.9781611972757.50>.

RUIZ, A. P.; FLYNN, M.; LARGE, J.; MIDDLEHURST, M.; BAGNALL, A. The great multivariate time series classification bake off: a review and experimental evaluation of recent algorithmic advances. **Data Mining and Knowledge Discovery**, Springer, v. 35, n. 2, p. 401–449, 2021. Disponível em: <https://doi.org/10.1007/s10618-020-00727-3>.

SCHÄFER, P.; HÖGQVIST, M. Sfa: a symbolic fourier approximation and index for similarity search in high dimensional datasets. In: **Proceedings of the 15th international conference on extending database technology**. [s.n.], 2012. p. 516–527. Disponível em: <https://doi.org/10.1145/2247596.2247656>.

SCHÄFER, P.; LESER, U. Fast and accurate time series classification with weasel. In: **Proceedings of the 2017 ACM on Conference on Information and Knowledge Management**. [s.n.], 2017. p. 637–646. Disponível em: <https://doi.org/10.1145/3132847.3132980>.

SHIFAZ, A.; PELLETIER, C.; PETITJEAN, F.; WEBB, G. I. Ts-chief: a scalable and accurate forest algorithm for time series classification. **Data Mining and Knowledge Discovery**, Springer, v. 34, n. 3, p. 742–775, 2020. Disponível em: <https://doi.org/10.1007/s10618-020-00679-8>.

SILVA, D. F.; GIUSTI, R.; KEOGH, E.; BATISTA, G. E. Speeding up similarity search under dynamic time warping by pruning unpromising alignments. **Data Mining and Knowledge Discovery**, Springer, v. 32, n. 4, p. 988–1016, 2018. Disponível em: <https://doi.org/10.1007/s10618-018-0557-y>.

SONG, H.; LI, G. Tourism demand modelling and forecasting—a review of recent research. **Tourism management**, Elsevier, v. 29, n. 2, p. 203–220, 2008. Disponível em: <https://doi.org/10.1016/j.tourman.2007.07.016>.

WEI, L.; KEOGH, E. Semi-supervised time series classification. In: **Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining**. [s.n.], 2006. p. 748–753. Disponível em: <https://doi.org/10.1145/1150402.1150498>.

XING, Z.; PEI, J.; KEOGH, E. A brief survey on sequence classification. **ACM Sigkdd Explorations Newsletter**, ACM New York, NY, USA, v. 12, n. 1, p. 40–48, 2010. Disponível em: <https://doi.org/10.1145/1882471.1882478>.

YANG, Q.; WU, X. 10 challenging problems in data mining research. **International Journal of Information Technology & Decision Making**, World Scientific, v. 5, n. 04, p. 597–604, 2006. Disponível em: <https://doi.org/10.1142/S0219622006002258>.

YE, L.; KEOGH, E. Time series shapelets: a novel technique that allows accurate, interpretable and fast classification. **Data mining and knowledge discovery**, Springer, v. 22, n. 1, p. 149–182, 2011. Disponível em: <https://doi.org/10.1007/s10618-010-0179-5>.

Apêndices

APÊNDICE **A**

Resultados Estendidos

Este capítulo apresenta a versão estendida dos resultados obtidos nos experimentos. Cada seção representa um experimento e exibe todos os resultados obtidos. Incluindo os resultados com pouca relevância, uma vez que estes também foram importantes na pesquisa. Eles auxiliaram na condução dos experimentos para que pudessem alcançar os resultados que alcançaram.

A.1 Otimização de Parâmetros

O experimentos com mais dados produzidos e com uma necessidade de uma quantidade maior de tabelas para os exibir foi a otimização de parâmetros. Dito isto, esta seção apresenta todos os conjuntos de parâmetros testados, junto com seus resultados, em cada variante. A definição de todos os parâmetros e os valores testados para cada um são apresentados na tabela 3. Note que nem todas as combinações dos valores de cada parâmetro foram testadas. As combinações foram definidas a partir de um conjunto aleatório que foi sendo ajustado de forma manual durante o experimento. O ajuste teve o mesmo objetivo do experimento, alcançar a maior eficiência AUROC, e não definiu todas as combinações devido às limitações do ambiente de experimentos e às limitações de tempo da pesquisa.

Os resultados estendidos da otimização de parâmetros são apresentados nas Tabelas 4 até a 11. Cada uma delas representa a otimização de uma variante que pode estar dividida em experimentos específicos. Apenas as variantes V1 e V3 dividiram suas otimizações. A V1 primeiramente buscou pelo melhor classificador (Tabela 5) para em seguida dividir a variante em duas, a V1-SG e a V1-kWS. Essa divisão ocorreu pelo fato de duas abordagens diferentes para a técnica de multirresolução terem sido implementadas. Da mesma forma, duas abordagens para a representação *ngram* foram implementadas, a V3-SU e a V3-NR. As quatro abordagens foram consideradas como variantes diferentes, tiveram seus parâmetros otimizados e rodaram os experimentos para a avaliação de classificação. Os pseudocódigos de cada variante (Seção 4.2.2) descrevem a variante considerando apenas a melhor abordagem de cada técnica. A melhor abordagem é aquela com a maior eficiência

na avaliação de classificação dentre as abordagens implementadas.

Tabela 3 – Definição dos parâmetros.

Parâmetro	Abbrv.	Valores	Definição
tamanho do alfabeto	ax	{2, 4}	Define o tamanho do alfabeto da discretização
declínio	Dc	Presente, Ausente	Seleciona mais palavras para a melhor janela e diminui a seleção gradualmente
ending feature selection	EFS	{Presente, Ausente}	Define o uso de uma seleção aleatória antes da seleção de características
fração de amostra	frac	{0.1, 0.2, 0.3, 0.4, 0.5, 0.6}	Define o tamanho de cada amostragem em relação aos dados originais
feature selection	FS	Presente, Ausente	Define o uso da seleção de características
feature selection ending selection	RSFS	{Presente, Ausente}	Define uma seleção de características final depois da seleção em cada resolução considerando todas as características
k-*window search*	kx-y	{5-mean, 5-max, 10-mean, 10-max}	Define a quantidade janelas selecionadas usando a métrica y
l máximo	l_x	$\{L/2, L\}$	Define o tamanho máximo das janelas
número de l	nl$x$	1, 10, 20, 30	Define a quantidade de janelas usadas
número de l sax e sfa	nl$x$-y	{1-4, 2-8, 2-10, 2-12, 4-10}	Define a quantidade de janelas usadas para o sax e sfa
número de w	nw	{50, 100, 150, 200, 500}	Define a quantidade de palavras selecionadas
p-*value*	px	{0.05, 0.005, 0.0005, 0.00005}	Define as palavras selecionadas usando o p-*value*
processadores	pc$x$	{1, 3, 5, 6}	Define a quantidade de processadores para treinar o *ensemble* em paralelo
random selection feature selection	RSFS	{Presente, Ausente}	Define o uso de uma seleção aleatória antes da seleção de características
seletores	Sx	{10, 20}	Define a quantidade de seletores no *ensemble*
seletores sax e sfa	Sx-y	{10-20, 7-23, 6-24, 13-27, 20-40}	Define a quantidade de seletores sax e seletores sfa do *ensemble*
sax	SAX	SAX	Define o SAX como discretizador
sfa	SFA	SFA	Define o SFA como discretizador
sax e sfa	SS	(sax, sfa)	Define os discretizadores sax e sfa para a técnica de multidomínio
tamanho da palavra	wx	{4, 6}	Define o tamanho da palavra discretizada
ngrams	$x$grams	{3, 5}	Define o tamanho de n para compor todas as *ngrams* de 1 a n
Classificadores			
balanceamento de classe	bc	{Presente, Ausente}	Ajusta pesos para as classes de acordo com a quantidade de cada uma no dados
logistic regression	LR-x	{*lbfgs*, *liblinear*}	Define o *logistic regression* como classificador usando a função de otimização x
random forest	RF-x	{*gini*, *entropy*}	Define a *random forest* como classificandor usando a função de qualidade x
quantidade de características *random forest*	mw$x$	$\{\sqrt{w}, 0.40\}$	máximo de características em cada árvore em relação ao total de características
quantidade de objetos *random forest*	ox	{1, 2}	Mínimo de objetos em cada folha da árvore
support vector machine	SVC-x	{ *rbf*, polinômio, sigmoide }	Define a SVM como classificador usando *kernels* do tipo x.

Tabela 4 – Variante 0 (V0)

Parâmetros	média (%) Acurácia	média (%) AUROC	média (s) treinamento	Eficiência Acurácia	Eficiência AUROC
SAX l_L nw...	44,29	50,00	121,97	0,02	0,01
SFA l_L nw...	52,39	63,69	33,26	0,23	0,35
SAX RSFS l_L/2 nw200	53,32	62,13	143,28	0,06	0,06
SAX RSFS l_L/2 nw100	53,32	62,13	142,01	0,06	0,06
SAX FS l_L/2 nw200	59,03	66,11	174,02	0,14	0,10
SAX FS l_L/2 nw100	59,03	66,11	173,38	0,14	0,10
SAX FS l_L/2 nw500	59,03	66,11	172,62	0,14	0,10
SAX l_L/2 nw...	59,03	66,11	175,39	0,14	0,10
SFA RSFS l_L/2 nw100	69,27	81,35	16,82	8,61	14,62
SFA RSFS l_L/2 nw200	69,68	81,65	18,67	8,33	13,89
SFA FS l_L/2 nw100	71,44	83,10	19,11	11,04	17,43
SFA FS l_L/2 nw200	72,28	84,01	22,06	11,06	17,69
SFA FS l_L/2 nw500	73,14	84,37	28,07	10,10	14,79
SFA l_L/2 nw...	73,89	84,53	54,60	5,91	7,81

Tabela 5 – Variante 1 *shotgun* com comparação de classificadores (V1-clf)

Parâmetros	média (%) Acurácia	média (%) AUROC	média (s) treinamento	Eficiência Acurácia	Eficiência AUROC
SVC-pl5_FS_nw100	66,14	83,29	8,63	9,77	39,90
SVC-pl3_FS_nw100	69,96	85,98	8,49	19,23	64,63
RF_o1_mw$\sqrt{w}$_FSES_nw100	73,96	86,44	9,23	35,36	64,42
SVC-rbf_RSFS_nw100	74,89	87,89	8,49	45,22	90,05
SVC-rbf_bc_FS_nw100	75,13	88,25	8,67	46,12	93,80
RF_o1_mw$\sqrt{w}$_RSFS_nw100	75,78	88,57	9,24	48,48	93,05
LR-lib_FS_nw100	75,89	87,44	8,34	54,72	84,63
SVC-rbf_FS_nw100	76,11	88,26	8,54	55,53	95,40
RF-e_o1_mw$\sqrt{w}$_FS_nw100	76,32	88,86	9,39	52,41	96,25
RF_o1_mw$\sqrt{w}$_FS_nw100	76,41	88,88	9,35	53,43	97,06
SVC-s_FS_nw100	76,47	86,36	8,46	59,63	69,28
LR-lbfs_RSFS_nw100	76,58	86,99	7,69	66,90	85,04
RF_o2_mw.40_FS_nw100	77,52	88,11	10,24	59,11	77,52
LR-lbfs_bc_FS_nw100	77,53	87,46	8,24	73,62	86,13
RF_o2_mw$\sqrt{w}$_FS_nw100	77,87	88,74	9,28	69,27	95,31

Tabela 6 – Variante 1 *shotgun* (V1-SG)

Parâmetros	média (%) Acurácia	média (%) AUROC	média (s) treinamento	Eficiência Acurácia	Eficiência AUROC
FS_p.05_nl10	66,57	86,65	5,95	15,24	103,54
FS_p.005_nl10	69,16	87,91	6,00	23,69	127,66
RSFS_p.0005_nl10	69,64	87,10	5,77	26,77	115,39
FS_p.0005_nl10	71,73	88,46	6,46	34,37	130,52
RSFS_nw200_nl10	74,97	88,11	5,48	70,93	144,74
FS_nw20_nl20	75,04	87,38	9,75	40,43	71,72
RSFS_nw200_nl20	75,26	88,27	9,37	43,68	87,07
FS_nw20_nl30	75,49	87,58	13,47	31,60	53,74
FS_nw50_nl10	76,06	88,21	6,22	75,60	129,90
EFS_nw300_nl10	76,22	88,17	6,23	77,64	128,70
FS_nw400_nl20	76,36	89,18	9,76	50,75	97,88
FS_nw100_nl20	76,41	88,88	9,36	53,34	96,89
FS_nw200_nl10	76,54	89,06	5,59	91,43	167,48
FS_nw100_nl10	76,55	88,74	5,67	90,19	156,19
FS_nw300_nl10	76,63	88,85	5,69	91,08	158,43
FS_nw200_nl20	76,66	89,25	9,52	54,78	101,62
FS_nw300_nl20	76,70	89,03	9,65	54,42	96,39
FS_nw50_nl20	76,74	88,47	9,35	56,60	90,36
FS_nw100_nl30	76,83	89,00	13,58	39,57	68,18
FS_nw200_nl30	77,05	89,15	13,75	40,54	69,06
FS_nw50_nl30	77,08	88,43	13,48	41,60	62,26

Tabela 7 – Variante 1 k-*best windows search* (V1-kWS)

Parâmetros	média (%) Acurácia	média (%) AUROC	média (s) treinamento	Eficiência Acurácia	Eficiência AUROC
RSFS_k10-max_Dc_nl20	75,36	87,97	9,33	44,65	83,11
RSFS_k10-max_nl20	75,64	88,41	9,41	46,40	88,80
FS_k5-mean_nl20	75,81	88,65	9,70	46,42	89,89
FS_k10-mean_nl20	76,27	89,01	10,09	48,30	91,94
FS_k10-max_Dc_nl20	76,34	88,71	9,89	49,84	89,07
FS_k5-max_nl20	76,65	88,79	9,63	54,01	92,67
FS_k10-max_nl20	76,86	89,03	9,81	55,07	94,89
FS_k10-max_nl30	76,93	88,88	14,19	38,49	63,97

Tabela 8 – Variante 2 (V2)

Parâmetros	média (%) Acurácia	média (%) AUROC	média (s) treinamento	Eficiência Acurácia	Eficiência AUROC
SS_FSRS_nw200_nl2-8	75,11	87,85	18,25	21,83	41,59
SS_RSFS_nw200_nl2-8	75,56	88,01	18,59	23,19	41,97
SS_FS_nw200_nl2-8	76,05	88,65	18,72	25,07	46,52
SS_FS_nw200_nl2-10	76,25	88,93	20,11	24,16	45,52
SS_FS_nw200_nl4-10	76,37	88,88	33,50	14,79	27,06
SS_FS_nw200_nl2-12	76,67	88,99	20,34	25,69	45,43

Tabela 9 – Variante 3 Seleção Unificada (V3-SU)

Parâmetros	média (%) Acurácia	média (%) AUROC	média (s) treinamento	Eficiência Acurácia	Eficiência AUROC
RSFS_3grams_w6_a4	75,81	88,29	22,51	19,99	36,39
RSFS_5grams_w6_a4	76,58	88,51	32,92	15,62	25,84
FS_5grams_w6_a4	76,85	88,91	34,33	15,70	26,56
FS_3grams_w6_a4	77,23	89,04	22,46	25,62	41,56
FS_5gram_w4_a4	77,94	88,75	23,16	28,10	38,27
FS_3gram_w4_a4	78,09	89,02	18,96	35,26	49,00
FS_5gram_w4_a2	78,67	88,81	18,68	39,55	47,96
FS_3gram_w4_a2	79,06	89,05	17,86	44,23	52,33

Tabela 10 – Variante 3 Ngram como Resolução (V3-NR)

Parâmetros	média (%) Acurácia	média (%) AUROC	média (s) treinamento	Eficiência Acurácia	Eficiência AUROC
5grams_nw50_w6_a4	76,59	88,52	91,38	5,63	9,32
5grams_nw100_w6_a4	76,68	88,94	93,24	5,61	9,82
3grams_nw100_w6_a4	76,86	89,01	53,31	10,12	17,40
3grams_nw50_w6_a4	77,02	88,66	52,75	10,52	16,56
5grams_nw200_w6_a4	77,02	89,12	93,73	5,92	10,09
3grams_Dc_nw200_w6_a4	77,06	89,06	54,03	10,34	17,32
3grams_nw200_w6_a4	77,09	89,15	53,11	10,58	17,89
3grams_Dc_nw150_w6_a4	77,13	89,03	52,46	10,78	17,76
3grams_nw200_w4_a4	78,47	89,31	48,77	14,62	20,03
3grams_nw200_w4_a2	78,91	89,06	47,06	16,37	19,89

Tabela 11 – Variante 4 (V4)

Parâmetros	média (%) Acurácia	média (%) AUROC	média (s) treinamento	Eficiência Acurácia	Eficiência AUROC
S13-27_nl1_frac.2_pc1	70,06	86,02	63,61	2,61	8,69
S20-40_nl1_frac.2_pc1	71,11	86,16	94,84	2,10	5,97
S13-27_nl1_frac.1_pc1	71,58	87,27	73,62	2,94	9,32
S20-40_nl1_frac.1_pc1	71,62	87,41	106,63	2,04	6,59
S13-27_nl1_frac.2_pc1	73,64	88,27	85,72	3,60	9,52
S10-20_nl1_frac.3_pc1	73,80	88,66	73,20	4,34	11,93
S10-20_nl1_frac.2_pc1	74,22	88,38	63,24	5,41	13,15
S20_nl1-4_frac.2_pc1	74,26	88,30	70,18	4,90	11,69
S10_nl1-4_frac.2_pc1	75,10	88,12	33,91	11,74	23,44
S10-20_nl1_frac.4_pc5	75,12	88,81	38,52	10,36	23,28
S10_nl2-8_frac.2_pc1	75,18	87,94	54,13	7,46	14,24
S10_nl1-4_frac.3_pc1	75,69	88,61	42,30	10,42	20,47
S7-23_nl1_frac.2_pc6	76,34	89,16	47,14	10,47	20,19
S10-20_nl1_frac.6_pc5	76,35	89,15	49,69	9,95	19,14
S10_nl1-4_frac.5_pc3	76,52	89,11	30,67	16,60	30,80
S10-20_nl1_frac.6_pc1	76,73	89,18	104,97	5,03	9,11
S6-24_nl1_frac.6_pc1	76,79	89,23	130,66	4,08	7,37
S10_nl1-4_frac.6_pc1	77,15	89,23	69,19	8,21	13,93
S10_nl1-4_frac.6_pc3	77,30	89,22	34,45	16,90	27,95