ALDOMAR PIETRO SANTANA SILVA

ANÁLISE E COMPARAÇÃO DE MODELOS DE CLASSIFICAÇÃO DE SÉRIES TEMPORAIS MULTIVARIADAS

São Paulo
2022

ALDOMAR PIETRO SANTANA SILVA

ANÁLISE E COMPARAÇÃO DE MODELOS DE CLASSIFICAÇÃO DE SÉRIES TEMPORAIS MULTIVARIADAS

Trabalho apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Engenheiro de Computação.

São Paulo
2022

ALDOMAR PIETRO SANTANA SILVA

ANÁLISE E COMPARAÇÃO DE MODELOS DE CLASSIFICAÇÃO DE SÉRIES TEMPORAIS MULTIVARIADAS

Trabalho apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Engenheiro de Computação.

Orientadora:
Anna Helena Reali Costa

São Paulo
2022

À Dona Rosa, minha avó, melhor amiga e eterna inspiração.

AGRADECIMENTOS

Gostaria de agradecer aos meus pais, por acreditarem em mim quando mais precisei e por me apoiarem nos maiores desafios da minha vida. À minha prima Sabrina, por todo o aprendizado passado e por me mostrar a importância dos estudos. À minha vó, por sempre me amparar.

Ao Centro de Ciências de Dados (C^2D), junto ao Programa de Bolsas do Itaú (PBI), pela bolsa de estudos e por toda a infraestrutura de apoio fornecida.

Gostaria de agradecer, também, ao Engenheiro Rodrigo da Silva Cunha, que me auxiliou com dicas importantes para a execução de todo o projeto.

Por último, mas não menos importante, agradeço, também, à minha orientadora, Professora Doutora Anna H. R. Costa por toda a mentoria fornecida e sem a qual este trabalho não teria sido possível.

"A matemática é a linguagem na qual Deus escreveu o Universo'

-- Galileu Galilei

RESUMO

Este projeto faz parte do Pré-Mestrado, que irá se estender para um Mestrado no ano seguinte e também se configura como tema de TCC do aluno. O projeto de pesquisa tem como objetivo resolver um problema que não possui somente um alto impacto em problemas de engenharia em geral, como também é complexo o suficiente para justificar o uso de técnicas avançadas de *Machine Learning*. Trata-se da Classificação de Eventos em Séries Temporais, problema no qual objetiva-se avaliar diversos modelos que buscam resolvê-lo, levando em consideração a precisão da classificação, o tempo de treinamento e, por consequência, a escalabilidade das abordagens. Por fim, a técnica de classificação de séries temporais será utilizada no contexto de um problema do mundo real: a identificação de falhas em linhas de amarração de plataformas petrolíferas *offshore*. Neste problema, as alterações nos dados de movimentação dessas unidades são utilizadas para detectar possíveis quebras. Os dados gerados pelo simulador Dynasim são usados para treinar e testar os modelos de *Machine Learning* selecionados. Três modelos foram selecionados: Dependent Dynamic Time Warping, ROCKET e InceptionTime. Bons resultados foram gerados em uma análise de sensibilidade, onde um ruído branco foi adicionado a um conjunto de dados, no qual o InceptionTime apresentou os melhores resultados, decaindo sua precisão apenas em altos níveis de ruído, seguido do ROCKET e, em seguida, do Dynamic Time Warping. Também obtivemos bons resultados no problema de falhas em linhas de amarração, onde todos os modelos apresentaram acurácia acima de 99%, exceto para o Dependent Dynamic Time Warping que ultrapassou o limite de tempo.

Palavras-Chave – *Machine Learning*, *Deep Learning*, Classificação de Séries Temporais, Classificação de Eventos.

ABSTRACT

This work is part of the "Pré-mestrado"program, which will be extended to a master's degree work in the following year and also configures itself as the student's completion of coursework theme. This research aims to solve a problem that has a high impact on general engineering problems and is complex enough to justify the usage of advanced Machine Learning techniques. We are talking about the Time Series Classification problem. We aim to evaluate several models that seek to solve this kind of issue, taking into account the classification's accuracy, training time and, consequently, these approaches' scalability. Finally, the time series classification technique will be used in the context of a real-world problem: the detection of mooring systems failures of floating oil production units. In this problem, changes in the movement data of these units will be used to detect possible breakage. Data generated by the Dynasim simulator will be used to train and test the selected Machine Learning models. We selected three models: Dependent Dynamic Time Warping, ROCKET e InceptionTime. We had good results in a sensitivity analysis, where white noise was added to a dataset, and InceptionTime has shown the best results, only decaying its accuracy in high noise levels, followed by ROCKET and, then, Dynamic Time Warping. We also had good results in the mooring line problem, where all the models had shown accuracy above 99%, except for Dependent Dynamic Time Warping which had exceeded the time limit.

Keywords – Machine Learning, Deep Learning, Time Series Classification, Event Classification.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Nos últimos anos observou-se um grande aumento na relevância e na presença de dados de séries temporais. Segundo [1], empresas que variam desde cuidados com a saúde, até sensoreamento remoto, todas produzem dados de séries temporais de forma nunca antes vista, tanto em termos de quantidade quanto em termos de tamanho dos dados, gerando a necessidade de uma forma de classificação automática desse tipo de informação. Esse problema é conhecido como classificação de séries temporais, do inglês *Time Series Classification* (TSC) e a sua resolução é feita, idealmente, através de algoritmos de *Machine Learning* (ML) que possam realizar esse tipo de classificação de forma precisa e com um tempo de treinamento relativamente curto, sendo este último atributo aquele que permite que os algoritmos sejam utilizados em larga escala.

Essa necessidade de classificar séries temporais também está presente nos mais diversos assuntos e temas, podendo ter seu uso em áreas como: o mercado financeiro, no qual a TSC pode ser utilizada para detectar fraudes ou transações anômalas a partir de análise de séries financeiras; a área da saúde, onde eventos expúrios podem ser detectados com TSC; aplicações de IoT, nas quais a TSC pode ser usada para avaliar dados de sensoriamento remoto. Dessa forma, pode-se perceber que diversas aplicações relevantes carecem da tarefa de classificar séries temporais de dados e podem se beneficiar diretamente de um modelo de TSC preciso e com tempo de treinamento mais rápido.

1.1 Objetivos e Justificativas

Este projeto tem como objetivo estudar e comparar métodos de *Machine Learning* que buscam resolver o problema de Classificação de Séries Temporais Multivariadas, do inglês *Multivariate Time Series Classification* (MTSC). Esses métodos são construídos com o propósito de, a partir de um conjunto de exemplos de séries temporais de múltiplas dimensões, realizar a identificação do evento que tem como característica a série apresentada.

Para que esse tipo de classificação seja feito, o modelo precisa ser treinado e testado, seguindo o paradigma de aprendizado supervisionado, um processo que demanda um determinado tempo computação, o qual pode ser diferente para cada modelo. Assim, para cada modelo, há um tempo destinado ao treinamento do modelo e outro, da execução do modelo. Além dos tempos de treinamento e execução, a acurácia de cada técnica também é relevante, ou seja, os modelos não realizam a classificação com 100% de eficácia, possuindo uma taxa de erro que depende, além do próprio modelo, do tipo de dado usado por ele. Por fim, a quantidade de memória auxiliar utilizada pelos modelos também varia, o que nos faz prestar atenção nesses tipos de métricas, que influenciam diretamente na escolha ou não de um determinado modelo para realizar a solução de algum problema envolvendo MTSC.

Assim, pretende-se, com este trabalho, realizar um estudo detalhado e aprofundado de alguns dos algoritmos de MTSC a partir do treinamento e teste desses modelos em dados de *benchmark* apropriados, realizando a medição de cada uma das métricas que se mostrarem adequadas para a comparação desses métodos entre si. Dessa forma, espera-se apresentar, no final deste projeto, uma análise detalhada de cada modelo estudado, com um possível levantamento de seus prós e contras.

Outro fator relevante na análise de modelos de MTSC é a robustez de cada modelo, isto é, é importante avaliar a sensibilidade de cada modelo a diferentes graus de ruídos nas séries de entrada. Desta forma, as respostas dos modelos em função do nível de ruído nas entradas também serão avaliadas.

Por fim, a partir dos resultados da análise comparativa, os modelos serão empregado em uma aplicação real, utilizando dados do problema de detecção de falhas na amarração de plataformas petrolíferas *offshore*. Serão analisadas as alterações da movimentação de plataformas para prever se algum cabo de amarração encontra-se rompido e, dessa forma, demonstrar a utilidade do uso de algoritmos de MTSC para a resolução de problemas das mais diversas áreas.

1.2 Organização do trabalho

Esta monografia está organizada da seguinte forma. No segundo capítulo, são apresentados os fundamentos relacionados ao estudo de séries temporais, suas características e aplicações. Também são apresentados os modelos de classificação de séries temporais multivariadas, além de uma revisão da literatura no tocante às métricas comumente uti-

lizadas para avaliar os modelos. Por fim, o capítulo também conta com uma introdução ao problema de falha de linhas de amarração de plataformas petrolíferas offshore, um importante problema que atinge a indústria de petróleo e que se mostra uma grande possibilidade de aplicação no ramo de séries temporais multivariadas.

Já o terceiro capítulo trata da especificação de materiais e métodos utilizados no projeto, descrevendo os dados e as tecnologias utilizadas, assim como fornecendo as especificações dos requisitos funcionais e não funcionais do projeto.

O quarto capítulo, por sua vez, traz uma discussão a respeito do desenvolvimento do projeto, detalhando as etapas dos experimentos realizados, o que vai desde os critérios de seleção dos modelos até a etapa de planejamento dos experimentos. Por fim, o processo relacionado à aplicação dos modelos selecionados no problema de detecção de falhas de amarração em plataformas flutuantes também é detalhado, explicando o processamento feito nos dados, bem como a alteração dos hiperparâmetros dos modelos.

O quinto capítulo traz os experimentos realizados com os dados de *benchmark* e seus respectivos resultados, bem como a análise de sensibilidade dos métodos. Além disso, neste capítulo, os modelos selecionados por apresentarem os melhores resultados nos experimentos anteriores são aplicados nos dados do problema de detecção de falhas no sistema de amarração de plataformas *offshore*. Finalmente, no sexto capítulo ocorrem as considerações finais, passando pelos tópicos de conclusões do projeto, contribuições e perspectivas de continuidade.

2 FUNDAMENTOS E MODELOS

Neste capítulo são descritos alguns dos conceitos mais importantes envolvendo o contexto de Classificação de Séries Temporais. Além disso, os modelos de classificação de séries temporais multivariadas que mais se destacaram durante a análise preliminar da literatura são brevemente descritos, sendo estes modelos selecionados para uso nos experimentos que serão realizados ao longo deste trabalho.

2.1 Características dos Dados de Séries Temporais

Dados de séries temporais diferenciam-se de dados convencionais pelo fato de serem ordenados. Uma característica muito importante deste tipo de dado é que as observações vizinhas são dependentes, o que levanta o interesse de analisar e modelar essa dependência.

Os dados de séries temporais podem ser classificados entre univariados e multivariados: em séries univariadas os casos dependem de apenas uma variável e, em séries multivariadas, os casos dependem de mais de uma variável que podem ou não se relacionar entre si, impactando no resultado final. Esse tipo de dado possui características que norteiam os modelos no processo de classificação, sendo alguns deles:

Linearidade: indica que a forma da série depende do seu estado atual de modo que esse estado atual determine o modelo da série. Portanto, se uma série é linear, então ela pode ser representada por funções lineares de valores presentes e valores passados.

Tendência: Está relacionado ao comportamento de longo prazo da série, ou seja, quando vemos um gráfico de uma série onde os dados estão subindo, descendo ou estão constantes e com que velocidade esse comportamento muda, estamos observando uma tendência. Os comportamentos mais comuns são: tendência constante, tendência linear e tendência quadrática.

Sazonalidade: Uma série temporal se apresenta com alguns padrões de comportamento. Um determinado padrão pode se repetir em épocas específicas ao longo do tempo

e esse padrão é chamado de sazonalidade.

Além dessas características, há uma suposição básica que norteia a análise de séries temporais: a ideia de que há um sistema causal mais ou menos constante, relacionado com o tempo, que exerceu influência sobre os dados no passado e pode continuar a fazê-lo no futuro. Este sistema causal costuma atuar criando padrões não aleatórios que podem ser detectados em um gráfico da série temporal, ou mediante algum outro processo estatístico.

Assim, desenvolve-se modelos com o objetivo de analisar essas séries temporais e, dessa forma, identificar padrões não aleatórios em uma série temporal de interesse. A observação deste comportamento passado permite fazer previsões sobre o futuro, orientando a tomada de decisões nas mais diversas áreas como finanças, marketing, economia, saúde, entre outros.

Definindo matematicamente, uma série temporal é estabelecida como uma sequência de n observações ao longo do tempo de uma determinada quantidade medida, representada por um vetor $\mathbf{x} = (x_1, \ldots, x_n)$. Já uma série temporal multivariada pode ser representada por uma matriz $n \times m$ de m variáveis, com n observações cada, por exemplo:

$\mathbf{X} = (\mathbf{x}_1^\top, \ldots, \mathbf{x}_m^\top)$, $\mathbf{x}_j = (x_{1,j}, \ldots, x_{n,j})$, $1 \leq j \leq m$.

O conjunto de observações é dado por X, com $\mathbf{X} \in X$.

Já no contexto de classificação, cada série temporal é associada com um rótulo y, que é um valor pertencente à uma variável discreta Y. Uma instância é definida como um par $\langle \mathbf{X}, y \rangle$, e k pares formam um dataset $S = \{\langle \mathbf{X_1}, y_1 \rangle, \ldots, \langle \mathbf{X_k}, y_k \rangle\}$. Assim, um classificador pode ser tanto uma função f que vai de sequências à rótulos ($(f : X \to Y)$ ou uma função g que mapeia o espaço de possíveis entradas com uma probabilidade de distribuição sobre os valores de classe da variável ($g : X \times Y \to [0, 1]$). Achar a função que melhor aproxima f ou g usando o dataset S é chamado de "treinar o classificador".

2.2 Aprendizado Supervisionado

Aprendizado supervisionado é um tipo de aprendizado de máquina em que um modelo é treinado com um conjunto de dados rotulados de forma que se possa aprender a prever ou classificar novos dados com base nos exemplos fornecidos. No aprendizado supervisionado, os exemplos de treinamento são rotulados com as saídas desejadas, o que permite que o modelo aprenda a relação entre as entradas e as saídas e, em seguida, aplique o que aprendeu para prever ou classificar novos dados. Isso é diferente do aprendizado não

supervisionado, onde o modelo é treinado com um conjunto de dados não rotulados e é responsável por descobrir padrões e relações nos dados por conta própria.

2.3 Modelos e Arquiteturas de Classificação de Séries Temporais

O problema de Classificação de Séries Temporais tem sido considerado como um dos mais desafiadores na área de ciência de dados durante as duas últimas décadas [1]. Além disso, grande parte dos dados presentes no mundo real têm uma componente temporal, independente de serem processos naturais, como clima e ondas de som, ou de serem processos feitos pelo homem, como robôs ou a própria bolsa de valores [2]. Dessa forma, a análise de séries temporais tem sido alvo de pesquisas ativas por décadas, devido as suas propriedades únicas.

De acordo com [3], a tarefa de classificação de séries temporais pode ser vista como o aprendizado ou como a detecção de sinais ou padrões associados a classes relevantes das séries. Para isso, diferentes métodos para a classificação de séries temporais representam diferentes abordagens para a extração de características úteis. Essas abordagens geralmente focam em um único tipo de característica, como frequência ou variância de sinal.

Um estudo feito em [4] consagrou 3 métodos como os classificadores mais precisos presentes no arquivo da Universidade da Califórnia, Riverside: o *BOSS*, que é um dentre os vários métodos baseados em dicionários, o *HIVE-COTE* que é um enorme conjunto de outros classificadores, incluindo o próprio BOSS e, por fim, o *Shapelet Transform* um dentre os diversos métodos baseado em encontrar subséries discriminativas, todos métodos elaborados com princípios diferentes.

Estes modelos, porém, considerados estados-da-arte, possuem uma complexidade computacional muito elevada, o que os fazem lentos, mesmo para conjuntos de dados pequenos, e praticamente intratáveis para grandes conjuntos de dados. Isso motiva o desenvolvimento de novos modelos que sejam tão precisos quanto o estado da arte, porém que tenham um tempo de treinamento menor e, portanto, sejam mais escaláveis. Motivado por esse contexto, [5] propõe uma comparação de modelos de Classificação de Séries Temporais Multivariadas, que é, também, o propósito deste trabalho. No artigo, é selecionada uma grande variedade de algoritmos classificadores de séries temporais, sendo eles direcionados para séries multivariadas e outros já conhecidos no contexto de classificação de

séries temporais univariadas.

O principal foco do artigo foi comparar os diversos algoritmos selecionados com um classificador mais simples, denominado *Dynamic Time Warping* (modificado para uso em MTSC por [6]) e que, por muito tempo, foi considerado o "padrão de ouro" no contexto de classificação de séries temporais. O artigo ainda apresenta *datasets* interessantes com dados multivariados, retirados do próprio repositório formado por uma parceria de pesquisadores da Universidade da Califórnia, Riverside(UCR) e a Universidade da Ânglia Oriental, denominada UCR/UEA. Apesar de propor uma avaliação dos algoritmos em diferentes métricas, com a leitura podemos perceber que o artigo apenas cita os resultados dos algoritmos que melhor se saíram, se apegando muito mais aos resultados de precisão. Além disso, o artigo não leva muito em consideração as questões de eficiência, apenas apresentando uma tabela com os tempos de execução e afirmando que não levariam em conta esse tipo de parâmetro para o resultado final.

Dessa forma, neste trabalho pretende-se utilizar o conhecimento gerado por [5] (e todos os outros artigos já citados), levando-se em conta *datasets* utilizados (detalhados na seção de Materiais e Infraestrutura) e recomendações de algoritmos que se demonstraram promissores (detalhados a seguir). Contudo, nas análises a serem realizadas neste documento, pretende-se levar em conta outros tipos de métricas, que vão desde medidas de desempenho até medidas de eficiência (detalhadas ainda nesta sessão) , além de incluir outros algoritmos que não fazem parte da coletânea do arquivo construído pelos pesquisadores da UCR/UEA.

Em suma, existem diversos modelos de classificação de séries temporais que são baseados em diferentes abordagens como redes neurais, redes convolucionais, distância entre elementos, entre outros. Nesta seção, detalha-se alguns dos modelos que se tornaram inspiração para o projeto e que serão analizados em passos futuros.

2.3.1 InceptionTime

O modelo *InceptionTime* é composto por cinco modelos de *Deep Learning*: cada um criado cascateando múltiplos módulos de *Inception* [1]. Cada classificador individual neste conjunto terá exatamente a mesma arquitetura, mas com valores iniciais de peso aplicados aleatoriamente e diferentes entre si.

O conceito do módulo de *Inception*, proposto por [7], tem o objetivo de realizar classificações fim-a-fim de imagens e que acabou sendo trazido para a abordagem de TSC

quando [8] transformou os dados de séries temporais em imagens usando o Campo de Diferença Angular Gramatical, do inglês *Gramian Angular Difference Field* (GADF) e então alimentou um modelo Inception que já havia sido pré-treinado para o reconhecimento de imagens.

A principal ideia de um módulo de *Inception* é aplicar múltiplos filtros simultaneamente a uma entrada de série temporal. O módulo inclui filtros de larguras variadas, que permitirão que a rede extraia automaticamente atributos relevantes de séries temporais longas e curtas [1]. Em [9], porém, foi percebido que o resultado obtido por um único módulo de *Inception* exibe um alto desvio padrão na precisão, o que pode gerar análises muito ricas ou muito pobres, sendo a combinação de vários módulos uma forma de nivelar esta instabilidade. À essa combinação de vários módulos *Inception* deu-se o nome de *InceptionTime*.

A rede *Inception*, componente do *InceptionTime*, é composta por dois blocos residuais, cada um composto por 3 módulos *Inception*, como visto na Figura 1. Neles, é empregada uma camada *Global Average Polling* (GAP) que calcula a média das séries temporais ao longo de toda a dimensão do tempo [1] e, por fim, é usada uma camada totalmente conectada com função *softmax*, onde o número de neurônios corresponde ao número de classes no conjunto de dados.

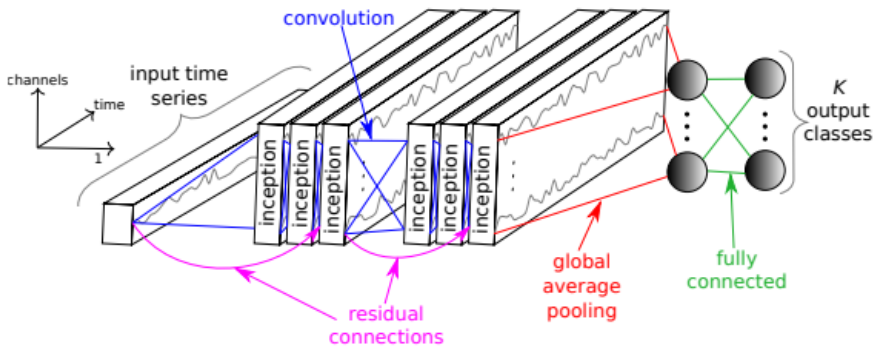

Figura 1: Rede Inception para classificação de séries temporais, retirada de [1].

2.3.2 Random Convolutional Kernel Transform (ROCKET)

O ROCKET, abreviação do inglês *Random Convolutional Kernel Transform* [3] foi escolhido por sua notável acurácia e velocidade em uma grande variedade de datasets do repositório UCR & UEA [5]. Este modelo usa um grande número de **kernels** convolucionais aleatórios de diferentes larguras, pesos, viéses, dilatações e preenchimentos, a fim de transformar a série temporal para que ela possa ser usada como entrada em um

classificador linear. Todo *kernel* é aplicado para cada instância e, do mapa resultante de atributos, o valor máximo (`max`) e a proporção de valores positivos (`ppv`) são retornados. Esses valores, então, são usados no treinamento de modelos classificadores lineares, como o Ridge Regression [10].

Uma operação de convolução é aplicada entre os kernels e as séries e os valores resultantes, como já dito, são `max` e `ppv`, sendo este último um resumo da proporção das séries correlacionadas ao kernel e uma medida que, como descoberto, aumenta significativamente a acurácia da classificação [5]. No caso de séries univariadas, a convolução entre uma instância e um *kernel* pode ser interpretada como o produto escalar entre dois vetores. O valor padrão do número de kernels aleatórios gerados é 10,000 e, nesse caso, 20,000 atributos são gerados para cada série. O dataset de atributos resultante é, então, usado para treinar um classificador linear qualquer, sendo o Ridge Classifier o mais utilizado.

O único hiperparâmetro do modelo ROCKET é o número de kernels K, que estabelece uma troca entre acurácia na classificação e tempo de processamento computacional: quanto maior K, maior o valor da acurácia de classificação, porém maior se torna o tempo de processamento. Para conjuntos de dados multivariados, os kernels são gerados automaticamente com diferentes dimensões atribuídas, sendo agora matrizes, e com pesos sendo atribuídos à diferentes dimensões. A convolução, nesse caso, pode ser interpretada como o produto escalar entre duas matrizes enquanto o *kernel* convoluciona "horizontalmente" pelas séries. os valores `max` e `ppv` são, agora, calculados entre mais dimensões, porém ainda produzem os mesmos 20,000 atributos no caso de K = 10,000.

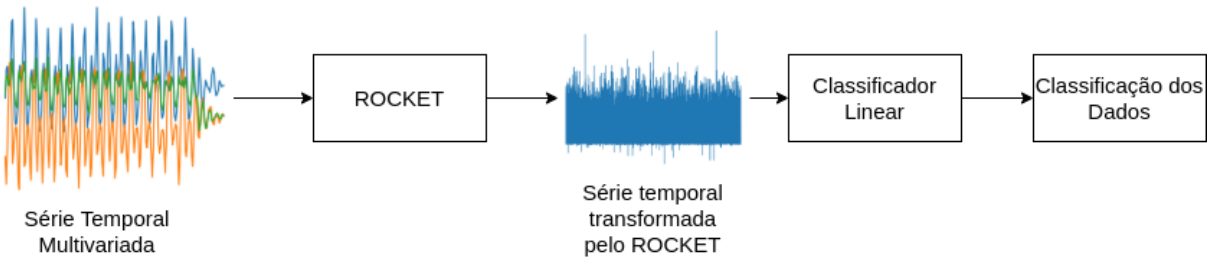

Figura 2: Diagrama esquemático representando o funcionamento do modelo ROCKET.

2.3.3 Dependent Dynamic Time Warping

Dynamic Time Warping (DTW) é uma medida de distância que emergiu como a principal escolha para muitas aplicações em séries temporais. Ela é usada em conjunto com o classificador do vizinho mais próximo (*1-nearest neighbourhood*), onde a classe atribuída ao objeto de estudo é a classe da observação da base de dados mais próxima.

O modelo *Dinamic Time Warping* (DTW) é a função de distância mais popular para esse propósito e pode ser usada com séries que possuem comprimentos desiguais, além de compensar possíveis desvios de confusão, permitindo algum realinhamento da série. A distância DTW resume a distância Euclidiana como um caso especial.

Para o cálculo de uma instância DTW no caso de uma série univariada, uma $n \times n$ matrix is constructed, cujos elementos (i, j) contêm a distância Euclidiana ao quadrado $d(q_i, c_j) = (qi - cj)^2$ entre dois pontos q_i e c_j das séries univariadas $\mathbf{q}$ e $\mathbf{c}$. Então, um caminho de distorção P, um conjunto contíguo de elementos da matriz definindo um mapa entre $\mathbf{q}$ e $\mathbf{c}$, é calculado comumente sujeito a diversas restrições, como: (i) os passos no caminho distorcido estão restritos à celulas adjacentes; (ii) o caminho deve iniciar e terminar em células diagonais opostas da matriz e (iii) os pontos do caminho distorcido devem ser monotonicamente espaçados no tempo. Também, é comum restringir o caminho limitando o quão longe ele pode ficar da diagonal [11]. Assim, o caminho de distorção que minimiza o custo é escolhido, sendo o custo D o valor que consiste na soma dos pontos do caminho, que pode ser calculado com a função recursiva [12]:

$$D(i, j) = d(q_i, c_j) + min\{D(i-1, j-1), D(i-1, j), D(i, j-1)\}. \tag{2.1}$$

Para o caso de séries multivariadas, duas generalizações do DTW foram propostas: a forma dependente e a independente. Neste projeto será utilizada a versão dependente, como sugerido em [5] e por isso apenas essa forma será detalhada. Para a versão dependente, o valor de $d(q_i, c_j)$ é redefinido e assume o valor da distância euclidiana quadrada cumulativa de M pontos:

$$d(q_i, c_j) = \sum_{l=1}^{M} (q_{i,l} - c_{j,l})^2, \tag{2.2}$$

onde i e j são pontos da l-ésima dimensão da série temporal multivariada. Antes de calcular a distância DTW, uma normalização z de cada dimensão da série é necessária para fazer a medida de distância e deslocamento. Finalmente, as distâncias calculadas são usadas no modelo *1-Nearest-Neighbour classifier.*

2.4 Métricas de Avaliação dos Modelos

As métricas são utilizadas para avaliar a qualidade dos resultados emitidos por um modelo classificador. Neste trabalho, como além de avaliar os modelos de aprendizado de máquina, também iremos compará-los a partir de seus resultados em diferentes datasets,

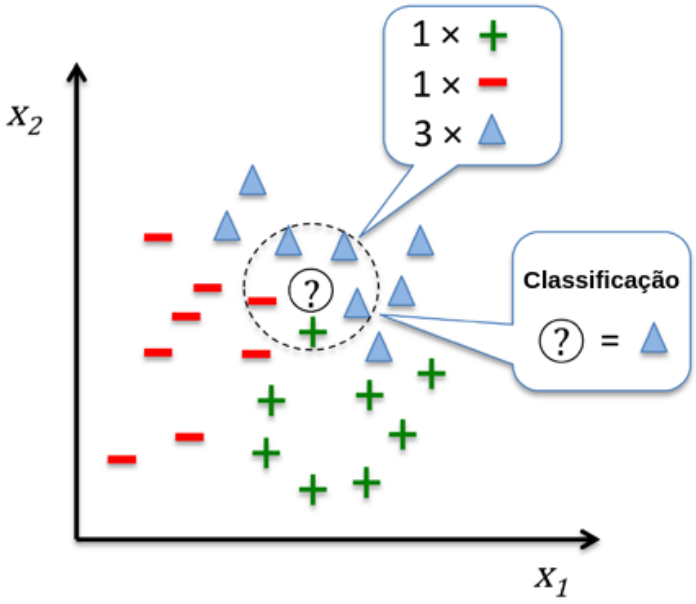

Figura 3: Exemplo de funcionamento do modelo de vizinho mais próximo (KNN), onde é medida a distância entre os exemplos presentes no dataset de treino e os dados a serem testados. Na figura, é retratado o modelo 5NN, onde as classes dos 5 modelos mais próximos são considerados e é realizada uma espécie de votação, com a classificação do modelo sendo dada pela maioria. No caso do modelo DDTW, apenas uma observação é considerada nessa votação e a medida de distância utilizada para selecioná-la é o próprio DTW.

as métricas se tornam ainda mais importantes, visto que os modelos, por terem estratégias diferentes, podem oscilar seus resultados dependendo da métrica escolhida. Dessa forma, [13] reuniu as principais métricas a serem utilizadas nesse contexto.

2.4.1 Matriz de Confusão

A matriz de confusão é uma tabela usada para avaliar a corretude de um modelo de classificação. Ela é frequentemente usada em problemas de aprendizado de máquina, onde o objetivo é prever a classe a qual um item pertence com base em um conjunto de dados de treinamento.

A matriz de confusão é uma tabela que contém as previsões do modelo em uma linha e as classes verdadeiras em uma coluna. Cada célula da tabela indica quantas vezes o modelo previu corretamente ou incorretamente uma classe. Isso permite a visualização de como o modelo está se saindo em relação às diferentes classes e quantas vezes ele está confundindo uma classe com outra. Dela, pode-se retirar diferentes métricas, como veremos a seguir.

Por exemplo, suponha que você esteja usando um modelo de classificação para prever se uma pessoa tem ou não uma doença. A matriz de confusão pode ajudá-lo a entender quantas vezes o modelo previu corretamente que uma pessoa tem a doença (verdadeiro positivo) ou não tem a doença (verdadeiro negativo), bem como quantas vezes ele previu incorretamente que uma pessoa tem a doença (falso positivo) ou não tem a doença (falso negativo). Isso pode ajudá-lo a identificar quais ajustes precisam ser feitos no modelo para melhorar sua precisão.

	Real **Positivo = 1**	**Real** **Negativo = 0**
Predito **Positivo = 1**	**VP**	*FP*
Predito **Negativo = 0**	*FN*	**VN**

Figura 4: Representação da matriz de confusão. Nela, VP, VN, FP e FN significam Verdadeiro Positivo, Verdadeiro Negativo, Falso Positivo e Falso Negativo, respectivamente.

2.4.2 Acurácia

A acurácia é considerada uma das métricas mais simples e importantes, pois avalia o percentual de acertos que o modelo obteve. Assim, pode ser obtida pela razão entre a quantidade de acertos e o total de instâncias avaliadas:

$$acuracia = \frac{\texttt{Total de Acertos}}{\texttt{Total de itens}}. \tag{2.3}$$

Quando utilizamos como base a matriz de confusão, a fórmula pode ser reescrita como:

$$acuracia = \frac{VP + VN}{VP + FN + VN + FP}. \tag{2.4}$$

2.4.3 Precisão

A precisão é uma métrica que avalia a quantidade de verdadeiros positivos sobre a soma de todos os valores positivos, sendo eles verdadeiros ou não:

$$precisao = \frac{VP}{VP + FP}. \tag{2.5}$$

2.4.4 Revocação

Também chamada de sensibilidade ou *recall*, essa métrica avalia a capacidade do método de detectar com sucesso resultados realmente positivos. É obtida através da equação:

$$revocacao = \frac{VP}{VP + FN}. \tag{2.6}$$

2.4.5 F1-Score

O F1-Score é uma métrica que resume as duas últimas métricas: precisão e revocação,

$$F1 - Score = 2 \times \frac{Precisao \times Revocacao}{Precisao + Revocacao}. \tag{2.7}$$

Ele se baseia em uma média harmônica entre as duas medidas, se tornando uma opção de substituição para ambas. Sendo uma média harmônica, o F1-Score assume um valor muito mais próximo da medida que possui menor valor do que uma média simples, por exemplo e, dessa forma, caso este valor esteja baixo é um indicativo de que pelo menos uma das métricas (Precisão ou Revocação) está aquém do ideal.

2.5 O problema de amarração de plataformas flutuantes

Em busca de expandir a busca por recursos energéticos por conta da crescente demanda por esse tipo de riqueza, as indústrias foram atrás de uma forma de geração de energia em alto mar. Para isso, foram criadas as Unidades Flutuantes de Produção, Armazenamento e Transferência, ou *Floating Production Storage and Offloading* (FPSO), um tipo de embarcação utilizado pela indústria petrolífera para a exploração e armazenamento de petróleo e/ou gás natural, além do escoamento da produção por navios cisterna. Esse tipo de estrutura se enquadra dentro do que é conhecido como Plataformas Flutuantes, ou *Floating Production Units*, das quais, além de extração de petróleo, também são usadas para abrigar turbinas eólicas para a produção de energia elétrica ou até mesmo fazendas de criação de peixes em alto mar.

Uma das estruturas mais importante que compõem as FPSO são os cabos de amarração: essas plataformas necessitam de estabilidade para que todo o processo de extração seja considerado seguro, o que é garantido pela ancoragem através de cabos de aço.

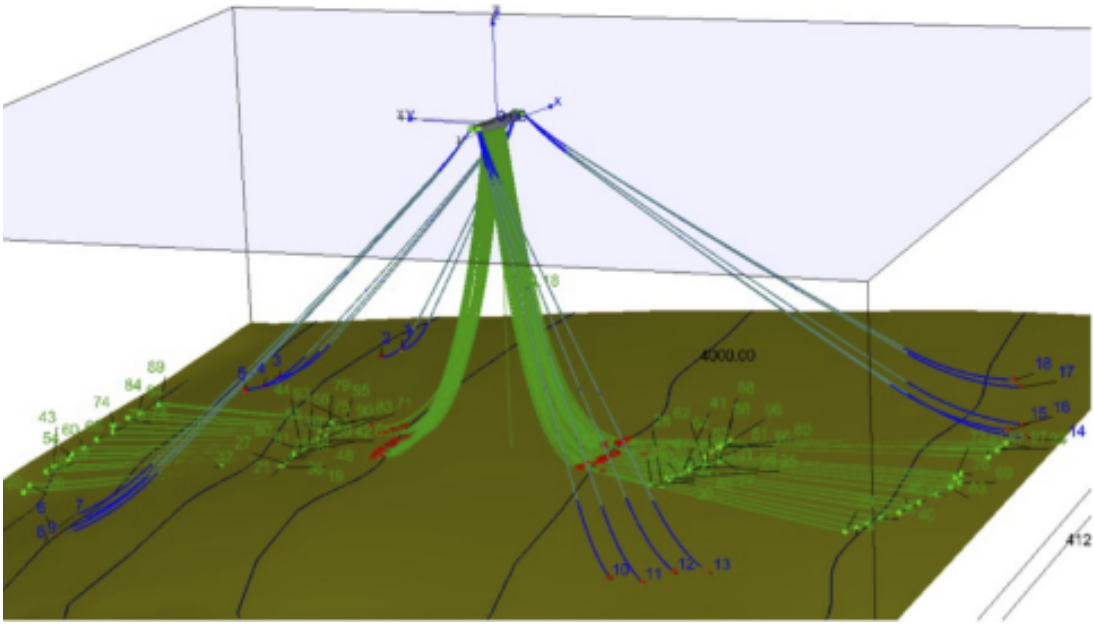

Figura 5: Desenho esquemático representando uma plataforma petrolífera flutuante e suas estruturas. Os fios verdes representam os Risers, dutos pelos quais é feita a extração do petróleo. Já os cabos com pontas azuis são os cabos de amarração, responsáveis pela fixação da plataforma.

Os sistemas de amarração são extremamente importantes, visto que as embarcações estão constantemente enfrentando diversas condições ambientais como correntes marítimas, vento e ondas que podem retirá-las de sua localização original.

Os cabos de amarração estão dispostos em grupos ao redor da plataforma, como mostra a figura 6, e possuem sua outra extremidade fixada no leito submarino de forma a fixar a plataforma na posição escolhida, realizando sua ancoragem. A integridade das linhas de amarração são cruciais para o bom funcionamento das embarcações de petróleo mas, além disso, elas garantem, também, a segurança das pessoas, visto que uma falha nos sistemas de ancoragem pode vir causar acidentes que, por sua vez, podem causar risco à vida não apenas dos tripulantes da embarcação mas também da população como um todo.

Além do risco à vida humana, outro risco que pode ser provocado devido à falhas no sistema de amarração de plataformas petrolíferas é o risco ambiental. Um acidente com a embarcação pode ocasionar o derramamento do petróleo em alto mar, o que pode resultar na morte de diversos animais marinhos, além de interferir significativamente nesse ecossistema.

Os cabos de amarração estão constantemente sujeitos à falhas: por estarem inseridos dentro água marinha, as linhas de ancoragem estão sujeitas à corrosão e, somando isso

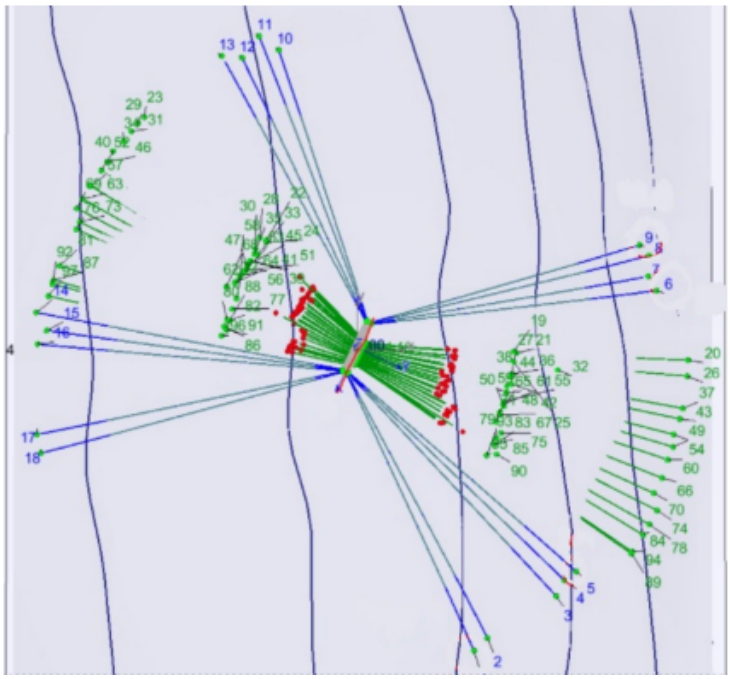

Figura 6: Desenho esquemático representando uma plataforma petrolífera flutuante e os grupos de amarração. Os fios verdes representam os Risers, pelos quais é feita a extração do petróleo. Já os fios azuis são os cabos de amarração, que são organizados em grupos e responsáveis pela fixação da plataforma.

ao fato da possibilidade de tensões excessivas ocasionadas por alguma condição ambiental serem aplicadas, as quebras nos cabos são uma possibilidade. De fato, estudos realizados demonstram que pelo menos 45% das falhas em sistemas de amarração estão relacionados à corrosão e fadiga [14, 15]. Além disso, quando há um rompimento em uma dessas estruturas, as outras passam a ter de suportar uma carga adicional, o que contribui para maiores rompimentos: segundo [14], quando há falha em uma das linhas, a taxa de degradação das demais é aumentada.

Uma questão alarmante a respeito desse contexto é o fato da dificuldade relacionada à identificação dos casos de rompimento das linhas de amarração: grande parte das plataformas localizadas no Mar do Norte, por exemplo, não possuem um método efetivo para essa tarefa, sem a possibilidade de monitorar as linhas em tempo real ou de mensurar o deslocamento da plataforma [16], o que pode resultar em um longo tempo de operação com uma ou mais linhas rompidas. Isso pode se tornar ainda mais urgente, visto que, como mencionado acima, a operação de embarcações com ao menos uma linha rompida aumenta ainda mais a possibilidade de mais linhas se romperem.

Uma forma encontrada de resolver esse problema é através do uso de aprendizado supervisionado para a identificação das falhas nas linhas de amarração da plataforma através do monitoramento da série temporal resultante da movimentação da plataforma

em seus graus de liberdade [17]. Esse tipo de solução foi proposto visto que o processo de verificação dos cabos é feito de forma inteiramente manual, o que se configura um processo lento e com custo elevado.

As abordagens recentes propõem considerar a sequência de movimentações da plataforma retirada de sensores localizados na embarcação como uma série temporal multivariada e utilizar algoritmos de aprendizado de máquina para a previsão desses dados. Em seguida, a série originada do modelo previsor é comparada com a série original oriunda dos sensores e, por fim, essa diferença é usada para treinar um classificador linear que irá, então, definir se houve algum rompimento nos cabos de amarração. Outra abordagem, mais desenvolvida, segue o mesmo caminho, porém propondo, também, identificar em qual dos grupos de amarração que houve o rompimento.

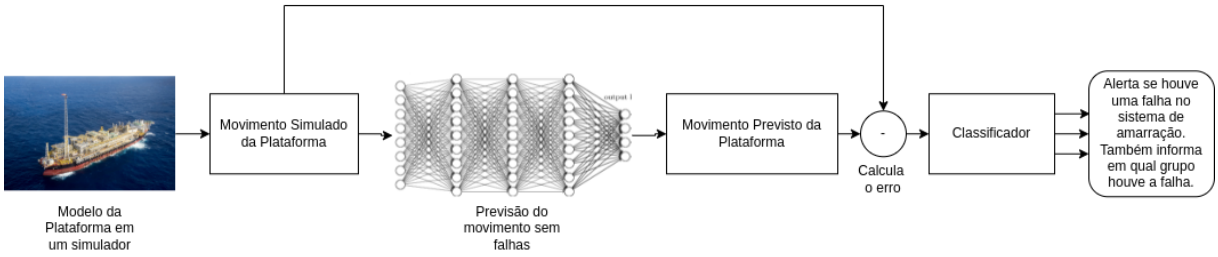

Figura 7: Figura exemplificando o pipeline proposto por abordagens recentes.

Na abordagem que será utilizada neste trabalho, porém, o problema de detecção de falhas em linhas de amarração de plataformas offshore será tratado como um problema de classificação de séries temporais, propondo uma abordagem end-to-end, sem muitas etapas, para esse problema. Mais detalhes sobre o projeto dessa solução estão na sessão 5.

3 MÉTODOS E MATERIAIS

Para realizar a comparação entre modelos classificadores de séries temporais multivariadas, alguns recursos são necessários. Além dos recursos computacionais, ou seja, um computador com memória e poder de processamento suficientes para executar redes neurais e arquiteturas similares, dados de *benchmark* também se tornam uma necessidade, visto que são exemplos padronizados utilizados no treinamento de modelos para que, assim, seja possível compará-los no que diz respeito à métricas de eficácia e eficiência.

Outros recursos importantes estão relacionados à infraestrutura, como sistemas e plataformas desenvolvidos por terceiros que, de alguma forma, dão suporte ao projeto desenvolvido. Dessa forma, neste capítulo são listados e detalhados os principais recursos utilizados durante a realização deste projeto.

3.1 Dados de Benchmark

Os dados de *benchmark* levantados para a realização da comparação entre os modelos de MTSC foram selecionados a partir do conjunto de datasets utilizados em [5]. Assim, nos tópicos a seguir serão descritos de forma mais detalhada os datasets selecionados para uso nos experimentos desse projeto.

3.1.1 Epilepsy

Os dados foram coletados a partir de participantes usando um acelerômetro 3D no pulso dominante. Todos os exemplos das quatro classes, caminhando, correndo, serrando e simulação de convulsão (enquanto sentado), foram gravados por durações diferentes de tempo. A frequência de amostragem foi de 16Hz. Cada participante realiza cada atividade pelo menos dez vezes. As simulações de convulsão foram treinadas e controladas seguindo um protocolo definido por um especialista médico.

Algumas atividades duraram cerca de 30 segundos, outras 1 minuto e outras cerca de

2 minutos. Esses dados foram truncados para o comprimento da série mais curta. Depois dos ajustes no conjunto de dados, foi gerado um total de 275 casos.

3.1.2 BasicMotions

Os dados foram coletados por estudantes da Universidade da Ânglia Oriental. Nesse dataset, os participantes foram convidados a colocar um smartwatch e realizar atividades básicas como correr, andar, descansar e praticar badminton. Essas atividades são, também, as classes do dataset.

Os dados originados foram retirados a partir de acelerômetros e giroscópios presentes no smartwatch. Cada participante realizou a gravação de cada atividade cinco vezes, com a duração de 10 segundos cada e frequência de amostragem de 10 Hz.

3.1.3 Cricket

Os dados foram coletados a partir de dois acelerômetros colocados nos pulsos direito e esquerdo de quatro árbitros do jogo de críquete. Durante a coleta de dados, os árbitros desempenham sinais de sinalização comuns no jogo: por exemplo, o pedido do árbitro para ver o replay de uma jogada é sinalizado pela mímica do contorno de uma televisão, enquanto o sinal de "no-ball"é sinalizado tocando cada ombro com a mão oposta.

Cada acelerômetro grava dados correspondentes a três dimensões: x, y e z. Assim, o dataset conta com dados em 6 dimensões (3 de cada sensor). Os dados consistem em cada árbitro realizando os 12 sinais, com 10 repetições cada. A frequência de registro desses dados foi em 184 Hz.

3.2 Dados do Problema Real

A técnica MTSC será aplicada no problema real de detectção do rompimento no sistema de amarração de plataformas *offshore*. Para isso, dados provenientes do simulador Dynasim serão usados, os quais descrevem as séries temporais dos seis graus de liberdade de movimentação da plataforma.

3.2.1 Dynasim

O Dynasim é um sistema de simulações hidrodinâmico elaborado pelo Tanque de Provas Numérico da Universidade de São Paulo. A partir da utilização do Dynasim, torna-se possível simular os movimentos de um navio a partir de seu modelo hidrodinâmico e do conjunto de condições ambientais que se deseja simular, como configuração do vento, ondas, correntes marítimas, entre outros. O resultado final dessa simulação é uma série temporal multivariada que representa posição, veocidade e aceleração de uma embarcação em seus seis graus de liberdade.

Neste projeto, o Dynasim será utilizado para simular o movimento da plataforma petrolífera P-50, a unidade flutuante de maior capacidade do Brasil e que possui a característica de produzir, processar, armazenar e escoar óleo e gás, segundo a própria Petrobras. Para gerar essa simulação, serão utilizados os modelos hidrodinâmicos da plataforma com diferentes níveis de carga do navio (chamados de calados), bem como conjuntos de condições ambientais provenientes de modelos matemáticos. A partir dessa combinação de modelo hidrodinâmico e modelos de condições ambientais, todos fornecidos pela Petrobras, gera-se séries temporais de movimento em diferentes situações, que serão analisadas a partir dos modelos estudados e comparados nesse projeto. Um ponto importante a ser destacado é que o processo de simulação realizado pelo Dynasim possui um custo computacional muito elevado. Dessa forma, para a realização do processo, os *clusters* do Tanque de Provas Numérico[1] são utilizados.

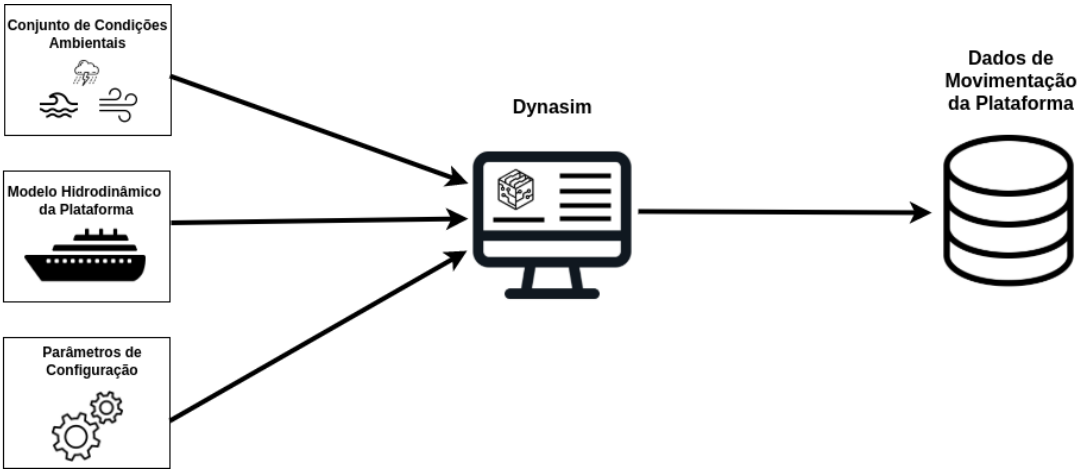

Figura 8: Diagrama esquemático demonstrando o funcionamento do Dynasim.

[1] https://tpn.usp.br

3.3 Tecnologias Utilizadas

Além de um hardware que comporte a execução dos modelos e, também, dos datasets que serão utilizados para treinar os modelos, algumas tecnologias auxiliares desempenham um papel importante na implementção do projeto. Assim, nesta seção iremos detalhar um pouco mais sobre elas.

Python: A linguagem Python foi escolhida como a linguagem de programação a ser utilizada para a implementação dos modelos e tratamento dos dados. A linguagem é comumente utilizada em aplicações de Machine Learning e Ciência de Dados, possuindo um grande número de bibliotecas e *frameworks* conhecidas que são amplamente difundidas, atualizadas e assistidas por uma grande comunidade.

Sistema de Gerenciamento de Simulações: O Sistema de Gerenciamento de Simulações (SGS) é utilizado para prover a automação das simulações realizadas pelo Dynasim. Para isso, o SGS mantém o registro das simulações e de todos os parâmetros utilizados para gerá-las, como calado e condição ambiental, garantindo que não haja nenhum tipo de confusão no momento de utilização dessas simulações por algum modelo.

3.4 Aplicação Prática: Problema de detecção de falhas em linhas de amarração de plataformas petrolíferas

Apesar de não ser o ponto principal deste projeto, que tem como objetivo realizar a comparação dos modelos classificadores de séries temporais multivariadas, a aplicação no problema de detecção de falhas em linhas de amarração de plataformas petrolíferas tem uma função importante: demonstrar a importância do uso desses classificadores em problemas presentes em nosso cotidiano. Por isso, os requisitos funcionais e não funcionais do problema foram levantados, com a intenção de reforçar os pontos mais importantes a serem observados nos resultados.

3.4.1 Requisitos Funcionais

Os requisitos funcionais definem as funções de um sistema. O requisito funcional representa o que o sistema faz em termos de tarefas ou serviços. Assim, os requisitos

funcionais do problema de detecção de falhas em linhas de amarração de plataformas petrolíferas offshore podem ser definidos como:

Realizar o pré-processamento dos dados: Os dados relacionados ao problema de amarração necessitam passar por uma etapa de processamento. O centro de coordenadas do sistema necessita ser transferido para a popa do navio para facilitar o cálculo por parte dos modelos. Além disso, as séries temporais precisam ser divididas em janelas, processo conhecido como janelamento, para que possam ser inseridos nos modelos. Por fim, todas as séries precisam ter o mesmo tamanho, de forma que alguns segundos que possuem efeito de transição são descartados. Dessa forma, esse processo necessita ser realizado pelo sistema todas as vezes que novas séries forem introduzidas no sistema.

Classificar o estado do sistema de amarração: Ao receber as séries temporais contendo o movimento da plataforma, o sistema deve identificar o estado do sistema de amarração. Dessa forma, o sistema deve detectar se há ou não uma falha em alguma linha de amarração a partir dos dados de movimento da plataforma.

Especificar grupo de rompimento: Além de classificar se houve ou não falha, o sistema deve, em caso de falhas, especificar em qual dos grupos houve o rompimento. Assim, quando houver a detecção do rompimento na linha de amarração, o sistema deve especificar se a linha rompida está no grupo 1, 2, 3 ou 4.

3.4.2 Requisitos Não Funcionais

Os requisitos não funcionais são os requisitos relacionados ao uso da aplicação. Assim, os requisitos não funcionais expressam como as funcionalidades do sistema devem ser entregues ao usuário.

Altos valores para as métricas de avaliação: A classificação do estado do sistema de amarração deve ser feito com alta confiabilidade. Dessa forma, os modelos classificadores devem atingir valores elevados nas diferentes métricas de avaliação, o que demonstra que a aplicação dos modelos é adequada e que utilizá-los é uma solução segura para o problema.

Capacidade de Generalização: O sistema deve possuir boa capacidade de generalização. É muito importante que o modelo consiga generalizar os movimentos aprendidos por movimentos gerados por diferentes configurações, como é o caso das condi-

ções ambientais, de forma que não haja diferença significativa entre as considerações realizadas nos diferentes tipos de movimento. Ou seja, os modelos devem conseguir aprender com os diferentes tipos de movimento e, no momento de teste, nao devem apresentar diferenças significativas na classificação, independente do tipo de condição ambiental considerada.

Tempo de treinamento: O tempo de treinamento do modelo é muito importante para a aplicação no problema. Caso uma linha de amarração realmente se rompa, outra deverá ser colocada no lugar e isso poderá modificar a movimentação da plataforma, afetando o funcionamento do modelo (problema conhecido como *drift*). Caso isso ocorra, o modelo deverá ser treinado novamente, o que deve acontecer em tempo hábil pois a plataforma continuaria em funcionamento enquanto isso. Por isso, em nossos experimentos limitamos o tempo de treinamento em 12 horas.

4 DESENVOLVIMENTO DO PROJETO

Este trabalho é, antes de tudo, um trabalho de pesquisa científica e, portanto, segue o método científico descrito em [18] como guia na condução do projeto. Dessa forma, o método de trabalho está dividido principalmente em duas etapas: a primeira busca comparar modelos classificadores de séries temporais multivariadas em diferentes dados de benchmark, enquanto a segunda busca empregar esses modelos na resolução do problema de amarração de plataformas petrolíferas offshore.

4.1 Comparação de Modelos Classificadores de Séries Temporais Multivariadas

Esta etapa pretende comparar alguns dos modelos classificadores de séries temporais a fim de observar seu comportamento frente a diferentes tipos de dados e, assim, realizar uma análise a respeito de suas características. Com o fim dessa etapa, pretende-se levantar as principais vantagens e desvantagens de cada modelo e, assim, obter informações técnicas mais aprofundadas de cada classificador.

Para realizar essa comparação, passos anteriores são necessários, como a seleção dos modelos classificadores que serão utilizados na comparação, além dos dados de *benchmark* que serão utilizados para treinar os modelos e das métricas que serão usadas para avaliar os resultados obtidos. Além disso, o projeto dos experimentos comparativos também é uma etapa necessária. Dessa forma, o método de escolha de cada um desses passos está descrito a seguir:

- **Escolha dos modelos:** A seleção dos modelos levou alguns critérios em consideração, como a proposta do modelo (se pretende superar algum algoritmo relevante já existente e em quais métricas isso ocorreria) ou a sua notoriedade (se é algum algoritmo já consagrado e que vem sendo amplamente utilizado pela comunidade científica) e até mesmo sua sinergia com o tema (modelos que normalmente não

seriam selecionados mas que, se acrescentados, podem proporcionar uma coerência maior aos experimentos). Nesse contexto, um artigo recente [5] que também compara modelos classificadores de séries temporais propõe que em estudos futuros fossem utilizados dois modelos: Rocket [3], um modelo recente que pretende superar o modelo estado-da-arte e DTW [6], um modelo que, apesar de clássico, proporciona um estudo mais aprofundado sobre os modelos por esse mesmo motivo. Esses dois modelos cumprem os critérios inicialmente propostos e, por isso, foram selecionados. Além deles, o InceptionTime [4] também foi selecionado, por mostrar resultados promissores em [5] e se apresentar como uma alternativa ao Rocket. Os modelos selecionados estão detalhados no capítulo 2.

- **Escolha dos dados de benchmark:** A escolha dos dados de benchmark é uma escolha importante, pois as características das observações que os compõem influenciam nos resultados que iremos obter dos modelos: cada modelo funciona a partir de um princípio diferente e, por isso, alguns modelos podem identificar características e padrões dos dados que outros não podem e vice-versa. Assim, selecionar dados com padrões diferentes nos garante que iremos diversificar os estímulos apresentados aos modelos, diminuindo o risco de apresentar dados com viéses que venham a beneficiar somente um tipo de classificador. Dessa forma, os benchmarks escolhidos foram retirados do repositório das Universidades UCR/UEA, um repositório com uma grande variedade de datasets com características distintas. Os dados selecionados estão descritos no capítulo 3.

- **Escolha de métricas:** As seleção das métricas segue as comumente usadas no contexto de *Machine Learning*, como acurácia e precisão. No contexto de comparação de modelos, porém, algumas outras métricas acabam se destacando, como indica [13], um artigo que descreve o método para a comparação de modelos classificadores em múltiplos datasets. Mais detalhes sobre as métricas selecionadas são descritos em 2.

- **Projeto dos experimentos:** Os experimentos de comparação entre os modelos classificadores exigem, além dos tópicos levantados acima, de um projeto que aborde de forma mais elaborada como esses experimentos serão executados: como será a divisão entre dados de treino e dados de teste, se algum tipo de pré-processamento será acrescentado em cada dataset e se a avaliação dos resultados será feita em cima de uma única resposta do modelo ou se será calculada a média entre uma quantidade de resultados em um mesmo dataset. Dessa forma, projetou-se dois experimentos: o primeiro busca comparar o resultado dos modelos nos dados da

forma que estão disponíveis no repositório e o segundo busca acrescentar ruído nesses dados e verificar como os modelos reagem. Para o primeiro experimento, a divisão entre dados de treino e dados de teste foi usada conforme sugerido no repositório e os resultados obtidos são provenientes de múltiplas execuções dos modelos nos dados de benchmark, um procedimento que diminui o impacto de possíveis variáveis aleatórias nos modelos. Já o segundo experimento utiliza a mesma divisão de teste/treino, porém terá um pré-processamento para acréscimo de ruído nos dados, como em [19] e, ao contrário do primeiro experimento, as métricas serão obtidas a partir de uma única execução dos modelos. As figuras 9 e 10 ilustram cada uma das etapas desses experimentos. Maiores detalhes de cada experimento estão descritos a seguir.

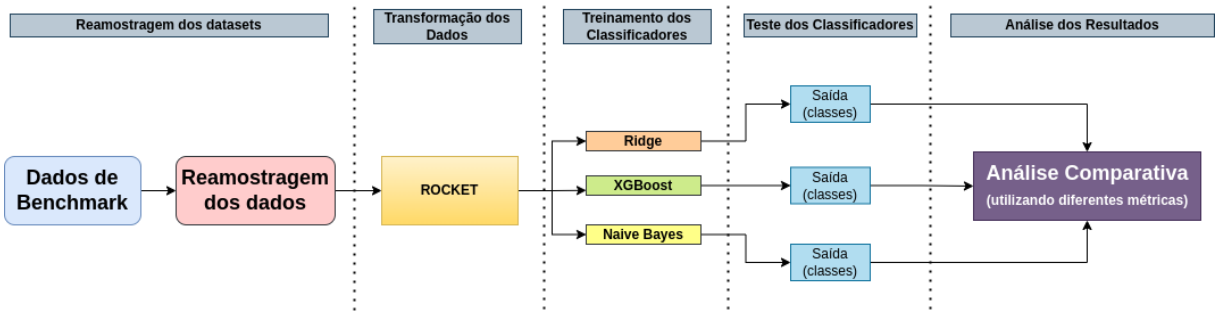

Figura 9: Desenho esquemático representando as etapas do experimento 1: A análise de sinergia com o transformador ROCKET.

Na figura 9, temos representadas as etapas que compõem o nosso primeiro experimento: a análise de sinergia com o modelo transformador ROCKET. Ao todo, são 4 as etapas: (i) Reamostragem dos dados, onde cada dataset é usado para gerar conjuntos de dados formados por amostras do dataset original; (ii) Transformação dos dados, em que esses datasets reamostrados são utilizados pelo modelo ROCKET para transformar os dados de séries temporais em dados com características lineares; (iii) Treinamento dos classificadores lineares, em que os dados com caracteristicas lineares obtidos com o passo ii são utilizados para treinar os modelos; (iv) Teste dos classificadores, em que dados préviamente separados são usados para testar o resultado do classificador, a partir da coleta de diferentes métricas e (v) Análise dos resultados, em que as métricas obtidas no passo iv são analisadas estatísticamente e comparadas entre si.

Já na figura 10, o experimento de análise de sensibilidade ao ruído dos modelos classificadores é dividido em 4 etapas: (i) Atribuição de Ruídos, onde um ruído aleatório é gerado e acrescentado à série objeto de estudo; (ii) Treinamento dos modelos, onde os modelos são treinados com esses dados ruidosos; (iii) Teste dos Classificadores, onde um conjunto de dados de teste é utilizado para avaliar a capacidade de decisão dos modelos treinados e

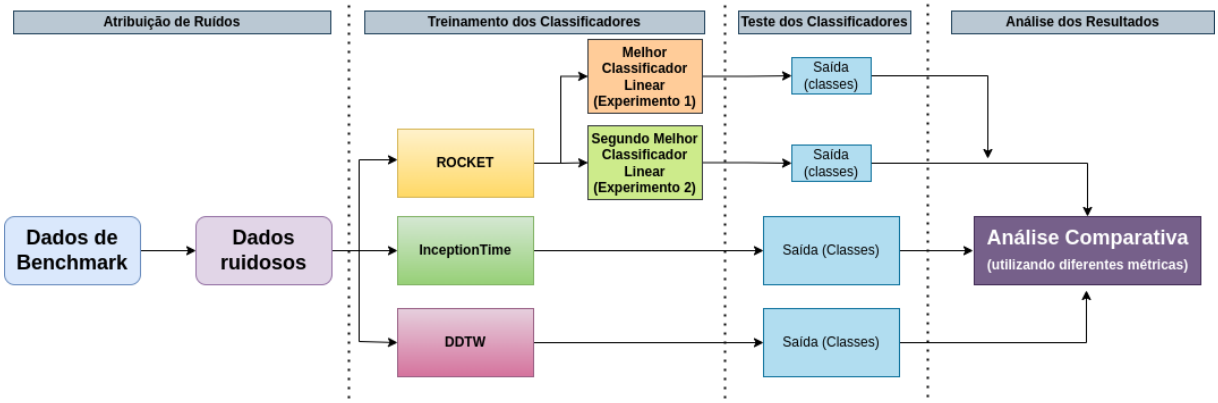

Figura 10: Desenho esquemático representando as etapas do experimento 2: A análise de sensibilidade dos modelos ao ruído.

onde, também, utilizamos diferentes métricas para coletar o desempenho desses modelos e (iv) Análise dos resultados, onde as métricas de cada modelo são comparadas entre si.

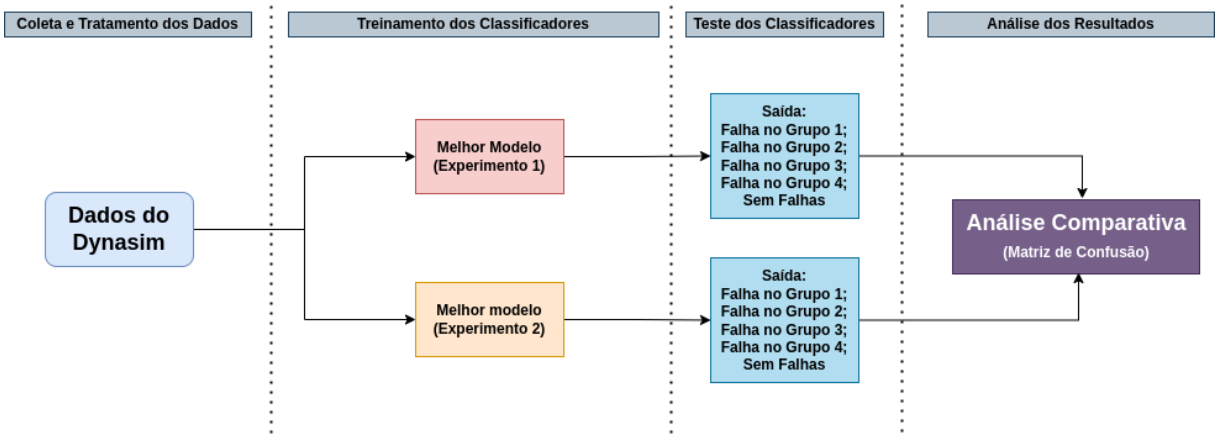

Figura 11: Prova de Conceito: Modelos Classificadores de Séries Temporais Multivariadas aplicados em uma aplicação real.

Por fim, na figura 11, o exemplo de aplicação prática de modelos classificadores de séries temporais multivariadas é representado. Nesse sentido, são 4 as etapas que compõem o experimento que ilustra a utilidade desse tipo de classificador: (i) Coleta e tratamento dos dados, onde os dados são coletados do simulador Dynasim e processados para eliminar algumas partes das séries que não interessam para a classificação; (ii) Treinamento dos classificadores, onde os melhores modelos de cada um dos experimentos são selecionados e treinados nos dados provenientes da etapa 1; (iii) teste dos classificadores, onde um conjunto de dados de teste é usado para avaliar a saída de cada um dos classificadores testados, que indica em qual dos grupos de cabo de amarração da plataforma apresentou falha (ou se não há falha nenhuma) e (iv) Análise dos resultados, onde utilizamos a matriz de confusão e as métricas que dela se originam para comparar os resultados entre os classificadores.

4.1.1 Experimento 1: Análise de sinergia com o modelo ROCKET

Conforme explicado na seção 2, o modelo ROCKET é um modelo que transforma o dataset de séries temporais em um dataset de características lineares, para que então esse novo dataset possa ser processado por um classificador linear. Dessa forma, três classificadores lineares foram selecionados para realizar um estudo comparativo em busca de entender qual seria o melhor modelo linear para combinar com o modelo ROCKET: o classificador Ridge, Naive Bayes ou o XGBoost.

4.1.1.1 Ridge Classifier

O Classificador Ridge [10] é um modelo classificador baseado no método de regressão Ridge. O método é o mesmo que o da regressão, porém os rótulos são convertidos em [-1, 1] antes da implementação do modelo. A regressão Ridge é um método para estimar os coeficientes de modelos de regressão múltipla em cenários onde as variáveis linearmente independentes são altamente correlacionadas. Ele tem sido usado em muitos campos, incluindo econometria, química e engenharia.

Na regressão linear padrão, um vetor-coluna $n \times 1$ y é projetado em um espaço de colunas $n \times p$ de uma matriz X em que as colunas são altamente correlacionadas. O estimador clássico de mínimos quadrados dos coeficientes (beta pertencente a reais elevado a px1), pelos quais as colunas são multiplicadas para obter a projeção ortogonal (Xbeta) é representado por:

$$\hat{\beta} = (X^T X)^{-1} X^T y, \tag{4.1}$$

em que X^T é a transposição de X.

Em situações em que as variáveis dependentes do problema de regressão (as colunas de X) são altamente correlacionadas, a matriz inversa acima pode ser um cálculo muito complexo a ser realizado. Nesse caso, a regressão Rige é utilizada, onde os coeficientes são calculados utilizando a fórmula alternativa:

$$\hat{\beta}_{ridge} = (X^T X + kI_p)^{-1} X^T y, \tag{4.2}$$

onde I_p é a matriz identidade $p \times p$ e $k > 0$ é pequeno.

4.1.1.2 Naive Bayes

O classificador Naive Bayes [20] é um classificador probabilístico baseado no Teorema de Bayes. Para isso, o modelo gera uma tabela de probabilidades de acordo com o dataset em mãos, de acordo com a seguinte expressão:

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)} \tag{4.3}$$

- P (B | A) significa a probabilidade de B acontecer já que o evento A se confirmou;
- P (A) é a probabilidade de A acontecer;
- P (B) é a probabilidade de B acontecer.

É importante ressaltar que o modelo Naive Bayes desconsidera qualquer relação entre as variáveis do dataset, recebendo o nome de Naive (ingênuo) por conta desta característica. Por exemplo, se temos como objetivo classificar frutas, tivermos como classe a fruta "Maçã"e tivermos como *features* as características "Vermelha", "Redonda"e "10 centímetros de diâmetro", a correlação entre esses fatores não será considerado e essas características serão tratadas individualmente.

4.1.1.3 XGBoost

O Extreme Gradient Boosting, ou XGBoost [21], é um classificador baseado em árvores de decisão e aumento de gradiente proposto como um projeto de pesquisa na Universidade de Washington. O aumento de gradiente se baseia na utilização do algoritmo Gradient Descent, que busca minimizar o erro quadrático conforme novos exemplos vão sendo adicionados. Já as árvores de decisão consistem em métodos onde existe uma função que recebe um vetor de valores (de atributos) como entrada e retorna uma decisão (de saída), executando uma série de etapas, ou testes, criando várias ramificações ao longo desse processo.

Uma das principais vantagens do XGBoost é a sua alta escalabilidade, tendo apresentado resultados 10 vezes mais rápidos que soluções populares [21]. Isso se deve a diversas otimizações sistemicas e algorítmicas realizadas, que incluem um novo algoritmo de aprendizado de árvore para lidar com dados esparsos e uma ponderação teoricamente justificada no procedimento de esboço de quantil que permite lidar com pesos de instância na aprendizagem aproximada da árvore.

4.1.1.4 O experimento

Para realizar o experimento, os 3 datasets descritos em 2 (Epilepsy, Cricket e BasicMotions) foram usados para gerar 30 novos datasets cada, de forma que esses datasets são reamostras do dataset original. Para isso, foi utilizado o método *resample()* da biblioteca *scikit-learn* [1].

Cada modelo foi, então, treinado e testado para cada dataset gerado (sendo os dados de treino diferentes dos dados de teste), gerando resultados de Acurácia e F1-Score intermediários, que foram depois utilizados para calcular o valor médio do desempenho do modelo no dataset, como exemplifica a figura x. Esse processo foi realizado para evitar que o efeito de variáveis aleatórias interferissem no resultado. Para o treinamento dos modelos, foram utilizados os parâmetros padrão de cada modelo.

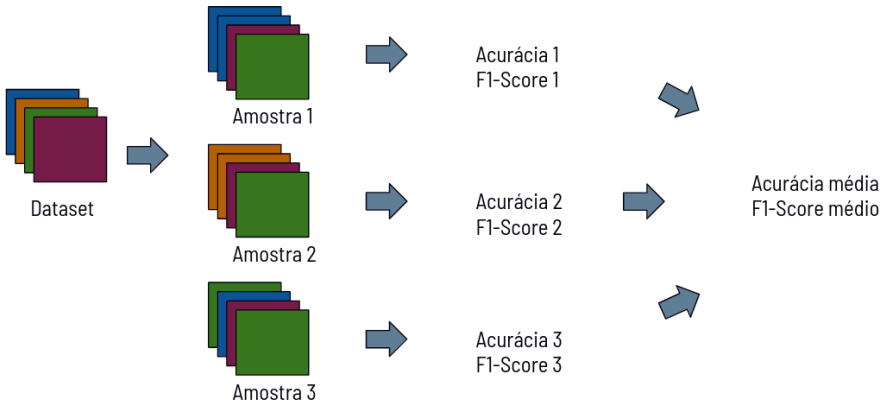

Figura 12: Desenho esquemático mostrando o processo de reamostragem feito em cada dataset.

4.1.2 Experimento 2: Análise de Sensibilidade a Ruído dos Modelos

Dados presentes no mundo real contêm ruído (valores irrelevantes, expúrios ou faltantes) que afetam significativamente os resultados de algoritmos de classificação de séries temporais multivariadas, fazendo com que sejam uma preocupação essencial [22]. Esse tipo de evento nos dados acaba gerando diversos estudos acerca da sensibilidade de modelos na área de Machine Learning, como em [19] e em [23], especificamente em classificação de séries temporais.

Nesta etapa, alguns experimentos de sensibilidade ao ruído foram aplicados aos modelos classificadores selecionados e, para isso, foi utilizado apenas o dataset Epilepsy, citado

[1] https://scikit-learn.org/stable/

na sessão 3. Este é um dataset em que todos os modelos de MTSC apresentaram alta precisão ao serem avaliados, o que nos ajuda na degradação do sinal com diferentes níveis de ruído. Além disso, o dataset é similar à uma das aplicações práticas que serão utilizadas neste trabalho: o problema de amarração de plataformas petrolíferas.

O método empregado em [19] foi também empregado neste experimento para a análise de sensibilidade. Para isso, cada instância do dataset epilepsy é somada a um ruído branco. Ruído branco é um tipo de sinal com distribuição normal, média zero e desvio padrão finito σ, conhecido como nível de ruído. Assim, para cada intervalo de tempo t de cada instância do dataset, o ruído s_t é amostrado do intervalo $\mathcal{N}(0, \sigma)$ e os novos valores $x'_{t,j} = x_{t,j} + s_t,\ j = 1, 2, 3$ são calculados.

Nosso experimento empregou valores de nível de ruído tal que:

$$\sigma \in (0, 0.1, 0.2, 0.3, 0.4, 0.5, 1.0). \tag{4.4}$$

Os geradores de números aleatórios possuem uma semente fixa, com propósitos de reprodução. Dessa forma, 7 datasets diferentes foram criados para o experimento de sensibilidade ao ruído: um sem ruído e outros com níveis de ruídos variados. A figura a seguir mostra um exemplo de como o nível de ruído impacta em uma única série temporal.

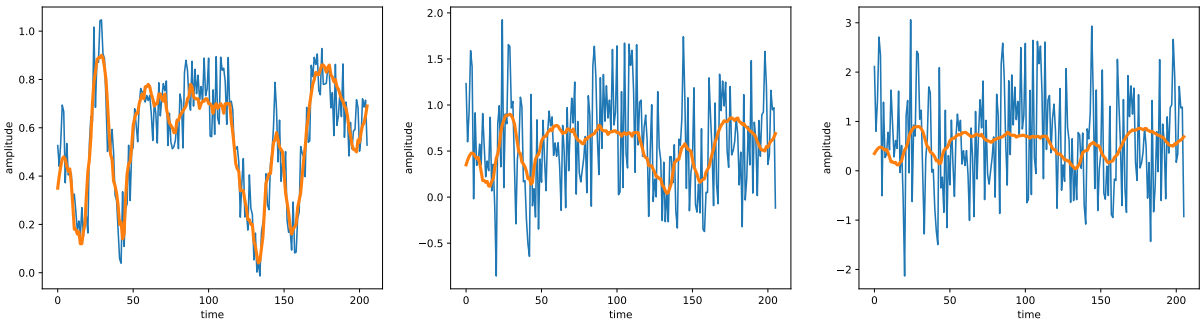

Figura 13: Exemplo de uma dimensão de uma instância do conjunto de dados de epilepsia (em laranja) com diferentes níveis de ruído (da esquerda para a direita, 0,1, 0,5 e 1).

4.2 Identificação de falhas em cabos de amarração com modelos classificadores de séries temporais multivariadas

Como uma prova de conceito, foi realizada a aplicação dos diferentes modelos estudados nos experimentos 1 e 2 em um problema real, que possui grande importância no mercado de plataformas flutuantes: a detecção de falhas em cabos de amarração. Assim,

pretende-se, além de comparar esses modelos em um contexto que possui uma quantidade massiva de dados e ver seu comportamento, também provar a utilidade de modelos classificadores de séries temporais multivariadas nos diversos problemas presentes no dia a dia.

Para concluir esta etapa, os dados utilizados foram obtidos do simulador Dynasim, um simulador desenvolvido pela equipe do Tanque de Provas Numéricas da USP, em parceria com a empresa Petróleo Brasileiro S.A.. O simulador recebe um conjunto de condições ambientais, como velocidade do vento e velocidade das ondas marítimas, além do modelo hidrodinâmico de uma embarcação e da condição da linha de amarração (se houve falha em um dos grupos ou nenhuma falha). No caso de haver o rompimento em algum dos grupos de amarração, o grupo também é especificado, bem como o momento em que essa quebra ocorrerá. Com esses dados, é produzido um conjunto de séries temporais multivariadas que descrevem a movimentação nos seis graus de liberdade da embarcação que teve seu modelo hidrodinâmico utilizado. A figura 14 representa os movimentos horizontais de uma plataforma, que são os movimentos mais afetados quando uma quebra ocorre.

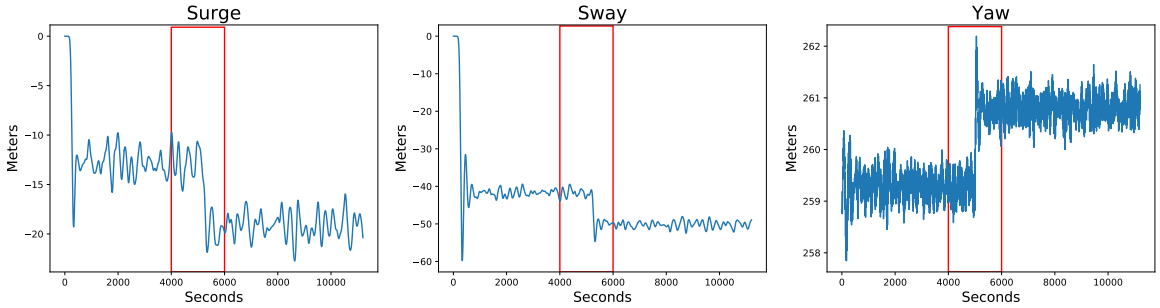

Figura 14: Exemplo de séries representando os três movimentos horizontais da plataforma. O retângulo vermelho destaca o momento de transição resultante de uma quebra em algum cabo de amarração.

Para este experimento, foi utilizada uma condição ambiental específica: o tamanho do calado, que representa a profundidade em que o navio está submerso na água, como mostrado na figura 15. O valor escolhido para o calado foi de 16 metros, o valor mais frequente para a plataforma nessa aplicação. O modelo hidrodinâmico utilizado nesta simulação foi fornecido pela empresa Petróleo Brasileiro S.A. e as condições ambientais foram retiradas de uma estação meteorológica localizada na bacia de Campos do Rio de Janeiro, de 2003 a 2006 em intervalos de 3 horas, o que totaliza 18000 condições ambientais. Para cada condição ambiental, o Dynasim origina uma série de 3 horas de movimento da plataforma.

Após a obtenção das séries, uma etapa de processamento das séries foi iniciada. Foram

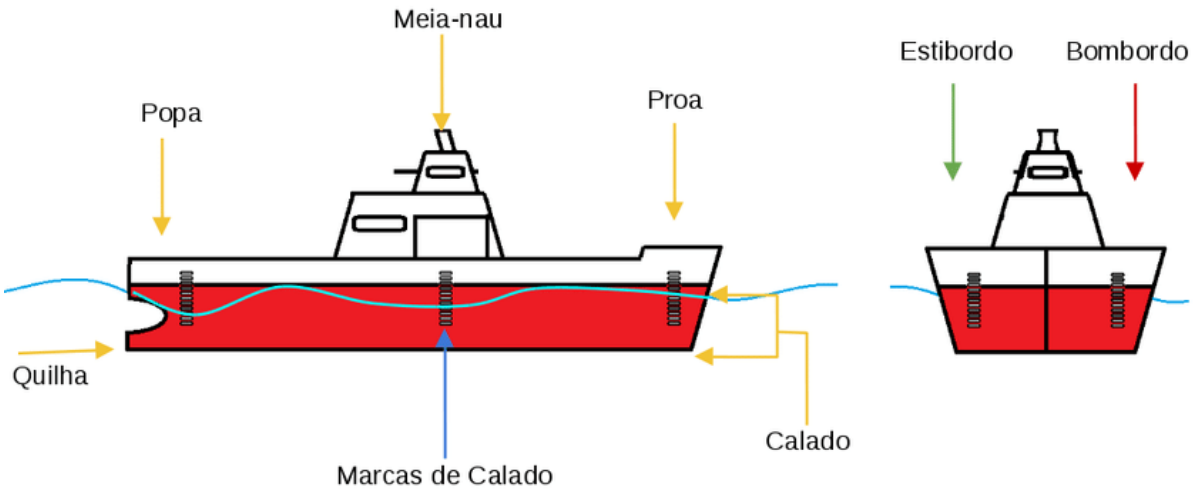

Figura 15: Figura exemplificando as nomenclaturas relacionadas a um navio.

descartados os primeiros 3600 segundos de cada série, por estarem sujeitos a efeitos de transição; 1400 segundos de algumas séries também foram removidos, para garantir que todos os dados (com ou sem falha) tenham o mesmo tamanho, já que as simulações com rompimento de linha de amarração foram ajustadas para ocorrer aos 5000 segundos de simulação. Uma mudança de eixo de coordenada também foi aplicada aos dados para que o seu centro estivesse na popa da plataforma. Por fim, os dados foram normalizados e cada série temporal foi dividida em janelas não sobrepostas de 600 segundos, cada uma anotada com uma das 5 classes definidas para o problema: sem rompimento ou rompimento em um dos 4 grupos de amarração.

As 18000 condições ambientais foram divididas em 7 grupos, das quais 7002 condições foram amostradas para serem simuladas no Dynasim, garantindo uma diversidade de situações nas quais a plataforma está sujeita. Após o janelamento das séries, obtêm-se um conjunto de dados com 63018 janelas igualmente distribuídas entre as 5 classes. Este conjunto de dados foi dividido em conjuntos de treinamento (60%), validação (20%) e teste (20%).

5 RESULTADOS E DISCUSSÕES

Nesta seção são apresentados os resultados obtidos a partir dos experimentos detalhados na seção 4. Além disso, os resultados também serão discutidos a fim de ressaltar possíveis conhecimentos construídos a partir das comparações realizadas, oferecendo contribuições científicas significativas. Dessa forma, o capítulo está dividido em três tópicos, referentes a cada um dos experimentos realizados.

5.1 Experimentos de análise e comparação

Os experimentos de análise e comparação são os experimentos relacionados à exploração dos modelos classificadores em diferentes situações utilizando os dados de benchmark. Os resultados apresentados são discutidos tendo em vista a natureza dos modelos selecionados e seu desempenho em cada etapa, buscando explorar as contribuições científicas obtidas.

5.1.1 Experimento 1: Análise de sinergia com o modelo ROCKET

Neste tópico, os resultados obtidos após a realização dos experimentos são apresentados. Além disso, uma discussão qualitativa e quantitativa desses resultados também será realizada. Abaixo, são apresentados os desempenhos de cada um dos classificadores lineares (Naive Bayes, Ridge e XGBoost), expressos pela média dos resultados nas 30 reamostras de cada dataset, seguindo métricas convencionais para a avaliação de classificadores, como a acurácia e o F1-Score. Junto à essas métricas, também foi acrescentado o valor de acurácia mínimo e máximo obtido nas 30 reamostras de cada dataset, pois expressam resultados interessantes para uma análise posterior.

A partir de uma análise qualitativa dos resultados obtidos, podemos perceber que o classificador Ridge obteve os melhores resultados, alcançando valores de acurácia superiores a 95% em todos os experimentos, sendo que em algumas execuções ele conseguiu a

Tabela 5.1: Resultado dos experimentos para os datasets Epilepsy, BasicMotions e Cricket, expressos em porcentagem (%)

Epilepsy Dataset			
Model	Accuracy	F1-Score	Accuracy (Min. - Max.)
Ridge	**95,74 ± 1,64**	**95,72 ± 1,66**	**92,02 - 99,27**
Naive Bayes	$92,87 \pm 2,10$	$92,96 \pm 2,04$	88,40 - 96,37
XGBoost	$87,48 \pm 3,77$	$87,36 \pm 3,87$	79,71 - 94,2
BasicMotions Dataset			
Model	Accuracy	F1-Score	Accuracy (Min. - Max.)
Ridge	**96,08 ± 2,98**	**96 ± 3,11**	**87,5 - 100**
Naive Bayes	$66,25 \pm 16,8$	$60,29 \pm 20,61$	25 - 97,5
XGBoost	$71,33 \pm 9,04$	$70,46 \pm 9,89$	55 - 90
Cricket Dataset			
Model	Accuracy	F1-Score	Accuracy (Min. - Max.)
Ridge	**98,56 ± 0,77**	**98,55 ± 0,77**	**97,22 - 100**
Naive Bayes	$64,9 \pm 8,08$	$62,86 \pm 8,3$	47,22 - 79,16
XGBoost	$68,61 \pm 6,88$	$68,11 \pm 7,3$	56,94 - 81,94

atingir o valor de 100%, como podemos observar na última coluna da tabela 5.1. Quanto aos outros classificadores, podemos observar que o Naive Bayes teve um desempenho em acurácia melhor no dataset epilepsy quando comparado ao XGBoost, o que não acontece nos outros datasets, nos quais o Naive Bayes obteve resultados inferiores aos outros classificadores.

Outro ponto importante pode ser destacado a partir da observação dos grandes desvios padrões apresentados pelos modelos XGBoost e Naive Bayes e, além disso, pela grande diferença entre os valores mínimo e máximo observados nos quais, para um mesmo dataset, o Naive Bayes chegou a apresentar um valor de acurácia de 25% (menor valor) e 97,5% (maior valor); já o XGBoost apresentou acurácia de 55% (menor valor) e 90% (maior valor). Nesse contexto, podemos entender que o sistema de suavização implementado pelo classificador Ridge acaba diminuindo possíveis inconsistÊncias em sua classificação, reduzindo também possíveis oscilações em seus resultados.

A fim de obter uma análise quantitativa dos resultados obtidos, o teste de Bonferroni-Dunn foi implementado. O teste leva em consideração a média dos rankings dos modelos nos 3 datasets, modelos considerados similares são ligados por um traço horizontal. O diagrama obtido consta a seguir.

O diagrama indica que, nesses experimentos, não podemos afirmar que há diferenças significativas entre os resultados obtidos pelo Ridge e pelo XGBoost, pois eles são considerados similares. Também não podemos afirmar que tivemos diferenças significativas entre

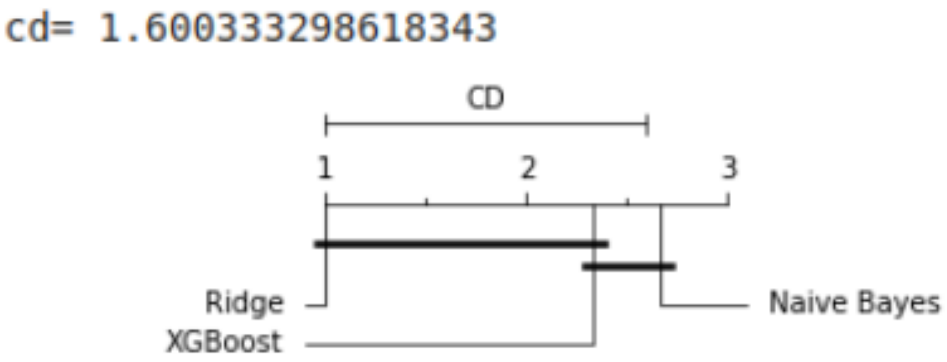

Figura 16: Diagrama de Diferença Crítica utilizando o teste de Bonferroni-Dunn.

os resultados apresentados pelo XGBoost e pelo Naive Bayes. O que podemos afirmar, segundo o diagrama, é que há resultados significativos entre o classificador Ridge e o classificador Naive Bayes, sendo o Ridge um classificador com melhor desempenho. Apesar de obterem resultados bem abaixo do ridge, é interessante observar que o Naive Bayes, um classificador mais simples e sem métodos sofisticados pôde se comparar ao XGBoost, mesmo que este último tenha sido treinado com os valores padrão de hiperparâmetros.

É importante ressaltar que, durante a execução desse experimento, não houve nenhum tipo de otimização de hiperparâmetros, o que pode explicar o baixo desempenho do XGBoost, um modelo considerado atualmente um dos melhores classificadores lineares, quando otimizado. Nesse sentido, resolvemos trazer, também, uma ilustração de como ele poderia desempenhar suas funções caso otimizado. Assim, o experimento 1.1 foi proposto, onde utilizamos o dataset Epilepsy para treinar e testar o modelo XGBoost, com as devidas otimizações, além de incluir os outros modelos descritos na seção 2.

5.1.1.1 Experimento 1.1: Análise de desempenho com o XGBoost otimizado

Inicialmente, mostramos na Tabela 5.2 os resultados dos experimentos usando apenas o dataset Epilepsy, o dataset no qual os modelos apresentaram melhores resultados. Os resultados alcançados nestes experimentos demonstram que DDTW, InceptionTime e ROCKET + Ridge apresentam desempenhos semelhantes, conforme esperado de resultados anteriores apresentados em [5]. Um fato interessante, porém, é que enquanto ROCKET + XGBoost apresenta os piores resultados em todas as métricas, ROCKET + XGBoost Otimizado apresenta os maiores valores nas métricas da Tabela 5.2.

Nesse contexto, o ROCKET é frequentemente usado combinado com o classificador Ridge e nossos resultados podem sugerir que combinar o ROCKET com outros classificadores lineares pode ser uma boa alternativa para resolver problemas. Mas não podemos

Tabela 5.2: Epilepsy metrics for each model over 30 resamples in percentage points. The best mean values are in bold.

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**
DDTW	90.36 ± 2.28	91.57 ± 1.89	90.36 ± 2.28	90.31 ± 2.27
ROCKET + Ridge	95.74 ± 1.64	95.81 ± 1.42	95.70 ± 1.45	95.69 ± 1.46
ROCKET + XGBoost	87.48 ± 3.77	88.56 ± 3.54	88.02 ± 3.72	87.94 ± 3.76
ROCKET + Opt. XGBoost	**97.15 ± 1.46**	**97.23 ± 1.41**	**97.15 ± 1.46**	**97.13 ± 1.48**
InceptionTime	96.52 ± 1.67	96.65 ± 1.59	96.52 ± 1.67	96.51 ± 1.67

esquecer o fato de que a pequena melhoria que o ROCKET+Opt. O XGBoost entrega sobre outros modelos como ROCKET+ Ridge, por exemplo, pode não valer os recursos computacionais extras empregados para otimizar os modelos: o ROCKET é um modelo classificador pensado e desenvolvido especificamente para a questão da eficiência de treinamento e, nesse caso, a utilização de um modelo que necessite gastar muito mais recursos computacionais para apresentar uma melhora tão pequena com relação ao modelo padrão pode não vir a ser a melhor decisão.

Assim, mesmo que tenhamos demonstrado que aliar o ROCKET a um novo classificador linear pode aumentar a classificação em comparação ao classificador linear padrão, essa troca deve ser analisada de forma a verificar se o ganho em acurácia compensa o gasto em recursos computacionais. Essa troca deve compensar em aplicações em casos que a precisão da classificação é muito mais importante que o seu tempo de resposta.

Finalmente, apesar da acurácia variada, todos os classificadores se encaixam bem no problema, o que nos ajudará na próxima etapa, que é a análise de sensibilidade ao ruído.

5.1.2 Experimento 2: Análise de sensibilidade ao ruído

As figuras 17 e 18 mostram como cada modelo é sensível a diferentes níveis de ruído introduzidos nos dados de treinamento e teste. Os elementos em azul representam valores mais baixos de precisão e os elementos em vermelho têm valores mais altos de precisão. A figura 19 resume a sensibilidade dos modelos quando o ruído é inserido apenas nos dados de teste.

Fazendo uma análise qualitativa, podemos ver na Figura 19 que InceptionTime tem os maiores valores de precisão ao lidar com ruído branco nos dados de teste, com degradação significativa de precisão apenas para $\sigma > 0.5$, enquanto o desempenho de outros modelos degradam-se continuamente com o aumento dos níveis de ruído. Isso também pode ser visto nas Figuras 17 e 18, mostrando que a maior quantidade de cor vermelha está no mapa classificador InceptionTime. Para comparar os modelos neste experimento, geramos

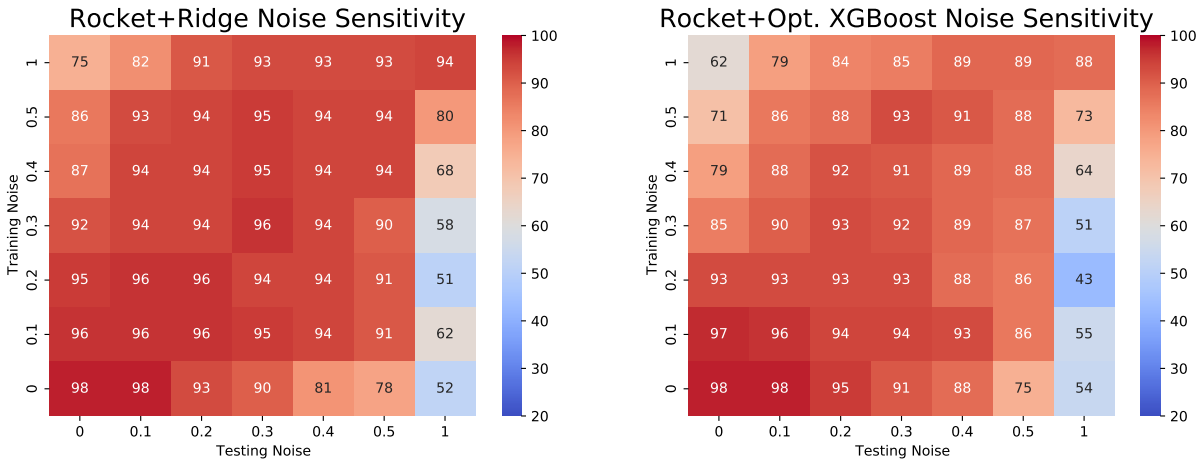

Figura 17: Sensibilidade ao ruído do ROCKET com Ridge e com o XGBoost Otimizado em relação à sensibilidade ao ruído. Sem ruído em ambos os dados de treino e teste, os clssificadores possuem o melhor valor de acurácia (representado em vermelho mais escuro). Os numeros dentro de cada célula representam acurácia.

um Diagrama de Diferenças Críticas usando a análise post-hoc de Wilcoxon-Holm para detectar a significância pareada.

Para gerar o diagrama, consideramos cada combinação de ruído de treinamento e ruído de teste como um único conjunto de dados. Por exemplo, a combinação de treinamento sem ruído e teste com $\sigma = 0,1$ de ruído representa um conjunto de dados, enquanto treinamento sem ruído e teste com $\sigma = 0,2$ de ruído representa outro. Assim, acabamos com 49 conjuntos de dados para comparar os modelos.

De fato, os resultados apresentados pelos modelos são todos estatisticamente diferentes entre si, sendo o InceptionTime o melhor modelo nesta abordagem. InceptionTime é uma CNN e seu melhor desempenho pode ser explicado por sua capacidade de extrair feições devido às convoluções feitas em suas camadas de entrada. Depois disso, todos os recursos são filtrados pelas camadas ocultas da rede, reduzindo ainda mais os efeitos nocivos do ruído na classificação. O diagrama também mostra que ROCKET+Ridge é o segundo melhor classificador. Isso reforça a possibilidade de que classificadores como o XGBoost associado ao ROCKET possam ter menos sinergia com o transformador do que o classificador Ridge, dado seu desempenho bastante sólido. Por fim, o DDTW teve o pior desempenho, com valores de precisão abaixo de 30, quando todos os outros modelos apresentaram valores de pelo menos 43.

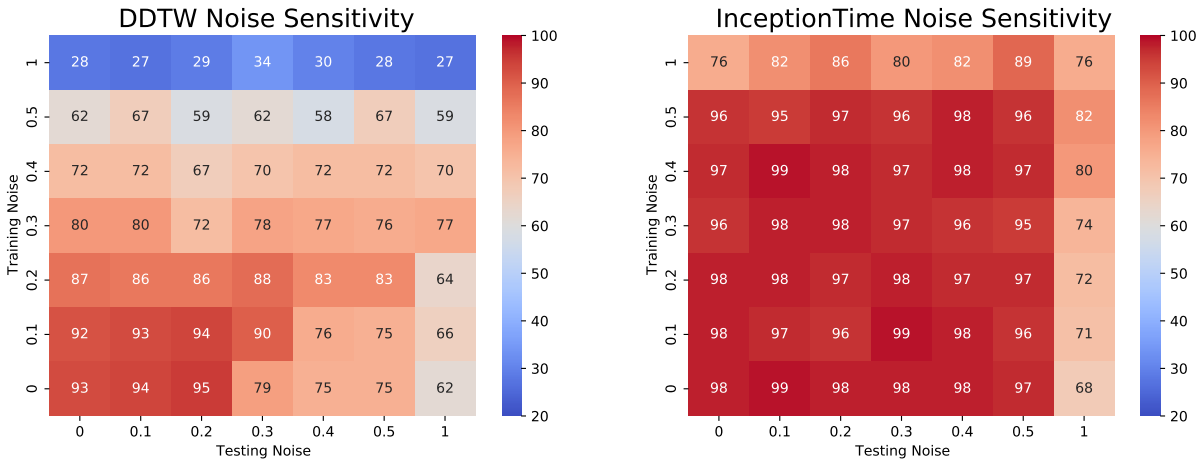

Figura 18: Precisão do Dependent Dynamic Time Warping e InceptionTime em relação à sensibilidade ao ruído. Sem ruído tanto no treinamento quanto no teste, os classificadores têm a melhor precisão (representada pelo vermelho mais escuro). Os números dentro de cada célula indicam acurácia.

5.2 Prova de Conceito: Problema de amarração de plataformas petrolíferas

Os experimentos executados no problema real consistem no treinamento dos 3 modelos (DDTW, ROCKET e InceptionTime) com os dados de movimento da plataforma P-50, conforme explicado acima. É importante observar que, embora este conjunto de dados seja simulado, ele representa um fenômeno natural e, além disso, ainda contém ruído em seus dados.

Alguns ajustes foram feitos com os modelos para executar os experimentos. Para o InceptionTime, a maioria dos parâmetros foram mantidos iguais aos apresentados no artigo original [Ismail Fawaz et al. 2020]. O número de épocas de treinamento foi definido como 500 e o tamanho de janela foi mantido em 64; além disso, o otimizador Adam foi usado com uma taxa de aprendizado de 10^{-3} para coletar os melhores hiperparâmetros para o experimento. Outra configuração importante adicionada ao treinamento foi um escalonador de taxa de aprendizado, que reduziu a taxa de aprendizado quando o modelo parou de melhorar por um certo número de épocas; este componente foi mantido igual à proposição original do InceptionTime. O modelo foi então avaliado no conjunto de teste, resultando em uma precisão de teste de 99,77%. Para o modelo ROCKET, mantemos o número de kernels convolucionais aleatórios em 10.000, como padrão do artigo original. Para os classificadores associados ao ROCKET, foi utilizado o classificador Ridge como padrão, e o XGBoost foi testado com seus valores padrão e com os otimizados.

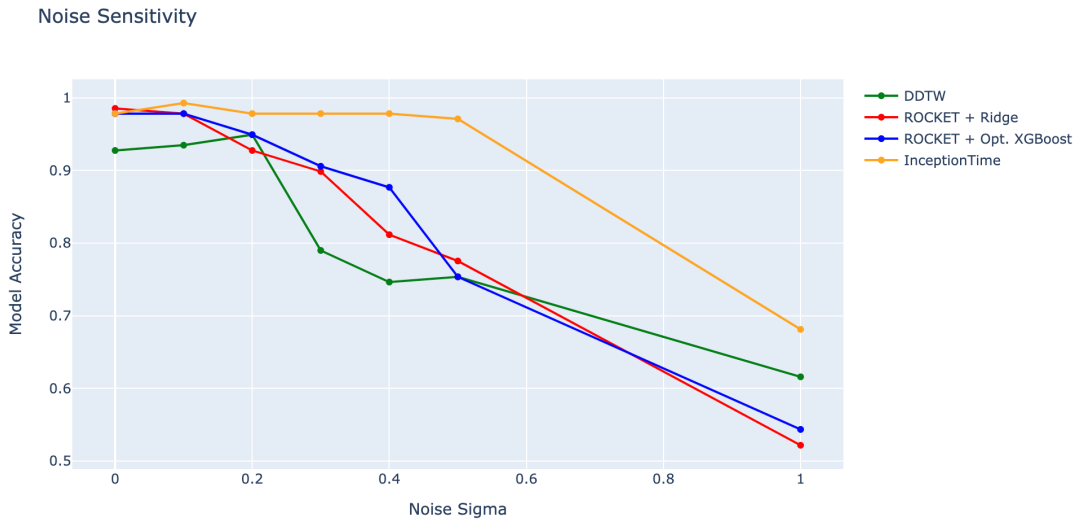

Figura 19: Sensibilidade dos classificadores em relação ao desvio padrão do ruído. Este gráfico mostra a sensibilidade dos modelos quando o ruído é adicionado aos dados de teste, com o treinamento realiado sem ruído.

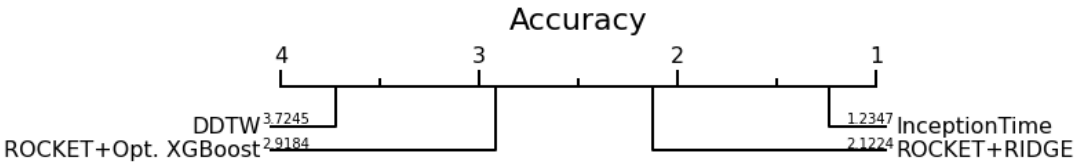

Figura 20: Diagrama de Diferença Crítica acerca do experimento de sensibilidade ao ruído. Modelos mais próximos de "1" apresentam melhores resultados, enquanto modelos mais próximos de "4" apresentam os piores. Uma linha horizontal conectando dois modelos significa que seus resultados não possuem diferença estatística. Aqui, vemos que o modelo InceptionTime apresentou os melhores resultados.

Na Tabela 5.3 reunimos os resultados obtidos nos três modelos. Os resultados alcançados para ROCKET (associado ao classificador Ridge, XGBoost ou Optimized XGBoost) e InceptionTime são muito promissores. Quase todos os exemplos no conjunto de dados foram classificado corretamente com qualquer um desses modelos. Por uma pequena margem, ROCKET com o classificador Ridge teve uma acurácia maior. O Dynamic Time Warping, no entanto, não apresentou resultados positivos: o treinamento DDTW não terminou dentro de um prazo aceitável (12 horas), o que foi muito mais do que o ROCKET e o InceptionTime levaram, destacando a inadequação dos métodos tradicionais de última geração para conjuntos de dados massivos.

Tabela 5.3: Acurácia (%) dos três modelos no experimento com calado igual a 16 metros.

Modelo	Acurácia de Teste	Modelos com ROCKET	Acurácia de Teste
		+ XGBoost	99.77
DDTW	*timeout*	+ Opt. XGBoost	99.91
InceptionTime	99.77	+ Ridge Classifier	**99.97**

6 CONCLUSÃO E TRABALHOS FUTUROS

Este trabalho teve como objetivo estudar mais profundamente modelos classificadores de séries temporais multivariadas, levantando características desses modelos com base nos experimentos realizados por meio de sua comparação frente a diferentes datasets e diferentes tipos de dados. Esse trabalho fez parte do projeto de pesquisa referente ao módulo de Pesquisa Científica em Engenharia de Computação (programa de pré-mestrado) da Escola Politécnica da Universidade de São Paulo. Dessa forma, esse projeto tem um propósito científico.

A partir dos conceitos e experimentos apresentados nesse projeto foi possível observar a grande importância dos estudos realizados na área de séries temporais. Esse tipo de dado se diferencia de dados convencionais por conta de suas características únicas e isso os torna mais desafiadores. Por outro lado, séries temporais possuem grande abundância no mundo atual, estando presentes nos mais diversos tipos de aplicações e a necessidade de classificação desses dados se torna, então, urgente.

Os modelos classificadores que se originam desse contexto vêm evoluindo ao longo dos anos, com a história nos ajudando a perceber que os modelos antes considerados estado-da-arte se tornaram obsoletos por conta de serem extremamente lentos e difíceis de serem aplicados em uma situação real. Atualmente, novos modelos foram desenvolvidos, sendo muito mais rápidos comparados aos modelos anteriores e obtendo valores de precisão parecidos, como é o caso do ROCKET e do InceptionTime, modelos estudados neste trabalho. A área de classificação de séries temporais multivariadas ainda é, porém, considerada uma área em estágio embrionário.

Os experimentos de comparação aqui realizados buscaram, então, contribuir para a comunidade científica detalhando com maior profundidade os modelos que se destacam na área de classificação de séries temporais multivariadas, podendo assim fornecer maiores artifícios para o desenvolvimento da área. Dessa forma, o primeiro experimento buscou explorar o modelo que mais vem se destacando na área de séries temporais multivariadas: o modelo ROCKET. Esse modelo é comumente aliado com um classificador linear, deno-

minado RIDGE, embora outros classificadores lineares também pudessem ser utilizados. Nesse sentido, buscamos associar o ROCKET a outros modelos classificadores para verificar se podemos melhorar ainda mais suas métricas de desempenho e verificar se outros classificadores lineares realmente são compatíveis com o ROCKET.

Já o segundo experimento buscou avaliar o comportamento dos classificadores melhor avaliados em um artigo recente, porém colocando-os frente à dados ruidosos. O ruído é um tipo de sinal decorrente de ínúmeras adversidades que, como demonstrado nos experimentos realizados, representa um desafio para os modelos classificadores. Identificar o comportamento dos modelos nesse tipo de problema nos ajuda a entender os mecanismos utilizados por cada técnica para processar os sinais das séries temporais e, além disso, nos ajuda a entender quais modelos são mais indicados em ambientes com muitas interferências e ruídos.

Já o último experimento foi na verdade uma aplicação prática dos modelos estudados. Esses modelos foram implementados no problema de detecção de falhas de linha de amarração de plataformas petrolíferas *offshore*, um problema complexo, que não foi inteiramente resolvido e que se beneficia significativamente do uso de métodos automáticos para verificação dos cabos de ancoragem das plataformas. Buscou-se, com esta etapa, provar a eficácia e o potencial de aplicação dos modelos classificadores de séries temporais multivariadas, destacando ao mesmo tempo a importância de seu estudo e disseminação e, também, a importância de estruturar os problemas no contexto adequado: nesse caso, a estruturação da questão de falhas de cabos de ancoragem em um problema de séries temporais multivariadas rendeu ótimos resultados, que provavelmente solucionam o problema de amarração.

Assim, podemos concluir com o primeiro experimento, a partir da combinação do ROCKET com outros modelos classificadores, que é possível, sim, utilizar o ROCKET com outros classificadores lineares além do RIDGE e até mesmo obter valores de acurácia maiores, como foi o caso do XGBoost otimizado. Esse aumento em acurácia, porém, não significa que podemos afirmar que o XGBoost otimizado é melhor que o RIDGE para ser associado ao ROCKET: a melhora na classificação adquirida com outro modelo pode não ser grande o suficiente para justificar o gasto em recursos computacionais necessários para otimizar o classificador XGBoost. Dessa forma, com esse experimento também concluímos que, apesar de ser possível associar o ROCKET com outros classificadores lineares, o RIDGE geralmente desempenha um ótimo trabalho, generalizando muito bem para diversos casos, com custo computacional baixo.

Já com o experimento de sensibilidade ao ruído podemos concluir que os ruídos realmente representam um grande desafio a ser superado pelos modelos classificadores: todos os modelos classificadores tiveram sua classificação profundamente alterada conforme o nível de ruído era aumentado. Nesse sentido, o modelo mais robusto, ou seja, o modelo mais tolerante ao ruído, foi o InceptionTime, uma Rede Neural Convolucional, que teve o valor de sua acurácia alterado somente com níveis de ruído maiores que 0.5, o que pode indicar a grande capacidade dessa rede de filtrar as características das séries e extrair informações precisas delas. Dessa forma, em aplicações mais ruidosas, o InceptionTime se mostra um modelo com maior afinidade.

Por fim, com a etapa de aplicação no problema de detecção de falhas de linha de amarração de plataformas petrolíferas, podemos concluir que há um amplo espaço de aplicação para modelos classificadores de séries temporais multivariadas. O excelente desempenho desses modelos no problema de linhas de amarração nos mostra como as séries temporais vão além de um conceito teórico, no qual os conhecimentos construídos auxiliam na criação de ferramentas que acabam se mostrando importantes em nosso dia a dia.

Como trabalho futuro, podemos incluir a aplicação de técnicas de análise de séries temporais para descobrir outras características dos dados que podem impactar os modelos e analisar como eles se sobressaem. Nesse sentido, conforme demonstrado neste trabalho, o ruído se apresenta como uma propriedade do dado que influencia diretamente a classificação dos modelos. Dessa forma, averiguar como os modelos são afetados por outras características é uma maneira de não apenas apontar qual o melhor classificador em determinado contexto, mas também auxiliar, fornecendo informações, para o desenvolvimento de novos modelos.

Outro trabalho que pode ser desenvolvido futuramente é a exploração de outros problemas que possam ser resolvidos com o problema de classificação de séries temporais multivariadas. Como já dito, diversos dados podem ser estruturados como séries temporais e, dessa forma, há diversos problemas que ainda não foram completamente resolvidos que podem se beneficiar de modelos classificadores de séries temporais multivariadas.

REFERÊNCIAS

1 FAWAZ, H. I. et al. Inceptiontime: Finding alexnet for time series classification. *Data Mining and Knowledge Discovery*, v. 34, n. 1936–1962, 2020.

2 LäNGKVIST, M.; KARLSSON, L.; LOUTFI, A. A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognition Letters*, v. 42, p. 11–24, jun. 2014. ISSN 01678655. Disponível em: <https://linkinghub.elsevier.com/retrieve/pii/S0167865514000221>.

3 DEMPSTER, A.; PETITJEAN, F.; WEBB, G. I. ROCKET: Exceptionally fast and accurate time series classification using random convolutional kernels. *Data Mining and Knowledge Discovery*, v. 34, n. 5, p. 1454–1495, set. 2020. ISSN 1384-5810, 1573-756X. ArXiv: 1910.13051. Disponível em: <http://arxiv.org/abs/1910.13051>.

4 BAGNALL, A. et al. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, v. 31, n. 3, p. 606–660, maio 2017. ISSN 1384-5810, 1573-756X. Disponível em: <http://link.springer.com/10.1007/s10618-016-0483-9>.

5 RUIZ, A. P. et al. The great multivariate time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, v. 35, n. 2, p. 401–449, mar. 2021. ISSN 1384-5810, 1573-756X. Disponível em: <http://link.springer.com/10.1007/s10618-020-00727-3>.

6 SHOKOOHI-YEKTA, M. et al. Generalizing DTW to the multi-dimensional case requires an adaptive approach. *Data Mining and Knowledge Discovery*, v. 31, n. 1, p. 1–31, jan. 2017. ISSN 1384-5810, 1573-756X. Disponível em: <http://link.springer.com/10.1007/s10618-016-0455-0>.

7 C, S. et al. Going deeper with convolutions. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. Boston, USA: [s.n.], 2015.

8 S, K.-B.; F, M.; A, M. Scalable classification of univariate and multivariate time series. In: *IEEE International Conference on Big Data*. Seattle, USA: [s.n.], 2018.

9 S, S.; D, W. Randomness in neural networks: an overview. *Data Mining and Knowledge Discovery*, v. 7, n. 2, p. e1200, 2017.

10 MCDONALD, G. C. Ridge regression. *WIREs Computational Statistics*, v. 1, n. 1, p. 93–100, 2009. Disponível em: <https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/wics.14>.

11 DING, H. et al. Querying and mining of time series data: Experimental comparison of representations and distance measures. *PVLDB*, v. 1, p. 1542–1552, 08 2008.

12 KRUSKAL, J.; LIBERMAN, M. The symmetric time-warping problem: From continuous to discrete. *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*, 01 1983.

13 DEMSAR, J. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, v. 7, p. 1–30, 01 2006.

14 MA, K.-t. et al. A historical review on integrity issues of permanent mooring systems. In: *All Days*. OTC. p. OTC–24025–MS. Disponível em: <https://onepetro.org/OTCONF/proceedings/13OTC/All-13OTC/Houston,\\%20Texas,\\%20USA/37626>.

15 FONTAINE, E. et al. Industry survey of past failures, pre-emptive replacements and reported degradations for mooring systems of floating production units. In: *Day 4 Thu, May 08, 2014*. OTC. p. D041S047R002. Disponível em: <https://onepetro.org/OTCONF/proceedings/14OTC/4-14OTC/Houston,\\%20Texas/172127>.

16 BROWN, M. et al. Floating production mooring integrity JIP - key findings. In: *Offshore Technology Conference*. Offshore Technology Conference. Disponível em: <http://www.onepetro.org/doi/10.4043/17499-MS>.

17 SAAD, A. M. et al. Using neural network approaches to detect mooring line failure. *IEEE Access*, v. 9, p. 27678–27695, 2021.

18 WAZLAWICK, R. S. *Metodologia de Pesquisa para Ciência da Computação*. Second. Elsevier, 2014. ISBN 978-85-352-7782-1. Disponível em: <https://www.sciencedirect.com/book/9788535277821>.

19 KALAPANIDAS, E. et al. Machine learning algorithms: a study on noise sensitivity. In: *Proc. 1st Balcan Conference in Informatics*. [S.l.: s.n.], 2003. p. 356–365.

20 WEBB, G. Naïve bayes. In: _____. [S.l.: s.n.], 2016. p. 1–2.

21 CHEN, T.; GUESTRIN, C. Xgboost: A scalable tree boosting system. In: . [S.l.: s.n.], 2016. p. 785–794.

22 GUPTA, S.; GUPTA, A. Dealing with Noise Problem in Machine Learning Data-sets: A Systematic Review. *Procedia Computer Science*, v. 161, p. 466–474, 2019. ISSN 18770509. Disponível em: <https://linkinghub.elsevier.com/retrieve/pii/S1877050919318575>.

23 SCHäFER, P. The boss is concerned with time series classification in the presence of noise. *Data Mining and Knowledge Discovery*, v. 29, n. 6, p. 1–23, 11 2015.