Usando aprendizado de máquina para predição de falhas no processo de desodorização de alimentos

Luis Gabriel Caetano Diniz

Disciplina: Monografia em Sistema de Informação II

Orientador: Adriano Alonso Veloso

Departamento de Ciência da Computação - Universidade Federal de Minas Gerais - Belo Horizonte, Brasil

lgcaetano@ufmg.br

Abstract. *This paper explores predicting vacuum pressure loss in food deodorization using machine learning. Techniques for a wide range of purposes including exploratory data analysis and data preprocessing were utilized on time series data. Many machine learning techniques were utilized from simple models based on the ARIMA algorithm to complex transformer neural networks, with model validation techniques. The balance between the earliness and quality of prediction was a focus. The study analyzes the performance of the models developed using methods that show this relationship between prediction horizon and prediction accuracy.*

Resumo. *Este artigo explora a predição da perda de pressão a vácuo em processos de desodorização de alimentos usando aprendizado de máquina. Técnicas para uma série de propósitos incluindo análise exploratória de dados e pré-processamento de dados foram utilizadas em dados de séries temporais.Diversos algoritmos de aprendizado de máquina foram utilizados, desde modelos simples baseados no algoritmo ARIMA até redes neurais complexas baseadas na arquitetura transformer. O equilíbrio entre a antecipação e a qualidade da previsão foi um foco. O estudo analisa o desempenho dos modelos desenvolvidos através de métodos que ilustram a relação entre horizonte de previsões e acurácia das previsões.*

1. Introdução

1.1 Caracterização do problema

É muito comum em diversas indústrias, existirem uma série de processos de produção que estão sujeitos a falhas que, se fossem evitadas, tornariam a cadeia mais eficiente e econômica. Existe um custo considerável incorrido pelo desperdício de tempo e insumos. Na indústria alimentícia, não é diferente e foi assim que este trabalho surgiu, com o propósito de buscar soluções para um problema constatado em processos dessa cadeia de produção.

O problema encontrado nas cadeias de produção de alimentos diz respeito ao processo de desodorização de alimentos, processo comumente aplicado na produção de óleos e gorduras comestíveis, responsável por remover alguns compostos indesejáveis desses produtos. Este processo possui diversas etapas e é importante que algumas

condições sejam mantidas dentro do desodorizador para que o processo ocorra sem problemas, por exemplo, durante o processo de desodorização de alguns óleos é necessário que estes sejam aquecidos a temperaturas relativamente elevadas e precisam ficar em contato com vapor, que é responsável por remover os sabores e cheiros desagradáveis. Uma das condições que afetam significativamente o processo é a pressão encontrada dentro do desodorizador durante o processo, que geralmente deve estar baixa ou até mesmo próxima ao vácuo.

Contudo, apesar de se ter conhecimento dessa necessidade de manter a pressão em níveis baixos para a realização do processo, é difícil controlar a pressão dentro do desodorizador e prever quando ocorrerá um aumento significativo da mesma que causaŕa falhas no processo. Essa situação foi embrionária para o tema do trabalho, já que foi neste contexto que se viu a oportunidade de se coletar dados sobre o ambiente de desodorização para a aplicação de técnicas da ciência de dados e do aprendizado de máquina a fim de prever a ocorrência de falhas, encontrar padrões estatísticos que indiquem o porquê dessas falhas acontecerem para que elas possam ser evitadas no futuro.

1.2 Motivação e objetivos

Estes são os principais objetivos do trabalho, encontrar maneiras de utilizar da ciência de dados e aprendizado de máquina para ajudar a mitigar um problema real que causa desperdício e ineficiência em uma indústria muito relevante como a alimentícia. Contudo, para que isso seja feito, dados sobre o ambiente tiveram de ser coletados para serem utilizados nas análises e treinamentos dos modelos. Estes dados formam séries temporais, sequências de medições automáticas realizadas nos desodorizadores periodicamente. Dados que possuem esta característica têm de ser tratados de forma específica, a separação da base dados em dados de treino e teste tem que considerar a ordem em que as medições foram feitas, por exemplo. Além disso, o problema em questão acaba tendo outras características também, o que levanta questões interessantes que este trabalho buscará responder.

Uma das questões mais interessantes a se considerar diz respeito ao intervalo entre o momento que o modelo realiza uma previsão de ocorrência de uma falha no processo e a concretização da mesma, já que existe um trade-off que pode ser observado: é desejável que as previsões de falhas ocorram o mais cedo possível, evitando perda de tempo e insumos, mas também é muito provável que seja observada uma relação inversamente proporcional entre o intervalo citado e a acurácia do modelo, ou seja, uma previsão do modelo que indique que o desodorizador irá falhar em algumas horas provavelmente tem menos chance de estar correta do que uma falha idêntica ocorrerá em alguns minutos, contudo, se ambas as previsões estiverem corretas, a primeira terá sido muito mais útil, pois terá evitado maiores desperdícios.

Portanto, além dos objetivos principais, que buscam atacar o problema encontrado no processo de desodorização, existem questões relacionadas ao aprendizado de máquina que se deseja observar e se levar em consideração durante o processo de desenvolvimento dos modelos, como este trade-off citado, seria muito enriquecedor se fosse possível encontrar uma espécie de equilíbrio nessa situação, por exemplo.

1.3 Estrutura do trabalho

O Capítulo 2 fornece um referencial teórico, identificando trabalhos correlatos e estabelecendo a base acadêmica para este estudo.

No Capítulo 3, apresentamos nossa contribuição para este campo de pesquisa. Nele, descrevemos em detalhe as atividades realizadas, desde o entendimento dos dados, passando pelo seu pré processamento, até a modelagem e validação dos modelos de aprendizado de máquina. Também discutimos as técnicas usadas para lidar com séries temporais e a abordagem adotada para lidar com o trade-off entre antecipação e precisão na previsão de falhas.

O Capítulo 4 conclui o estudo, destacando as principais conclusões e sugerindo direções para trabalhos futuros.

O trabalho inclui também uma lista de referências bibliográficas.

2. Referencial teórico e trabalhos correlatos

Para este trabalho, foi escolhido um problema bastante específico, que, apesar de relevante, pouco foi tratado do ponto de vista acadêmico na área da ciência de dados e aprendizado de máquina, algo que não é surpreendente, já que representa apenas parte de um processo produtivo que possui diversas fases e que não é muito conhecido por pessoas leigas. Contudo, apesar da escassez de referências bibliográficas tratando deste problema específico, foi possível encontrar muito material relacionado a problemas que podem ser considerados similares. Os dados neste caso formam uma série temporal com medidas de condições do ambiente de desodorização, essas medidas incluem temperatura de componentes do desodorizador, pressão de componentes, vazão de insumos e produtos secundários, etc., e o objetivo principal é criar modelos de aprendizado de máquina que sejam capazes de realizar previsões de perda vácuo neste processo. São estas características que indicam as maiores similaridades com outros tipos de problema.

Pode-se dividir os trabalhos correlatos em certos grupos de acordo com como eles se relacionam com este trabalho e porque eles foram considerados relevantes.

2.1 Manutenção Preventiva utilizando Aprendizado de Máquina

Dentre os trabalhos e materiais correlatos encontrados, aqueles que mais se assemelham a este, são os que tratam de problemas de manutenção preventiva no contexto da ciência de dados e aprendizado de máquina, já que são trabalhos que lidam com séries temporais e tentam prever a ocorrência de uma falha em um processo e que, portanto, acabam tendo de lidar com questões e obstáculos extremamente similares aos encontrados no desenvolvimento deste trabalho.

Estes trabalhos não só suportam a viabilidade do propósito deste trabalho, já que

tratam de exemplos reais o uso de aprendizado de máquina para previsão de falhas em processos industriais como trazem conhecimento sobre a utilização de diversas técnicas neste contexto. Por exemplo, foram encontrados trabalhos que demonstram a eficácia do uso de redes neurais recorrentes para manutenção preventiva em motores de aeronaves (Hermawan et al., 2020), uma situação onde a tolerância a falhas é praticamente nula, trabalhos que dão uma visão mais ampla do tema de manutenção preventiva, explorando diversas técnicas de aprendizado de máquina clássico e aprendizado profundo e apontando por exemplo as vantagens de cada abordagem, apontando por exemplo que o uso de técnicas de aprendizado profundo pode ser mais eficaz e preciso, mas menos explicável (Sohaib et al., 2021), o que é muito relevante neste contexto, já que evidentemente é desejável que exista a possibilidade de se explicar as previsões eventualmente realizadas pelo modelo, até mesmo para fins de correções a serem feitas no processo analisado. Essa conclusão foi um dos motivos para uma decisão tomada neste trabalho de se experimentar com estas duas classes de modelos e de comparar os resultados. O desafio de lidar com os problemas resultantes de se lidar com dados captados por sensores industriais foi ressaltado também, já que há muito ruído neste tipo de dado, o que faz com que um processo de limpeza dos mesmo seja altamente recomendado (Sohaib et al., 2021).

Apesar da grande utilidade deste tipo de trabalho, poucos foram os exemplares encontrados que se enquadram nesta classe, como mesmo ressalta o trabalho de Sohaib et al., ainda há muito o que se explorar pela comunidade acadêmica se tratando de manutenção preventiva utilizando aprendizado de máquina.

2.2 Séries Temporais

A maior parte do referencial teórico do trabalho pode ser englobada nesta categoria, já que existem muitos e muitos trabalhos tratando de séries temporais e aplicando aprendizado de máquina para realizar a previsão das mesmas, o que foi muito útil, pois foi encontrado conhecimento em todas as fases do desenvolvimento de um modelo de aprendizado de máquina, desde o processo de compreensão e análise dos dados utilizados até a fase de validação, passando pelo processamento dos dados e modelagem.

Na fase de compreensão dos dados, haviam muitas técnicas já preestabelecidas e consolidadas na literatura, como a utilização de gráficos para visualização dos dados e de certas estatísticas derivadas dos mesmos, como média, variância, moda, desvio padrão, etc. Contudo, dada a natureza temporal dos dados em questão, buscou-se encontrar técnicas que permitissem observar caracterĩsticas dos dados, como sazonalidade, tendência, enfim, dependências temporais dos dados. Alguns dos artigos encontrados apresentavam uma ampla gama de técnicas de decomposição de séries temporais para fins de análise (Plummer, 2020) que acabaram sendo utilizadas nesta fase do trabalho, como a decomposição aditiva e a visualização através de ACF (função de autocorrelação) e PACF (função de autocorrelação parcial). Outros conhecimentos foram adquiridos, como técnicas para identificação de relações causais em séries temporais (McCracken, 2016), mas não necessariamente implementados neste trabalho, contudo sua aplicação em trabalhos futuros seria extremamente enriquecedora.

Quanto à fase de pré-processamento dos dados, diversas técnicas foram

encontradas na literatura e analisadas, especialmente relacionadas a seleção de features e engenharia de dados. Neste contexto, foram identificadas diversas técnicas que se vêm muito úteis no contexto das séries temporais, como o uso de médias móveis das features, algo que se vê necessário para utilizar algoritmos clássicos de aprendizado de máquina, que não conseguem lidar com médias temporais automaticamente, features *lags*, que representam valores recentes obtidos na série temporal (Surakhi, 2021) (Cerliani, 2022), além da possível utilização de técnicas mais complexas para criação de novas features, como a utilização de autoencoders (Ntakaris et al., 2019) e outras técnicas de aprendizado não-supervisionado (Längkvist et al., 2014) (Erhan et al., 2010) (Gamboa, 2017). Além disso, foram analisadas técnicas para remoção de possíveis features, para evitar lidar com o problema de *overfitting*, que ocorre quando a dimensionalidade dos dados é grande demais (Pudjihartono et al., 2022).

Um problema encontrado no desenvolvimento do trabalho consistia na dificuldade de encontrar técnicas de *resampling*, já que foram identificados dados sujos na base, que tiveram de ser retirados e *outliers* que podem trazer grande impacto negativo no poder preditivo dos modelos. Como "completar" os buracos remanescentes desse processo de limpeza dos dados? Para isso, foram procurados artigos que identificassem técnicas robustas e mais complexas para preencher estas lacunas (Moniz et al., 2017) (Camponovo et al., 2010), apesar de ter-se decidido por utilizar técnicas triviais, como *backward fill*, já que se mostraram mais viáveis de se utilizar, por serem menos computacionalmente custosas.

A fase de modelagem talvez seja a que mais trouxe conhecimento novo e onde foram encontrados mais materiais relacionados a este trabalho. Muitos deles descrevendo modelos específicos para uso em séries temporais como Prophet (Taylor e Letham, 2017), autoregressão de vetores (Zivot e Wang, 2003), métodos *ensemble* no geral (Gastinger et al., 2021), que levaram à utilização do algoritmo XGBoost neste trabalho (um algoritmo de *boosting*, e portanto, também *ensemble*), além de diversas arquiteturas de redes neurais, especialmente redes neurais recorrentes, como LSTMs, GRUs e redes neurais convolucionais (Karpathy, 2015) (Fawaz et al., 2018) (Sohaib et al., 2021) (Hermawan et al., 2020) (Parmezan et al., 2019) (Jordan et al., 2019) (Han et al., 2019). Além da análise da performance de modelos, outros artigos tratam também da seleção de modelos, sugerindo meta-aprendizado para selecionar modelos *ensemble* (Gastinger et al., 2021) e analisando diferentes formatos de arquitetura de redes neurais para séries temporais (Peter, et al., 2019). Além disso, alguns artigos recentes se mostraram muito pertinentes para o contexto do trabalho, já que tratam da utilização de transformers para previsões de séries temporais. A comunidade HuggingFace se mostrou uma grande aliada na realização desse trabalho, já que sua biblioteca *transformers* para a linguagem Python já trazia consigo a implementação de alguns algoritmos descritos nestes trabalhos. Um desses algoritmos é a implementação original da arquitetura (Vaswani et al., 2017), mas além desta, implementações de algoritmos mais complexos e com arquitetura desenhada especificamente para a resolução de problemas de previsões de séries temporais como o Autoformer(Wu, Haixu, et al., 2021) e o Informer (Zhou, Haoyi, et al., 2021) cujo artigo foi eleito o melhor da conferência AAAI em 2021. Apesar de apresentarem ferramentas interessantes para o contexto do trabalho, havia a dúvida de como seria o desempenho dos transformers considerando algumas ressalvas feitas pelos autores dos artigos citados. Primeiramente, é importante ressaltar que os artigos encontram resultados melhores quando utilizando apenas dados

univariados, o que no nosso caso significaria desperdiçar uma grande abundância de dados que poderiam ser utilizados para modelar relações mais complexas nos dados. Levando isso em consideração, foi tomada a decisão de construir uma variante univariada para cada algoritmo transformer utilizado a fim de verificar se o desempenho destes é de fato superior neste caso. Como é destacado no artigo sobre o Autoformer(Wu, Haixu, et al., 2021), em alguns casos é comum que modelos de séries temporais univariadas superem modelos multivariados devido à dificuldade de algoritmos de modelar relações cruzadas entre as diferentes séries temporais. Alguns artigos recentes apresentam algoritmos cuja arquitetura foi desenhada exatamente com este problema em mente, como o Crossformer (Zeng, Chen et al. 2023), contudo, essas técnicas não puderam ser exploradas neste trabalho, mas parecem ser extremamente pertinentes para este caso.

Finalmente, foram encontrados alguns artigos tratando da fase de validação dos modelos. Evidentemente, os artigos que tratam da seleção de modelos e que comparam a performance de algoritmos no contexto de séries temporais, já tratam parcialmente do processo de validação, já que ela é necessária para estas tarefas, não há como comparar a performance de um modelo sem validá-lo de alguma forma. Geralmente, no caso de séries temporais, se separa uma parte inicial da base de dados para o treinamento do modelo e uma parte final para validação (Bergmeier e Benítez, 2012), garantindo assim, que o modelo não é testado em dados anteriores aos seus dados de treinamento. Contudo, isso tra limitações na fase de validação, já que em problemas clássicos de aprendizado de máquina, pode-se utilizar da validação cruzada para validação, uma técnica que permite que o modelo seja avaliado utilizando quase todo o dado disponível para treino. Por isso, buscou-se encontrar técnicas similares a essa que seriam aplicáveis no contexto do trabalho e foi encontrada uma forma de conciliar o "melhor dos dois mundos": utilizando validação cruzada em bloco (Bergmeier e Benítez, 2012) mantendo a cronologia correta dos dados de treino e teste, mas utilizando o máximo possível dos dados para treino. Contudo, esta técnica não se vê tão útil neste contexto considerando um dos objetivos do trabalho, o de observar e analisar o *trade-off* entre antecipação da previsão e qualidade da mesma. Com a forma de validação original, é possível observar como a performance do modelo se comporta conforme ele realiza previsões cada vez mais distantes no tempo.

3. Desenvolvimento do trabalho

Para a realização do trabalho, foi utilizada a linguagem Python com auxílio de diversas bibliotecas comumente utilizadas em projetos de ciência de dados e aprendizado de máquina, como *scikit-learn*, *pandas*, *statsmodels*, *numpy*, *matplotlib*, *seaborn*, etc. Essas bibliotecas foram utilizadas ao longo do projeto para auxiliar na visualização, análise e preprocessamento dos dados, além da fase de modelagem e validação.

Na fase de modelagem, a biblioteca *keras*, utilizada para modelar a rede neural LSTM e a biblioteca *transformers*, do HuggingFace, utilizada para criar os modelos *transformers*, se mostraram essenciais para o desenvolvimento do trabalho, já que facilitaram muito o desenvolvimento de redes neurais com arquiteturas complexas. A biblioteca *transformers* já possui implementações de três arquiteturas diferentes de *transformers* especificamente criadas para modelar séries temporais, por exemplo.

3.1 Análise exploratória dos dados

O primeiro passo a ser realizado no projeto foi a limpeza de alguns dados problemáticos do *dataset*, estes dados consistiam de medições que possuíam valores para a variável alvo (uma das medições de pressão no desodorizador) acima de 50. Esses dados foram removidos, pois foi constatado que essas medições somente ocorriam em situações atípicas e problemáticas como na ausência de eletricidade, portanto, sua presença poderia ter impacto negativo significativo na performance dos modelos.

Após a remoção destes dados problemáticos, se iniciou o processo de análise e visualização da base de dados. Primeiramente, a base de dados inteira foi plotada em forma cronológica mostrando a variação da variável alvo:

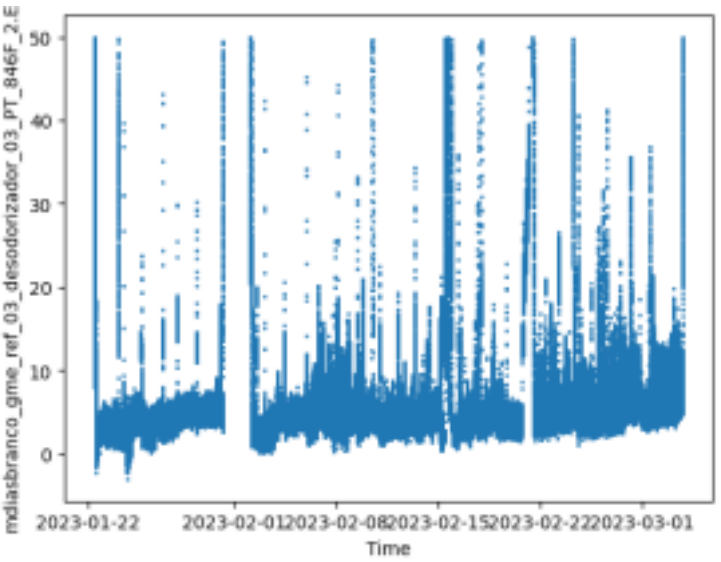

Figura 1. Pressão de vácuo do desodorizador ao longo do tempo

Como pode ser observado na figura, pode-se observar, após a remoção dos dados faltosos, lacunas muito claras ficaram evidentes nos dados, isso é um problema quando lidando com séries temporais, já que trás alguns obstáculos para o processo de criação dos modelos. Um exemplo disso é o fato de que alguns algoritmos para previsão de séries temporais presumem um intervalo temporal uniforme entre os diversos pontos dos dados, o que claramente não ocorre nessa situação. Além disso, foi constatado que apesar da grande maioria dos intervalos entre as medições serem de apenas 3 segundos, haviam diversas ocorrências de intervalos de 6, 9 ou 12 segundos.

Essas observações acabaram levando à decisão de realizar um processo de *resampling* sobre os dados, preenchendo as lacunas existentes e uniformizando o intervalo de tempo entre os diversos dados coletados. Este processo será melhor relatado nas próximas subseções já que faz parte da fase de tratamento dos dados e não da análise exploratória.

Felizmente, não haviam dados com valores faltantes, ou seja, todas as medições ainda remanescentes na base de dados após a filtragem inicial possuíam uniformidade quanto às features.

Em seguida, foram visualizadas e analisadas a média, mediana e desvio padrão da variável alvo:

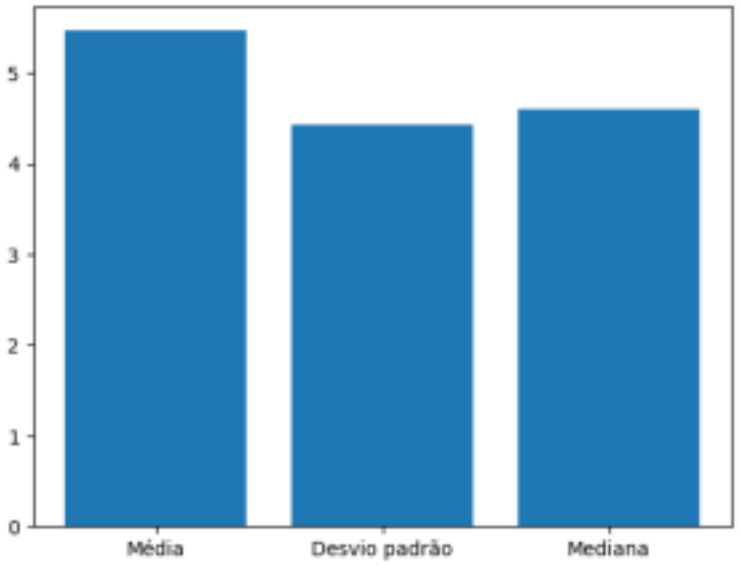

Figura 2. Média, desvio padrão e mediana da variável alvo

Como pode ser observado na figura, o desvio padrão é relativamente grande em relação à média e à mediana, sendo equivalente à 81,12% e 96,26% de cada, respectivamente. indicando grande variedade nos dados, que poderia ser explicada pela presença de *outliers* os dados, possivelmente resultantes de medições errôneas. Um sinal disso é a grande presença de dados que possuem a variável alvo com medições próximas de 50, que foi justamente a quantidade utilizada para delimitar os dados "sujos" presentes no *dataset*. Considerando que a média e mediana da variável alvo ficaram próximas de 5, não é de se estranhar que a presença de anomalias poderia distorcer significativamente os dados, já que a magnitude delas é muito maior do que a dos dados corretamente coletados.

Essa constatação também motivou a utilização de um algoritmo para detecção de outliers que seriam removidos da base futuramente antes do processo de *resampling*.

Já pensando no processo de seleção de *features* e possível enriquecimento dos dados a partir da criação de novas características para os dados, especialmente algumas estatísticas relevantes e comumente usadas no contexto das séries temporais, como variáveis de atraso ("lags"), médias móveis, etc., foi feita a visualização de duas funções muito relevantes para dados em formato de série temporal: a função de autocorrelação (ACF) e a função de autocorrelação parcial (PACF). A ACF mede a correlação linear entre pontos de dados que são separados por um intervalo de tempo específico, o 'lag'. Em outras palavras, a ACF fornece a correlação de uma série com a sua própria versão

defasada. Ela é útil para identificar padrões sazonais ou cíclicos em uma série temporal. Por outro lado, a PACF fornece a correlação de uma série com a sua própria versão defasada, mas após controlar ou eliminar as correlações de todas as defasagens menores. Ou seja, a PACF mede a correlação direta entre pontos de dados que são separados por um número de intervalos de tempo, ignorando qualquer correlação potencial que possa ser explicada por pontos de dados entre esses intervalos.

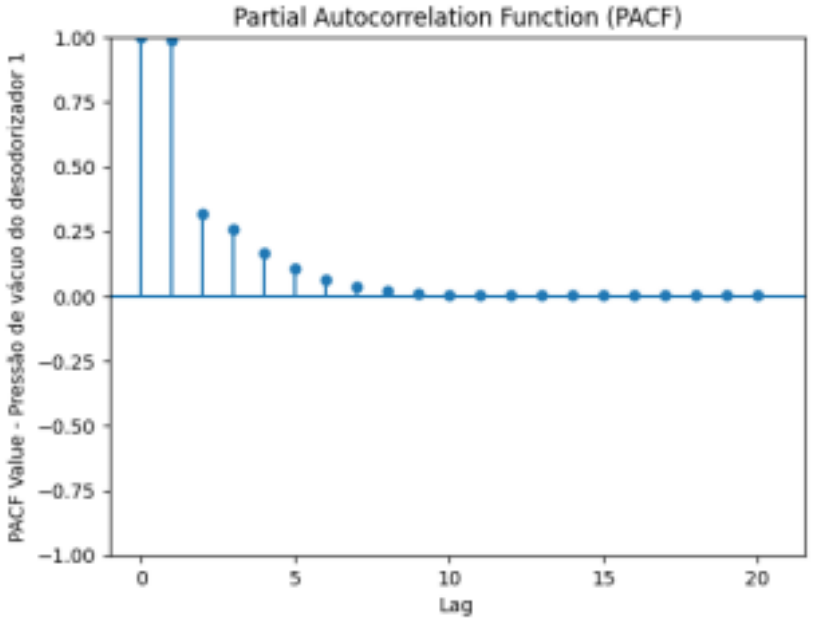

Figura 3. Visualização da PACF da variável alvo

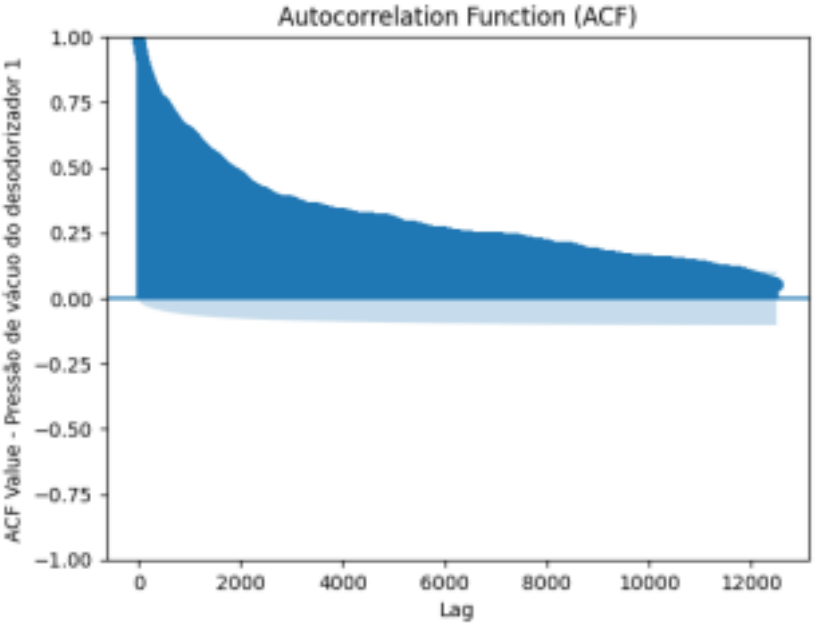

Figura 4. Visualização do ACF da variável alvo, obtido com auxílio da transformada rápida de Fourier

As visualizações dessas funções foram obtidas com auxílio da biblioteca *statsmodels*, e, como ressaltado abaixo da figura 4, o cálculo da função ACF teve de ser

feito com auxílio da transformada rápida de Fourier, já que a forma tradicional consumia muitos recursos computacionais para uma base de dados tão grande quanto a utilizada neste projeto. Além disso, como pode ser observado na mesma figura 4, a biblioteca adiciona automaticamente um intervalo de relevância para os dados sobre a autocorrelação que pode ser observado em um leve tom de azul. Levando isso em consideração, chega-se à conclusão de que as autocorrelações da variável alvo são muito significativas, ou seja, os valores anteriores da variável alvo estão altamente correlacionados com seus valores futuros, já que até uma defasagem de mais de 12000 períodos a autocorrelação ainda se encontra acima do limiar de relevância. Como o período nessas circunstâncias é de 3 segundos, até um intervalo de 10 horas entre medições apresenta autocorrelação relevante. Já a função PACF traz dados diferentes, a autocorrelação parcial fica abaixo do mesmo limiar de relevância após 6 períodos.

Outra forma de visualizar relações relevantes nos dados é observar as correlações entre as features. A visualização da correlação pode ajudar a identificar quais recursos estão mais fortemente relacionados à variável alvo, o que pode ser extremamente útil para fins de seleção de *features*. Além disso, se houverem muitas características altamente correlacionadas, podemos considerar a remoção de algumas delas para reduzir a dimensão do espaço de características, o que pode ajudar a evitar o overfitting e melhorar a eficiência. A correlação entre as *features* pode ser observada no mapa de calor a seguir:

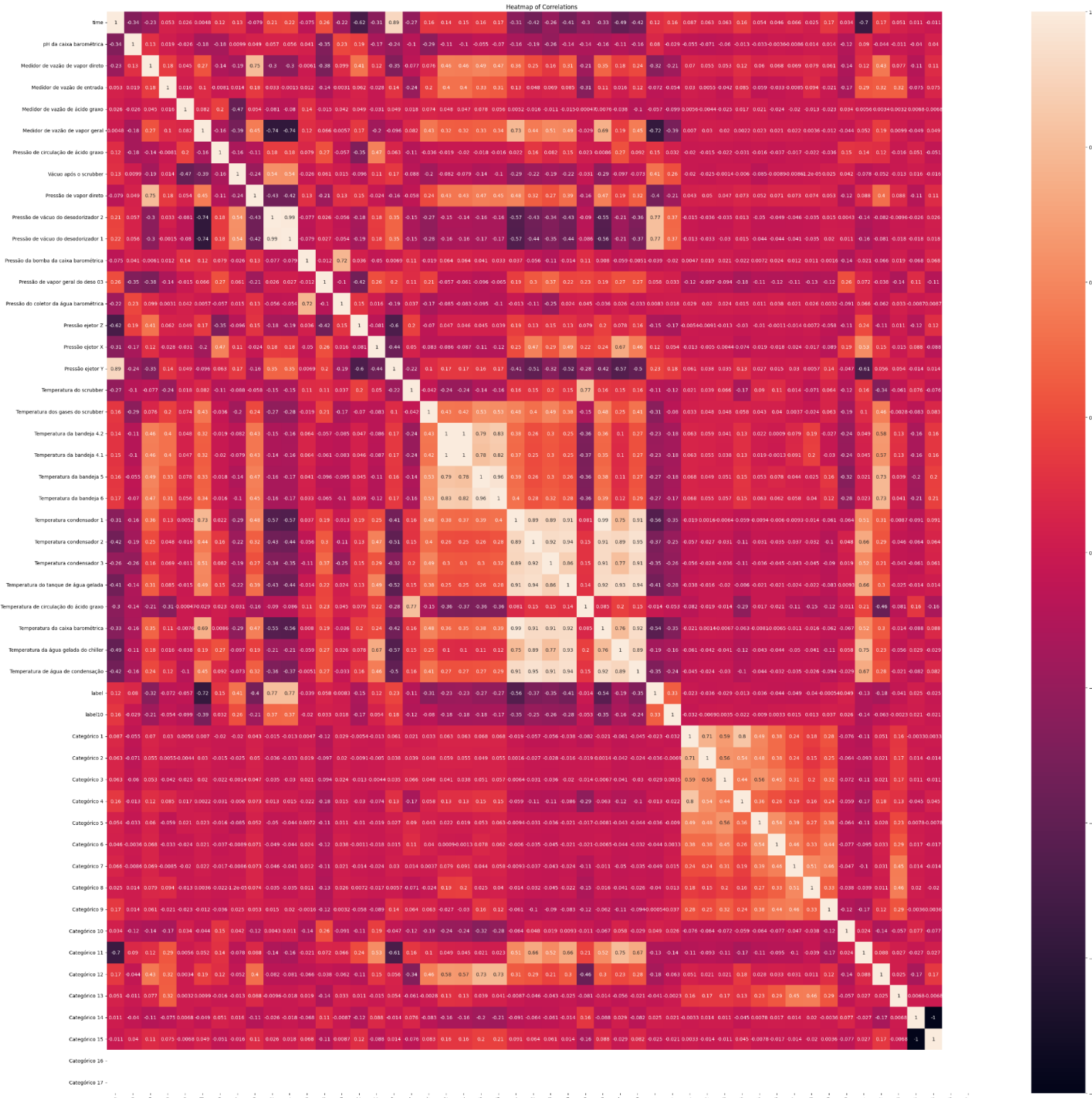

Figura 5. Mapa de correlações das features da base de dados

Como pode ser observado no mapa de correlações, algumas *features* apresentam correlações significativas com a variável alvo, especialmente a temperatura dos condensadores, das bandejas, dos gases do *scrubber*, além da pressão dos ejetores e de vapores, vácuo após o *scrubber* e vazão de vapores. É sempre importante ressaltar que correlação não implica em causalidade, portanto, não se pode afirmar que estes fatores sejam de alguma forma os causadores da pressão alta e da perda de vácuo nos desodorizadores, contudo, somando essas observações ao conhecimento que se tem do domínio do problema, é inevitável que surjam algumas teorizações. Por exemplo, é de conhecimento geral que temperatura e pressão possuem uma relação bem próxima na física, já que substâncias e materiais que possuam temperaturas mais altas tendem a se expandir, ocupar mais espaço. Essa relação é tão importante que até hoje é utilizada na produção de energia elétrica a partir do vapor d'água, cuja pressão move válvulas que por sua vez produzem carga elétrica. Portanto, não seria surpreendente se a causa das perdas de vácuo fosse a temperatura alta de alguns componentes.

A vazão dos vapores também apresenta uma situação similar, como o exemplo dado anteriormente evidencia. O vapor d'água é utilizado muitas vezes em processos exatamente devido à pressão causada por ele.

Por outro lado, é provável que a correlação da variável alvo com outras *features* não represente nenhuma relação implícita de causalidade, a pressão nos ejetores pode estar correlacionada com a pressão de vácuo no desodorizador pelo fato de que algo está causando aumento de pressão em todos os componentes, por exemplo.

3.2 Transformação dos dados

Após o processo de análise exploratória da base de dados, foi iniciado o processo de transformação efetiva dos dados. Primeiramente, foi realizada a análise e remoção de *outliers*. Estes *outliers* foram identificados com a utilização do algoritmo Isolation Forests. Inicialmente, desejava-se utilizar o algoritmo DBSCAN, mas o custo computacional era muito grande e muito tempo também era despendido neste processo, ao contrário do Isolation Forests, que é mais eficiente e simples. O algoritmo encontrou mais de 10000 *outliers* (quantidade varia por execução devido à natureza do algoritmo, já foram encontrados 10682, 10731, 11092, etc.) que foram removidos da base de dados. Estes *outliers* representam cerca de 1% da base, que possui mais de 1 milhão de medições coletadas.

Esta foi a última remoção feita sobre a base de dados, após a realização deste processo foi feito o procedimento de *resampling* dos dados mencionado anteriormente. Como dito anteriormente, a uniformidade do intervalo entre os dados em uma série temporal é fundamental para muitos métodos de análise de séries temporais, que pressupõem que as observações sejam espaçadas igualmente ao longo do tempo. Isso se deve a vários fatores:

1. Muitos modelos estatísticos para séries temporais, como ARIMA, pressupõem observações em intervalos regulares. Se esse pressuposto for violado, as estimativas dos parâmetros do modelo podem ser enviesadas, levando a previsões imprecisas.
2. A estacionariedade é uma propriedade desejável em muitas análises de séries temporais, pois permite que o modelo generalize padrões passados para o futuro.

No entanto, se os intervalos entre as observações forem irregulares, pode ser difícil determinar se a série é estacionária.

3. Ter dados em intervalos regulares simplifica a manipulação de dados, pois permite usar métodos padrão para agregar, interpolar, deslocar e outras operações comuns em séries temporais. Se os intervalos não forem regulares, essas operações podem requerer métodos mais complexos e propensos a erros.

Em resumo, ter um intervalo uniforme em uma série temporal facilita a análise, a modelagem e a interpretação dos dados, tornando mais provável que as conclusões tiradas dos dados sejam precisas e úteis.

Inicialmente, o *resampling* foi feito utilizando uma técnica que pode ser considerada trivial, o *backwards fill*, que preenche as lacunas encontradas na série temporal com o próximo valor válido na série. Esta tática pode não apresentar nenhum problema aparente de início, mas após inspeção um pouco mais minuciosa dos dados apresentados na Figura 1, podemos observar que muitas das lacunas nos dados ficam rodeadas por *outliers*, ou seja, lacunas nos dados são geralmente sucedidas por medições com valores bem altos da variável alvo, o que fez com que as lacunas fossem preenchidas com valores muito distantes da média e mediana dos dados originais. Em suma, este processo produziu novos *outliers* e distorceu a característica dos dados. Por esta razão, decidiu-se por adotar outra estratégia: os dados foram preenchidos a partir de médias móveis das medidas obtidas anteriormente. O resultado dessa operação pode ser observado a seguir:

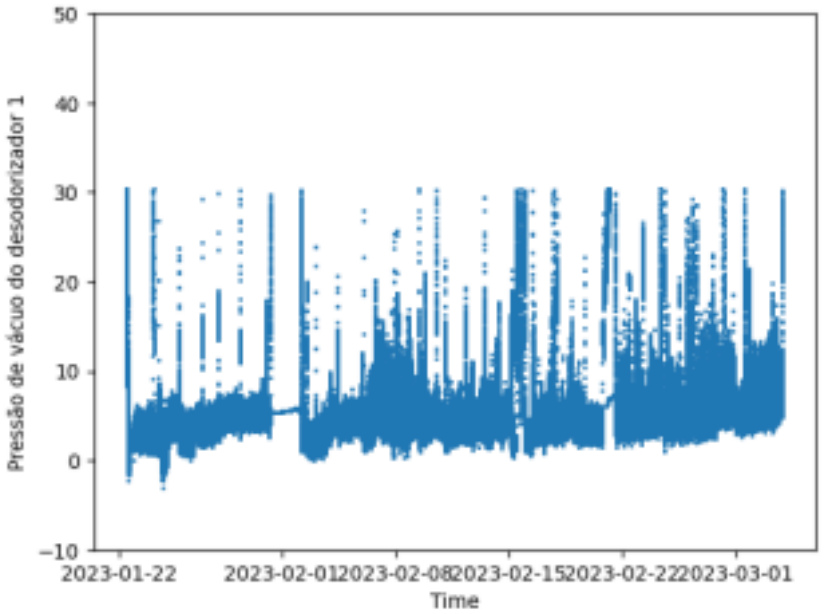

Figura 6. Variável alvo após *resampling* e remoção de *outliers*

Ao observar a Figura 6, fica evidente que existem problemas na estratégia escolhida, as lacunas foram preenchidas sem apresentar os mesmos padrões de outras regiões dos dados, que possuem variação muito maior da variável alvo. Além disso, pode-se observar que a remoção de *outliers* teve um efeito visível nos dados da variável

alvo, que anteriormente possuíam muitos exemplares que se aproximavam de 50 e após este processo de transformação mal possuem dados acima de 30. Apesar desses problemas, essas transformações foram mantidas para o desenvolvimento dos modelos. As mudanças causadas por esses processos ficam evidentes ao se observar as alterações causadas na média, mediana e desvio padrão:

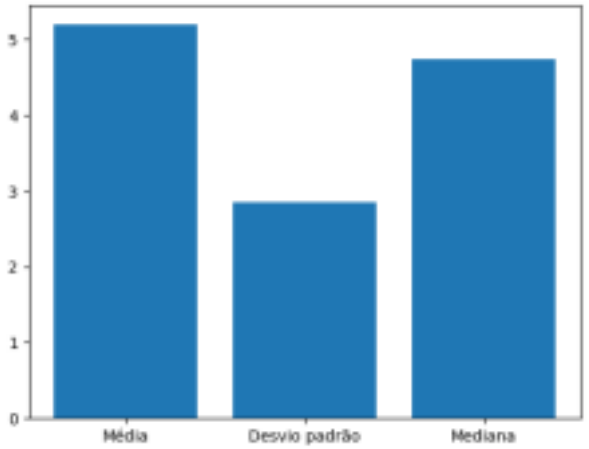

Figura 7. Estatísticas da variável alvo após *resampling* e remoção de *outliers*

Agora o desvio padrão representa apenas 54,68% da média (antes eram 81,12%) e 60,26% da mediana (antes, 96,26%).

Após este pequeno processo de transformação dos dados, foram realizados processos já pensando na fase de modelagem do projeto. Uma técnica importante quando se tratando de séries temporais é a decomposição das mesmas em sazonalidade, tendência e valores residuais. A tendência mostra a direção geral dos dados ao longo do tempo, independentemente das flutuações. A sazonalidade mostra as variações periódicas nos dados - por exemplo, um aumento nas vendas de sorvete durante o verão. Finalmente, o resíduo é o que resta depois de remover a tendência e a sazonalidade dos dados.

No caso deste projeto e devido a algumas características da base de dados, foi realizada a decomposição aditiva dos mesmos. Visualizar a decomposição aditiva de uma série temporal é útil por várias razões. Em primeiro lugar, ajuda a identificar se existe uma tendência subjacente ou um padrão sazonal nos dados, o que pode ser importante ao escolher o tipo de modelo de série temporal a ser utilizado. Além disso, ao decompor uma série temporal em seus componentes aditivos - tendência, sazonalidade e resíduo - somos capazes de entender melhor se a série é estacionária ou não. Se a tendência é não-constante ou se existe uma sazonalidade que varia com o tempo, a série temporal não será estacionária. A importância de observar a estacionariedade de uma série temporal reside no fato de que muitos modelos de séries temporais, incluindo o ARIMA, pressupõem estacionariedade.

Portanto, a decomposição aditiva nos ajuda a entender e a tratar a não estacionariedade em uma série temporal. Ao remover a tendência e a sazonalidade (através da diferenciação dos dados, por exemplo), podemos transformar uma série não estacionária em uma série estacionária, que é mais adequada para a modelagem e

previsão. A decomposição da série temporal da variável alvo pode ser observada a seguir:

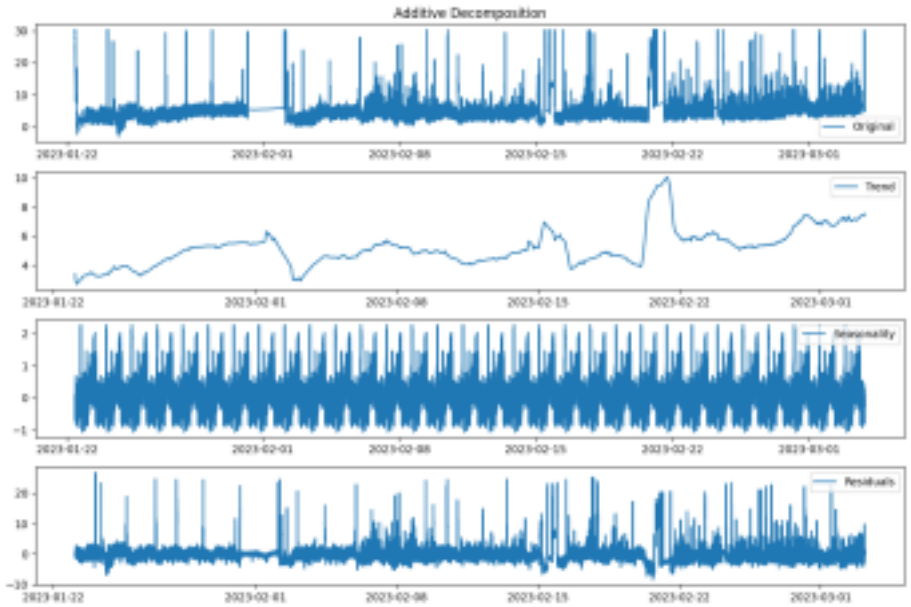

Figura 8. Decomposição aditiva da série temporal da variável alvo

Para realizar a decomposição aditiva, é necessário ao menos estimar o número de ciclos ocorrentes na série a ser decomposta. No caso deste projeto, o número de ciclos foi estimado em 31 a partir da visualização dos dados. Como pode ser observado na figura 8, os dados inicialmente não formam uma série temporal estacionária, isso é evidenciado pelas características da tendência, que não é constante e é sempre positiva, indicando que a média e outras propriedades estatísticas apresentam variações significativas ao longo do tempo. Como o modelo ARIMA e outros algoritmos para modelagem de séries temporais demandam que os dados sejam estacionários, foi preciso fazer com que os dados passassem por processos de diferenciação até que eles adquirissem as propriedades desejadas. A seguir pode-se observar o resultado da decomposição após uma diferenciação da série da variável alvo.

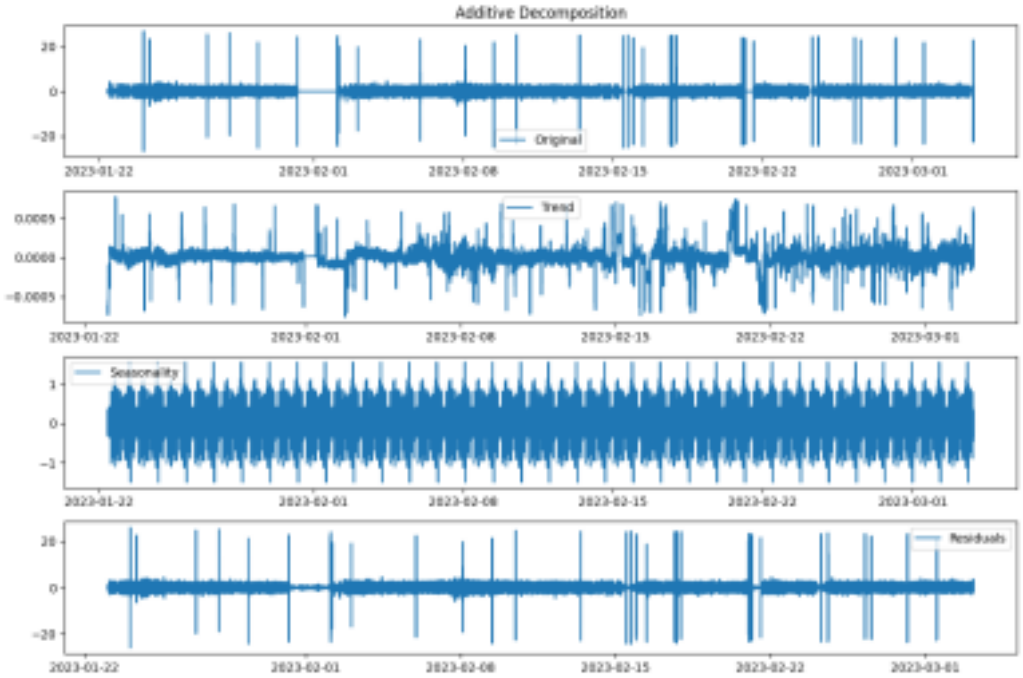

Figura 9. Decomposição aditiva da série da variável alvo diferenciada 1 vez

Como pode ser observado na Figura 9, após realizar a diferenciação da variável alvo, observa-se grandes diferenças nos três componentes da série. Primeiramente, é possível observar que a tendência se comporta de forma completamente diferente, apresentando mais variações de baixíssima magnitude (o valor absoluto da tendência não passa de 0.001), indicando que a média, variância e demais estatísticas passam por pequenas variações ao longo do tempo, indicando que a série é pelo menos fracamente estacionária após passar por este processo. Para nos assegurarmos da estacionariedade da série temporal, seria necessário aplicar um teste como o teste de Dickey-Fuller, algo que foi tentado, mas que se mostrou inviável devido ao consumo muito alto de recursos computacionais. Além da tendência, pode-se observar que os valores residuais apresentaram mudanças significativas. Antes da diferenciação eles eram quase sempre positivos e apresentavam maiores variações. Após a diferenciação, eles passaram a variar menos e diversas vezes assumem valores negativos.

Esse processo de diferenciação seguido de decomposição aditiva para fins de verificação da estacionariedade foi repetido mais algumas vezes, contudo, os resultados não apresentaram grandes diferenças para estes observados após um passo de diferenciação e portanto, quando foi necessário utilizar dados estacionários nas fases seguintes do projeto, optou-se por utilizar apenas um passo de diferenciação, ou diferenciação direta.

3.3 Modelagem e Validação

3.3.1 Modelagem com o algoritmo ARIMA

O modelo ARIMA, que significa Modelo AutoRegressivo Integrado de Médias Móveis, é uma classe popular de modelos para prever séries temporais. Em sua forma básica, ele usa a ideia de autocorrelação, diferenciação (para tornar a série estacionária), e uma média móvel para prever valores futuros. O modelo ARIMA é tipicamente

univariado, ou seja, é projetado para prever uma série temporal com base em seus próprios valores passados, sem considerar variáveis exógenas (demais *features* dos dados, que não a variável alvo). Este é o caso do modelo que foi construído neste projeto. Ou seja, o modelo construído a partir do algoritmo ARIMA prevê os futuros valores da pressão de vácuo no desodorizador utilizando apenas os valores anteriores observados da pressão. Os parâmetros em um modelo ARIMA são geralmente denotados como ARIMA(p, d, q), onde:

- **p** é a ordem do termo auto regressivo. Ele permite incorporar o efeito de valores passados na previsão. Por exemplo, se p é 5, os cinco valores anteriores da série temporal são usados para prever o próximo valor.
- **d** é a ordem de diferenciação. Diferenciação é o processo de subtrair a observação atual da observação anterior para tornar a série estacionária. Por exemplo, se d é 1, estamos usando uma primeira diferença, que é exatamente o que fizemos na subseção anterior ao entre os processos de decomposição aditiva.
- **q** é a ordem do termo da média móvel. Uma média móvel permite que incorporemos o efeito de erros passados na previsão.

Existem várias técnicas para selecionar esses valores, o parâmetro, uma delas envolve a análise do ACF e PACF da série temporal APÓS A DIFERENCIAÇÃO. Em geral, o valor de 'p' pode ser estimado pelo atraso em que o PACF corta o limite de confiança superior pela primeira vez. Da mesma forma, 'q' pode ser estimado pelo atraso em que o ACF cruza o limite de confiança superior pela primeira vez. Devido à simplicidade desta técnica, tanto em termos de compreensão de seu funcionamento quanto em consumo de recursos computacionais, ela foi a selecionada para determinar os parâmetros **p** e **q**. O parâmetro **d** já foi determinado na seção anterior ao averiguarmos que apenas um passo de diferenciação já aparentou ser suficiente para tornar a série ao menos fracamente estacionária.

Para obtermos os valores para os outros parâmetros, foram visualizadas as funções ACF e PACF novamente, agora aplicadas aos dados diferenciados uma vez.

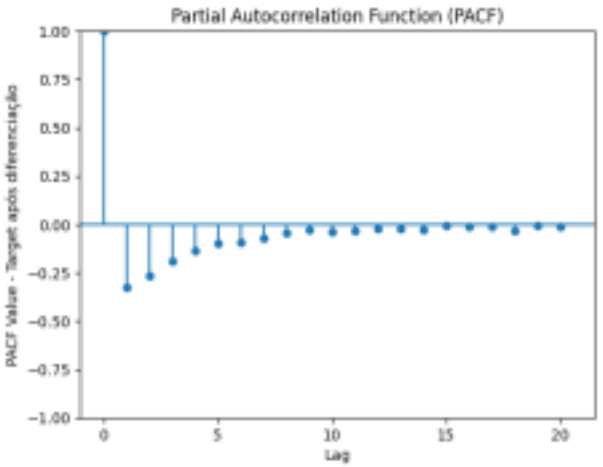

Figura 10. Visualização da PACF da variável alvo após diferenciação

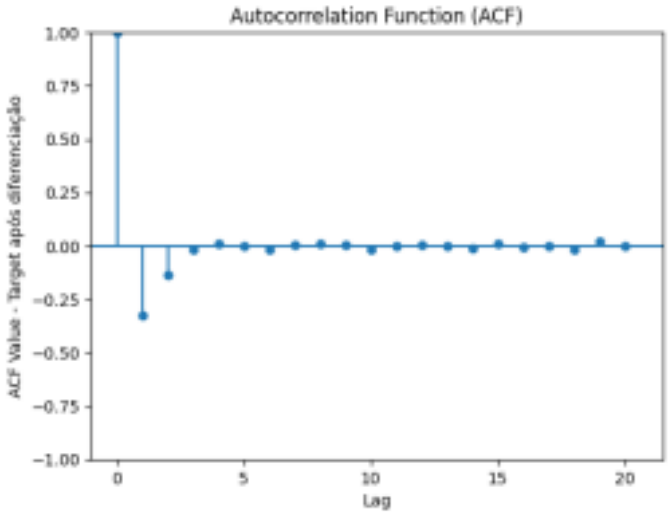

Figura 11. Visualização da ACF da variável alvo após diferenciação

Após realizar a observação das funções, os parâmetros escolhidos foram 2 e 8 para **p** e **q**, respectivamente.

3.3.2 Validação do modelo ARIMA

A fase de validação dos modelos apresentou obstáculos devido a diversas características deste trabalho. Primeiramente, o caráter dos dados, que formam uma série temporal, já que nestes casos, uma das principais técnicas utilizadas na validação de modelos construídos para lidar com problemas tradicionais de aprendizado de máquina, a validação cruzada, não pode ser utilizada neste caso, já que ela presume que os dados podem ser divididos separados em dados de teste e de treino de forma randômica e ao validar uma série temporal é necessário que a ordem dos dados seja cronológica e que nenhum dado de teste seja anterior a um dado de treino. Portanto, a única forma de realizar a validação é seccionando a base de dados em um ponto e utilizando os dados anteriores àquele ponto como treino e os posteriores como teste.

Contudo, outro desafio é introduzido pelo objetivo de observar como a performance do modelo se comporta conforme o mesmo realiza previsões mais distantes no futuro, observar o *trade-off* entre antecipação das previsões e qualidade das mesma. Por isso, foi selecionada uma forma talvez pouco convencional para visualização da performance dos modelos. Geralmente, se separariam os dados entre treino e teste, se realizariam as previsões sobre os dados de teste e se obteriam estatísticas como erro quadrático médio. Contudo, partindo desse desejo de observar a variação da performance em maiores horizontes de tempo, se decidiu calcular o erro quadrático médio sucessivas vezes, adicionando uma nova previsão de teste por iteração, além disso foram plotados o erro absoluto, o valor da previsão e o valor real observado.

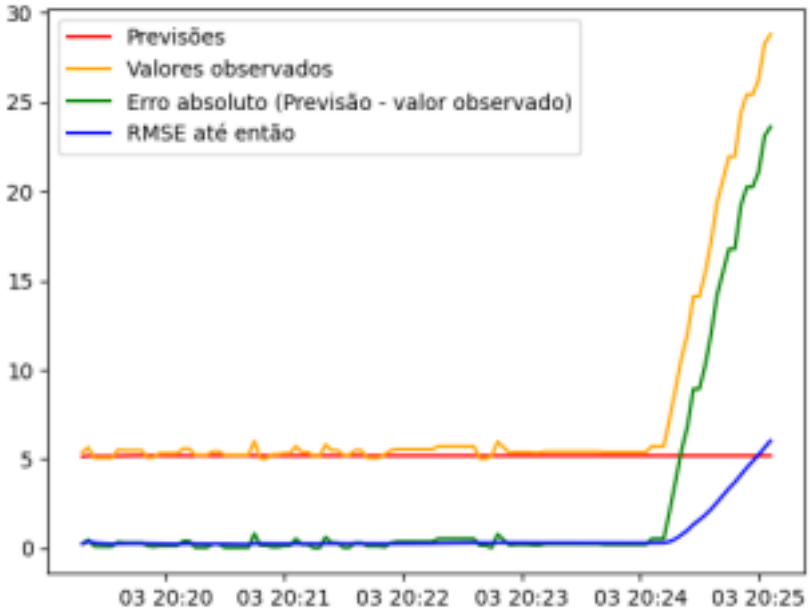

Figura 12. Resultados da validação do modelo ARIMA

Logo ao observar o gráfico da Figura 12, percebe-se alguns problemas com o modelo construído. Primeiramente, apesar de seu erro inicialmente ser baixo, ele aparenta realizar previsões praticamente constantes (não são totalmente constantes, como pode ser observado na Figura 13, localizada abaixo). Isso não é um problema para o início dos dados de teste, que pouco distam desse valor quase constante, por isso o erro absoluto e o erro quadrático médio permanecem bem baixos até que chega um ponto em que os valores reais de teste aumentam significativamente, uma tendência que o modelo ARIMA foi completamente incapaz de prever. A partir deste momento, o erro absoluto e quadrático médio disparam, mostrando que o modelo ARIMA provavelmente é simples demais para este problema. Isso não era surpreendente considerando que o modelo em questão é univariado, ou seja, somente utiliza de dados anteriores da variável alvo para realizar suas previsões, não pode levar em consideração alterações, valores e tendências observadas em variáveis exógenas (as demais *features* dos dados, que não são o *target*).

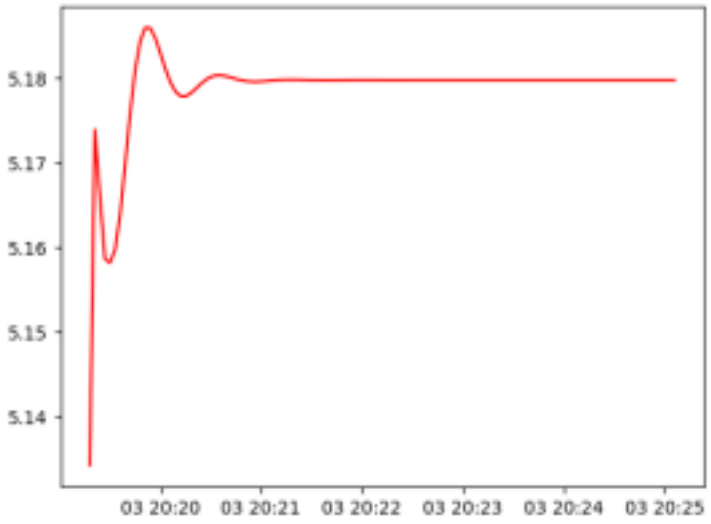

Figura 13. Previsões do modelo ARIMA isoladas

3.3.3 Modelagem com modelo de *boosting*

Como foi ressaltado na última subseção, o algoritmo ARIMA tem suas limitações. Ele é projetado apenas para séries temporais univariadas, o que significa que só pode levar em consideração uma única série temporal de cada vez. Se existem múltiplas séries temporais relacionadas que interagem entre si, o ARIMA pode não ser capaz de capturar essas interações. Além disso, o ARIMA assume que a série temporal é linear e segue uma distribuição normal. Se essas suposições não são atendidas, o desempenho do ARIMA pode ser comprometido, o que pode explicar sua performance bem ruim.

Em contraste, o XGBoost (Extreme Gradient Boosting) é uma técnica de aprendizado de máquina que pode lidar com problemas de aprendizado supervisionado multivariados. Ele pode levar em consideração várias séries temporais ao mesmo tempo e capturar as interações entre elas. Além disso, o XGBoost não faz suposições sobre a linearidade ou a distribuição dos dados, tornando-o uma ferramenta de previsão mais flexível e robusta. Uma abreviação de eXtreme Gradient Boosting, ele é um algoritmo de aprendizado de máquina que pertence à classe dos algoritmos de boosting. O boosting é um método de aprendizado em conjunto que visa a criar um modelo preditivo forte e preciso a partir da combinação de vários modelos preditivos fracos.

Em problemas de séries temporais, as observações são dependentes do tempo, ou seja, a observação em um determinado momento é muitas vezes influenciada pelas observações em momentos anteriores. Portanto, ao lidar com séries temporais, é essencial capturar essa dependência temporal nos dados. O XGBoost, assim como muitos outros algoritmos de aprendizado de máquina, são originalmente projetados para problemas onde as observações são independentes umas das outras. Isso significa que, se os dados de séries temporais forem fornecidos a esses algoritmos sem qualquer modificação, eles não conseguirão capturar as dependências temporais inerentes aos dados, resultando em modelos que provavelmente terão um desempenho inferior. Para superar isso, uma abordagem comum é transformar os dados de séries temporais para incluir recursos que possam capturar a dependência temporal. Isso é feito criando recursos derivados como lags e médias móveis. Essa foi a abordagem utilizada neste projeto.

3.3.4 Validação do modelo XGBoost

Um problema não mencionado na seção sobre a modelagem com o algoritmo ARIMA, é a de que o ARIMA não é um modelo típico de aprendizado de máquina, ele não funciona como os demais modelos implementados neste trabalho, por isso foi necessário realizar uma forma de validação um pouco diferente para observar como o seu erro de previsão aumentava conforme o horizonte de previsões era expandido. Contudo, os demais modelos podem ser validados de forma mais convencional, possibilitando observar a relação do erro com a antecedência da previsão de forma clara e permitindo compará-los, já que eles serão treinados com os mesmos dados e treinados com os mesmos dados. Para que o mesmo fosse feito com o modelo ARIMA, ele teria de ser retreinado várias vezes, já que não é um modelo que recebe uma entrada na hora da previsão, somente na hora do treinamento.

A métrica utilizada para esta validação e que será utilizada para realizar a

validação dos demais modelos foi a raiz quadrada do erro quadrático médio. Diversos fatores tornam esta escolha adequada neste caso: primeiramente, essa métrica devido ao cálculo do erro quadrático inicialmente, pune mais erros grandes de previsão do que erros pequenos, o que é muito pertinente nessa situação, já que erros grandes na previsão da pressão dentro do desodorizador são muito mais problemáticos, já que as situações de perdas de vácuo representam variações bem grandes no valor da pressão, ou seja, um erro grande de previsão geralmente ocorrerá quando a perda de vácuo ocorrer, que é justamente a situação que está tentando se prever aqui. Não à toa, a função de perda escolhida para os modelos de redes neurais desenvolvidas é justamente o erro quadrático médio. Além disso, como estamos utilizando a raiz quadrada do erro quadrático médio, obtemos uma métrica mais interpretável, já que o erro está na mesma escala dos dados, permitindo-nos compará-lo com estatísticas da variável alvo como a média e o desvio padrão.

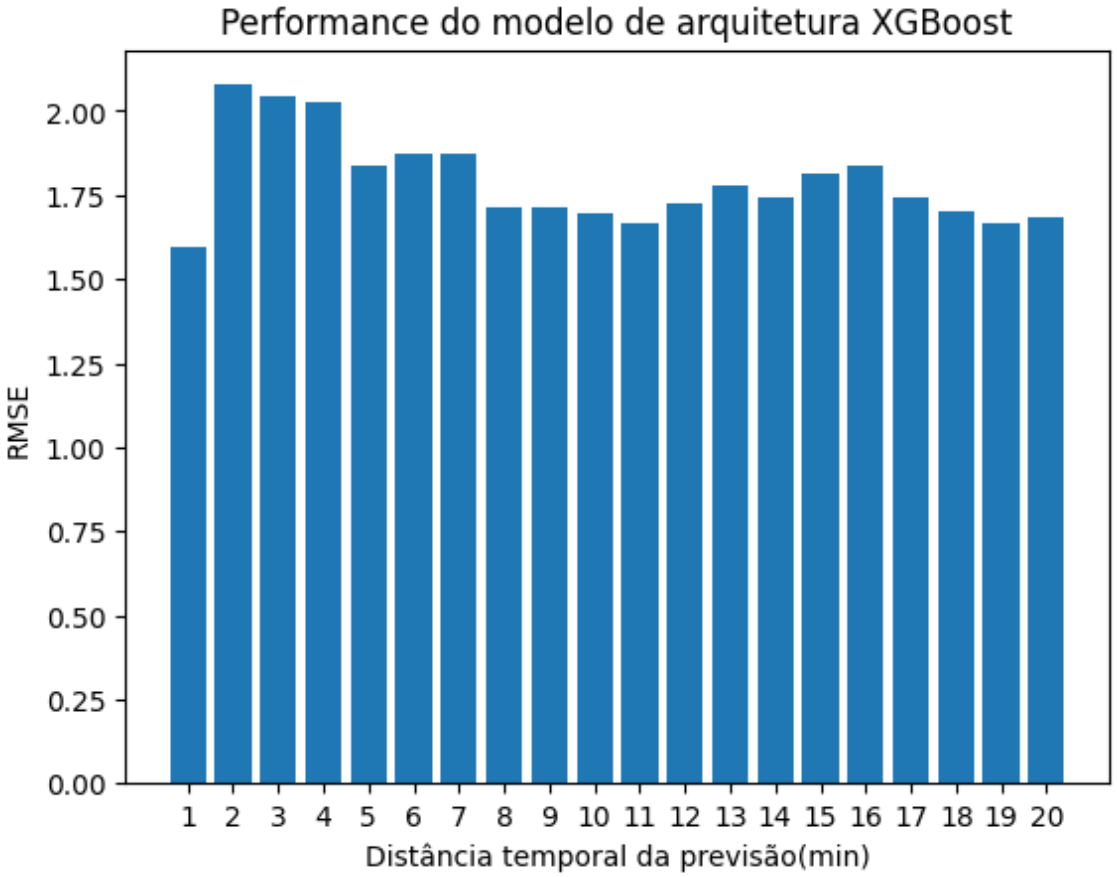

Figura 14. Resultados da validação do modelo XGBoost

Como pode ser visto na Figura 14, a performance do modelo XGBoost apresentou erro bastante estável conforme o horizonte de previsões ia sendo aumentado. Isso foi inesperado, mas pode ser explicado por alguns fatores, é provável que se o horizonte de previsões continuasse sendo expandido (a base de dados contém vários meses de medições feitas a cada 3 segundos, e portanto o horizonte de previsões poderia ter uma escala muito maior) o erro iria adquirir uma tendência de crescimento, mas isso não foi feito, pois não fazia muito sentido desenvolver um modelo que realizasse previsões com antecedências muito maiores do que as já feitas, já que 20 minutos é uma

janela de tempo suficiente para prevenção de falhas no desodorizador.

Considerando que o desvio padrão está próximo de 3, temos que o desempenho deste modelo é pelo menos superior ao que se esperaria de uma estratégia ingênua de sempre prever a média, já que se seguíssemos esta estratégia, o RMSE deveria ser o desvio padrão (ou próximo, já que não necessariamente o desvio padrão da base toda é igual ao desvio padrão dos dados de teste).

Contudo, este tipo de algoritmo não é considerado o estado da arte para séries temporais e, portanto, foram desenvolvidos modelos utilizando redes neurais com arquiteturas especificamente criadas para lidar com dados sequenciais como séries temporais.

3.3.5 Modelagem com rede neural LSTM

As redes neurais Long Short-Term Memory (LSTM) são uma categoria especial de redes neurais recorrentes (RNNs), projetadas para lidar com sequências de dados. Elas são particularmente adequadas para prever séries temporais devido à sua capacidade de capturar dependências de longo prazo em dados sequenciais.

Uma rede LSTM é composta por unidades chamadas células LSTM. Cada célula tem três portas: a porta de esquecimento, a porta de entrada e a porta de saída. Essas portas controlam o fluxo de informações dentro e fora da célula. A porta de esquecimento decide quanta informação do estado anterior será mantida ou descartada. Isso é crucial para evitar o problema de dependências de longo prazo, onde a rede não consegue aprender conexões entre eventos separados por longos intervalos de tempo. A porta de entrada controla a quantidade de informação nova que será adicionada ao estado da célula. O estado da célula é atualizado com base nas informações que passam pela porta de esquecimento e pela porta de entrada. Este estado é o "coração" da LSTM, carregando informações ao longo das sequências. Além disso, a porta de saída decide quais informações do estado da célula serão usadas para gerar a saída da célula. Em cada passo de tempo, a LSTM recebe um input, atualiza seu estado interno (memória de longo prazo) e produz um output. Este processo é repetido para cada elemento da sequência de entrada.

Além disso, elas têm sido usadas com sucesso em uma variedade de tarefas de séries temporais, como previsão do mercado de ações, previsão meteorológica e análise de tendências de consumo.

Como estamos tratando de redes neurais neste caso, temos diversas questões a serem decididas antes de construir um modelo. Qual deve ser a estrutura da rede? Quantas camadas e quantos neurônios devem existir em cada camada? Qual otimizador deve ser utilizado? Após diversas experimentações, a configuração de três camadas LSTM, contendo 50, 30 e 20 neurônios cada, com camadas de dropout de 0.2 interpolando-as se mostrou a melhor. Uma boa regra para se definir a quantidade de neurônios da primeira camada de uma rede como esta é utilizar o tamanho do dado, que neste caso é exatamente 50, ou seja, a regra se provou útil no contexto do trabalho. O otimizador utilizado foi o Adam e a função de perda foi o erro quadrático médio.

3.3.5 Validação da rede neural LSTM

A validação da rede neural LSTM desenvolvida foi feita da mesma maneira utilizada para validar o modelo XGBoost. Abaixo estão os resultados encontrados:

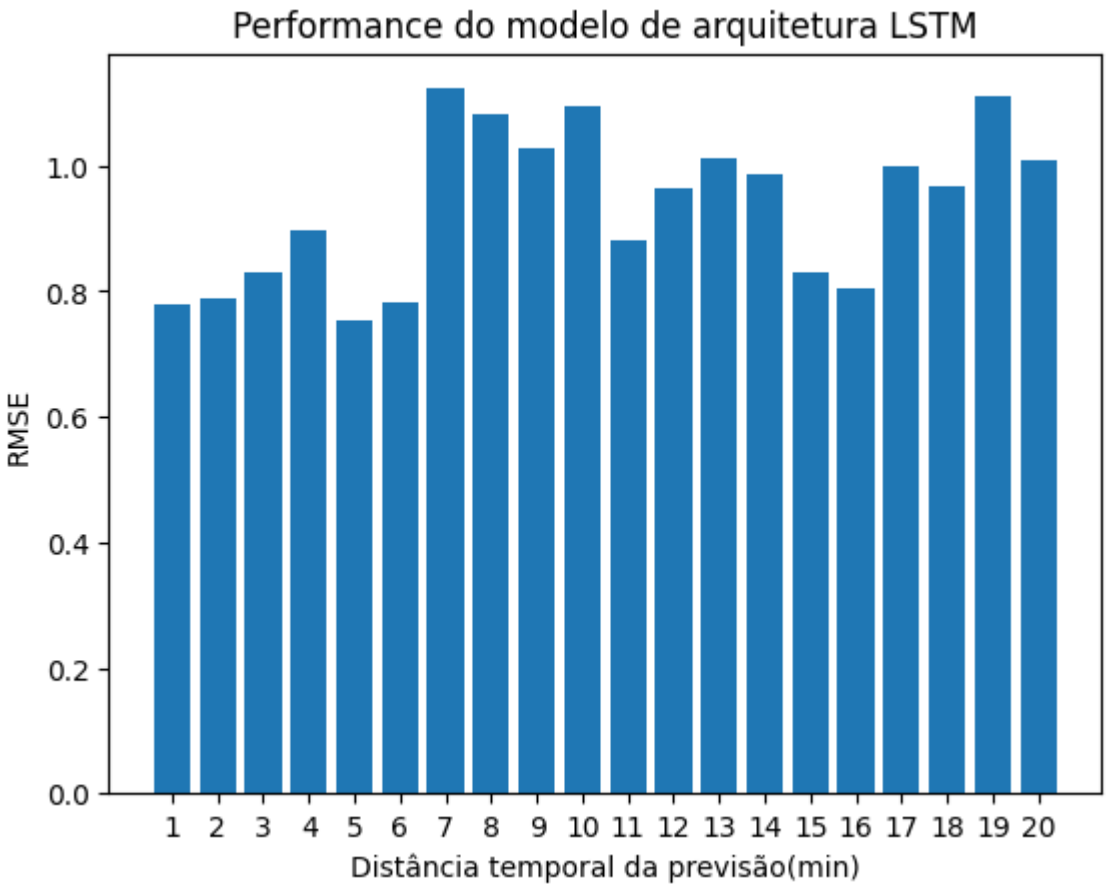

Figura 15. Resultados da validação do modelo de rede neural LSTM

Como pode ser observado na figura 15, o RMSE da rede neural oscila conforme a distância temporal da previsão aumenta, apresentando erros maiores quando a previsão é feita com antecedência de 7 a 10 minutos, mas os melhores resultados foram encontrados nas menores distâncias temporais, como esperado. Contudo, é notável a estabilidade do erro, já que o aumento observado entre as menores distâncias temporais e as maiores é pequeno e o erro é menor que o observado para o modelo XGBoost, que apresentou valores de raiz do erro quadrático médio próximos a 2, enquanto a LSTM apresentou resultados abaixo de 1 e todos os erros ficaram abaixo de 1,2. Considerando o valor do desvio padrão, percebe-se que os resultados têm alguma relevância, já que o desempenho é ao menos melhor que uma estratégia ingênua.

Esperava-se que o aumento da janela de previsões tivesse grande impacto no erro do modelo, mas o aumento foi relativamente pequeno, e, considerando o contexto de possível aplicação destes modelos, atuando em sistemas de prevenção de falhas na indústrias, é muito relevante que se tenha um sistema que não só seja capaz de prever a ocorrência de problemas, mas de fazê-lo com certa antecedência.

3.3.6 Modelagem com redes neurais Transformer

Apesar de terem sido desenvolvidas com o intuito de tratar dados sequenciais e terem sido utilizadas em larga escala para a criação de modelos de previsão de séries temporais, as redes neurais LSTM têm sido substituídas por redes neurais de arquitetura Transformer em diversos contextos, especialmente na área de processamento de linguagem natural. Modelos de linguagem extremamente complexos têm se mostrado extremamente capazes de resolver problemas de linguagem e existem diversas semelhanças entre problemas envolvendo linguagem e problemas envolvendo śeries temporais. Essas semelhanças ajudam a explicar porque o uso de redes neurais com essa arquitetura é adequado para o contexto deste trabalho.

A grande semelhança entre linguagem e séries temporais é que ambos são dados sequenciais, a ordem das palavras importam, assim como a ordem cronológica das medições das condições dentro do desodorizador importam também. É neste contexto que as redes Transformer se inserem, já que a arquitetura foi criada com o intuito de modelar relações entre diferentes partes de uma sequência (Vaswani et al., 2017) através do mecanismo de atenção, especificamente o "self-attention". Este mecanismo permite que o modelo pondere a importância de diferentes partes da entrada ao processar cada palavra (ou parte de uma sequência). Em outras palavras, ele permite que o modelo aprenda contextos e relações dentro da sequência.

Um Transformer típico é composto por um encoder e um decoder. O encoder processa a sequência de entrada e gera uma representação rica em contexto e o decoder gera a sequência de saída, passo a passo, usando a saída do encoder e o que foi gerado até o momento. Tanto o encoder quanto o decoder são compostos por várias camadas idênticas que contêm mecanismos de atenção e redes neurais *feedforward*. Além disso, diferentemente das LSTMs, os Transformers não processam os dados sequencialmente. Isso permite o processamento paralelo dos dados, tornando-os muito mais eficientes em termos de tempo de treinamento. Isso pode soar estranho devido ao fato de que estamos tratando de séries temporais, mas apesar de processarem todos os elementos simultaneamente, os Transformers ainda são capazes de entender a ordem dos dados. Isso é alcançado através do uso de *positional encodings*. Estes são vetores adicionados à entrada que fornecem informações sobre a posição de cada elemento na sequência.

Portanto, devido a essas inúmeras vantagens apresentadas pelos Transformers em relação às LSTMs, se decidiu por implementar diferentes modelos a fim de atingir melhores resultados nas previsões e comparar os modelos de diferentes arquiteturas.

3.3.6.1 Modelagem do Time Series Transformer Multivariado

A primeira implementação de modelo transformer feita no trabalho utiliza do algoritmo Time Series Transformer da biblioteca *transformers* do HuggingFace. Este algoritmo implementa a arquitetura originalmente proposta pelo artigo “Attention is all you need” (Vaswani et al., 2017) que introduziu a arquitetura. O algoritmo foi criado com foco no problema univariado originalmente, mas suporte para problemas

multivariados, como o caso deste trabalho, foi adicionado após algum tempo. Isso mostra um dos problemas enfrentados nesse trabalho: diversos algoritmos e trabalhos relacionados a séries temporais presumem um problema de previsão univariada, o que neste caso é ruim, pois há uma abundância de dados relevantes que podem ser utilizados para modelar relações complexas entre as *features*. Por isso, decidiu-se por criar duas implementações diferentes para cada arquitetura transformer utilizada, uma variante univariada e outra multivariada. O artigo que introduz o Autoformer (Wu, Haixu, et al., 2021) destaca que o desempenho da implementação univariada do estudo superou a implementação multivariada em um problema de regressão, o que mostra a relevância desta questão.

O algoritmo possui diversos hiperparâmetros, a melhor configuração neste caso possuía 8 camadas no encoder e no decoder, com 32 neurônios em cada camada e o treinamento foi feito utilizando o otimizador Adam, por 4 épocas, utilizando *batches* com 512 instâncias de treino.

3.3.6.2 Validação do modelo Time Series Transformer Multivariado

A validação foi feita da mesma maneira que os dois modelos anteriores. A seguir estão os resultados encontrados:

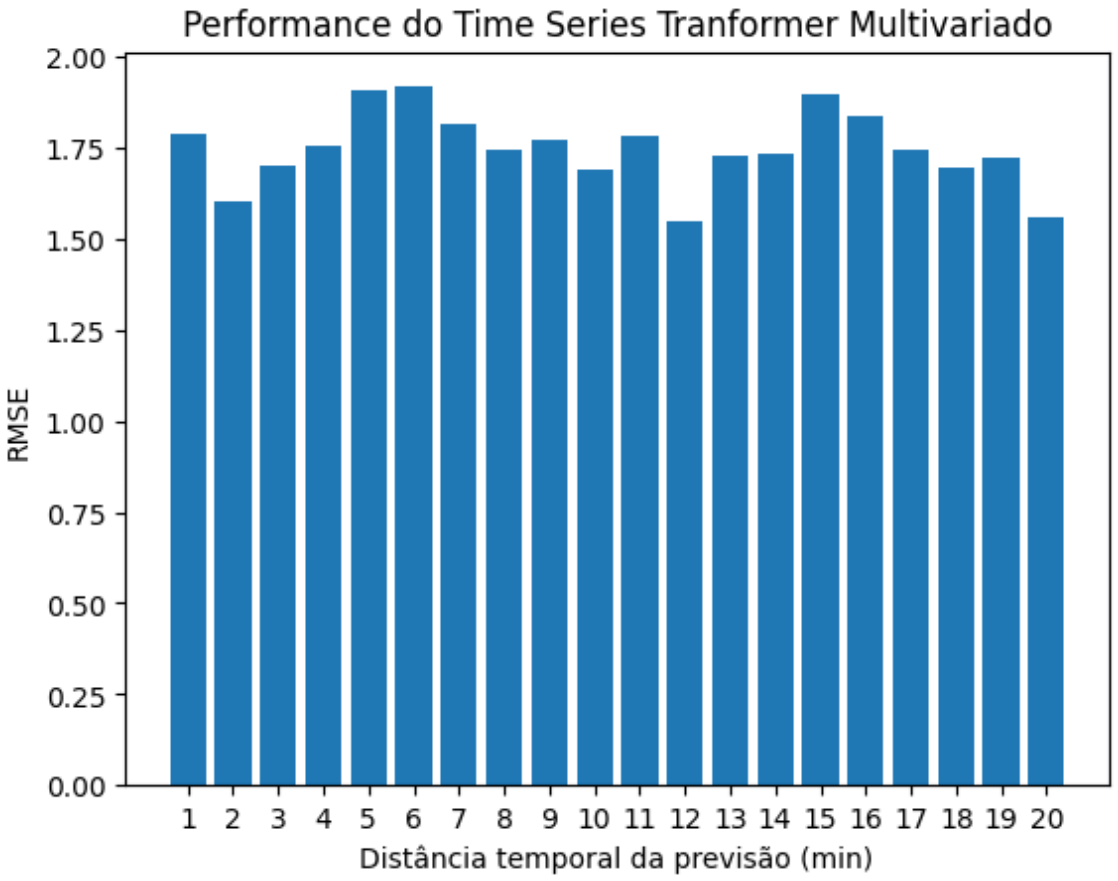

Figura 16. Resultados da validação do Time Series Transformer Multivariado

Como pode ser observado na Figura 16, o RMSE do modelo Time Series Transformer oscila entre valores de 1,5 a 2. Os valores encontrados são similares aos observados na validação do modelo XGBoost, mas são piores que os encontrados na validação da rede neural LSTM. Novamente, temos um modelo que apresenta resultado surpreendente e contra intuitivo quanto à variação do erro conforme a distância temporal da previsão cresce, o erro do modelo quando a distância temporal é de 1 minuto é menor que quando a distância é de 20 minutos.

Apesar do desempenho ser pior quando comparado à rede neural LSTM, o modelo ainda apresenta erros menores que o desvio padrão da variável alvo, que seria o erro esperado da estratégia ingênua descrita anteriormente.

3.3.6.3 Modelagem do modelo Autoformer Multivariado

O Autoformer é uma arquitetura de rede neural transformer criada especificamente para lidar com séries temporais. Ele incorpora diversos mecanismos de decomposição das séries temporais e um mecanismo de autocorrelação. Assim como no caso do Time Series Transformer, o algoritmo foi criado e testado majoritariamente em séries temporais univariadas e apresentou resultados piores quando incorporando múltiplas variáveis do que modelando apenas a variável alvo em bases de dados que permitiam essa comparação (Wu, Haixu, et al., 2021). Apesar dessa ressalva, o Autoformer apresentou resultados relevantes à época de publicação do artigo que introduziu a arquitetura, como uma melhora de 38% em *benchmarks* que cobriam uma série de aplicações de previsão a longo prazo, como meteorologia e tráfego urbano. O fato de que a arquitetura se saiu bem em previsões a longo prazo o torna muito relevante neste caso, devido ao desejo de se observar como a distância temporal afeta o erro, e por isso o algoritmo se mostrou promissor.

Apesar das expectativas iniciais, a modelagem do Autoformer multivariado foi muito problemática, o modelo durante o treinamento demorava muito a convergir, os erros de treinamento eram muito grandes e os de validação mais ainda. Não foi possível descobrir ao certo o que causou os problemas, mas os problemas parecem estar relacionados com a dimensionalidade dos dados, como dito anteriormente, o modelo apresentou problemas para modelar as relações entre as diferentes séries temporais no estudo original que o propôs.

A validação do modelo foi realizada, mas os erros se mostraram muito acima de um limiar razoável, apesar de várias modificações nos hiperparâmetros. Estes problemas não se repetiram nos outros casos, e a arquitetura pode ser testada com uma variante univariada.

3.3.6.4 Modelagem do modelo Informer Multivariado

Assim como o Autoformer, o Informer é uma arquitetura de rede neural transformer criada com o intuito de lidar com séries temporais, com enfoque especial a problemas de precisão a longo prazo. O artigo que introduziu a arquitetura venceu o prémio de melhor artigo da conferência da AAAI em 2021. A arquitetura é baseada na arquitetura transformer original, mas com algumas modificações: mecanismo de

auto-atenção *ProbSparse*, que reduz significativamente a complexidade de tempo e memória associada com o mecanismo de auto-atenção dos transformer convencionais e apresenta desempenho comparável em termos de alinhamento de dependências sequenciais; um processo de "destilação" para redução do tamanho do *input* entre camadas da rede e um decoder que, ao contrário dos modelos transformer convencionais e do modelo anterior, realize todas as previsões do horizonte de previsões pré-determinado em apenas um passo, ao contrário dos demai modelos que o fazem passo-a-passo, geralmente de maneira autoregressiva (utilizando das próprias previsões como input para as previsões seguintes). Essas modificações tornam o modelo muito mais flexível e rápido, a complexidade da previsão pelo decoder, por exemplo, cai de O(h), onde h é o tamanho do horizonte de previsões, para O(1).(Zhou, Haoyi, et al., 2021). Todas as modificações feitas tentam tratar de problemas de complexidade de tempo e de memória, o que por si só não é evidência de que o modelo captura relações temporais longas nas séries de forma melhor que a implementação original da arquitetura, então por que o modelo apresentaria melhores resultados para previsões de longo prazo? Porque as reduções de complexidade tornam mais viável a utilização de mais dados. Por isso, este modelo foi treinado com mais *lags* que os demais, pois seu treinamento e validação se mostraram extremamente eficientes em termos de memória e tempo, consumindo a mesma quantidade de tempo, mas processando quantidades maiores de dados.

Os hiperparâmetros a serem determinados eram os mesmos do algoritmo Time Series Transformer, e, portanto, se encontrou a melhor configuração para o modelo, com 8 camadas de neurônios no encoder e decoder e 32 neurônios em cada camada. O fato de a melhor configuração encontrada ter sido a mesma para ambos os modelos não é uma coincidência considerando os resultados encontrados na validação do Informer, que mostraram que os dois modelos multivariados possuem muitas similaridades em seu comportamento.

3.3.6.5 Validação do modelo Informer Multivariado

A validação foi feita da mesma maneira que nos casos anteriores, a seguir estão os resultados encontrados:

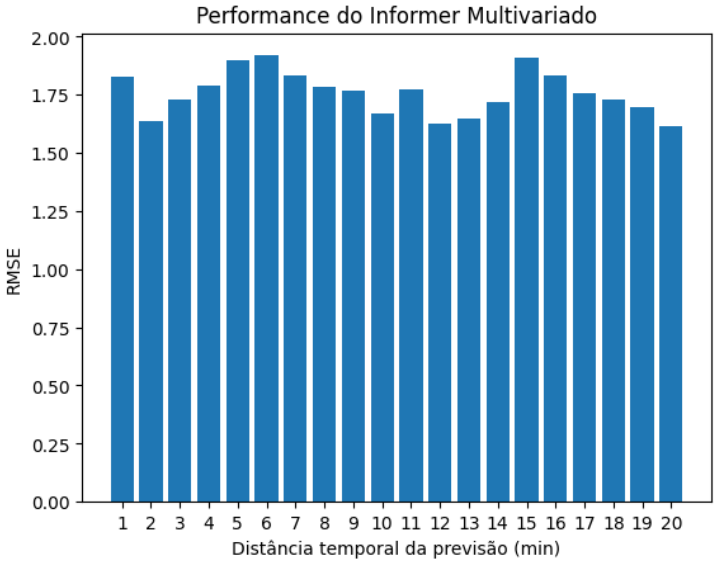

Figura 17. Resultados da validação do Informer Multivariado

Como pode ser observado na Figura 17, os resultados encontrados para o modelo Informer são muito similares aos encontrados na validação do modelo transformer sem modificações, mesmo com o Informer tendo sido treinado com mais *lags* no treino. É claro que os erros encontrados não foram idênticos nos dois casos, mas a semelhança tanto na magnitude dos erros, que variam entre 1,5 e 2 em ambos os casos, quanto nas variações ao longo do gráfico são muito parecidas. Essa constatação é no mínimo curiosa, mas pode ser explicada: o modelo transformer original já incorporava uma grande quantidade de *lags*, e, como ressaltado anteriormente, a arquitetura do modelo Informer introduz modificações que teoricamente impactaram apenas a complexidade de tempo e memória do transformer original, portanto funcionalmente, a arquitetura do modelo é a mesma, e os dados utilizados para treinamento foram os mesmos e estes foram processados na mesma ordem em ambos os treinamentos. Portanto, o que aparenta ter ocorrido neste caso é que o modelo Informer não aprendeu relações significativas entre os *lags* adicionais, e encontrou relações semelhantes às encontradas pelo modelo transformer original. As pequenas diferenças observadas podem ser explicadas pelos *lags* adicionais.

3.3.6.6 Modelagem do Time Series Transformer Univariado

Como ressaltado anteriormente, é muito comum em séries temporais que modelos univariados, aqueles que levam em consideração apenas valores passados da variável alvo para realizar suas previsões, comumente superam modelos multivariados, que teoricamente seriam capazes de modelar relações mais complexas nos dados. Além disso, alguns artigos que propõem arquiteturas transformer, como o Autoformer (Wu, Haixu, et al., 2021), mostram redes neurais transformer univariadas tendo desempenho superior às variantes multivariadas. Além disso, por levarem em consideração apenas os valores da variável alvo, os dados utilizados para treinamento dos modelos univariados ocupam menos memória e seu processamento é mais rápido, perimitindo que essas redes sejam treinadas de formas diferentes, com mais *lags*. Devido a estes fatores, foram construídos modelos univariados.

O Time Series Transformer univariado possuía os mesmos hiperparâmetros que o modelo multivariado. A rede com melhor configuração possui 6 camadas no encoder e decoder com 48 neurônios em cada camada.

3.3.6.7 Validação do Time Series Transformer Univariado

A validação foi feita da mesma maneira que nos casos anteriores, a seguir estão os resultados encontrados:

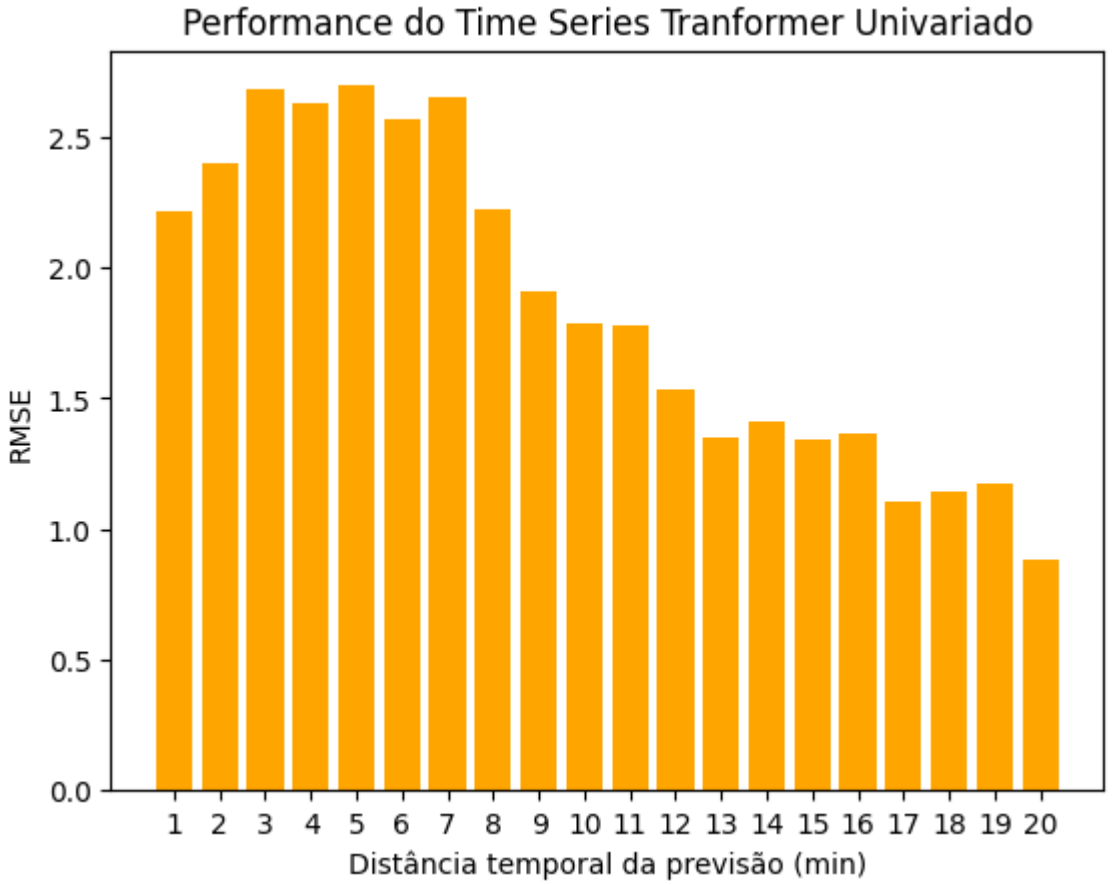

Figura 18. Resultados da validação do Time Series Transformer Univariado

Como pode ser observado na Figura 18, ao contrário do que foi observado nas demais validações, o Time Series Transformer não apresenta erros estáveis conforme o horizonte de previsões vai sendo expandido, mas isso não torna os resultados menos curiosos, já que ao contrário do que esperaria intuitivamente, os erros parecem cair conforme a distância temporal vai aumentando, os resultados foram bem inesperados. O que eles podem indicar é que o modelo conseguiu modelar as relações a longo prazo melhor que as de curto prazo. O resultado é ainda mais surpreendente quando se considera que o erro quando a distância temporal da previsão é 20 minutos chega a ficar abaixo de 1, o melhor de todos os resultados. Apesar de surpreendentes e positivos, o menor erro encontrado ainda não supera os resultados da LSTM. Além disso, quando a distância temporal é menor, como a 3 minutos, o erro supera 2,7, se aproximando o desvio padrão da variável alvo, que seria o erro esperado da estratégia ingênua que prevê

a média todas as vezes, o que coloca em cheque o desempenho do modelo. Apesar dos resultados relativamente positivos com grandes distâncias temporais, (afinal, os dois transformers multivariados construídos apresentaram desempenhos piores para janelas deste tamanho) o desempenho para janelas menores parece ser bem insatifastório. Considere por exemplo, que a média da variável alvo é cerca de 5 e ela permanece em valores similares por boa parte do tempo e que quando a variável alvo supera 10, uma falha está configurada, fica evidente que um erro de 2,7 na previsão é muito significativo.

3.3.6.8 Modelagem do Autoformer Univariado

Ao contrário do que ocorreu com a variante multivariada do algoritmo Autoformer, a versão univariada foi implementada sem maiores problemas, sua melhor configuração consistiu de 8 camadas no encoder e decoder com 48 neurônios em cada camada.

3.3.6.9 Validação do Autoformer Univariado

A validação foi feita da mesma maneira que nos casos anteriores, a seguir estão os resultados encontrados:

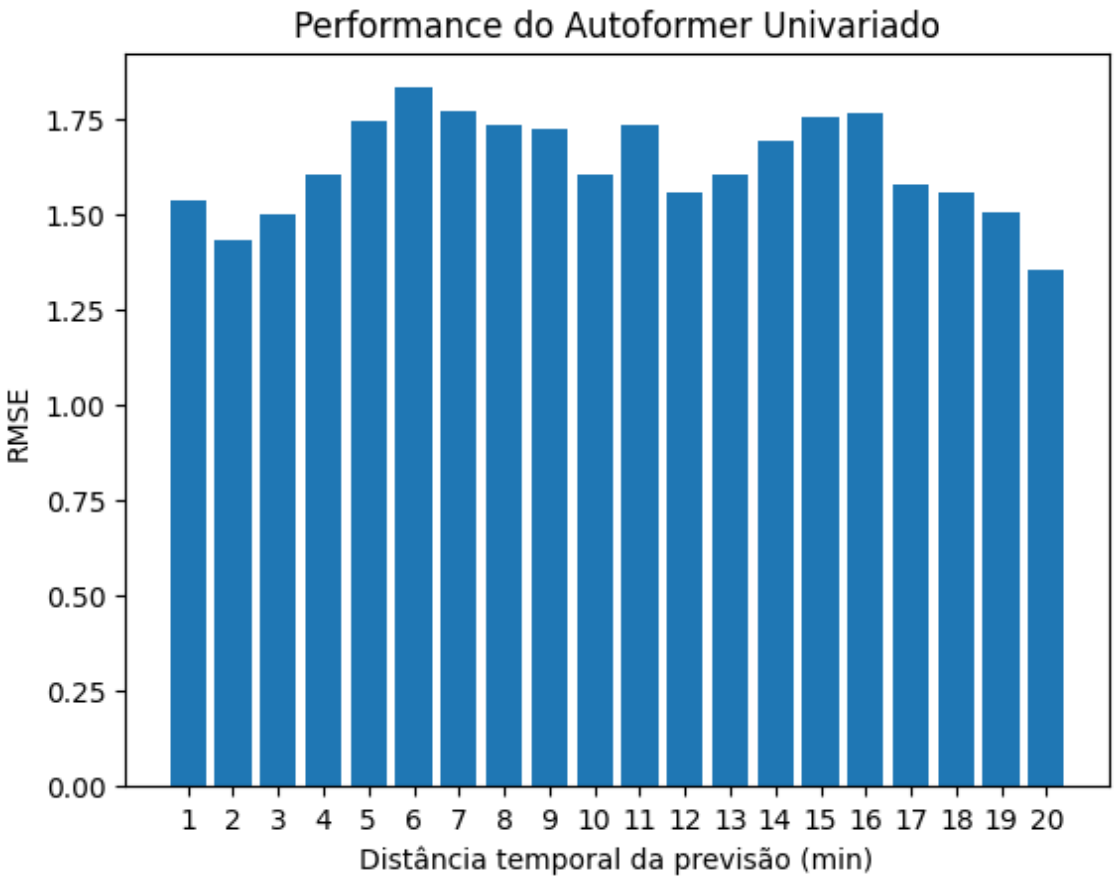

Figura 19. Resultados da validação do Autoformer Univariado

Como pode ser observado na Figura 19, os resultados encontrados para o Autoformer univariado são semelhantes aos encontrados na validação do Time Series Transformer Multivariado e do Informer Multivariado, contudo, os resultados foram um pouco melhores neste caso, mesmo que isso não esteja visível neste gráfico. Assim como com estes algoritmos anteriores, o modelo apresentou um erro estável, pouco afetado pela distância temporal das previsões, e que ao menos é menor que o desvio padrão do alvo. Apesar da melhora nos resultados em relação aos transformer predecessores, o Autoformer Univariado apresentou resultados significativamente piores que a LSTM construída.

3.3.6.10 Modelagem do Informer Univariado

Assim como foi feito com os outros algoritmos, se implementou uma versão univariada do modelo Informer, que foi construído com 6 camadas no encoder e decoder e 48 neurônios em cada camada, a mesma configuração do Time Series Transformer Univariado.

3.3.6.11 Validação do Informer Univariado

A validação foi feita da mesma maneira que nos casos anteriores, a seguir estão os resultados encontrados:

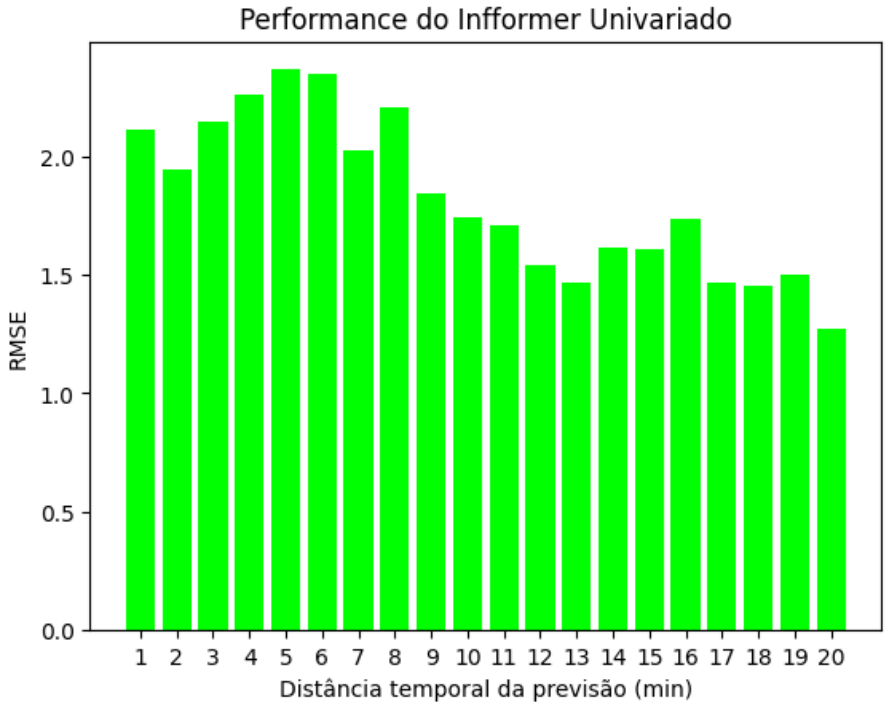

Figura 20. Resultados da validação do Informer Univariado

Assim como o Informer Multivariado apresentou validação com resultados muito semelhantes ao Time Series Transformer Multivariado, o Informer Univariado também apresenta semelhanças no comportamento do seu erro com o Time Series Transformer

Univariado. Apesar das diferenças entre os erros serem muito maiores neste caso, ambos apresentam tendência de queda conforme a distância temporal vai sendo aumentada. Por outro lado, o erro do Informer Univariado é bem mais estável que o do Time Series Transformer, enquanto o erro do Informer nas menores distâncias temporais é bem menor, para maiores distâncias o Time Series Transformer apresenta desempenho bem superior. O maior erro encontrado nas janelas de previsão do Informer foi cerca de 2,4 e o menor cerca de 1,3 (superando os demais *transformers* na maior janela testada).

3.3.6.12 Comparação geral dos modelos

A fim de visualizar o desempenho dos modelos comparativamente, foi plotado um gráfico com a performance de cada um para cada distância temporal de previsão. O modelo ARIMA e o modelo Autoformer Multivariado foram excluídos dessa comparação, por terem apresentado desempenhos muito ruins em comparação com os demais. Além disso, em subseções anteriores foram destacadas peculiaridades do modelo ARIMA que inviabilizaram a realização da sua validação da mesma maneira que foi feita com os demais modelos. Uma estratégia de *mean guessing* foi incluída. Esta estratégia é a estratégia ingênua mencionada anteriormente, ou seja, o erro de um modelo que sempre prevesse a média da variável alvo.

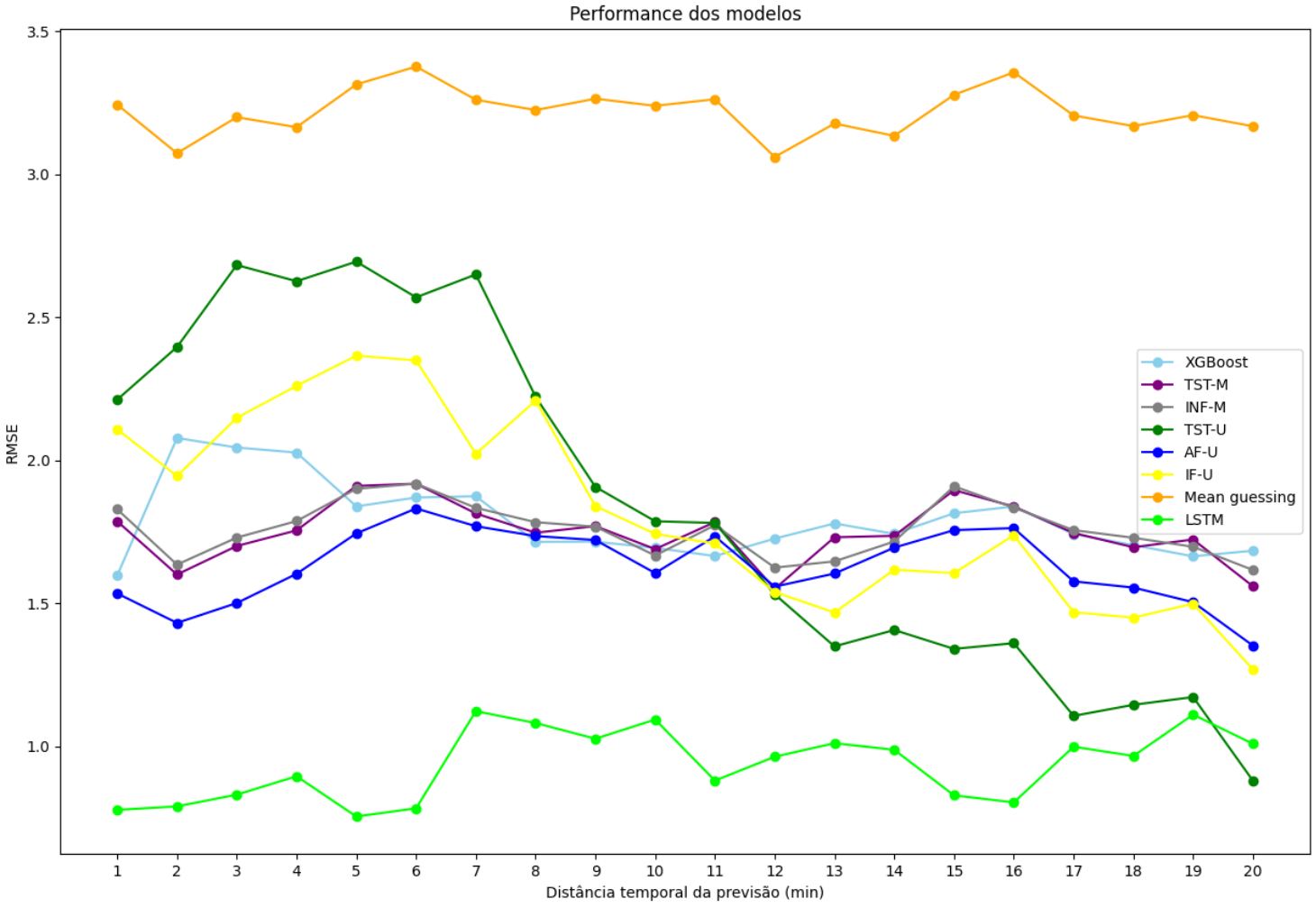

Figura 21. Quadro comparativo dos modelos desenvolvidos

O quadro mostra claramente a superioridade da rede neural LSTM em relação aos demais modelos. O erro deste modelo é o menor de todos para quase todas as janelas de previsão, com exceção relevante da maior janela de todas, quando a LSTM é superada pelo Time Series Transformer Univariado. Além dos resultados impressionantes da LSTM em comparação com os demais, é notável também a tendência de queda no erro apresentada pelos Time Series Transformer e Informer univariados. É possível que estes modelos tenham sido capazes de modelar relacionamentos a longo prazo na variável alvo melhor do que os relacionamentos de curto prazo, por mais contraintuitivo isso seja. O fato de que esse comportamento não foi observado no outro modelo univariado, o Autoformer, é um indício de que o mecanismo de auto-atenção tradicional teve melhor capacidade de modelar as relações de longo prazo do que o Autoformer, mesmo sendo uma arquitetura desenhada especificamente para tarefas de previsão a longo prazo.

Os modelos transformers multivariados, o Autoformer Univariado e o modelo XGBoost apresentaram desempenho bastante próximo, sugerindo que os transformers não foram capazes de modelar relações muito complexas nos dados, considerando que não se saíram muito melhor que um modelo de aprendizado de máquina clássico, o mecanismo de auto-atenção não parece ter sido potencializado pelos modelos.

Por outro lado, como o gráfico evidencia, todos os modelos se saíram melhor que a estratégia ingênua, que fornece um *baseline* bem útil neste caso, já que pode ser utilizada para futuramente determinar a viabilidade da aplicação de alguns destes modelos.

4. Conclusão e trabalhos futuros

Este trabalho explorou a aplicação do aprendizado de máquina para a previsão de falhas no processo de desodorização de alimentos. A partir da análise exploratória de dados e do pré-processamento de dados de séries temporais, foram desenvolvidos modelos utilizando o modelo ARIMA, o algoritmo XGBoost e redes neurais de arquitetura LSTM e Transformer. Os resultados iniciais indicaram que o modelo ARIMA não foi tão eficaz, o que era esperado devido à natureza simples e univariada do algoritmo. Além disso, o modelo multivariado baseado na arquitetura Autoformer apresentou muitos problemas no processo de implementação e erros muito acima do esperado e não se mostrou nem um pouco viável.

Os demais modelos foram implementados e avaliados como planejado, buscando-se observar o *trade-off* existente entre antecedência de uma previsão e qualidade da mesma. As validações apresentaram resultados bastante surpreendentes e intrigantes: todos os modelos mostraram-se ao menos tão efetivos nas maiores distâncias temporais quanto nas menores e alguns se mostraram bem melhores quando lidando com horizontes de previsão maiores, caso do Time Series Transformer Univariado e do Informer Univariado. Como ressaltado anteriormente, estes dois algoritmos possuem diversas similaridades e não foi surpreendente que eles tenham apresentado desempenho tão similar tanto no caso univariado quanto no multivariado.

Apesar de nas menores distâncias de previsões os modelos multivariados terem

apresentado superioridade em relação aos seus equivalentes univariados, os resultados encontrados parecem ao menos em parte corroborar uma tendência apresentada em outros trabalhos com o artigo introdutor do Autoformer: *transformers* apresentando dificuldades em modelar relações cruzadas nas diferentes séries temporais. Isso fica muito evidente quando se observa o desempenho do Autoformer Univariado em comparação com os *transformers* multivariados, os resultados são similares, mas em todas as janelas de previsões há uma pequena vantagem do Autoformer Univariado, que por sinal, é bem mais eficiente em termos de recursos computacionais pela dimensionalidade muito reduzida do dado.

De todos os resultados encontrados, o mais surpreendente foi o desempenho da rede neural LSTM. Além de superar os modelos Transformer, que vêm substituindo redes LSTM em diversas aplicações, a LSTM apresentou erro bastante estável, e enquanto os demais modelos se mostraram no máximo duas vezes melhores que a estratégia ingênua, a LSTM apresenta erro cerca de três vezes menor para a maioria das distâncias temporais, o que evidencia maior significância dos resultados. Isso não significa que seria viável implantar este modelo na cadeia de produção da indústria alimentícia, apenas indica que a rede LSTM se mostrou melhor que os demais modelos. É possível que a LSTM tenha se saído melhor devido a dificuldades dos modelos *transformers* implementados de modelar relações entre as diferentes *features*, enquanto a LSTM tenha sido capaz de fazer o mesmo, afinal, vale ressaltar, a rede LSTM desenvolvida era multivariada, ou seja, as diversas *features* sobre temperatura dos componentes do desodorizador, pressão de vapor, etc. são levadas em consideração pelo modelo.

Seria interessante em trabalhos futuros explorar arquiteturas que atacam exatamente este problema mencionado, que possivelmente explica o porquê dos modelos Transformer multivariados não terem se saído tão bem: a dificuldade de modelar relações entre diferentes séries temporais. Uma das arquiteturas mais interessantes que foram introduzidas recentemente é o CrossFormer (Zeng, Chen et al. 2023), cuja aplicação em um contexto similar ao deste trabalho seria extremamente pertinente.

5. Referências bibliográficas

Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller 2018. Deep learning for time series classification: a review. *arXiv: Learning*.

Ade Pitra Hermawan, Dong-Seong Kim, and Jae-Min Lee 2020. Predictive Maintenance of Aircraft Engine using Deep Learning Technique. *Information and Communication Technology Convergence*.

Nicholas Pudjihartono, Tayaza Fadason, Andreas W. Kempa-Liehr, and Justin M. O'Sullivan 2022. A Review of Feature Selection Methods for Machine Learning-Based Disease Risk Prediction. *Frontiers in bioinformatics*.

Chong Chen, Ying Liu, Xianfang Sun, Carla Di Cairano-Gilfedder, and Scott Titmus 2019. Automobile maintenance prediction using deep learning with GIS data. *Procedia CIRP*.

James M. McCracken 2016. Exploratory Causal Analysis with Time Series Data. *Exploratory Causal Analysis with Time Series Data*.

Christoph Bergmeir, and José Manuel Benítez 2012. On the use of cross-validation for time series predictor evaluation. *Information Sciences*.

John Cristian Borges Gamboa 2017. Deep Learning for Time-Series Analysis. *arXiv: Learning*.

Muhammad Sohaib, Muhammad Sohaib, Shiza Mushtaq, Jia Uddin, and Jia Uddin 2021. Deep Learning for Data-Driven Predictive Maintenance. *null*.

Julia Gastinger, Sebastien Nicolas, Dusica Stepic, Mischa Schmidt, and Anett Schulke 2021. A study on Ensemble Learning for Time Series Forecasting and the need for Meta-Learning. *IEEE International Joint Conference on Neural Network*.

Gladilin Peter, Gladilin Peter, and Maria Matskevichus 2019. Hyperparameters Tuning for Machine Learning Models for Time Series Forecasting. *International Conference on Social Networks Analysis, Management and Security*.

Nuno Moniz, Paula Branco, and Luís Torgo 2017. Resampling strategies for imbalanced time series forecasting. *Journal of data science*.

Lorenzo Camponovo, Olivier Scaillet, and Fabio Trojani 2010. Robust Resampling Methods for Time Series. *null*.

Ola Surakhi, Ola Surakhi, Martha A. Zaidan, Pak Lun Fung, Naser Hossein Motlagh, Sami Serhan, Mohammad AlKhanafseh, Rania M. Ghoniem, and Tareq Hussein 2021. Time-Lag Selection for Time-Series Forecasting Using Neural Network and Heuristic Algorithm. *Electronics*.

Dumitru Erhan, Aaron Courville, Yoshua Bengio, and Pascal Vincent 2010. Why Does Unsupervised Pre-training Help Deep Learning?. *International Conference on Artificial Intelligence and Statistics*.

Antonio Rafael Sabino Parmezan, Vinícius Mourão Alves de Souza, Vinícius Mourão Alves de Souza, and Gustavo E. A. P. A. Batista 2019. Evaluation of statistical and machine learning models for time series prediction: identifying the state-of-the-art and the best conditions for the use of each model. *Information Sciences*.

Lei Li, Yihang Ou, Yabin Wu, Qi Li, Qi Li, and Daoxin Chen 2018. Research on feature engineering for time series data mining. *IEEE International Conference on Network Infrastructure and Digital Content*.

Martin Längkvist, Lars Karlsson, and Amy Loutfi 2014. A review of unsupervised

feature learning and deep learning for time-series modeling ☆. *Pattern Recognition Letters*.

Ian D. Jordan, Piotr Aleksander Sokol, and Il Memming Park 2019. Gated recurrent units viewed through the lens of continuous time dynamical systems. *arXiv: Learning*.

Zhongyang Han, Jun Zhao, Jun Zhao, Henry Leung, King Ma, King Fai Ma, Wei Wang, Wei Wang, Wei Wang, and Wei Wang 2019. A Review of Deep Learning Models for Time Series Prediction. *IEEE Sensors Journal*.

Eric Zivot, and Jiahui Wang 2003. Vector Autoregressive Models for Multivariate Time Series. *null*.

Sean J. Taylor, and Benjamin Letham 2017. Forecasting at Scale.. *PeerJ*.

Adamantios Ntakaris, Giorgio Mirone, Juho Kanniainen, Moncef Gabbouj, and Alexandros Iosifidis 2019. Feature Engineering for Mid-Price Prediction with Deep Learning. *arXiv: Statistical Finance*.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, Wancai Zhang 2021. Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting *arXiv: Machine Learning*.

Haixu Wu, Jiehui Xu, Jianmin Wang, Mingsheng Long 2021. Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting *arXiv: Machine Learning*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin 2017. Attention Is All You Need *arXiv: Computation and Language*

Yunhao Zhang, Junchi Yan 2023.Crossformer: Transformer Utilizing Cross-Dimension Dependency for Multivariate Time Series Forecasting *ICLR 2023*

Marco Cerliani 2022. Time Series Forecasting with Feature Selection: Why you may need it.
https://towardsdatascience.com/time-series-forecasting-with-feature-selection-why-you may-need-it-696b23ecc329

Andrej Karpathy 2015. The Unreasonable Effectiveness of Recurrent Neural Networks https://karpathy.github.io/2015/05/21/rnn-effectiveness/

Andrew Plummer 2020. Different Types of Time Series Decomposition https://towardsdatascience.com/different-types-of-time-series-decomposition-396c09f92693