UNIVERSIDADE FEDERAL DE JUIZ DE FORA
FACULDADE DE ENGENHARIA
PROGRAMA DE PÓS-GRADUAÇÃO EM MODELAGEM COMPUTACIONAL

Victor Hugo Soares Pereira

Aplicação de Algoritmos de Aprendizado de Máquina na Previsão do Indicador de Tempo de Permanência de Veículos em uma Usina Siderúrgica

Juiz de Fora

2024

Victor Hugo Soares Pereira

Aplicação de Algoritmos de Aprendizado de Máquina na Previsão do Indicador de Tempo de Permanência de Veículos em uma Usina Siderúrgica

Dissertação apresentada ao Programa de pós-graduação em Modelagem Computacional da Universidade Federal de Juiz de Fora como requisito parcial à obtenção do título de Mestre em Modelagem Computacional.

Orientador: Prof. D. Sc. Eduardo Pestana de Aguiar

Juiz de Fora

2024

Ficha catalográfica elaborada através do Modelo Latex do CDC da UFJF

com os dados fornecidos pelo(a) autor(a)

Pereira, Victor Hugo Soares.

Aplicação de Algoritmos de Aprendizado de Máquina na Previsão do Indicador de Tempo de Permanência de Veículos em uma Usina Siderúrgica / Victor Hugo Soares Pereira. – 2024.

58 f. : il.

Orientador: Eduardo Pestana de Aguiar

Dissertação (Mestrado) – Universidade Federal de Juiz de Fora, Faculdade DE ENGENHARIA. Programa de pós-graduação em Modelagem Computacional, 2024.

1. Aprendizado de máquina. 2. Logística . 3. Indústria. I. de Aguiar, Eduardo Pestana, orient. II. Título.

Victor Hugo Soares Pereira

Aplicação de Algoritmos de Aprendizado de Máquina na Previsão do Indicador de Tempo de Permanência de Veículos em uma Usina Siderúrgica

Dissertação apresentada ao Programa de Pós-Graduação em Modelagem Computacional da Universidade Federal de Juiz de Fora como requisito parcial à obtenção do título de Mestre em Modelagem Computacional. Área de concentração: Modelagem Computacional.

Aprovada em 18 de dezembro de 2024.

BANCA EXAMINADORA

Prof. Dr. Eduardo Pestana de Aguiar - Orientador
Universidade Federal de Juiz de Fora

Prof. Dr. Rafael Antunes Nóbrega
Universidade Federal de Juiz de Fora

Prof. Dr. Jorge Luís Machado do Amaral
Universidade do Estado do Rio de Janeiro

Juiz de Fora, 17/12/2024.

Documento assinado eletronicamente por **Eduardo Pestana de Aguiar**, **Professor(a)**,

em 20/12/2024, às 16:22, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Rafael Antunes Nobrega**, **Professor(a)**, em 06/01/2025, às 17:06, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Jorge Luís Machado do Amaral**, **Usuário Externo**, em 22/01/2025, às 07:58, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no Portal do SEI-Ufjf (www2.ufjf.br/SEI) através do ícone Conferência de Documentos, informando o código verificador **2163026** e o código CRC **EB96C2D2**.

RESUMO

O Tempo de Permanência de Veículos (TPV) é um dos principais indicadores de eficiência operacional do processo de recebimento de sucata metálica de uma usina siderúrgica. Uma previsão precisa do TPV permite não apenas uma gestão mais eficiente de recursos, mas também a minimização de gargalos, o que resulta em uma operação mais ágil e econômica. Nesse sentido, diversos são os modelos de aprendizado de máquina que tratam da previsão de séries temporais. Este estudo tem o objetivo de remodelar uma base de dados transacional do processo de recebimento de caminhões de descarga de sucata metálica, com dados de Janeiro a Setembro de 2021, em uma série temporal do indicador de TPV e aplicar algoritmos de aprendizado de máquina, tanto clássicos quanto baseados em lógica fuzzy, para prevê-la. Foram aplicados métodos de *grid search* para busca dos melhores hiperparâmetros para os modelos, além de avaliar suas generalizações através de validação-cruzada. O desempenho dos modelos foi avaliado com base em métricas como *Mean Average Error* (MAE) e *Root Mean Average Error* (RMSE), e os resultados mostraram que o modelo *NTSK-wRLS*, baseado em lógica fuzzy, apresentou a maior precisão, destacando-se por sua capacidade de adaptação às dinâmicas do TPV e capturando, de forma eficaz, as variações sazonais e operacionais do processo. Por fim, este trabalho também contribui para a aplicação prática do aprendizado de máquina em ambientes industriais, evidenciando seu potencial para aprimorar processos de tomada de decisão.

Palavras-chave: **aprendizado de máquina**; **logística**; **usina siderúrgica**; **previsão de séries temporais**

ABSTRACT

Length of Stay (LOS) is one of the key process indicators of operational efficiency in the scrap metal inbound process of a steel plant. Accurate prediction of LOS not only allows for more efficient resource management but also minimizes bottlenecks, resulting in a more agile and cost-effective operation. In this context, various machine learning models address time series forecasting. This study aims to reshape a transactional database of the scrap metal inbound process, with data from January to September 2021, into a time series of the LOS indicator and apply machine learning algorithms, both classical and fuzzy logic-based, to forecast it. Grid search methods were used to find the best hyperparameters for the models, and their generalizations were evaluated through cross-validation. The models' performance was evaluated based on metrics such as Mean Average Error (MAE) and Root Mean Average Error (RMSE), and the results showed that the fuzzy logic-based NTSK-wRLS model demonstrated the highest accuracy, standing out for its ability to adapt to the dynamics of the LOS series and effectively capturing the seasonal and operational variations of the process. Finally, this work also contributes to the practical application of machine learning in industrial environments, highlighting its potential to enhance data-driven decision-making processes.

Keywords: **machine learning**; **logistics**; **steel plant**; **time series forecasting**

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

1.1 Contexto do estudo

A crescente complexidade das cadeias de suprimentos em nível global tem impulsionado empresas de grande porte a investirem continuamente em soluções tecnológicas como um diferencial competitivo (1). Em um cenário caracterizado por alta dinamicidade e interconectividade, a adoção de tecnologias como automação, inteligência artificial e análise de *big data* permite não apenas a otimização dos processos logísticos, mas também a redução de custos operacionais, a melhoria da rastreabilidade dos materiais e uma resposta mais ágil às flutuações da demanda e às mudanças nas condições de mercado (2, 3).

A logística em uma usina de produção de aço é caracterizada por ser uma operação de alto volume e complexidade, envolvendo múltiplos agentes em sua cadeia de suprimentos (4). O sucesso operacional de uma usina depende diretamente da eficiência com que sua cadeia de suprimentos é gerenciada, desde a aquisição da matéria-prima até a expedição dos produtos acabados. Uma coordenação inadequada ao longo desse processo pode resultar em impactos negativos, como aumento dos custos logísticos, desperdício de recursos, interrupções na linha de produção e, consequentemente, perda de competitividade.

Dentre os materiais envolvidos nesta cadeia logística, o recebimento da sucata metálica se destaca como um dos mais representativos, com um volume que pode ultrapassar milhares de toneladas diárias, além de se destacar como um recurso crítico em um modelo de economia circular, uma vez que o aço tem a vantagem de ser totalmente reciclável (5).

Entretanto, a logística de recebimento de sucata metálica apresenta diversos desafios. O fluxo de entrada e saída de caminhões carregados de sucata metálica é de grande importância para a eficiência logística e operacional, e deve ocorrer de forma eficiente a fim de evitar atrasos no recebimento de matéria-prima (6). Dessa forma, com o objetivo de monitorar a eficiência da logística de recebimento, diversos indicadores de desempenho são empregados, sendo o Tempo de Permanência de Veículos (TPV) um dos mais relevantes (7, 8, 9, 10, 11). Esse indicador mensura o intervalo total de tempo decorrido entre a chegada do veículo à usina e sua saída após a conclusão do descarregamento. Valores elevados de TPV podem sinalizar ineficiências operacionais, tais como falta de sincronização entre a programação de recebimento e a capacidade de descarga, atrasos em procedimentos burocráticos ou baixa disponibilidade de infraestrutura para manuseio do material. Em

contrapartida, a redução do TPV pode trazer benefícios diretos, como maior previsibilidade nas operações, aumento da capacidade de processamento de carga, menor necessidade de estacionamento prolongado dentro das dependências da usina e redução de custos associados a períodos ociosos.

Além dos impactos operacionais e financeiros, a otimização do TPV possui implicações diretas nas relações trabalhistas e na conformidade com a legislação vigente. A regulamentação do setor de transporte rodoviário de cargas, como estabelecido pela Lei 13.103/2015 no Brasil (12), determina limites para o tempo de espera de motoristas durante o processo de carga e descarga, buscando garantir melhores condições de trabalho e reduzir os riscos associados à fadiga e à exaustão profissional. O descumprimento dessas normativas pode acarretar penalidades legais para as empresas e prejudicar as relações com transportadoras e fornecedores, tornando ainda mais relevante o monitoramento e a melhoria contínua do TPV.

Figura 1 – Exemplo de caminhões com sucata metálica sendo descarregados

Fonte: DB Metal Arabia

1.2 Justificativa do estudo

Em estudo prévio (13), foi verificado que a previsão do TPV por veículo na empresa em que esta pesquisa foi conduzida - uma grande planta siderúrgica brasileira - baseia-se em uma suposição simplificada, utilizando um valor fixo de vazão (toneladas por minuto) para todas as configurações de caminhão e material. Os resultados evidenciaram que o uso de técnicas de aprendizado de máquina para o cálculo do TPV permite uma redução significativa das métricas de erro das previsões.

A partir da previsão mais precisa dos valores de TPV para cada caminhão, é possibilitada a criação de diversas estratégias que podem minimizar os impactos no indicador gerencial, como a redistribuição de baias de descarga, re-priorização de caminhões e re-dimensionamento das horas de mão-de-obra contratada. O sucesso obtido nesse estudo inicial justifica uma exploração mais aprofundada dos dados oriundos do processo de descarga de sucata metálica, considerando diferentes hipóteses e abordagens analíticas.

Uma dessas possíveis abordagens consiste em considerar o valor do indicador gerencial de TPV e reformular o problema de regressão como um problema de previsão de série temporal. Dessa forma, é possível obter uma base de dados com o valor do indicador em diversas janelas de tempo ao longo do período em que os dados foram coletados. E esta é a abordagem utilizada neste estudo.

Mais uma vez são utilizadas técnicas de aprendizado de máquina para agregar valor a um processo real. Os modelos de análise preditiva se destacam pela capacidade de se adaptarem às mudanças de um ambiente dinâmico e com muitas partes envolvidas, como: flutuação de demanda, filas, problemas operacionais como indisponibilidade de máquinas, greves, entre outros. A utilização destes métodos não só aprimora as previsões, como também contribui para a criação de um sistema de tomada de decisão mais robusto e proativo, permitindo à empresa antecipar e mitigar possíveis gargalos logísticos.

1.3 Declaração do objetivo geral

O objetivo deste estudo é avaliar modelos de aprendizado de máquina para a tarefa de previsão da série temporal do indicador gerencial de TPV do processo de recebimento de sucata de ferro.

1.4 Escopo do estudo

O escopo deste trabalho possui os seguintes objetivos específicos:

- Remodelar a base de dados transacional do processo de descarga de caminhões de sucata para criar uma série temporal do indicador gerencial de TPV, em janelas de 30 minutos.
- Realizar engenharia de variáveis para identificar e criar atributos que possam contribuir para a melhoria da acurácia das previsões.
- Avaliar e comparar o desempenho de modelos clássicos de aprendizado de máquina e uma abordagem baseada em lógica *Fuzzy*
- Explorar e otimizar hiperparâmetros com o objetivo de minimizar os erros de previsão dos modelos avaliados.
- Implementar a validação cruzada dos modelos selecionados para consolidar os resultados através de múltiplas iterações de previsão.
- Analisar e discutir os resultados obtidos, propondo direções para estudos futuros e possíveis aplicações práticas.

1.5 Divisão do trabalho

- Capítulo 2: Neste capítulo, é abordado o indicador gerencial de Tempo de Permanência de Veículos (TPV), explorando sua importância no contexto logístico da usina siderúrgica. Também são apresentados os detalhes da base de dados utilizada e as transformações necessárias para a construção da série temporal do indicador.
- Capítulo 3: Este capítulo discute os diferentes modelos de aprendizado de máquina que foram aplicados para prever o indicador TPV. São descritos tanto algoritmos clássicos quanto técnicas baseadas em lógica fuzzy, destacando suas características, vantagens e desvantagens.
- Capítulo 4: São apresentados os resultados obtidos a partir dos experimentos realizados com os modelos propostos. São discutidos os procedimentos de otimização de hiperparâmetros e validação-cruzada, bem como as métricas de desempenho

utilizadas para avaliar os modelos. Além disso, são demonstrados os resultados das avaliações dos modelos e os gráficos de suas previsões.

- Capítulo 5: Este capítulo conclui o trabalho, sintetizando as principais descobertas, discutindo as implicações dos resultados e propondo direções para estudos futuros e aplicações práticas.

2 Formulação do problema

Neste capítulo será explorado o indicador gerencial do TPV e apresentada sua importância para a empresa. Além disso, será apresentada a base de dados anonimizada do processo de recebimento de sucata metálica.

2.1 Tempo de permanência do veículo (TPV)

O Tempo de Permanência do Veículo (TPV) é um dos *Key Process Indicators* (KPI) de maior relevância no contexto da logística de recebimento, sendo amplamente utilizado para monitorar e avaliar a eficiência operacional das atividades de descarga de matérias-primas. Este indicador representa o intervalo de tempo total que um veículo permanece na usina, abrangendo desde o momento de sua entrada, com a matéria-prima destinada ao processamento, até sua saída, já descarregado e vazio.

O acompanhamento contínuo do TPV faz parte da gestão logística da usina, uma vez que ele fornece uma visão detalhada do desempenho dos processos operacionais e pode refletir gargalos, ineficiências ou problemas estruturais. Nesse sentido, qualquer variação inesperada ou anomalia identificada no comportamento desse KPI é tratada com prioridade pela administração da usina. Um comitê de crise é imediatamente convocado para conduzir análises detalhadas sobre as possíveis causas de possíveis intercorrências, propondo e implementando ações corretivas que visem a mitigação de problemas e a manutenção da fluidez do fluxo logístico.

Além do impacto operacional, o monitoramento do TPV também está diretamente relacionado ao cumprimento de normas e legislações que regem as condições de trabalho dos motoristas. A Lei 13.103/2015, conhecida como Lei do Caminhoneiro, estabelece diretrizes claras sobre os direitos dos motoristas, incluindo limites de tempo para espera e operação em locais de carregamento e descarregamento (12). Assim, desvios significativos no indicador podem indicar não apenas questões internas, mas também potenciais infrações legais, destacando a importância de sua manutenção dentro de padrões aceitáveis.

O cálculo do TPV como um indicador gerencial - diferente do TPV de cada caminhão - é realizado com base na média acumulada dos tempos de permanência de todos os caminhões recebidos ao longo do dia, até o momento de interesse. Em termos práticos, isso significa que, para qualquer horário dentro do período analisado, o TPV é obtido pela

média aritmética dos tempos registrados para cada caminhão que ingressou na usina até aquele instante. Essa metodologia de cálculo proporciona uma representação agregada do desempenho logístico diário, sendo amplamente utilizada para a identificação de tendências e o suporte à tomada de decisão em tempo real.

Por fim, é importante destacar que o TPV não apenas quantifica a eficiência do processo logístico, mas também serve como base para análises preditivas e avaliações de desempenho de diferentes setores da planta. Por exemplo, variações significativas no TPV entre turnos ou em diferentes dias da semana podem estar associadas a mudanças no volume de veículos, características das matérias-primas ou alterações nos procedimentos operacionais. Essa sensibilidade às condições operacionais faz do TPV uma métrica amplamente valorizada em estudos que buscam a otimização logística e o aperfeiçoamento de sistemas industriais.

2.2 A base de dados original

A base de dados utilizada neste estudo foi anonimizada com o intuito de proteger informações sensíveis e preservar a confidencialidade das operações. Essa base compreende um total de 24.974 registros detalhados relacionados à operação de descarga de caminhões, abrangendo o período de janeiro a setembro de 2021. Cada registro inclui as informações necessárias para o cálculo do TPV, como o horário de entrada e saída de cada veículo nas instalações. Além disso, a base de dados contém diversas variáveis que caracterizam o caminhão utilizado no transporte , como o tipo de composição utilizada (por exemplo, caminhões simples ou bitrens), o Peso Bruto Total (PBT) de cada veículo e o peso líquido das cargas transportadas. Outro aspecto importante são as variáveis que caracterizam o processo, como a alocação de pontos de descarga e a presença de múltiplos materiais sendo transportados por um mesmo veículo, o que adiciona um nível adicional de complexidade ao processo de descarga, uma vez que o caminhão deve descarregar um material por vez, realizar mais pesagens e entrar mais vezes nas filas.

Além disso, a base inclui duas variáveis muito interessantes para o estudo do TPV: a estimativa original do indicador, calculada pelo método tradicional utilizado pela usina, e os valores reais medidos do TPV, conforme visto em estudo anterior (13), em que foi proposta uma abordagem baseada em aprendizado de máquina para previsão do TPV de cada caminhão.

Os dados e códigos deste estudo podem ser encontrados no repositório: `https://github.com/VictorHSPereira/dissertacao_pgmc` (14)

2.3 Transformações de dados

Com o objetivo de transformar os dados originais para representarem o objeto de estudo, foi necessário implementar um método de agrupamento dos registros de saída dos caminhões em intervalos de 30 minutos. A escolha das janelas de 30 minutos se deu por conveniência para com o processo real, uma vez que janelas menores seriam muito tempestivas e janelas maiores não capturariam com tanta precisão as dinâmicas intra-diárias. Após o agrupamento foi necessário calcular, para cada janela de tempo, a média acumulada do TPV, resultando em uma série temporal que representa de forma consolidada o comportamento do indicador gerencial, tal como é realizado na rotina operacional da usina.

Além disso, com o intuito de enriquecer a análise da série temporal gerada e permitir a inclusão de informações que capturam dinâmicas mais complexas, foram construídas novas variáveis derivadas, conhecidas como variáveis de características ou *features*. Esse processo seguiu práticas consolidadas na literatura de séries temporais e incluiu várias estratégias de enriquecimento dos dados. Primeiramente, foi empregada a criação de defasagens (*lags*), que são variáveis que representam o valor do TPV em instantes de tempo anteriores com o objetivo de caputrar a autocorrelação inerente à série temporal. Em seguida, foram incorporadas variáveis para identificar dias especiais no calendário, como feriados ou datas específicas em que há uma alteração anômala no comportamento da série. A inclusão dessas variáveis possibilita capturar os efeitos sazonais e eventos pontuais que afetam diretamente o TPV, aumentando a capacidade do modelo de prever mudanças abruptas no padrão observado. Outro passo importante foi a inclusão de variáveis cíclicas para representar padrões temporais associados a ciclos diários, semanais ou sazonais. Para isso, as informações temporais, como o horário do dia ou o dia da semana, foram transformadas utilizando funções trigonométricas (seno e cosseno). Essa técnica, amplamente reconhecida na literatura, permite que variáveis temporais categóricas sejam representadas de forma contínua, preservando a estrutura cíclica do tempo e facilitando a captura de variações periódicas na série. Por fim, foram adicionadas variáveis que refletem o estado do processo operacional no momento de cada registro, como a quantidade de veículos presentes na

usina durante a janela de tempo, com o objetivo de modelar a possibilidade de filas e outros gargalos devido ao nível de ocupação do processo de descarga.

2.3.1 Lags temporais

A geração de lags temporais é uma técnica amplamente utilizada em problemas de previsão de séries temporais. Um *lag* refere-se ao atraso de uma variável em relação ao histórico do seu valor. Foram escolhidos os valores de lag de 1 hora, 1 dia e 1 semana para a variável de TPV pelo padrão cíclico de dias da semana e causalidade com o processo de descarga de metálicos.

- ***Lag* de 1 hora:** Este *lag* permite capturar variações de curto prazo do TPV, que podem ser influenciadas por flutuações imediatas no fluxo de caminhões, como geração de filas ou outros gargalos logísticos. Ao considerar o TPV de uma hora anterior, o modelo pode ajustar suas previsões com base em eventos que ocorreram no período anterior ao da previsão.
- ***Lag* de 1 dia:** A inclusão do *lag* de 1 dia visa capturar padrões diários que possam existir no processo logístico. É comum que haja uma dependência em relação ao horário, com padrões que se repetem diariamente, como trocas de turno. Considerando o TPV do mesmo período no dia anterior, o modelo pode reconhecer a ajustar a previsão a esses padrões recorrentes.
- ***Lag* de 1 semana:** O lag de 1 semana é interessante para capturar padrões semanais, em que a atividade da usina pode ser significativamente diferente com base no dia da semana. Esse lag permite que o modelo considere o TPV do mesmo período, na semana anterior, reconhecendo, por exemplo, como a operação pode ser afetada pelo início da semana ou seu fim.

2.3.2 Dias especiais

Para diferenciar registros referentes a feriados e fins de semana dos dias úteis, além de identificar a primeira quinzena do mês, foram criadas variáveis binárias específicas. É importante evidenciar estes períodos pois a operação logística apresenta comportamentos significativamente diferentes entre eles. Para obter os dados de feriados no Brasil, foi utilizada a biblioteca *holidays* (15) disponível para a linguagem Python.

Essas variáveis binárias, portanto, permitem a modelagem das diferenças entre os padrões observados durante dias úteis, dias especiais e primeira e segunda quinzenas do mês, como escala de trabalhadores, diferença na atividade industrial e variação na demanda, por exemplo.

2.3.3 Quantidade de veículos dentro da usina

Foi criada a variável que recebe a quantidade de veículos dentro da usina para cada registro, uma vez que a quantidade de veículos pode influenciar diretamente a formação de filas e, consequentemente, o TPV. Quando a quantidade de veículos é alta, pode haver um aumento na probabilidade de congestionamentos e filas nas áreas de descarga. Isso pode levar a maiores tempos de espera e a uma possível diminuição da eficiência operacional. Períodos de alta concentração de veículos podem estar associados a um aumento na frequência de incidentes, como atrasos e quebras de equipamentos, por exemplo. Por outro lado, um número reduzido de veículos pode indicar menor carga sobre a infraestrutura e processos mais ágeis.

2.3.4 Variáveis temporais periódicas

Para representar a natureza cíclica das variáveis periódicas como hora do dia e dia da semana, foram aplicadas transformações utilizando as funções seno e cosseno. Esta abordagem é mais eficaz do que a representação linear, uma vez que preserva a continuidade dos dados (16). Por exemplo, na representação linear a diferença entre as horas 23 e 0 é de 23 unidades, apesar de, na realidade, ser de apenas uma hora. A Figura 2 ilustra a disposição dos valores em um círculo unitário, refletindo a proximidade cíclica comentada.

De maneira análoga, a mesma representação foi construída para os dias da semana, como é possível verificar na Figura 3:

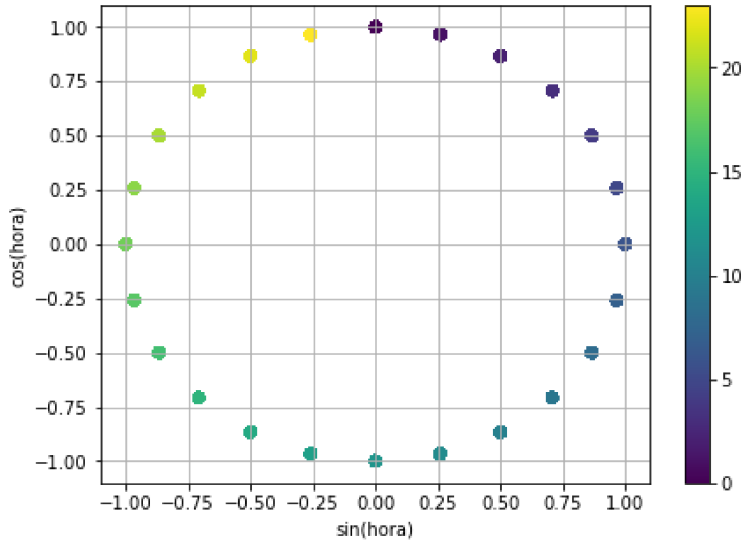

Figura 2 – Representação das horas do dia em um círculo trigonométrico

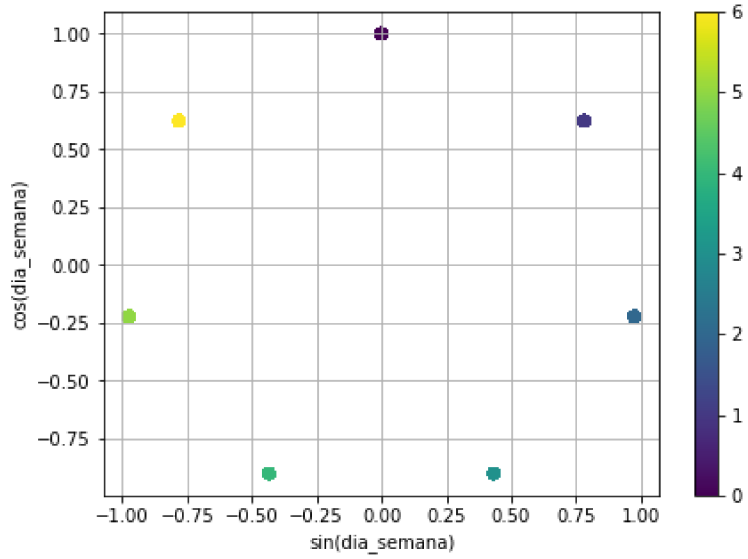

Figura 3 – Representação dos dias da semana em um círculo trigonométrico

2.3.5 Exemplo da base de dados final

A base final do projeto possui 10.271 entradas sequenciais de valores de TPV para cada janela de 30 minutos, de janeiro a setembro de 2021. Para demonstrar a estrutura e a composição da base, a Tabela 1 apresenta uma amostra aleatória composta por três registros extraídos do conjunto de dados gerado a partir do indicador gerencial de Tempo de Permanência do Veículo (TPV). Esses registros incluem variáveis originais e derivadas, as quais foram criadas ou transformadas durante o processo de preparação e enriquecimento dos dados.

	Registro 1	Registro 2	Registro 3
Hora_sin	-1.000000	0.965926	0.258819
Hora_cos	-1.836970e-16	2.588190e-01	9.659258e-01
Dia_semana_sin	0.000000	-0.433884	-0.433884
Dia_semana_cos	1.000000	-0.900969	-0.900969
QTD_VEICULOS_USINA	34	9	9
Primeira_quinzena	0	1	1
Feriado	0	1	1
Fim_semana	0	1	0
Lag_1h	1.669976	7.327900	6.805000
Lag_1d	0.000000	6.722042	8.109762
Lag_7d	1.982072	9.366667	5.091667
TPV	1.787069	7.469917	8.953889

Tabela 1 – Amostra aleatória de 3 registros da base de dados

A Tabela 2 apresenta as estatísticas descritivas da variável dependente, agrupadas por dia da semana, como média, percentis (25%, mediana e 75%) e desvio-padrão.

Dia da Semana	Média	25% Percentil	Mediana	75% Percentil
Segunda-feira	1.08 ± 1.04	0.00	0.90	1.99
Terça-feira	5.66 ± 2.97	4.90	6.14	7.55
Quarta-feira	5.21 ± 2.68	4.34	5.70	6.81
Quinta-feira	5.32 ± 2.83	4.26	5.82	6.94
Sexta-feira	5.25 ± 2.80	4.67	5.76	6.89
Sábado	5.62 ± 4.85	3.74	5.72	6.88
Domingo	5.05 ± 4.77	0.00	6.75	8.84

Tabela 2 – Estatísticas descritivas do indicador gerencial TPV, agrupadas por dia da semana.

Observa-se que as segundas-feiras apresentam o menor valor médio de TPV (1,08 horas), acompanhado de uma baixa variabilidade (desvio-padrão de 1,04 horas). Este comportamento pode ser explicado pela retomada das operações após o final de semana, quando o fluxo de caminhões é geralmente mais uniforme e as operações logísticas tendem a ocorrer de maneira mais controlada. Além disso, o percentil 25% igual a 0,00 horas indica que, em muitos casos, não há registros de atrasos ou longos períodos de espera, sugerindo uma operação bastante eficiente.

Por outro lado, a maior média de TPV é observada na terça-feira (5,66 horas), com um desvio-padrão de 2,97 horas. Os dias úteis intermediários (quarta, quinta e sexta-feira) mostram padrões relativamente homogêneos, com médias de TPV variando entre 5,21 e 5,32 horas e desvios-padrão entre 2,68 e 2,83 horas. Esses valores indicam um fluxo

regular, sem grandes oscilações, mas ainda assim com tempos de permanência superiores ao observado nas segundas-feiras. Esse comportamento sugere que a operação logística nesses dias é mais consistente, embora ainda sujeita a possíveis gargalos específicos.

Já os finais de semana (sábado e domingo) apresentam um padrão distinto. O sábado possui a maior variabilidade no TPV, com um desvio-padrão de 4,85 horas, e uma média de 5,62 horas, semelhante à terça-feira, enquanto o domingo possui um média de 5,05 horas com um desvio de 4,77 horas. Essa alta dispersão durante o fim de semana pode estar relacionada à redução de equipes operacionais, à priorização de determinados tipos de carga ou a demandas específicas de transporte em função do perfil de operação da usina. Outro ponto relevante é a análise dos percentis. Os dias úteis apresentam distribuições mais simétricas, com os valores de mediana próximos à média. Nos finais de semana, a assimetria fica mais evidente, sugerindo que há maior incidência de tempos de permanência extremos, o que corrobora a percepção de instabilidade operacional nesses períodos.

Essas análises evidenciam que o comportamento do TPV está fortemente influenciado por fatores sazonais e operacionais, além de variações na demanda ao longo dos dias da semana. Estes são insumos importantes para a modelagem e entendimento do problema.

A Figura 4 complementa a visualização, demonstrando a média do TPV por horário e dia da semana e exemplificando que há padrões sazonais e diários que influenciam o TPV, evidenciando, por exemplo, picos de atividade durante determinados horários ou dias específicos da semana.

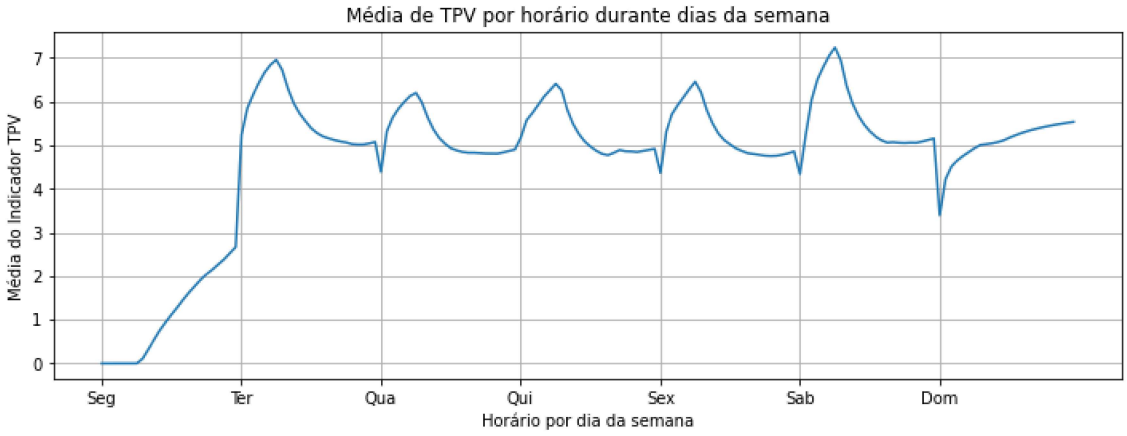

Figura 4 – Média de TPV por horário e dia da semana (dados agrupados)

3 Modelos propostos

Esta seção descreve os aspectos gerais dos modelos de aprendizado de máquina que foram utilizados neste trabalho para a tarefa de prever o indicador gerencial do TPV. Foram utilizados tanto algoritmos clássicos quanto algoritmos baseados em lógica *fuzzy*.

3.1 Modelos clássicos

3.1.1 *K-Nearest Neighbors(KNN)*

O *K-Nearest Neighbors* (*KNN*) (17) é um algoritmo de aprendizado de máquina supervisionado que se baseia no conceito de proximidade: cada ponto de dados é analisado com base nos valores das observações mais próximas no espaço de características. A distância entre os pontos é geralmente calculada usando a métrica Euclidiana (3.1), embora outras métricas, como *Manhattan* (3.2) ou *Minkowski* (3.3), possam ser também ser utilizadas.

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{n}(y_i - x_i)^2} \tag{3.1}$$

$$d(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n}|y_i - x_i| \tag{3.2}$$

$$d(\mathbf{x}, \mathbf{y}) = \left(\sum_{i=1}^{n}|y_i - x_i|^p\right)^{\frac{1}{p}} \tag{3.3}$$

onde x e y são dois vetores, sendo n o seu tamanho e p é um inteiro maior que zero. A distância Euclidiana e a *Manhattan* são casos especiais da distânica de *Minkowski*, com o valor de p sendo igual a 1 e 2, respectivamente.

A técnica se baseia na premissa de que dados que estão próximos no espaço de características provavelmente compartilham valores semelhantes. Para cada previsão, o algoritmo seleciona os k vizinhos mais próximos e decide o valor com base na maioria destes vizinhos. Este modelo é amplamente utilizado para tarefas de previsão (18, 19, 20, 21), principalmente para definir um *benchmark* inicial.

Vantagens:

- Algoritmo de lógica simples, excelente ponto de partida para tarefas de modelagem.

- Baixo tempo de treinamento quando comparado a algoritmos mais complexos.
- Tem boa adaptação a tipos de dados misturados, como dados contínuos e categóricos no mesmo espaço de características.

Desvantagens:

- Um alto número de dimensões reduz a significância da distância entre os pontos, o que pode reduzir o desempenho do modelo.
- As instâncias utilizadas no treinamento são mantidas em memórias para a fase de teste, o que aumenta o custo computacional do algoritmo e pode ser problemático em grandes conjuntos de dados.

3.1.2 Árvores de regressão

A árvore de regressão é um caso particular do algoritmo de árvores de decisão, amplamente utilizado para problemas de regressão (22, 23, 24). Seu algoritmo baseia-se na lógica de dividir recursivamente o espaço de características da base de dados em intervalos cada vez menores, de modo que os valores dentro de cada intervalo sejam os mais homogêneos possíveis. Por fim, durante a fase de treinamento, é construída uma árvore binária baseada em um conjunto de regras (25), como podemos ver no exemplo da Figura 5.

O algoritmo realiza a divisão de um intervalo de modo a minimizar a variabilidade dos valores da variável resposta dentro dos nós resultantes. Este processo continua até que um critério de parada seja atingido, como a profundidade máxima da árvore ou um número mínimo de observações em um nó terminal. Para a previsão, as regras de decisão são aplicadas à medida em que a árvore é percorrida até um nó terminal e então o valor médio das observações daquele nó é atribuído como valor previsto.

Apesar de sua simplicidade e interpretabilidade, as árvores de regressão sofrem com a tendência ao *overfitting*. Este problema surge quando o modelo se ajusta excessivamente aos dados de treinamento, capturando variações que não se generalizam para novos dados. Para mitigar este problema, uma técnica frequentemente utilizada é a poda da árvore, que consiste em remover ramos de uma árvore que possuem baixa importância para a tarefa de previsão, de forma a diminuir a complexidade do modelo e favorecer a generalização.

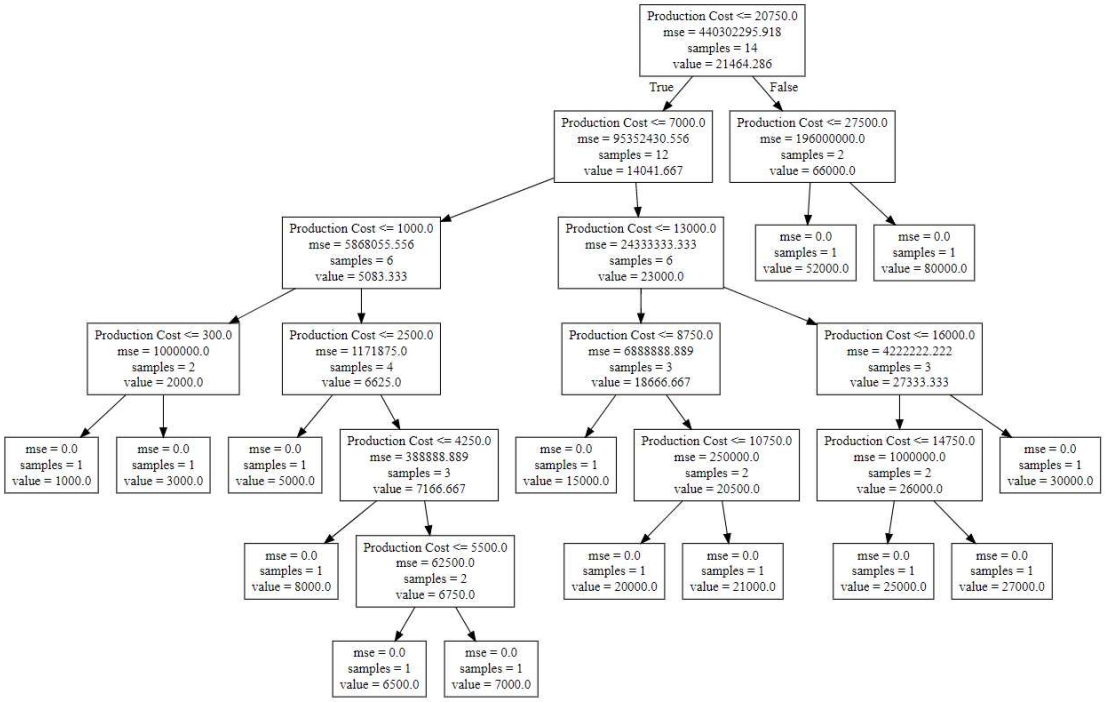

Figura 5 – Exemplo de uma Árvore de Regressão

Fonte: geeksforgeeks.org/python-decision-tree-regression-using-sklearn

Vantagens:

- Por ser baseado em um conjunto de regras de decisão, o algoritmo permite alta explicabilidade.
- As regras podem refletir interações entre variáveis que poderiam ser complexas de modelar com métodos lineares.

Desvantagens:

- Este algoritmo possui alta propensão ao *overfitting* caso não sejam aplicados critérios de parada, o que prejudica a generalização.
- Alta sensibilidade no período de treinamento. O re-treino com os mesmos dados re-ordenados pode gerar uma árvore completamente diferente.

3.1.3 *Gradient Boosting*

Gradient Boosting é uma técnica de *ensemble* que constrói modelos de forma iterativa, onde cada novo modelo tem objetivo de corrigir os erros cometidos pelos modelos predecessores (26, 27), método este chamado de *boosting* (28). Este processo é baseado em gradientes que guiam a função de erro, de forma a melhorar continuamente a precisão do modelo. O algoritmo se baseia na premissa de que a combinação ponderada de todos os modelos tende a gerar uma previsão final mais precisa.

No problema de regressão, o objetivo do algoritmo é encontrar uma função $F(x)$ que minimize a função de perda $L(y, F(x))$, dado um conjunto de treinamento $(x_1, y_1), \ldots, (x_T, y_T)$ com tamanho T, variáveis de entrada (ou preditores) x_t e o valor de saída correspondente y_t. O método aditivo para encontrar a solução ótima $\hat{F}(x)$ pondera os modelos fracos $h(x_t)$ gradualmente ao longo do procedimento de descida. A inicialização do algoritmo é feita com uma função constante $F_0(x)$, conforme descrito a pela Equação 3.4:

$$F_0(x) = \arg\min_{\gamma} \sum_{t=1}^{T} L(y_t, \gamma) \tag{3.4}$$

O algoritmo é então otimizado pelo gradiente negativo, conforme mostrado pela Equação 3.5:

$$z_m(x_t) = -\frac{\partial L(y_t, F_{m-1}(x_t))}{\partial F_{m-1}(x_t)} \tag{3.5}$$

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x) \tag{3.6}$$

onde $F_m(x)$ é a integração dos valores das árvores de regressão básicas, $h_m(x)$ é a m-ésima árvore de regressão, e γ_m é o coeficiente de ponderação da m-ésima árvore.

A próxima árvore de regressão $h_m(x)$ é construída com base nos valores de $z_m(x_t)$ e x. Os coeficientes γ_m são determinados pela equação 3.7:

$$\gamma_m = \arg\min_{\gamma} \sum_{t=1}^{T} L(y_t, F_{m-1}(x_t) - \gamma_m h_m(x_t)) \tag{3.7}$$

O desempenho do *Gradient Boosting Regressor* pode ser influenciado pelo número máximo de árvores, pela taxa de aprendizado e também pela profundidade máxima da árvore (29). A melhor combinação desses parâmetros possibilita o resultado ótimo do modelo.

O primeiro parâmetro refere-se ao número total de árvores (ou seja, modelos fracos) integradas ao *Gradient Boosting Regressor*. O segundo parâmetro define a contribuição de cada modelo fraco para os resultados finais, com valores entre 0 e 1. O terceiro parâmetro expressa a complexidade da árvore. O *Gradient Boosting Regressor* é um modelo forte formado pela combinação de diversos modelos fracos. Portanto, a profundidade máxima de cada árvore deve ser controlada para limitar a complexidade do sistema como um todo (29).

Vantagens:

- O *Gradient Boosting* generaliza bem em problemas complexos e de alta dimensão.
- Parâmetros como taxa de aprendizado e número máximo de iterações controlam o *overfitting* - caso em que o modelo se ajusta excessivamente aos dados de treinamento, prejudicando sua capacidade de generalização.

Desvantagens:

- Depenendo do critério de parada escolhido, o treinamento do método de *Gradient Boosting* pode ser relativamente demorado.
- Por conta de sua característica recursiva e complexa, a saída do modelo possui baixa explicabilidade.

3.1.4 *LightGBM*

O *LightGBM* é um algoritmo de *boosting* com capacidade de ter boa performance com grandes bases de dados. O método se baseia na construção de árvores de decisão tal como o *Gradient Boosting*, porém de forma mais eficiente (30). Este ganho de eficiência está relacionado principalmente a dois fatores: o método *Gradient-based One-Side Sampling* (*GOSS*), que consiste em priorizar amostras com gradientes altos para reduzir o esforço computacional e no crescimento da árvore de forma a priorizar as folhas (*leaf-wise*), que consiste em expandir as folhas com maior ganho primeiro, resultando em árvores assimétricas e mais profundas, consequentemente melhorando a precisão.

O GOSS é, basicamente, um algoritmo de amostragem para rebalanceamento. As amostras com grandes gradientes - aquelas em que há grande erro durante o treinamento - são mantidas, enquanto as amostras com pequenos gradientes são selecionadas aleatoriamente

e recebem pesos constantes. Dessa forma, o GOSS concentra-se mais nas amostras subtreinadas, sem alterar a distribuição dos dados originais. Com as amostras selecionadas e reponderadas, a árvore de decisão é construída como de costume, mas com um conjunto de dados muito menor, resultando em um processo de treinamento mais rápido (30, 31).

Vantagens:

- Por conta de técnicas como *GOSS* e o crescimento *leaf-wise*, o custo computacional é otimizado e o algoritmo é ágil na fase de treinamento.
- O *LightGBM* é projetado para lidar com volumes de dados extensos, sendo otimizado para consumo de memória.

Desvantagens:

- Devido ao método *leaf-wise* de construção de árvores, o *LightGBM* pode ser mais propenso ao *overfitting* se não forem aplicadas técnicas de regularização adequadas.
- O algoritmo possui uma grande quantidade de parâmetros, o que pode tornar a tarefa de *fine-tunning* muito custosa.

3.2 Modelos baseados em lógica fuzzy

Os sistemas baseados em lógica fuzzy surgiram como uma ferramenta para lidar com problemas em domínios em que a incerteza e subjetividade estão presentes. Baseados no trabalho de Zadeh em 1965 (32), esses sistemas se fundamentam no conceito de conjuntos fuzzy, que permitem a representação de elementos com graus de pertinência contínuos a um determinado conjunto ao contrário da lógica booleana clássica.

Na teoria tradicional dos conjuntos, a associação de um elemento a um conjunto é binária - é 0 ou 1. No exemplo a seguir, considerando um conjunto A em um universo X, os elementos de X simplesmente estão ou não estão em A, de acordo com a função $f_A(x)$:

$$f_A(x) = \begin{cases} 1 & \text{se e somente se } x \in A \\ 0 & \text{se e somente se } x \notin A \end{cases} \tag{3.8}$$

O estudo de Zadeh sugere que $f_A(x)$ é capaz de assumir infinitos valores dentro do intervalo $[0, 1]$. De maneira similar ao exemplo previamente apresentado, um conjunto fuzzy A em um universo X é caracterizado por uma função de pertinência $\mu_A(x) : X \to [0, 1]$ e é representado por um conjunto de pares ordenados da seguinte forma:

$$A = \{\mu_A(x)/x\} \quad x \in X \tag{3.9}$$

onde $\mu_A(x)$ indica o grau de pertinência de x com o conjunto A. Além disso, é relevante destacar que x pode pertencer a outros conjuntos fuzzy distintos de A, apresentando diferentes graus de pertinência para cada um dos conjuntos.

Entre os sistemas fuzzy mais amplamente utilizados, é possível citar o modelo Takagi-Sugeno, introduzido em 1985 ((33)). Este modelo cria regras do tipo "se-então"com uma função - linear ou não linear - como saída, conforme a equação a seguir:

$$\mathcal{R}_i: \quad \textbf{SE} \quad \underbrace{x \quad \text{é} \quad \mathcal{A}_i}_{\text{Antecedente}} \quad \textbf{ENTÃO} \quad \underbrace{y_i = f_i(x, \theta_i)}_{\text{Consequente}} \tag{3.10}$$

onde $\mathcal{R}_i$ é a i-ésima regra *fuzzy*, $i = 1, 2, ..., R$, R é o número de regras *fuzzy*, $x = [x_1, \ldots, x_m]^T \in \mathbb{R}^m$ é o vetor de entrada, m é o número de atributos em x, $\mathcal{A}_i$ é o conjunto *fuzzy* da i-ésima regra *fuzzy* e y_i é a saída da i-ésima regra calculada como uma função da do vetor de entrada e dos parâmetros consequentes.

A saída global de um sistema fuzzy (3.11) é obtida através de um processo de agregação das saídas individuais das regras, ponderadas por seus graus de ativação w_i, normalmente obtidos por meio das funções de pertinência dos antecedentes:

$$\hat{y} = \frac{\sum_{i=1}^{R} w_i y_i}{\sum_{i=1}^{R} w_i} \tag{3.11}$$

Essa abordagem garante que a saída final seja uma combinação linear das diferentes saídas das regras fuzzy. A popularidade dos modelos Takagi-Sugeno decorre da sua capacidade de capturar dinâmicas não lineares de forma computacionalmente eficiente, tornando-os adequados para aplicações em controle automático, sistemas de previsão, análise de dados e aprendizagem de máquina. Pesquisas recentes têm explorado extensões deste modelo, como os sistemas fuzzy evolutivos, que incorporam técnicas de aprendizagem

para adaptação dinâmica das regras e parâmetros, ampliando ainda mais seu potencial de aplicação.

3.2.1 *evolving Takagi-Sugeno* (*eTS*)

O eTS (34, 35) é um modelo de sistemas *fuzzy* evolutivos que se adapta dinamicamente a novos dados, atualizando suas regras *fuzzy* de maneira contínua e incremental. Este modelo é baseado no conceito Takagi-Sugeno, onde as regras *fuzzy* são do tipo "se-então"(Equação 3.10), e o consequente das regras é uma função linear dos valores de entrada. A capacidade de evolução do *eTS* permite que ele se ajuste a mudanças no padrão de dados ao longo do tempo, tornando-o adequado para aplicações em ambientes dinâmicos. Porém, modelos do tipo *eTS* geralmente possuem diversos hiperparâmetros que regulam diferentes aspectos do algoritmo, como a criação e remoção de regras, bem como a taxa de atualização. Como consequência, a busca por uma combinação adequada desses hiperparâmetros, que equilibre a precisão do modelo e a complexidade representada pelo número de regras, pode ser um desafio considerável, podendo até mesmo se tornar inviável em determinadas situações (36).

Vantagens:

- O *eTS* é capaz de se adaptar continuamente a novas condições, mantendo a precisão do modelo em cenários onde os dados mudam ao longo do tempo.
- A estrutura *fuzzy* oferece interpretabilidade, permitindo uma compreensão das regras de decisão geradas.

Desvantagens:

- Em ambientes altamente dinâmicos, o modelo pode gerar um número excessivo de regras, o que pode aumentar a complexidade do modelo e o custo computacional.
- A necessidade de ajuste contínuo da base de regras pode também aumentar o custo computacional.

3.2.2 *Simpl_eTS*

O *Simpl_eTS* é uma variação simplificada do modelo *eTS*, projetada para reduzir a complexidade e melhorar a eficiência computacional. Enquanto mantém a essência da

adaptação contínua e incremental do *eTS*, o *Simpl_eTS* introduz simplificações na estrutura das regras *fuzzy*, visando reduzir o número de parâmetros e operações necessárias para a evolução do modelo. São exemplos da simplificação o uso da função de pertinência do tipo Cauchy em vez de Gaussiana, a dispersão como medida de densidade e capacidade de resumo em vez de potencial, além do uso da idade da regra como representação da estacionariedade das regras (37). Essas simplificações tornaram o modelo mais ágil em termos de processamento, mantendo um bom desempenho preditivo.

Vantagens:

- A simplificação da estrutura *fuzzy* reduz o tempo de processamento, tornando o *Simple_TS* mais eficiente
- Mantém a capacidade de adaptação a novos dados, permitindo uma evolução contínua com menor custo computacional quando comparado ao *eTS*.

Desvantagens:

- As simplificações podem levar a uma redução na capacidade do modelo de capturar padrões complexos nos dados.
- Em alguns casos, a redução da complexidade pode comprometer a precisão do modelo, especialmente em casos não-lineares.

3.2.3 *extended Takagi-Sugeno* (*exTS*)

O *exTS* é uma extensão do modelo *eTS*, incorporando técnicas adicionais para melhorar a robustez e a precisão do modelo em cenários com grande variabilidade de dados (38). Em relação ao *eTS*, há um novo método recursivo para adaptação do raio (dispersão das funções de pertinência), que resulta na construção de clusters mais flexíveis que representam melhor a distribuição dos dados. Além disso, também é introduzida uma nova condição para substituição de clusters, a fim de evitar regras contraditórias na base de regras (39). O *exTS* aprimora a capacidade de evolução do modelo base, permitindo a incorporação de diferentes tipos de informações contextuais e a adaptação não apenas das regras *fuzzy*, mas também das funções consequentes. Isso torna o *exTS* mais flexível e capaz de capturar relações mais complexas nos dados, sendo amplamente utilizado para a

tarefa de previsão de séries temporais (40, 41, 42).

Vantagens:

- A capacidade de incorporar informações contextuais e ajustar tanto as regras *fuzzy* quanto as funções consequentes aumenta a flexibilidade e a precisão do modelo.
- O *exTS* é mais robusto em relação a variações nos dados.

Desvantagens:

- A maior flexibilidade e a incorporação de técnicas adicionais aumentam a complexidade computacional, exigindo mais recursos para treinamento e atualização do modelo.
- A complexidade adicional pode tornar o modelo mais difícil de interpretar em comparação com suas variantes mais simples.

3.2.4 *ePL-KRLS-DISCO*

O *ePL-KRLS-DISCO* (43) é um modelo de sistema *fuzzy* evolutivo baseado em regras, projetado para a previsão de séries temporais (44, 45, 46, 47). Ele se origina do *evolving Takagi-Sugeno* (eTS) e incorpora melhorias significativas para superar as limitações dos modelos anteriores, como o *ePL-KRLS* ((48)).

As principais inovações do modelo incluem o cálculo da saída global com base na regra mais compatível, diferentemente do seu predecessor, o *ePL-KRLS*, que considerava as saídas locais de todas as regras. Essa abordagem previne situações em que saídas de regras divergentes poderiam comprometer a precisão do modelo. Além disso, a introdução da correlação de distância (*DISCO*) possibilita a formação de *clusters* mais bem definidos, resultando em regras com menor desvio padrão e, consequentemente, em uma maior qualidade.

Como visto anteriormente, as regras baseadas em Takagi-Sugeno (3.10) possuem uma parte antecedente e uma parte consequente. A parte antecedente é constituída por regras, que são construídas a partir da entrada do modelo. Os autores propõem a construção dessas regras *fuzzy* utilizando técnicas de *partipatory learning*, além de verificar a utilidade de regras não utilizadas. Por fim, os parâmetros da parte consequente das regras é definida

através do método *kernel Recursive Least Square* (KRLS) de forma recursiva a cada nova entrada atribuída à uma regra.

O modelo *ePL-KRLS-DISCO* proposto por Kaike Sa Teles Rocha Alves e Eduardo Pestana de Aguiar é descrito em detalhes a seguir, desde a construção das regras *fuzzy* até sua saída.

Primeiramente, é importante pontuar que cada regra possui um centroide, que funciona como uma estimativa de seu média. Dado um vetor de entrada $x^k = [x_1, ..., x_m]^T$ com m elementos, o algoritmo irá atribuir essa entrada a uma regra ou, caso não seja semelhante o suficiente às regras já existentes, irá criar uma nova. A semelhança entre um vetor de entrada e as regras existentes se dá através de duas medidas: *compatibility* e *arousal*. A definição de centroides de novas regras se dá por $v_i^k = x^k$, onde v_i^k é o centro da i-ésima regra na k-ésima iteração. A atribuição de uma nova entrada à uma regra existente atualiza a posição do centroide da regra, segundo a Equação (3.12):

$$v_i^k = v_i^{k-1} + \alpha(c_i^k)^{(1-a_i^k)}(x^k - v_i^{k-1}) \tag{3.12}$$

onde $\alpha \in [0, 1]$ é a taxa de aprendizado.

As medidas de *compatibility* e *arousal*, utilizadas para quantificar a semelhança entre o vetor de entrada e as regras já existentes e indicar a necessidade da criação de uma nova regra de forma a reduzir o impacto de *outliers*, respectivamente. Essas são definidas pelas Equações (3.13) e (3.14):

$$c_i^k = \left(1 - \frac{||x^k - v_i^k||}{m}\right)\left(\frac{\rho_{x^k, v_i^k} + 1}{2}\right) \tag{3.13}$$

onde $\rho_{x^k, v_i^k} \in$ [-1,1] é a correlação entre x^k e v^k_1.

$$a_i^k = a_i^{k-1} + \beta(1 - c_i^k - a_i^{k-1}) \tag{3.14}$$

onde $\beta \in [0, 1]$ controla a taxa de crescimento de a_i^k.

O algoritmo gera uma nova regra se o menor índice de *arousal* for maior que um limiar τ. Ou seja, $a_i^k > \tau$, onde $i = argmin_i\{a_i^k\}$ e $\tau = \beta$ (hiperparâmetro do algoritmo) e se nenhuma regra já foi excluída. Caso essas condições não sejam atendidas, o vetor da iteração será atribuído à regra com maior medida de *compatibility*.

Uma das inovações propostas pelos autores é a implementação do $DISCO$ ao modelo *ePL-KRLS*. Enquanto o modelo base utiliza a distância euclidiana para formar as regras, o *ePL-KRLS-DISCO* utiliza da correlação das distâncias, o que melhora a eficácia dos *clusters* criados para as regras.

O valor da correlação é dado pela Equação 3.15 para vetores A e B genéricos, com tamanho m. O valor de $\rho(A, B)$ pertence ao intervalo [-1,1] em que uma correlação de -1 indica que os vetores são fortemente associados de forma inversa, 0 indica que não há evidência de associação e 1 indica que os vetores são fortemente proporcionais.

$$\rho(A, B) = \frac{\text{Cov}(A, B)}{\sqrt{\text{Var}(A)} \cdot \sqrt{\text{Var}(B)}} \tag{3.15}$$

onde $\text{Cov}(A, B)$ é a covariância entre os vetores A e B, calculados através da Equação 3.16, e $Var(A)$ e $Var(B)$ são as variâncias de A e B, calculadas através da Equação 3.17.

$$\text{Cov}(A, B) = \frac{\sum_{l=1}^{m}(A_l - \bar{A})(B_l - \bar{B})}{m - 1} \tag{3.16}$$

$$\text{Var}(X) = \frac{\sum_{l=1}^{m}(X_l - \bar{X})^2}{m - 1} \tag{3.17}$$

Considerando a medida de *compatibility* apresentada pela Equação (3.13), é possível observar que c_i^k será grande para valores pequenos da distância euclidiana ($||x^k - v_i^k||$) e valores grandes de correlação entre o vetor de entrada de índice k (x^k) e o centroide da regra na k-ésima iteração (i v_i^k). Temos então que utilizar a correlação cria regras com menores desvio-padrão, uma vez que valores de correlação maiores vêm de maior covariância (Equação 3.16) e menores variâncias (Equação 3.17) dos vetores. Segundo os autores, como a acurácia da saída do modelo está diretamente ligada à média e desvio-padrão dos *clusters* formados, o método *DISCO* é responsável por aprimorar a performance do *ePL-KRLS*.

Com o objetivo de prover mais eficiência computacional ao algoritmo e evitar problema de *overfitting*, regras são constantemente avaliadas por uma medida de utilidade. Quando a medida é avaliada como menor que um limiar ϵ (hiperparâmetro do modelo), a regra é eliminada. A utilidade é dada pela Equação (3.18).

$$U_i^k = \frac{\sum_{l=1}^{k} \lambda_i^l}{k - l_i} \tag{3.18}$$

onde λ_i^l é o grau de ativação normalizado da i-ésima regra da l-ésima iteração (calculado conforme a Equação 3.19), k é o índice da iteração atual e I_i é a iteração de quando a i-ésima regra foi criada.

$$\lambda_i^k = \frac{\tau_i}{\sum_{regra=1}^{R^k} \tau_{regra}} \tag{3.19}$$

onde τ_i é o grau de ativação da i-ésima regra (Calculado conforme a Equação 3.20 e R^k é o número de regras presentes na iteração atual.

$$\tau_i = \mu_{i1} \times \mu_{i2} \times \cdots \times \mu_{ij} \tag{3.20}$$

onde μ_{ij} é calculado conforme a Equação 3.21.

$$\mu_{ij} = \frac{e^{-||x_j^k - v_{ij}^k||^2}}{2\sigma^2} \tag{3.21}$$

onde v_{ij}^k é o j-ésimo elemento do i-ésimo centroide de regra, $j = 1, 2, ..., m$ e σ define a dispersão da parte antecedente.

Após concluída a parte relativa ao antecedente, o algoritmo estima os parâmetros da parte consequente. A estimativa dos parâmetros é dada através do algoritmo *kernel Recursive Least Square (KRLS)*. Um dicionário local é formado pelas entradas passadas, denotado por $\mathcal{D}_i^k = [d_1, \ldots, d_{n_i}]$, onde $\mathcal{D}_i^k$ representa o dicionário da i-ésima regra na k-ésima iteração, n_i é o número de vetores de entrada armazenados, e $d_{ij} = [d_1, \ldots, d_m]^T$ é o j-ésimo vetor de entrada.

A inclusão irrestrita de todas as entradas no dicionário acarretaria em um aumento substancial no custo computacional. Consequentemente, o modelo implementa um procedimento de esparsificação, retendo apenas os vetores de entrada mais relevantes nos dicionários locais. A relevância de cada entrada é definida com base na sua distância em relação ao elemento mais próximo no dicionário local, conforme a Equação 3.22:

$$dis_x = \min_{d_{ij} \in \mathcal{D}_i^k} \|x^k - d_{ij}\| \tag{3.22}$$

Se a distância exceder um limiar predefinido, o vetor de entrada é incorporado ao dicionário local. Formalmente, se $dis_x \geq 0.1v$, onde $j = \arg\min_j \|x^k - d_{ij}^k\|$, então $\mathcal{D}_i^k = [\mathcal{D}_i^{k-1} \cup x^k]$, v_i^k é o tamanho do *kernel* do dicionário, e n_i é o número de vetores de entrada no dicionário local. O valor de v_i^k é inicializado com σ e atualizado ao longo das iterações. Seu valor influencia diretamente o número de entradas armazenadas no dicionário, sendo que valores menores resultam em um maior número de entradas.

Após a adição de um vetor ao dicionário, os parâmetros da parte consequente são atualizados de acordo com a Equação 3.27.

$$\Theta_i^k = \begin{bmatrix} \Theta_i^{k-1} - Z_i^k [r_i^k]^{-1} \hat{e}^k \\ [r_i^k]^{-1} \hat{e}^k \end{bmatrix} \tag{3.23}$$

onde $\Theta_i^{k-1} = [\Theta_{i1}^{k-1}, \ldots, \Theta_{in_1}^{k-1}]^T$, $z^k = Q_i^{k-1} g^k$, Q_i^k é atualizado de acordo com a Equação 3.28, $g^k = [\kappa\langle d_{i1}^k, x^k\rangle, \ldots, \kappa\langle d_{in_i}^k, x^k\rangle]^T$, $\kappa(\cdot,\cdot)$ é a função de *kernel* Gaussiana calculada conforme a Equação 3.25, $r_i^k = \lambda + \kappa\langle x^k, x^k\rangle - (z^k)^T g^k$, e $\hat{e}^k = y^k - g^k \Theta_i^{k-1}$ é o estimador de erro.

$$Q_i^k = (r^k)^{-1} \begin{bmatrix} Q_i^{k-1} r^k + Z^k (Z^k)^T & -Z^k \\ -(Z^k)^T & 1 \end{bmatrix} \tag{3.24}$$

$$\kappa(x^i, x^j) = \exp\left(-\frac{\|x^i - x^j\|^2}{2\sigma^2}\right) \tag{3.25}$$

onde σ é a largura do *kernel* e controla a linearidade do modelo. Quanto maior o σ, mais linear será a função. Segundo os autores, os valores iniciais sugeridos para este hiperparâmetro podem ser $\sigma \in [0{,}2,\ 0{,}5]$.

A matriz P é atualizada da seguinte forma:

$$P_i^k = \begin{bmatrix} P_i^{k-1} & 0 \\ 0^T & 1 \end{bmatrix} \tag{3.26}$$

onde P_i^k é inicializada com 1.

Caso o vetor de entrada não seja armazenado no dicionário, os parâmetros consequentes são atualizados conforme a Equação 3.27, e a matriz P conforme a Equação .

$$\Theta_i^k = \Theta_i^{k-1} + Q_i^k q_i^k \hat{e}^k \tag{3.27}$$

onde Q_i^k é obtido da Equação 3.28 e q_i^k da Equação 3.29.

$$Q_i^k = Q_i^{k-1} \tag{3.28}$$

$$q_i^k = \frac{P_i^{k-1} z^k}{1 + (z^k)^T P_i^{k-1}(z^k)} \tag{3.29}$$

$$P_i^k = P_i^{k-1} - \frac{P_i^{k-1} z^k (z^k)^T P_i^{k-1}}{1 + (z^k)^T P_i^{k-1}(z^k)} \tag{3.30}$$

O parâmetro consequente de uma nova regra é inicializado da seguinte forma:

$$\Theta_i^k = [\lambda + \kappa \langle x^k, x^{*k} \rangle]^{-1} y^k \tag{3.31}$$

onde $\lambda \in [0, 1]$ é um parâmetro de regularização e y^k é a saída desejada.

Quando o modelo atualiza uma regra existente, o tamanho do *kernel* é atualizado da seguinte forma:

$$v_i^k = \sqrt{(v_i^{k-1})^2 + \frac{||x^k - v_i^k||^2 - (v_i^{k-1})^2}{N_i^k} + \frac{(N_i^{k-1})||v_i^k - v_i^{k-1}||^2}{N_i^k}} \tag{3.32}$$

onde N_i^k é o número de entradas na i-ésima regra na k-ésima iteração.

Caso contrário, se uma nova regra for criada, o modelo inicializa o tamanho do *kernel* da regra da seguinte forma:

$$v_{R+1}^k = \frac{||x^k - v_i^k||}{\sqrt{-2\ log(\eta_{max})}} \quad (24) \tag{3.33}$$

onde η_{max} é o valor máximo de η^K para $K = 1, 2, ..., k$, k é a iteração atual, e η^k é calculado recursivamente como na Equação 3.34.

$$\eta^k = e^{-0.5}(\frac{2}{1 + e^{-\tilde{e}^k}} - 1) \tag{3.34}$$

onde $\tilde{e}^k = 0.8\tilde{e}^{k-1} + ||y^k - \hat{y}^k||$ e $\tilde{e}^1 = 0$.

Segundo os autores, como podemos observar nas equações acima, o modelo atualiza os o tamanho do *kernel* conforme a métrica de erro. Dessa forma, a eficiência do modelo é melhorada, uma vez que quanto menor o erro, menos dados são armazenados no dicionário.

E finalmente, o ePL-KRLS-DISCO calcula a saída usando a regra mais compatível, ou seja, $\hat{y} = \hat{y}_i | i = \arg\max_i \{c_i^k\}$.

O cálculo da saída do modelo se dá pela Equação 3.35:

$$\hat{y}_i = \sum_{j=1}^{n_i} \Theta_{ij}^k \times \kappa(x^k, d_{ij}^k) \tag{3.35}$$

O pseudo-código do algoritmo é apresentado a seguir (Algoritmo 1), de forma a entender o fluxo das transformações:

Algorithm 1 ePL-KRLS-DISCO

Entrada: $x, y, \alpha, \beta, \lambda, \tau, \omega, \sigma, \epsilon$
Saída: $\hat{y}$

1. **Inicialização:** $v_1^1 = x^1$, ${\mathcal{D}_\infty}^1 = x^1$, $v_1^1 = \sigma$, $P_1^1 = 1$, $\theta_1^1 = \left[\lambda + k(x^1, x^1)\right]^{-1} y^1$, $a_1^1 = 0$, `Regra_Excluida` = `falso`
2. **Para** $k = 2, 3, \ldots, n$ **faça**
   a) **Para** $i = 1, 2, \ldots, R$ **faça**
      i. Calcule a medida de compatibilidade: **Equação 3.13**
      ii. Calcule o *arousal index*:
      $$a_i^k = a_i^{k-1} + \beta\left(1 - c_i^k - a_i^{k-1}\right)$$
   b) **Se** $\left(a_i^k > \tau | i = \arg\min\{a_i^k\} \wedge$ `Regra_Excluida` = `falso`$\right)$ **então**
      i. Crie uma nova regra: $R = R + 1$
      ii. Inicialize $v_R^k, \mathcal{D}_R^k, \theta_i^k$: $v_R^k = [x^k]$, $\mathcal{D}_R^k = x^k$
      iii. $\theta_R^k = \left[\lambda + k\langle x^k, x^k\rangle\right]^{-1} y^k$
      iv. Inicialize v_R^k: Equação (24)
   c) **Senão**
      i. Encontre a regra mais compatível: $i = \arg\max_i\{c_i^k\}$
      ii. Atualize o centroide da regra:
      $$v_i^k = v_i^{k-1} + \alpha(c_i^k)^{(1-a_i^k)}\left(x^k - v_i^{k-1}\right)$$
      iii. Calcule g, z e r:
      $$g^k = \left[\kappa(d_{i1}^k, x^k), \ldots, \kappa(d_{in_i}^k, x^k)\right]^\top, \quad z^k = Q_i^{k-1} g^k, \quad r^k = \lambda + \kappa\langle x^k, x^k\rangle - (z^k)^\top g^k$$
      iv. **Se** $\min_{(\forall d_{ij} \in \mathcal{D}_i^k)} \|x^k - d_{ij}^k\| \geq 0.1 v_{ij}^k$ **então**
         A. Inclua x^k no dicionário: $\mathcal{D}_i^k = \mathcal{D}_i^k \cup x^k$
         B. Calcule $Q_i^k, P_i^k, \theta_i^k, v_i^k$: **Equações 3.28, 3.26, 3.27, 3.32**
      v. **Senão**
         A. Calcule $Q_i^k, P_i^k, q_i^k, \theta_i^k$: **Equações 3.28, 3.30, 3.29, 3.27**
3. **Para** $i = 1, 2, \ldots, R$ **faça**
   a) **Se** $U_i^k < \epsilon$ **então**
      i. Remova regras pouco utilizadas: `Remover (i)`
      ii. `Regra_Excluida` = `verdadeiro`
4. Calcule a saída:
$$\hat{y} = \sum_{j=1}^{n_i} \theta_{ij} \kappa\langle d_{ij}^k, x^k\rangle | i = \arg\max_i\{c_i^k\}$$

Vantagens:

- O uso do *DISCO* permite a criação de regras mais precisas, o que melhora a capacidade do modelo de lidar com dados complexos.
- O *ePL-KRLS-DISCO* possui um alto nível de adaptação, capaz de ter bom desempenho com séries dinâmicas.

Desvantagens:

- A complexidade do modelo pode aumentar bastante o tempo de execução, especialmente quando comparado a outros modelos menos sofisticados.
- A necessidade de ajuste fino dos parâmetros pode tornar a implementação do *ePL-KRLS-DISCO* desafiadora.

3.2.5 *NTSK-wRLS*

O *NTSK-wRLS* (36) é uma variação do modelo TSK que propõe uma nova forma de construção de regras *fuzzy*, em que o número de regras é definido previamente pelo usuário. Este modelo visa melhorar a interpretabilidade e simplificar o ajuste de hiperparâmetros, uma vez que o único hiperparâmetro é o número de regras. A estrutura de regras é baseada na variação do valor alvo, utilizando a tangente desses valores para agrupar amostras com características lineares dentro de cada regra.

Para calcular os parâmetros consequentes, o *NTSK-wRLS* utiliza o algoritmo *weighted Recursive Least Squares* (*wRLS*), que ajusta os graus de ativação w_i que indicam o quanto cada regra é relevante para a entrada atual. Além disso, o método ajusta os parâmetros consequentes de forma recursiva: a cada nova amostra de dados, os parâmetros são ajustados de acordo com o erro entre o valor previsto pelo modelo e o valor real da saída. A atualização é ponderada pelo peso w_i, o que significa que regras com maior relevância (maior w_i) têm um impacto maior na atualização.

O objetivo do *wRLS* é minimizar o erro quadrático entre as saídas reais e as previstas, ajustando os parâmetros consequentes para melhorar a precisão do modelo ao longo do tempo, permitindo que ele se adapte dinamicamente.

No pseudo-código abaixo, fornecido por Kaike Sa Teles Rocha Alves, Rosangela Ballini e Eduardo Pestana de Aguiar (36), é possível entender melhor como é a estrutura do novo algoritmo. As equações são mostradas logo em seguida, com explicações.

Algorithm 2 NTSK-wRLS

Entrada: $x, y, R_{\max}$
Saída: $\hat{y}$
for $k = 1, 2, \ldots, n-1$ **do**
    Calcular a variação entre dois valores consecutivos da saída desejada: Equação (3.36)
end for
Calcular o tamanho do intervalo: Equação (3.37)
for $k = 1, 2, \ldots, n-1$ **do**
    Calcular i_{S^k}: Equação (3.39)
end for
Agrupar as amostras com base em i_{S^k}
for $i = 1, 2, \ldots, R_{\max}$ **do**
    Calcular os parâmetros dos conjuntos fuzzy Gaussianos para cada atributo: Equações (3.40) e (3.41)
end for
for $k = 1, 2, \ldots, n$ **do**
    Calcular os parâmetros do consequente: Equação (3.42) para wRLS
    Calcular o grau de ativação: Equação (3.43)
    Calcular a saída do modelo: Equação (3.44)
end for

Como visto anteriormente, $R_{\max}$ é fornecido pelo usuário como uma das entradas do algoritmo e é definido como o número máximo de regras que devem ser criadas. Em seguida, o modelo calcula a variação entre cada dois valores consecutivos, o que representa a tangente dos valores do vetor de saída (y) como pode ser visto na Equação (3.36). Esta é uma das novidades propostas pelos autores, pois enquanto modelos clássicos do tipo fuzzy criam as regras com base no *input* dos atributos, o método *NTSK* cria as regras com base nos *outputs*. O objetivo é agrupar elementos com tangentes próximas para a criação das regras.

$$\tan(y^k) = \frac{\Delta y^k}{k - (k-1)} = \Delta y^k = y^k - y^{k-1} \tag{3.36}$$

Para a criação desses grupos, é necessário definir os intervalos, inferior e superior, para as tangentes do *output*. O tamanho dos intervalos é dado pela Equação (3.37) e o intervalo para cada regra até o limite $R_{\max}$ é dado pela Equação (3.38).

$$IS = \frac{\bar{m}_\Delta - \underline{m}_\Delta}{R_{max}} \tag{3.37}$$

onde $\bar{m}_\Delta = \max(\Delta y^k)$ e $\underline{m}_\Delta = \min(\Delta y^k)$.

$$Range_i = [m_\Delta + (i-1) \times IS, \quad m_\Delta + i \times IS] \tag{3.38}$$

onde $Range_i$ representa a i-ésima regra, para $i = \{1, 2, ..., R_{\max}\}$.

Após calculados os intervalos dos valores de cada uma das regras, os elementos y são devidamente atribuídos às regras criadas, de acordo com a Equação (3.39).

$$i_{S^k} = \begin{cases} \left\lfloor \frac{\Delta y^k - \underline{m}_\Delta}{IS} \right\rfloor, & \text{caso } \Delta y^k < \bar{m}_\Delta \\ R_{max}, & \text{c.c.} \end{cases} \tag{3.39}$$

onde i_{S^k} é a regra em que o k-ésimo elemento será incluído. Por exemplo, um $i_{S^6} = 3$ significa que o sexto elemento pertence à terceira regra.

A última etapa do algoritmo $NTSK$ compreende a definição dos conjuntos *fuzzy*. Mais uma das novidades implementadas no artigo, o algoritmo proposto cria as regras antes mesmo de os conjuntos *fuzzy* estarem propriamente definidos. Os autores propõem a utilização de funções de pertinência do tipo gaussiana. Logo, os únicos parâmetros a serem definidos são a média e o desvio-padrão, segundo as Equações (3.40) e (3.41), respectivamente:

$$v_{i,j} = \bar{x}^l_{i,j} \tag{3.40}$$

$$\sigma_{i,j} = \text{std}(x^l_{i,j}) \tag{3.41}$$

onde $v_{i,j}$ é a média e $\sigma_{i,j}$ é o desvio-padrão das amostras da i-ésima regra, para o j-ésimo atributo, para $l = 1, 2, ..., z$ e z é o número de elementos da i-ésima regra.

Por fim, após aplicado o $NTSK$, o algoritmo para o *Weighted Recursive Least Squares (wRLS)* é aplicado para calcular a parte consequente de cada regra.

O algoritmo para o *wRLS* faz uso de pesos para atualizar os parâmetros da parte consequente da regra, proporcionalmente à compatibilidade de cada regra com vetor de

entrada. Os pesos são dados pelo grau de ativação (w). O algoritmo do $wRLS$ é dado pela Equação (3.42):

$$\begin{cases} P_i^k = P_i^{k-1} - \frac{w_i P_i^{k-1} xe^k (xe^k)^T P_i^{k-1}}{1+w_i (xe^k)^T P_i^{k-1} xe^k} \\ \theta_i^k = \theta_i^{k-1} + P_i^k xe^k w_i (y^k - (xe^k)^T \theta_i^{k-1}) \end{cases} \tag{3.42}$$

onde w_i é o grau de ativação normalizado da i-ésima regra. P é a matriz de covariância, K é uma variável auxiliar, $xe^k = [1, (x^k)^T]^T, \theta$ é o conjunto de parâmetros estimados da parte consequente e y é a saída real.

Finalmente, após definidas a estrutura de base de regras e os parâmetros consequentes, o modelo é capaz de realizar previsões a partir de dados não vistos. O processo de inferência se dá em duas partes: cálculo dos graus de ativação e a saída calculada do modelo, como vistos nas Equações (3.43) e (3.44), respectivamente.

$$w_i = \frac{\prod_{j=1}^{p} \exp\left[-\frac{1}{2}\frac{(x_j^k - v_{i,j})^2}{\sigma_{i,j}^2}\right]}{\sum_{j=1}^{R}\left(\prod_{j=1}^{p} \exp\left[-\frac{1}{2}\frac{(x_j^k - v_{i,j})^2}{\sigma_{i,j}^2}\right]\right)} \tag{3.43}$$

onde $v_{i,j}$ é a média da i-ésima regra e do j-ésimo atributo, $v_i = [v_{i,1}, ..., v_{i,p}]^T$, p é a dimensão das entradas, x_j^k é o k-ésimo vetor de entrada para o j-ésimo atributo, e $\sigma_{i,j}$ é o desvio-padrão da i-ésima regra para o j-ésimo atributo, para $\sigma_i = [\sigma_{i,1}, ..., \sigma_{i,p}]^T$.

$$\hat{y} = \sum_{i=1}^{R_{max}} w_i (xe^k)^T \theta_i \tag{3.44}$$

Vantagens:

- O *NTSK-wRLS* permite ao usuário definir diretamente o número de regras, favorecendo a interpretabilidade.
- O modelo possui um número reduzido de hiperparâmetros, facilitando o ajuste e a implementação.

Desvantagens:

- A eficiência do modelo depende da correta definição do número de regras, que pode não ser trivial em situações com grande variabilidade nos dados.

4 Resultados experimentais

Esta seção descreve o método e os resultados obtidos nos experimentos com a base remodelada para o problema de previsão de séries temporais. Foram considerados todos os modelos mencionados na seção anterior. É importante relembrar que o código utilizado para implementar as técnicas abaixo foi disponibilizado no repositório do projeto (14).

A etapa de modelagem inclui a seleção e ajuste de hiperparâmetros por meio da técnica de *grid search*. Esse método consiste na avaliação sistemática de múltiplas combinações possíveis de valores para cada hiperparâmetro relevante dos modelos, permitindo a identificação das configurações que resultam no melhor desempenho preditivo. A busca por hiperparâmetros será conduzida dentro de um espaço de valores previamente definido com base em experimentação preliminar, de modo a equilibrar a complexidade computacional e a eficácia do ajuste do modelo aos dados.

Para realizar o *grid search*, além de definir o espaço de características (Tabela 3), é necessário realizar a divisão da base de dados. Os dados foram divididos inicialmente em três partes sequenciais (não aleatórias): treino, validação e teste, com a proporção de 50%, 30% e 20% da base. Esta divisão tem o objetivo de treinar cada modelo para cada conjunto de características e avaliar os resultados com o conjunto de validação. Após avaliar exaustivamente todos os conjuntos de características, o conjunto de parâmetros que retornar as melhores métricas de erro será utilizado com os dados de teste e seu resultado final será computado.

Além disso, para estender a avaliação dos resultados, será empregada a técnica de validação cruzada com os melhores parâmetros obtidos na etapa de *grid search*. Esse procedimento permite que cada modelo seja testado sob diferentes partições da base de dados, reduzindo a dependência dos resultados em uma única amostra e fornecendo uma estimativa de sua capacidade de generalização, sob a forma de média $\pm$ desvio-padrão das métricas. Uma segunda divisão dos dados será realizada, com a utilização da técnica de *TimeSeriesSplit*, ainda com a preocupação de preservar a estrutura temporal da série (6). É importante notar que, dado que teremos um conjunto ótimo de parâmetros para o conjunto de validação da etapa anterior, os resultados para o *split* que contiver os mesmos índices do conjunto de validação estejam inflados por uma questão de vazamento de dados.

Os resultados finais serão expressos por meio das métricas *Root Mean Squared Error*

(*RMSE*) (4.2) e *Mean Absolute Error* (*MAE*) (4.1), escolhidas por possuírem a mesma unidade de medida da variável dependente, o que facilita a interpretação dos erros preditivos no contexto do problema. O *RMSE* é uma métrica sensível a erros de maior magnitude, pois penaliza desvios maiores de forma mais expressiva devido à presença do termo quadrático. Já o *MAE* representa a média dos erros absolutos cometidos pelo modelo, proporcionando uma interpretação mais intuitiva e direta sobre a precisão das previsões. Essas métricas são amplamente empregadas na avaliação de modelos de previsão de séries temporais, uma vez que quantificam, de forma objetiva, a magnitude dos erros cometidos durante a fase de inferência.

$$MAE = \frac{1}{T}\sum_{k=1}^{T}|y^k - \hat{y}^k|, \tag{4.1}$$

$$RMSE = \sqrt{\frac{1}{T}\sum_{k=1}^{T}(y^k - \hat{y}^k)^2}, \tag{4.2}$$

onde $\hat{y}^k$ é $k-$ésimo valor previsto, y^k é o $k-$ésimo valor real e T é o tamanho da amostra.

4.1 *Implementação do Grid-search*

Para a implementação do *grid search*, foi estabelecido um conjunto discreto e finito de valores para cada um dos possíveis hiperparâmetros de cada modelo, apresentado na Tabela 3. Estes conjuntos de valores foram definidos de forma a equilibrar a abrangência da busca e a viabilidade computacional do processo, uma vez que a busca exaustiva pode se tornar inviável quando há um grande número de combinações possíveis.

Além disso, conforme exposto no início do capítulo, os dados foram divididos em três conjuntos: treino, validação e teste, com as proporções e quantidade de amostras segundo a Tabela 4.

Modelo	**Parâmetros**
KNN	n_neighbors: [2, 3, 4, 5, 10, 20]
Árvores de Regressão	max_depth: [2, 4, 6, 8, 10, 20, 30, 50, 100]
Gradient Boosting	learning_rate: [0.01, 0.05, 0.1, 0.5, 0.9] n_estimators: [2, 4, 8, 16, 32, 64, 100, 200]
LightGBM	learning_rate: [0.01, 0.05, 0.1, 0.5, 0.9]
eTS	InitialOmega: [50, 100, 250, 500, 750, 1000, 10000] r: [0.1, 0.3, 0.5, 0.7, 0.9, 1., 5, 10, 50]
Simpl_eTS	InitialOmega: [50, 250, 500, 750, 1000] r: [0.1, 0.3, 0.5, 0.7]
exTS	InitialOmega: [50, 250, 500, 750, 1000] mu_threshold: [0.1, 0.3, 0.5, 0.7]
ePL-KRLS-DISCO	alpha: [0.05, 0.1] beta: [0.01, 0.1, 0.25] lambda1: [0.0000001, 0.001] sigma: [0.5, 1, 10, 50] e_utility: [0.03, 0.05]
NTSK-wRLS	n_clusters: [1, 2, ..., 19, 20]

Tabela 3 – Modelos e Parâmetros Utilizados para Busca por Grid Search

Conjunto	**Percentual**	**Quantidade de amostras**
Treino	50%	5136
Validação	30%	3081
Teste	20%	2054
Total	**100%**	**10.271**

Tabela 4 – Distribuição dos dados nos conjuntos

A configuração de hiperparâmetros que apresentou o menor RMSE foi selecionada como o conjunto ótimo para cada um dos modelos avaliados, conforme apresentado na Tabela 5.

Modelo	**Parâmetros**
KNN	n_neighbors: 5
Árvores de Regressão	max_depth: 6
Gradient Boosting	learning_rate: 0.1 n_estimators: 64
LightGBM	learning_rate: 0.05
eTS	InitialOmega: 50 r: 5
Simpl_eTS	InitialOmega: 50 r: 0.7
exTS	InitialOmega: 50 mu_threshold: 0.1
ePL-KRLS-DISCO	alpha: 0.05 beta: 0.25 lambda1: 0.001 sigma: 10 e_utility: 0.05
NTSK-wRLS	n_clusters: 10

Tabela 5 – Melhores parâmetros encontrados para cada modelo

Após os modelos treinados com os parâmetros ótimos serem avaliados no conjunto de teste (dados não vistos na fase de treinamento), os resultados das métricas *MAE* e *RMSE* foram compilados na Tabela 6. Como pode ser observado, os valores de *RMSE* são maiores que os da métrica *MAE* devido à penalidade para grandes resíduos das previsões. Os modelos KNN, exTS e ePL-KRLS-DISCO demonstraram um ajuste relativamente pior, quando comparados aos demais modelos que obtiveram resultados bastante semelhantes.

Modelo	MAE	RMSE
KNN	0.86	1.53
Árvores de Regressão	0.25	0.55
Gradient Boosting	0.32	0.52
LightGBM	0.20	0.41
eTS	0.50	0.77
Simpl_eTS	0.36	0.65
exTS	1.69	2.99
ePL-KRLS-DISCO	1.11	1.49
NTSK-wRLS	0.35	0.64

Tabela 6 – Resultados dos modelos avaliados no conjunto de teste

4.2 Validação-cruzada

Após encontrados os primeiros resultados para o conjunto de teste com os parâmetros ótimos de cada um dos modelos, foi realizada a validação-cruzada com o objetivo de obter os valores médios e a dispersão das métricas de erro escolhidas para diversos pontos da série. Novamente, é importante pontuar que os dados utilizados nessa nova divisão não deveriam conter aqueles utilizados como validação durante o *grid-search*, uma vez que os resultados para estes índices dos dados não seriam confiáveis e atrapalhariam a avaliação da generalização do modelo.

4.2.1 Divisão dos dados para validação-cruzada

O método utilizado para a divisão da base de dados em conjuntos sucessivos de treino e teste foi o *TimeSeriesSplit*, disponível na biblioteca scikit-learn (49). Este método é específico para séries temporais, pois preserva a ordem cronológica dos dados, evitando a mistura entre observações do passado e do futuro. A base de dados foi dividida em 5 subconjuntos (*splits*) de forma que, a cada iteração, um novo bloco de dados era incluído no conjunto de treino, e o bloco subsequente era usado como conjunto de teste, como pode ser visualizado na Figura 6. A técnica de validação-cruzada permite que o modelo seja treinado e avaliado em múltiplos períodos de tempo, proporcionando uma avaliação mais confiável a respeito da sua generalização.

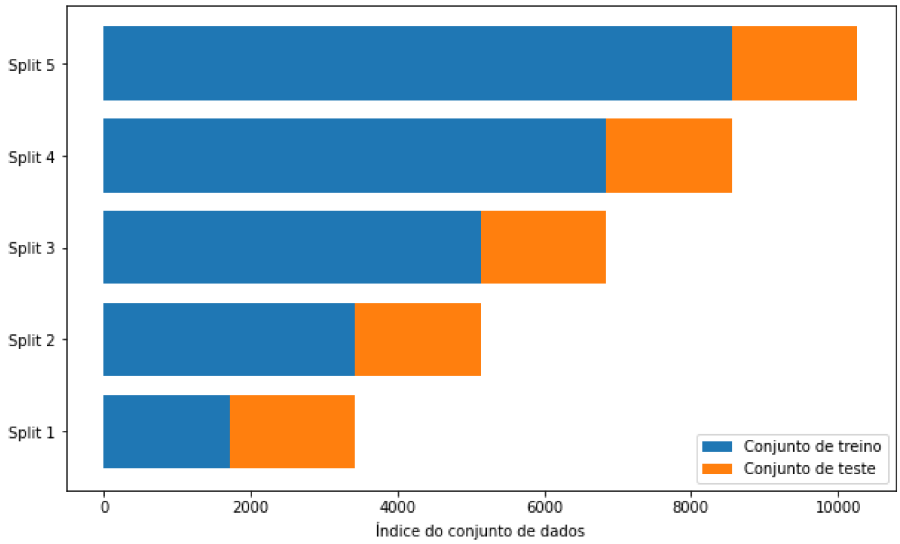

Figura 6 – Representação dos *splits*

4.2.2 Resultados da validação-cruzada

Os resultados da validação-cruzada são demonstrados na Tabela 7. Dentre os modelos clássicos de aprendizado de máquina, o *KNN* foi o que demonstrou pior ajuste à série temporal, com um *MAE* de 1.742 ± 0.646 e um *RMSE* de 2.655 ± 1.213. Os demais modelos da abordagem, Árvores de Regressão, Gradient Boosting e LightGBM, demonstraram resultados similares entre si. Dos modelos de abordagem *fuzzy*, o *ePL-KRLS-DISCO* registrou as maiores métricas de erro, com *MAE* de 1.044 ± 0.101 e *RMSE* de 1.483 ± 0.164. Dentre todos os modelos avaliados, o *NTSK-wRLS* se destaca com *MAE* de 0.355 ± 0.016 e *RMSE* de 0.720 ± 0.128, demonstrando valores médios de erro e também desvios-padrão consideravelmente menores que os demais.

Além disso, os resultados obtidos pelo modelo *NTSK-wRLS* destacam-se por oferecer uma margem de erro relativamente pequena para a previsão, considerando a estatística descritiva do indicador gerencial de TPV presente na Tabela 2.

Modelo	**MAE**	**RMSE**
KNN	1.742 ± 0.646	2.655 ± 1.213
Árvores de Regressão	0.377 ± 0.143	1.048 ± 0.837
Gradient Boosting	0.369 ± 0.147	1.016 ± 0.885
LightGBM	0.374 ± 0.131	1.036 ± 0.761
eTS	0.424 ± 0.058	0.776 ± 0.129
Simpl_eTS	0.378 ± 0.040	0.726 ± 0.154
exTS	0.635 ± 0.206	1.104 ± 0.266
ePL-KRLS-DISCO	1.044 ± 0.101	1.483 ± 0.164
NTSK-wRLS	**0.355 ± 0.016**	**0.720 ± 0.128**

Tabela 7 – Resultados de MAE e RMSE para os modelos avaliados

4.3 Discussões

Os resultados dos experimentos revelaram diferenças significativas no desempenho dos modelos testados para a tarefa de previsão da série temporal do indicador gerencial de TPV. Considerando os resultados de validação-cruzada, o modelo *NTSK-wRLS* destacou-se como o mais eficaz, apresentando os menores valores de erros e desvios. O destaque do modelo pode ser atríbuído à sua capacidade de adaptação às dinâmicas da série, que são influenciadas por fatores reais do processo, como sazonalidade, flutuações de demanda e intercorrências operacionais.

Por outro lado, o modelo *ePL-KRLS-DISCO*, embora tenha demonstrado um bom desempenho no trabalho anterior, foi menos eficaz neste contexto específico.

A Figura 7 demonstra as previsões realizadas pelos modelos em comparação com os valores reais de TPV. Para este caso, os modelos foram treinados no conjunto de treino do *split* 5, evidenciado na Figura 6, e as previsões foram realizadas com o respectivo conjunto de teste. A Figura 8 destaca as previsões do modelo *NTSK-wRLS*. Nela, é possível observar quão próximas são as previsões do modelo em comparação com os valores reais, o que corrobora os resultados demonstrados na Tabela 7.

Os resultados obtidos indicam que o *NTSK-wRLS* é um modelo que se ajusta bem a esta série. Esta acurácia nas previsões pode trazer benefícios significativos para a gestão logística, como uma alocação mais eficiente de recursos e uma redução no tempo de espera dos caminhões, o que, por sua vez, pode resultar em uma maior eficiência global do processo de recebimento de sucata metálica.

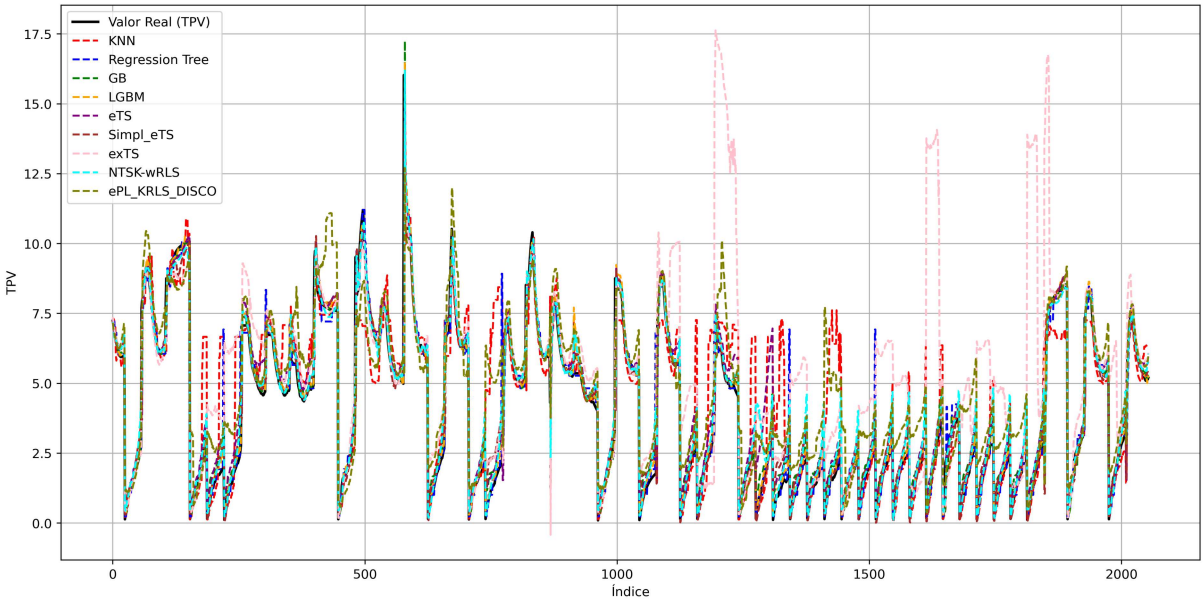

Figura 7 – Comparação das previsões dos modelos avaliados e o valor de TPV real

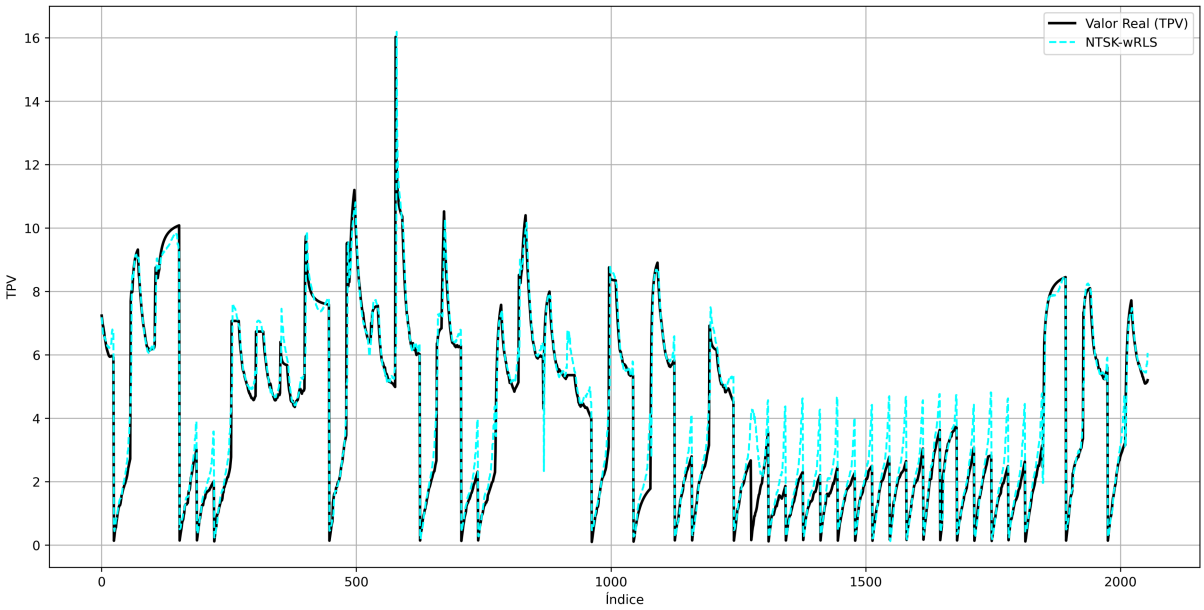

Figura 8 – Comparação das previsões do modelo NTSK-wRLS com o valor de TPV real

5 Conclusão

Este trabalho apresentou uma análise da aplicação de algoritmos de aprendizado de máquina para prever o TPV em uma usina siderúrgica. A pesquisa envolveu a remodelagem da base de dados original para um formato de série temporal e a implementação de diferentes modelos preditivos, tanto de abordagens clássicas quanto baseadas em lógica *fuzzy*.

Os resultados experimentais demonstraram que o modelo *NTSK-wRLS* se destacou significativamente em relação aos demais, apresentando os menores valores de *MAE* e *RMSE*. Essa performance superior do modelo pode ser atribuída à capacidade do *NTSK-wRLS* de adaptar-se às dinâmicas complexas da série temporal do TPV, refletindo de maneira mais fiel as variações operacionais e sazonais inerentes ao processo de recebimento de sucata metálica.

Além de validar a eficácia do modelo proposto por Alves, Ballini e de Aguiar (36), este trabalho contribui para a literatura ao destacar o potencial do aprendizado de máquina na solução de problemas em ambientes industriais reais. Isso é particularmente relevante em contextos onde a utilização da inteligência artificial pode resultar em significativas economias de custo e aumentos de produtividade.

Esta pesquisa abre caminho para futuras investigações que possam explorar a integração de outras variáveis operacionais e o desenvolvimento de sistemas híbridos que combinem múltiplos modelos preditivos, com o intuito de maximizar a precisão e a confiabilidade das previsões em ambientes industriais dinâmicos.

Por fim, é essencial que estudos futuros considerem a aplicação desses modelos em diferentes contextos industriais e com conjuntos de dados mais amplos e diversos, a fim de validar a generalização dos resultados obtidos e ampliar o escopo de aplicação das técnicas de aprendizado de máquina na otimização de processos logísticos.

6 Limitações do estudo

Em primeiro lugar, apesar de propor a comparação entre modelos clássicos de aprendizado de máquina e modelos baseados em lógica fuzzy, poderiam ter sido explorados modelos especializados em séries temporais, como ARIMA e suas derivações, incluindo SARIMA e ARIMAX.

Outra limitação diz respeito à escolha dos hiperparâmetros dos modelos. Embora tenha sido utilizado um processo de otimização baseado em *grid search*, abordagens mais sofisticadas, como *Bayesian Optimization* (50, 51) ou Algoritmos Genéticos (52, 53), poderiam ser exploradas para garantir uma busca mais eficiente por parâmetros.

Por fim, uma questão que diz respeito à forma como os hiperparâmetros dos modelos foram ajustados. Neste estudo, o procedimento de *grid search* foi realizado utilizando a base de validação para encontrar a melhor configuração de hiperparâmetros para cada modelo. No entanto, posteriormente, os modelos foram treinados e avaliados novamente com estes dados, ao passarem pela validação-cruzada utilizando *TimeSeriesSplit*. O problema que isso pode causar é o vazamento de dados (*data leakage)*, que pode resultar em uma avaliação excessivamente otimista do desempenho dos modelos, quando avaliados naqueles índices do conjunto de validação. Uma possível solução para isso seria implementar o processo de *nested cross-validation* (54).

Essas limitações abrem espaço para futuros trabalhos, permitindo o aprimoramento dos modelos utilizados e também a exploração dos dados do projeto - que se encontram no repositório GitHub (14).

REFERÊNCIAS

1 A Gunasekaran and E.W.T Ngai. Information systems in supply chain integration and management. *European Journal of Operational Research*, 159(2):269–295, 2004. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2003.08.016. URL https://www.sciencedirect.com/science/article/pii/S0377221703005186. Supply Chain Management: Theory and Applications.

2 Benjamín Riquelme Oyarzún. La logística 4.0. *Revista de Marina,(964)*, pages 39–44, 2018.

3 Robert N. Boute and Maxi Udenio. *AI in Logistics and Supply Chain Management*, pages 49–65. Springer International Publishing, Cham, 2023. ISBN 978-3-030-95764-3. doi: 10.1007/978-3-030-95764-3_3. URL https://doi.org/10.1007/978-3-030-95764-3_3.

4 Jorge Sala Minoves, Eloísa Helena Rodrigues Guimarães, Tarcísio Afonso, and Ester Eliane Jeunon. Logística direta e logística reversa na produção do aço: estudo de caso em uma empresa siderúrgica. *Revista Inovação, Projetos e Tecnologias*, 3(1): 137–151, 2015.

5 Reinol Josef Compañero, Andreas Feldmann, and Anders Tilliander. Circular steel: How information and actor incentives impact the recyclability of scrap. *Journal of Sustainable Metallurgy*, 7(4):1654–1670, 2021. ISSN 2199-3831. doi: 10.1007/s40831-021-00436-1. URL https://doi.org/10.1007/s40831-021-00436-1.

6 Ricardo Juliano Gouveia. Modelo de simulação da logística das portarias de entrada e saída de veículos de abastecimento e escoamento de produtos da usiminas em cubatão/sp. 2017.

7 Gustavo Nucci Franco, Marcellus Vinagre da Silva, Daniel Camargos Frade, Arthur Japiassu Marques Junior, and Pedro Cesar Sinhoroto Machado. Estudo para redução de permanência de veículos apoiado pela simulação em computador. In *35th Logistic Seminar*, pages 35–42, 2016.

8 Ana Ely de Souza FERREIRA, Vitória Caroline Santos PERES, Tailton de Oliveira MACIEL, and Silvio Akira HIRASSAKA. A aplicação de ferramentas da qualidade no auxílio da redução do tempo de permanência do veículo em uma indústria metalúrgica. *Observatorio de la Economía Latinoamericana*, (diciembre), 2018.

9 Wei Zhao, Jiali Mao, Shengcheng Cai, Peng Cai, Dai Sun, Cheqing Jin, and Ye Guo. Wtpst: Waiting time prediction for steel logistical queuing trucks. In Yunmook Nah, Bin Cui, Sang-Won Lee, Jeffrey Xu Yu, Yang-Sae Moon, and Steven Euijong Whang, editors, *Database Systems for Advanced Applications*, pages 790–794, Cham, 2020. Springer International Publishing. ISBN 978-3-030-59419-0.

10 Simone Tavares Fernandes and Fernando Augusto Silva Marins. Aplicação do lean six sigma na logística de transporte. *Revista produção online*, 12(2):297–327, 2012.

11 Sushant Sharma, Dong Hun Kang, Jose Rivera Montes de Oca, and Abhisek Mudgal. Machine learning methods for commercial vehicle wait time prediction at a border crossing. *Research in Transportation Economics*, 89:101034, 2021. ISSN 0739-8859. doi: https://doi.org/10.1016/j.retrec.2021.101034. URL https://www.sciencedirect.com/science/article/pii/S0739885921000068. Journal of the Transportation Research Forum, Volume 58.

12 Presidência da República do Brasil. Lei Nº 13.103, de 2 de março de 2015. https://www.planalto.gov.br/ccivil_03/_ato2015-2018/2015/lei/l13103.htm, 2015. [Acesso em: 29-jun-2024].

13 Victor Hugo Soares Pereira, Kaike Sa Teles Rocha Alves, and Eduardo Pestana de Aguiar. A machine learning approach to predict length of stay of vehicles in an inbound logistics operation. In E. Simas, D. D. Ferreira, and L. R. Oliveira, editors, *Anais do XVI Congresso Brasileiro de Inteligência Computacional (CBIC'2023)*, pages 1–6, Salvador, BA, 2023. SBIC. doi: 10.21528/CBIC2023-005.

14 Victor H. S. Pereira. Repositório da dissertação - ppgmc, 2024. URL https://github.com/VictorHSPereira/dissertacao_pgmc. Acessado em: 21 jan. 2025.

15 V. Vacanza. Holidays: Capturing public holidays in brazil, 2024. URL https://github.com/vacanza/holidays/. Accessed: 2024-08-27.

16 Eryk Lewinson. Three approaches to encoding time information as features for machine learning models, 2022. URL https://developer.nvidia.com/blog/three-approaches-to-encoding-time-information-as-features-for-ml-models/. Accessed: 2024-06-28.

17 S Yakowitz. Nearest-neighbour methods for time series analysis. *Journal of Time Series Analysis*, 8(2):235–247, 1987. doi: https://doi.org/10.1111/j.1467-9892.1987.tb00435.x.

18 Abdullah Albadrani, Mohamed A. Zohdy, and Richard Olawoyin. An approach to optimize future inbound logistics processes using machine learning algorithms. In *2020 IEEE International Conference on Electro Information Technology (EIT)*, pages 402–406, 2020. doi: 10.1109/EIT48999.2020.9208238.

19 Francisco Martínez, María Pilar Frías, María Dolores Pérez, and Antonio Jesús Rivera. A methodology for applying k-nearest neighbor to time series forecasting. *Artificial Intelligence Review*, 52(3):2019–2037, October 2019. ISSN 1573-7462. doi: 10.1007/s10462-017-9593-z. URL https://doi.org/10.1007/s10462-017-9593-z.

20 Chao Hu, Gaurav Jain, Puqiang Zhang, Craig Schmidt, Parthasarathy Gomadam, and Tom Gorka. Data-driven method based on particle swarm optimization and k-nearest neighbor regression for estimating capacity of lithium-ion battery. *Applied Energy*, 129: 49–55, 2014. doi: 10.1016/j.apenergy.2014.04.077.

21 Tao Ban, Ruibin Zhang, Shaoning Pang, Abdolhossein Sarrafzadeh, and Daisuke Inoue. Referential knn regression for financial time series forecasting. In *International Conference on Neural Information Processing*, pages 601–608. Springer, 2013. doi: 10.1007/978-3-642-42054-2_75.

22 Juan M. Scavuzzo, Francisco Trucco, Manuel Espinosa, Carolina B. Tauro, Marcelo Abril, Carlos M. Scavuzzo, and Alejandro C. Frery. Modeling dengue vector population using remotely sensed data and machine learning. *Acta Tropica*, 185: 167–175, 9 2018. ISSN 18736254. doi: 10.1016/j.actatropica.2018.05.003.

23 Hamidreza Saghafi and Milad Arabloo. Modeling of co2 solubility in mea, dea, tea, and mdea aqueous solutions using adaboost-decision tree and artificial neural network. *International Journal of Greenhouse Gas Control*, 58:256–265, 3 2017. ISSN 17505836. doi: 10.1016/j.ijggc.2016.12.014.

24 Sasanka Choudhury, Dhirendra Nath Thatoi, Jhalak Hota, and Mohan D. Rao. Predicting crack through a well generalized and optimal tree-based regressor. *International Journal of Structural Integrity*, 11:783–807, 9 2020. ISSN 17579872. doi: 10.1108/IJSI-09-2019-0086.

25 Shan Suthaharan. *Decision Tree Learning*, pages 237–269. Springer US, Boston, MA, 2016. ISBN 978-1-4899-7641-3. doi: 10.1007/978-1-4899-7641-3_10. URL `https://doi.org/10.1007/978-1-4899-7641-3_10`.

26 Alexey Natekin and Alois Knoll. Gradient boosting machines, a tutorial. *Frontiers in Neurorobotics*, 7, 2013. ISSN 1662-5218. doi: 10.3389/fnbot.2013.00021. URL `https://www.frontiersin.org/journals/neurorobotics/articles/10.3389/fnbot.2013.00021`.

27 Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, pages 1189–1232, 2001. URL `https://www.jstor.org/stable/2699986`.

28 Robert E Schapire. A brief introduction to boosting. In *Ijcai*, volume 99, pages 1401–1406. Citeseer, 1999. doi: 10.1.1.640.9942.

29 Xingbin Zhan, Shuaichao Zhang, Wai Yuen Szeto, and Xiqun Chen. Multi-step-ahead traffic speed forecasting using multi-output gradient boosting regression tree. *Journal of Intelligent Transportation Systems: Technology, Planning, and Operations*, 24: 125–141, 3 2020. ISSN 15472442. doi: 10.1080/15472450.2019.1582950.

30 Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: a highly efficient gradient boosting decision tree. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 3149–3157, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

31 Ronghua Wang, Yang Liu, Xi Ye, Quan Tang, Jing Gou, Mingzeng Huang, and Yunfeng Wen. Power system transient stability assessment based on bayesian optimized lightgbm. In *2019 IEEE 3rd Conference on Energy Internet and Energy System Integration (EI2)*, pages 263–268, 2019. doi: 10.1109/EI247390.2019.9062027.

32 L.A. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338–353, 1965. ISSN 0019-9958. doi: https://doi.org/10.1016/S0019-9958(65)90241-X. URL `https://www.sciencedirect.com/science/article/pii/S001999586590241X`.

33 Tomohiro Takagi and Michio Sugeno. Fuzzy identification of systems and its applications to modeling and control. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-15(1):116–132, 1985. doi: 10.1109/TSMC.1985.6313399.

34 P.P. Angelov and D.P. Filev. An approach to online identification of takagi-sugeno fuzzy models. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 34(1):484–498, 2004. doi: 10.1109/TSMCB.2003.817053.

35 Tomohiro Takagi and Michio Sugeno. Fuzzy identification of systems and its applications to modeling and control. *IEEE transactions on systems, man, and cybernetics*, (1):116–132, 1985.

36 Kaike Sa Teles Rocha Alves, Rosangela Ballini, and Eduardo Pestana de Aguiar. Financial series forecasting: A new fuzzy inference system for crisp values and interval-valued predictions. *Computational Economics*, July 2024. ISSN 1572-9974. doi: 10.1007/s10614-024-10670-w. URL `https://doi.org/10.1007/s10614-024-10670-w`.

37 Plamen Angelov and Dimitar Filev. Simpl_ets: A simplified method for learning evolving takagi-sugeno fuzzy models. In *The 14th IEEE International Conference on Fuzzy Systems, 2005. FUZZ'05.*, pages 1068–1073. IEEE, 2005.

38 O. Massol, X. Li, R. Gouriveau, J. H. Zhou, and O. P. Gan. An exts based neuro-fuzzy algorithm for prognostics and tool condition monitoring. In *2010 11th International Conference on Control Automation Robotics Vision*, pages 1329–1334, 2010. doi: 10.1109/ICARCV.2010.5707842.

39 Plamen Angelov and Xiaowei Zhou. Evolving fuzzy systems from data streams in real-time. In *2006 International Symposium on Evolving Fuzzy Systems*, pages 29–35, 2006. doi: 10.1109/ISEFS.2006.251157.

40 Marie-Danièle Gauvain, Rafael Gouriveau, Noureddine Zerhouni, and Mike Hessabi. Long term prediction approaches based on connexionist systems - a study for prognostics application. In *2011 IEEE Conference on Prognostics and Health Management*, pages 1–8, 2011. doi: 10.1109/ICPHM.2011.6024342.

41 Muhammad Asif Memon, Plamen Angelov, and Hasan Ahmed. An approach to real-time color-based object tracking. In *2006 International Symposium on Evolving Fuzzy Systems*, pages 86–91, 2006. doi: 10.1109/ISEFS.2006.251169.

42 Jose J. Macias, Plamen Angelov, and Xiaowei Zhou. A method for predicting quality of the crude oil distillation. In *2006 International Symposium on Evolving Fuzzy Systems*, pages 214–220, 2006. doi: 10.1109/ISEFS.2006.251167.

43 Kaike Sa Teles Rocha Alves and Eduardo Pestana de Aguiar. A novel rule-based evolving fuzzy system applied to the thermal modeling of power transformers. *Applied Soft Computing*, 112:107764, 2021. ISSN 1568-4946. doi: https://doi.org/10.1016/j.asoc.2021.107764. URL `https://www.sciencedirect.com/science/article/pii/S1568494621006852`.

44 Lei Hu, Xinghan Xu, Weijie Ren, and Min Han. Hierarchical evolving fuzzy system: A method for multidimensional chaotic time series online prediction. *IEEE Transactions on Fuzzy Systems*, 32(6):3329–3341, 2024. doi: 10.1109/TFUZZ.2023.3348847.

45 Eduardo Santos De Oliveira Marques, Kaike Sa Teles Rocha Alves, Direnc Pekaslan, and Eduardo Pestana De Aguiar. Kernel evolving participatory fuzzy modeling for time series forecasting: New perspectives based on distance measures. In *2022 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–8, 2022. doi: 10.1109/FUZZ-IEEE55066.2022.9882602.

46 Kaike Sa Teles Rocha Alves, Direnc Pekaslan, Christian Wagner, and Eduardo Pestana de Aguiar. A new evolving fuzzy system with mechanisms to deal with uncertainties in times series forecasting. In *2022 IEEE Latin American Conference on Computational Intelligence (LA-CCI)*, pages 1–6, 2022. doi: 10.1109/LA-CCI54402.2022.9981311.

47 Eduardo Santos de Oliveira Marques, Kaike Sa Teles Rocha Alves, Direnc Pekaslan, and Eduardo Pestana de Aguiar. Kernel evolving participatory fuzzy modeling for time series forecasting: New perspectives based on similarity measures. In *2022 IEEE International Conference on Evolving and Adaptive Intelligent Systems (EAIS)*, pages 1–8, 2022. doi: 10.1109/EAIS51927.2022.9787687.

48 R. Vieira, L. Maciel, R. Ballini, and Fernando Gomide. Stock market price forecasting using a kernel participatory learning fuzzy model. In Guilherme A. Barreto and Ricardo Coelho, editors, *Fuzzy Information Processing*, pages 361–373, Cham, 2018. Springer International Publishing. ISBN 978-3-319-95312-0.

49 F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

50 Ryan Turner, David Eriksson, Michael McCourt, Juha Kiili, Eero Laaksonen, Zhen Xu, and Isabelle Guyon. Bayesian optimization is superior to random search for machine learning hyperparameter tuning: Analysis of the black-box optimization challenge 2020. In Hugo Jair Escalante and Katja Hofmann, editors, *Proceedings of the NeurIPS 2020 Competition and Demonstration Track*, volume 133 of *Proceedings of Machine Learning Research*, pages 3–26. PMLR, 06–12 Dec 2021. URL `https://proceedings.mlr.press/v133/turner21a.html`.

51 Jia Wu, Xiu-Yun Chen, Hao Zhang, Li-Dong Xiong, Hang Lei, and Si-Hao Deng. Hyperparameter optimization for machine learning models based on bayesian optimizationb. *Journal of Electronic Science and Technology*, 17(1):26–40, 2019. ISSN 1674-862X. doi: https://doi.org/10.11989/JEST.1674-862X.80904120. URL `https://www.sciencedirect.com/science/article/pii/S1674862X19300047`.

52 Ismail Damilola Raji, Habeeb Bello-Salau, Ime Jarlath Umoh, Adeiza James Onumanyi, Mutiu Adesina Adegboye, and Ahmed Tijani Salawudeen. Simple deterministic selection-based genetic algorithm for hyperparameter tuning of machine learning models. *Applied Sciences*, 12(3), 2022. ISSN 2076-3417. doi: 10.3390/app12031186. URL `https://www.mdpi.com/2076-3417/12/3/1186`.

53 D. L. Shanthi and N. Chethan. Genetic algorithm based hyper-parameter tuning to improve the performance of machine learning models. *SN Computer Science*, 4(2):119, 2022. ISSN 2661-8907. doi: 10.1007/s42979-022-01537-8. URL `https://doi.org/10.1007/s42979-022-01537-8`.

54 Scikit-learn Developers. Nested cross-validation example with the iris dataset, 2024. URL https://scikit-learn.org/stable/auto_examples/model_selection/plot_nested_cross_validation_iris.html. Acessado em: 23 jan. 2025.