



INSTITUTO FEDERAL
Goiás

Câmpus
Anápolis

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE GOIÁS
IFG *CAMPUS* ANÁPOLIS
ESPECIALIZAÇÃO EM INTELIGÊNCIA ARTIFICIAL APLICADA

MARCOS RODRIGUES BRUGNARO

VIAGENS URBANAS INTELIGENTES: PREDIÇÃO DE TEMPO DE VIAGENS
URBANAS UTILIZANDO MODELOS DE MACHINE LEARNING E REDES NEURAIS
ARTIFICIAIS

ANÁPOLIS
2025

MARCOS RODRIGUES BRUGNARO

VIAGENS URBANAS INTELIGENTES: PREDIÇÃO DE TEMPO DE VIAGENS URBANAS
UTILIZANDO MODELOS DE MACHINE LEARNING E REDES NEURAIS ARTIFICIAIS

Trabalho de Conclusão de Curso apresentado ao curso de Especialização em Inteligência Artificial Aplicada do Instituto Federal de Educação, Ciência e Tecnologia de Goiás (IFG) – *Campus* Anápolis, como requisito parcial para obtenção do Título de Especialista em Inteligência Artificial Aplicada.

Orientador: Prof. Dr. Daniel Rosa Canedo.

Dados Internacionais de Catalogação na Publicação (CIP)

BRUGNARO, Marcos Rodrigues.	
B891v	Viagens urbanas inteligentes: predição de tempo de viagens urbanas utilizando modelos de <i>Machine Learning</i> e redes neurais artificiais./ Marcos Rodrigues Brugnaro. – Anápolis: IFG, 2025. 78 f. il. color.
	Orientador: Prof. Dr. Daniel Rosa Canedo.
	Trabalho de Conclusão de Curso – Instituto Federal de Educação, Ciência e Tecnologia de Goiás, Campus Anápolis – Especialização em Inteligência Artificial Aplicada, 2025.
	1. Transporte público urbano. 2. Estimativas precisas. 3. Modelos preditivos. 4. Regressão polinomial. 5. Redes neurais artificiais. I. CANEDO, Daniel Rosa (orient.). II. Título.
CDD 006.3	

Ficha catalográfica elaborada pelo bibliotecário Matheus Rocha Piacenti CRB1/02992
IFG Campus Anápolis



MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
DE GOIÁS CÂMPUS ANÁPOLIS

ATA DE DEFESA PÚBLICA DE TRABALHO DE CONCLUSÃO DE CURSO

(Modalidade da Sessão: Webconferência)

No dia 14 (quatorze) de fevereiro do ano de 2025, às 20 horas, no Programa de Pós-Graduação em Inteligência Artificial Aplicada, do Instituto Federal de Educação, Ciência e Tecnologia de Goiás (IFG) - Câmpus Anápolis, por meio de webconferência, deu-se a defesa do Trabalho de Conclusão de Curso (TCC) “**Viagens Urbanas Inteligentes: Predição de tempo de viagens urbanas utilizando modelos de Machine Learning e Redes Neurais Artificiais**”, de autoria de **Marcos Rodrigues Brugnaro**, como requisito para a conclusão do Curso de Especialização em Inteligência Artificial Aplicada.

Sob a presidência do Orientador e Presidente da Banca, **Prof. Dr. Daniel Rosa Canêdo** - IFG/Câmpus Luziânia, a Banca Examinadora teve como Avaliadores o **Prof. Dr. Sérgio Daniel Carvalho Canuto** - IFG/Câmpus Anápolis e o **Prof. Ms. Renan Rodrigues de Oliveira** - IFG/Câmpus Goiânia.

Em sessão pública, após a apresentação da pesquisa e dos seus resultados, assim como a defesa do Trabalho de Conclusão de Curso pelo discente, os integrantes da Banca Examinadora fizeram as suas arguições, considerações e avaliações. Depois de se reunir em sala separada para avaliação e deliberação, a Banca Examinadora retornou à sala de defesa pública para a proclamação do resultado. Assim, em conformidade com o Regulamento Geral dos Programas de Pós-Graduação Lato Sensu do Instituto Federal de Goiás (IFG), a Banca Examinadora manifestou-se pela **APROVAÇÃO** do Trabalho de Conclusão de Curso de **Marcos Rodrigues Brugnaro**.

Anápolis - GO, 14 de fevereiro de 2025.

Documento assinado eletronicamente por:

1. Prof. Dr. Daniel Rosa Canêdo - IFG/Câmpus Luziânia
2. Prof. Dr. Sérgio Daniel Carvalho Canuto - IFG/Câmpus Anápolis
3. Prof. Ms. Renan Rodrigues de Oliveira - IFG/Câmpus Goiânia
4. Marcos Rodrigues Brugnaro - Discente

Documento assinado eletronicamente por:

- **Sergio Daniel Carvalho Canuto, PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 30/06/2025 16:10:29.
- **Marcos Rodrigues Brugnaro, 20222060180075 – Discente**, em 28/06/2025 18:00:40.
- **Renan Rodrigues de Oliveira, PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 28/06/2025 11:21:54.
- **Daniel Rosa Canedo, PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 28/06/2025 09:33:07.

Este documento foi emitido pelo SUAP em 30/01/2025. Para comprovar sua autenticidade, faça a leitura do QRCode ao lado ou acesse

<https://suap.ifg.edu.br/autenticar-documento/> e forneça os dados abaixo:

Código Verificador: 612236

Código de Autenticação: 3b5b99bb4e



Instituto Federal de Educação, Ciência e Tecnologia de Goiás

Avenida Pedro Ludovico, s/ nº, S/N, Remy Cury, ANÁPOLIS / GO, CEP 75131-457

(62) 3703-3359 (ramal: 3359)



MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
SISTEMA INTEGRADO DE BIBLIOTECAS

TERMO DE AUTORIZAÇÃO PARA DISPONIBILIZAÇÃO
NO REPOSITÓRIO DIGITAL DO IFG - ReDi IFG

Com base no disposto na Lei Federal nº 9.610/98, AUTORIZO o Instituto Federal de Educação, Ciência e Tecnologia de Goiás a disponibilizar gratuitamente o documento no Repositório Digital (ReDi IFG), sem ressarcimento de direitos autorais, conforme permissão assinada abaixo, em formato digital para fins de leitura, download e impressão, a título de divulgação da produção técnico-científica no IFG.

Identificação da Produção Técnico-Científica

- | | |
|--|---|
| ☐ Tese | ☐ Artigo Científico |
| ☐ Dissertação | ☐ Capítulo de Livro |
| ☒ Monografia – Especialização | ☐ Livro |
| ☐ TCC - Graduação | ☐ Trabalho Apresentado em Evento |
| ☐ Produto Técnico e Educacional - Tipo: _____ | |

Nome Completo do Autor: Marcos Rodrigues Brugnaro

Matrícula: 20222060180075

Título do Trabalho: Viagens Urbanas Inteligentes: Predição de tempo de viagens urbanas utilizando modelos de Machine Learning e Redes Neurais Artificiais

Autorização - Marque uma das opções

1. (x) Autorizo disponibilizar meu trabalho no Repositório Digital do IFG (acesso aberto);
2. () Autorizo disponibilizar meu trabalho no Repositório Digital do IFG somente após a data ____/____/____ (Embargo);
3. () Não autorizo disponibilizar meu trabalho no Repositório Digital do IFG (acesso restrito).

Ao indicar a opção **2 ou 3**, marque a justificativa:

- () O documento está sujeito a registro de patente.
- () O documento pode vir a ser publicado como livro, capítulo de livro ou artigo.

() Outra justificativa: _____

DECLARAÇÃO DE DISTRIBUIÇÃO NÃO-EXCLUSIVA

O/A referido/a autor/a declara que:

- i. o documento é seu trabalho original, detém os direitos autorais da produção técnico-científica e não infringe os direitos de qualquer outra pessoa ou entidade;
- ii. obteve autorização de quaisquer materiais inclusos no documento do qual não detém os direitos de autor/a, para conceder ao Instituto Federal de Educação, Ciência e Tecnologia de Goiás os direitos requeridos, e que este material cujos direitos autorais são de terceiros está claramente identificados e reconhecido no texto ou conteúdo do documento entregue;
- iii. cumpriu quaisquer obrigações exigidas por contrato ou acordo, caso o documento entregue seja baseado em trabalho financiado ou apoiado por outra instituição que não o Instituto Federal de Educação, Ciência e Tecnologia de Goiás.

Anápolis, 30 de Junho de 2025.



Documento assinado digitalmente

MARCOS RODRIGUES BRUGNARO

Data: 02/07/2025 12:50:51-0300

Verifique em <https://validar.iti.gov.br>

Assinatura do Autor e/ou Detentor dos Direitos Autorais

RESUMO

Estamos imersos em uma era orientada por dados, especialmente nos grandes centros urbanos, tecnologias embarcadas em veículos monitoram e coletam dados diários, tornando o transporte público urbano uma fonte valiosa para extração e análise de dados (GRUS, 2016) ¹.

Nesse contexto, o transporte público enfrenta o desafio da imprevisibilidade na duração das viagens urbanas, o que afeta diretamente os passageiros, que dependem de estimativas precisas para planejar melhor seus deslocamentos. Para abordar essa questão, este trabalho propõe a exploração de bases de dados abertas do transporte público de Curitiba-PR e a realização de experimentos com modelos preditivos, visando estimar com maior precisão a duração das viagens de ônibus urbanos.

Foram realizados experimentos utilizando diferentes técnicas de aprendizado de máquina com o objetivo de avaliar a precisão na previsão da duração do tempo de viagem dos ônibus urbanos. Os resultados mostraram que, ao utilizar a abordagem tradicional Regressão Polinomial combinada com Validação Cruzada, o modelo proposto conseguiu prever os tempos de viagem com maior precisão do que os métodos baseados e Redes Neurais Artificiais. O menor erro registrado foi de 29,9729 segundos, indicando uma melhoria significativa na confiabilidade das estimativas.

Palavras chave: Transporte Público Urbano, Estimativas Precisas, Modelos Preditivos, Regressão Polinomial, Redes Neurais Artificiais,

ABSTRACT

We are immersed in a data-driven era, especially in large urban centers, technologies embedded in vehicles monitor and collect daily data, making urban public transport a valuable source for data extraction and analysis (GRUS, 2016) ¹.

In this context, public transport faces the challenge of unpredictability in the duration of urban trips, which directly affects passengers, who depend on accurate estimates to better plan their journeys. To address this issue, this work proposes exploring open public transport databases in Curitiba-PR and carrying out experiments with predictive models, aiming to more accurately estimate the duration of urban bus trips.

Experiments were carried out using different machine learning techniques to evaluate the accuracy in predicting the duration of urban bus travel times. The results showed that, when using the traditional Polynomial Regression approach combined with Cross-Validation, the proposed model was able to predict travel times with greater accuracy than methods based on Artificial Neural Networks. The lowest error recorded was 29,9729 seconds, indicating a significant improvement in the reliability of the estimates.

Keywords: Urban Public Transport, Accurate Estimates, Predictive Models, Polynomial Regression, Artificial Neural Networks,

LISTA DE SIGLAS

UFPR	Universidade Federal do Paraná
IOT	Internet das Coisas
API	Interface de Programação de Aplicações
IA	Inteligência Artificial
ML	Machine Learning
EDA	Análise Exploratória de Dados
KDD	Descoberta de Conhecimento em Bases de Dados
MAE	Erro Médio Absoluto
MSE	Erro Quadrático Médio
RMSE	Raiz do Erro Quadrático Médio
MAPE	Erro Absoluto Percentual Médio
R²	Coeficiente de Determinação
SVR	Regressão de Vetores de Suporte
MLP	Perceptron Multicamadas
RNA	Redes Neurais Artificiais
ANOVA	Análise de Variância
FWER	Family-Wise Error Rate

LISTA DE TABELAS

- Tabela 01** Métricas de Desempenho dos Algoritmos Tradicionais.
- Tabela 02** Métricas de Desempenho dos Algoritmos Baseados em Redes Neurais Artificiais.
- Tabela 03** Atributo `line_category` e suas regras numéricas.
- Tabela 04** Tabela de outliers.
- Tabela 05** Tabela de Coeficientes de Correlações entre Atributos.
- Tabela 06** Resultados do teste de Shapiro-Wilk para normalidade dos dados.
- Tabela 07** Resultados do teste de Friedman para comparação dos modelos.
- Tabela 08** Resultados do teste de Wilcoxon para comparações pareadas entre os modelos.
- Tabela 09** Resultados do teste de Nemenyi para comparações múltiplas entre os modelos.
- Tabela 10** Análise descritiva dos modelos baseados em RNA: média e desvio padrão.
- Tabela 11** Comparação Múltipla de Médias - Teste de Tukey HSD, FWER=0.05.
- Tabela 12** MAE para cada fold dos modelos tradicionais.
- Tabela 13** MAE para cada fold dos modelos baseados em RNA.
- Tabela 14** Comparação estatística entre Regressão Polinomial e Redes Neurais Artificiais (RNA).

LISTA DE FIGURAS

Figura 01 Atributos: duration, duration_min, station_initial_latitude.

Figura 02 Atributos: station_initial_longitude, station_final_latitude, station_final_longitude.

Figura 03 Atributos: itinerary_company_name, line_category, line_color, distance.

Figura 04 Atributos: distance_km, average_speed_ms, average_speed_kh.

Figura 05 Atributos: vehicle_time, accessibility_info_time.

Figura 06 Atributos: line_payment_method, accessibility_info_electric_ramp.

Figura 07 Atributos: line_color, line_category, accessibility_info_day.

Figura 08 Atributo: duration.

Figura 09 Gráfico de agrupamento do atributo: duration.

Figura 10 Gráfico de agrupamento do atributo: line_category.

Figura 11 Gráfico Matrix de Correlação.

Figura 12 Gráfico da Curva de Aprendizado dos Modelos RL Polinomial e SVR - Dataset Treino.

Figura 13 Gráfico da Curva de Aprendizado do Modelo MLP e Evolução do Erro Médio Absoluto (MAE) ao longo dos folds do modelo RL. RNA - Dataset Treino.

SUMÁRIO

1	INTRODUÇÃO	19
1.1	Contextualização	19
1.2	Objetivos	20
1.3	Justificativa do Estudo	21
1.4	Trabalhos Relacionados	21
2	REFERENCIAL TEÓRICO	23
2.1	Inteligência Artificial	23
2.2	Processamento de Dados	23
2.3	Avaliação dos Modelos	24
3	METODOLOGIA	25
3.1	Abordagem Metodológica	25
3.2	Etapas do Processo KDD	25
4	ANÁLISES DOS DADOS E EXPERIMENTOS	27
4.1	Aquisição dos Dados	27
4.2	Pré-processamento dos Dados	28
4.3	Extração de Atributos	29
4.4	Pós-Processamento	30
4.5	Análise Exploratória dos Dados	31
4.6	Algoritmos Tradicionais	37
4.7	Redes Neurais Artificiais	38
5	ANÁLISES DOS RESULTADOS	41
5.1	CrITÉrios de Avaliação	41
5.2	Análise Comparativa dos Experimentos	43
5.3	Análise Comparativa dos Modelos com Melhor Desempenho	45
5.4	Análise Comparativa com Trabalho Relacionado	45
6	CONCLUSÃO	47
	REFERÊNCIAS	49
	APÊNDICE A – CÓDIGOS	51
	APÊNDICE B – DICIONÁRIO DE DADOS	64
	APÊNDICE C – FIGURAS	69

APÊNDICE D – EQUAÇÕES 73

APÊNDICE E – TABELAS 75

1 INTRODUÇÃO

1.1 Contextualização

O avanço das tecnologias digitais tem impulsionado profundas transformações nas sociedades contemporâneas, especialmente nas áreas urbanas (LARSEN, 2020)². A popularização da Internet das Coisas (IOT) e o aumento da conectividade entre dispositivos e pessoas têm moldado cidades mais inteligentes, capazes de gerar e utilizar grandes volumes de dados para promover sustentabilidade, eficiência e qualidade de vida.

No contexto da mobilidade urbana, destaca-se a importância dos dados gerados pelo transporte público, em especial aqueles relacionados à duração das viagens de ônibus. A imprevisibilidade desse tempo é um dos principais desafios enfrentados pelos passageiros, pois gera insegurança, sensação de desamparo e dificulta o planejamento tanto para passageiros quanto para operadores do sistema, podendo resultar em superlotação e irregularidade nas linhas (LARSEN, 2020)².

Para enfrentar esses desafios, a coleta e análise de dados tornaram-se fundamentais para aprimorar a eficiência e a confiabilidade do transporte público. Dispositivos embarcados, como GPS e sensores de embarque, aliados a sistemas integrados, fornecem uma base de conhecimento detalhada sobre os padrões de deslocamento urbano. Essas informações permitem subsidiar o desenvolvimento de estratégias mais eficazes para o planejamento do transporte público.

Neste trabalho, são utilizados dados históricos do transporte público de Curitiba-PR, disponibilizados pela Universidade Federal do Paraná (UFPR). As informações de geolocalização das rotas foram integradas à Interface de Programação de Aplicações (API) do Google Maps, permitindo a extração de atributos como distância e duração dos trajetos. Após a coleta, os dados passaram por um processo de pré-processamento para garantir qualidade e consistência.

A análise exploratória dos dados constituiu uma etapa fundamental para o entendimento aprofundado do conjunto de informações coletadas. Foram aplicadas técnicas de estatística descritiva para examinar a distribuição dos principais atributos, identificar valores atípicos (outliers) e verificar a presença de dados ausentes ou inconsistentes. Foram também realizadas visualizações gráficas, que facilitaram a identificação de padrões, tendências e possíveis correlações entre variáveis. Essa análise preliminar foi essencial para orientar as etapas subsequentes do estudo, permitindo ajustes no pré-processamento, a exclusão ou correção de registros problemáticos e a seleção dos atributos mais relevantes para a construção dos modelos preditivos.

Na fase experimental, o estudo buscou avaliar o desempenho de diferentes técnicas de previsão da duração das viagens de ônibus. Inicialmente, foram aplicados algoritmos tradicionais de aprendizado de máquina, como regressão linear e regressão polinomial, re-

conhecidos por sua eficiência computacional e facilidade de interpretação dos resultados. Posteriormente, foram exploradas abordagens baseadas em redes neurais artificiais, como o Perceptron Multicamadas (MLP), que se destacam pela capacidade de modelar relações não lineares e capturar padrões complexos presentes nos dados. Para garantir a robustez e a generalização dos resultados, os experimentos foram conduzidos utilizando validação cruzada, permitindo avaliar o desempenho dos modelos em diferentes subconjuntos do conjunto de dados. A avaliação dos modelos foi realizada com base na métrica de Erro Médio Absoluto (MAE), possibilitando uma comparação objetiva da precisão das previsões geradas por cada abordagem.

Por fim, a análise estatística dos resultados permitiu uma comparação rigorosa entre os modelos avaliados. Foram aplicados testes como o de Shapiro-Wilk, para verificar a normalidade dos resíduos, e o teste de Friedman, para comparar o desempenho dos algoritmos de forma não paramétrica. Comparações pareadas foram realizadas com o teste de Wilcoxon, e o teste de Tukey HSD foi utilizado para múltiplas comparações. Essa abordagem quantitativa possibilitou identificar o modelo mais eficiente e robusto para a previsão da duração das viagens, fornecendo subsídios para futuras aplicações e aprimoramentos no sistema de transporte público.

1.2 Objetivos

O objetivo geral deste trabalho é analisar conjuntos de dados relacionados ao transporte público da cidade de Curitiba-PR, utilizando informações de geolocalização das rotas integradas à API do Google Maps. Essa integração possibilitará a extração de atributos relevantes, como distância e duração dos trajetos, que servirão de base para a realização de experimentos com algoritmos de aprendizado de máquina e redes neurais artificiais. O foco está na geração de previsões precisas para a duração das viagens, juntamente com a avaliação do desempenho dos modelos desenvolvidos.

Os objetivos específicos deste estudo incluem:

1. Aquisição e o pré-processamento dos dados, assegurando sua qualidade e consistência;
2. Extração de atributos a partir da integração com a API do Google Maps, com ênfase nos atributos de distância e duração dos trajetos;
3. Análise exploratória dos dados, visando identificar padrões, tendências e correlações relevantes;
4. Implementação e avaliação de modelos preditivos baseados em aprendizado de máquina e redes neurais artificiais;

5. Análise do desempenho dos modelos com base na métrica MAE, complementada por testes estatísticos para verificar a eficácia das previsões e identificar o modelo com melhor desempenho.

1.3 Justificativa do Estudo

O acesso a informações precisas sobre a duração das viagens de ônibus é fundamental tanto para os passageiros quanto para a gestão eficiente do transporte público. No entanto, prever com exatidão esses tempos de deslocamento ainda constitui um desafio significativo, uma vez que fatores como variações no trânsito, condições climáticas e eventuais falhas operacionais afetam diretamente a regularidade do serviço. Um exemplo notório desse problema é o fenômeno conhecido como bunching ou comboio, no qual longos períodos de espera são seguidos pela chegada simultânea de vários ônibus, comprometendo a eficiência do sistema e a experiência dos usuários.

Apesar dos avanços em sistemas de monitoramento, os métodos tradicionais de estimativa frequentemente não conseguem capturar toda a complexidade envolvida e acabam por fornecer previsões pouco confiáveis. Nesse contexto, modelos de aprendizado de máquina despontam como uma alternativa promissora, pois são capazes de identificar padrões ocultos nos dados e aprimorar a precisão das previsões. Contudo, ainda é necessário investigar quais técnicas são mais adequadas para esse cenário e de que forma a inclusão de diferentes variáveis pode impactar a qualidade das estimativas.

Diante desse panorama, este estudo justifica-se pela necessidade de explorar o potencial das técnicas de aprendizado de máquina na previsão da duração das viagens de ônibus urbanos, a partir da utilização de dados históricos e de atributos relevantes ao tempo dos deslocamentos. Ao analisar diferentes abordagens preditivas, busca-se avaliar o impacto dessas técnicas na confiabilidade das estimativas.

1.4 Trabalhos Relacionados

Diversos estudos vêm investigando a aplicação de técnicas de aprendizado de máquina na previsão da duração das viagens no transporte público urbano, com o objetivo de tornar o planejamento e a operação desses sistemas mais eficientes e precisos.

O artigo "Influências da Inteligência Artificial na Otimização do Tráfego Urbano", Machado e Silva (2022)³, apresenta uma revisão bibliográfica sobre a aplicação da Inteligência Artificial na melhoria do tráfego urbano. O estudo destaca projetos e soluções que utilizam IA para otimizar sistemas de controle viário, aumentar a segurança e promover maior eficiência na mobilidade nas cidades.

O artigo "Sistema Algorítmico de Rede Neural Artificial para Previsão da Localização de Ônibus", Civali (2021).⁴ propõe uma solução para estimar a posição de ônibus da UNICAMP em casos de falha na comunicação via GPS. A proposta utiliza redes neurais artifi-

ais alimentadas com dados históricos e recentes. O estudo detalha as etapas de filtragem, classificação e mineração dos dados. O objetivo é aprimorar a gestão do transporte e a experiência dos usuários no campus.

O artigo "Desenvolvimento de Sistema de Inteligência Artificial (IA) Baseado em Reconhecimento de Padrões Para Análise de Rotas Veiculares", ROPKE (2020), de Leonardo Luís Röpke e Manuel Osório Binelo⁵, descreve o estudo de desenvolvimento de um sistema de IA com K-means e Redes Neurais Artificiais para análise de rotas veiculares. As Redes Neurais Artificiais obtiveram 100% de acurácia em rotas e 61% em desvios, enquanto o K-means foi eficaz em rotas similares, visando auxiliar gerentes de frota.

A dissertação de mestrado de Gustavo Henrique Larsen "Predição de tempos de viagem em linhas de ônibus baseado em dados de tráfego e redes neurais artificiais", LARSEN (2020)², mostra resultados promissores na predição de tempos de viagem de ônibus utilizando redes neurais, avaliadas através das métricas Raiz do Erro Quadrático Médio (RMSE) e Erro Absoluto Percentual Médio (MAPE), atestando a alta acurácia. Um exemplo é o RMSE de 5.5 segundos, confirmando a eficácia do modelo.

Para uma compreensão aprofundada da literatura revisada, as análises dos trabalhos e artigos relevantes estão dispostas em tabelas nos apêndices (E.13) e (E.14).

2 REFERENCIAL TEÓRICO

O referencial teórico deste trabalho se constrói sobre os pilares da Inteligência Artificial (IA), com ênfase especial em Aprendizado de Máquina e Redes Neurais Artificiais, com o objetivo de contextualizar e evidenciar a relevância dessas técnicas no cenário da predição de tempo de viagens urbanas, um desafio complexo influenciado por múltiplas variáveis associadas ao ambiente do transporte público. Esta seção aborda os conceitos fundamentais que sustentam a capacidade dos modelos computacionais de aprender padrões a partir de dados históricos, destacando sua contribuição para o aprimoramento da eficiência das predições aplicadas à mobilidade urbana.

2.1 Inteligência Artificial

A Inteligência Artificial IA é um campo da ciência da computação que busca desenvolver sistemas capazes de realizar tarefas que normalmente exigiriam inteligência humana, como raciocínio, tomada de decisão, aprendizado e percepção. Dentro do escopo da IA, destacam-se técnicas como Aprendizado de Máquina (Machine Learning (ML)) e Redes Neurais Artificiais (Redes Neurais Artificiais (RNA)), que têm sido amplamente aplicadas em problemas de regressão e classificação.

No contexto da mobilidade urbana, a IA tem se mostrado uma aliada promissora na análise de grandes volumes de dados gerados por sensores, GPS, aplicativos de transporte e infraestrutura urbana. Tais dados, quando processados de maneira inteligente, possibilitam a geração de predições mais precisas sobre o comportamento do tempo estimado das viagens, contribuindo para um planejamento mais eficiente do transporte público.

2.2 Processamento de Dados

O processamento de dados é uma etapa determinante para o sucesso de qualquer modelo de predição baseado em IA. Envolve a coleta, limpeza, transformação e preparação dos dados brutos para que possam ser utilizados de forma eficaz pelos algoritmos. Dados inconsistentes, com ruídos ou incompletos, podem comprometer significativamente a qualidade dos resultados.

Na predição de tempo de viagens urbanas, os dados geralmente incluem atributos como localização geográfica, nome de rotas e características dos ônibus. Nessa etapa, são comuns práticas como a seleção de atributos relevantes e o tratamento de dados ausentes, que visam melhorar a qualidade do conjunto de dados utilizado na modelagem. Além disso, técnicas como a Análise Exploratória de Dados (EDA) auxiliam na compreensão da estrutura, distribuição e possíveis padrões presentes nos dados, permitindo decisões mais informadas na preparação e construção dos modelos preditivos.

2.3 Avaliação dos Modelos

Para validar a eficácia dos modelos preditivos, é fundamental utilizar métricas de avaliação adequadas ao tipo de tarefa. No caso da predição de tempo de viagem, que se configura como um problema de regressão, foram adotadas as métricas Erro Médio Absoluto (MAE), Erro Quadrático Médio (MSE), Raiz do Erro Quadrático Médio (RMSE) e Coeficiente de Determinação (R^2).

Cada uma dessas métricas fornece uma perspectiva distinta sobre o desempenho do modelo. A MAE oferece uma medida direta da média dos erros absolutos, refletindo o quão próximo, em média, as previsões estão dos valores reais. Já o RMSE penaliza com mais intensidade os erros maiores, sendo especialmente útil em cenários onde outliers têm impacto significativo. A escolha dessas métricas foi orientada pelos objetivos do projeto e pelas características específicas do conjunto de dados utilizado.

Complementando as avaliações dos modelos, a aplicação de testes estatísticos é fundamental para verificar se as diferenças nas métricas de erro entre os algoritmos preditivos são estatisticamente significativas ou apenas fruto do acaso. A Análise de Variância (ANOVA) é amplamente empregada para comparar médias entre grupos, sendo frequentemente seguida por testes post-hoc, como o de Tukey, que controla o erro tipo I em comparações múltiplas. Quando os pressupostos de normalidade, avaliados por testes como o de Shapiro-Wilk, não são atendidos, utilizam-se alternativas não paramétricas, como os testes de Friedman, Wilcoxon e Nemenyi. Tais abordagens estatísticas conferem maior rigor e confiabilidade à análise comparativa entre modelos.

Os conceitos explorados neste capítulo servem de base para os modelos preditivos desenvolvidos na seção 4, aplicados aos dados coletados na cidade de Curitiba-PR, evidenciando as técnicas de IA no contexto da predição de tempo de viagens urbanas.

3 METODOLOGIA

Para alcançar os objetivos propostos neste trabalho, foi essencial compreender o contexto da pesquisa e selecionar métodos adequados para a análise dos dados. Nesse processo, foram considerados os métodos adotados pela comunidade acadêmica nos últimos anos, com o intuito de identificar aqueles mais eficazes e compatíveis com o problema em estudo. As etapas desse processo estão ilustradas no apêndice (C.8).

3.1 Abordagem Metodológica

A abordagem metodológica adotada neste trabalho foi a Descoberta de Conhecimento em Bases de Dados (KDD) (Knowledge Discovery in Databases), devido à sua comprovada eficácia na extração de conhecimento a partir de dados estruturados, uma característica essencial para os objetivos desta pesquisa. Segundo Fayyad et al. (1996)⁶, esse processo é composto por etapas fundamentais que orientam de forma sistemática a descoberta de conhecimento em bases de dados, abrangendo desde a seleção e pré-processamento até a mineração e interpretação dos resultados.

3.2 Etapas do Processo KDD

- **Seleção dos Dados:** O conjunto de dados utilizado neste estudo foi obtido a partir de fontes confiáveis relacionadas ao transporte público urbano da cidade de Curitiba, localizada no estado do Paraná. A base de dados principal foi fornecida pela Universidade Federal do Paraná (UFPR), por meio do portal de dados abertos disponível em <https://dadosabertos.c3sl.ufpr.br/curitiba/urbs/>;
- **Pré-processamento dos Dados:** Esta etapa compreendeu a padronização e validação dos atributos, o tratamento de valores não numéricos, a ordenação e limpeza dos registros, e remoção de duplicatas. Também foi realizada a fusão e concatenação dos diferentes conjuntos de dados, assegurando a consistência e integridade da base final utilizada na modelagem;
- **Transformação dos Dados:** A etapa de transformação teve como foco a extração de atributos (feature engineering), com o objetivo de obter dados de distância e duração entre os pontos inicial e final de cada rota, a partir de suas coordenadas geográficas. Para isso, foi utilizada a Interface de Programação de Aplicações (API) do Google Maps, com o apoio das bibliotecas gmaps e googlemaps, e um algoritmo desenvolvido especificamente para essa finalidade;
- **Mineração de Dados:** Nesta etapa, foram conduzidos experimentos com algoritmos de Machine Learning (ML) tradicionais, incluindo Regressão Linear Simples,

Regressão Linear Múltipla, Regressão Polinomial e Regressão de Vetores de Suporte (SVR). Também foram aplicados modelos baseados em Redes Neurais Artificiais (RNA), como Perceptron Multicamadas (MLP), Regressão com Redes Neurais e Regressão com Tuning de Hiperparâmetros. Todos os modelos foram desenvolvidos com o objetivo de realizar tarefas de regressão, visando prever a duração das viagens de ônibus urbano;

- **Interpretação e Avaliação dos Resultados:** Os critérios definidos para avaliação dos modelos asseguram uma base consistente para a comparação entre diferentes abordagens de aprendizado supervisionado. A divisão dos dados, a validação cruzada e o uso padronizado dos atributos previsores garantem reprodutibilidade e confiabilidade nos experimentos. As métricas Erro Médio Absoluto (MAE), Erro Quadrático Médio (MSE), Raiz do Erro Quadrático Médio (RMSE) e Coeficiente de Determinação (R^2) permitem avaliar a qualidade do ajuste dos modelos, fornecendo uma análise abrangente do desempenho. Esses elementos serão fundamentais para a interpretação dos resultados, aplicação de testes estatísticos e identificação dos modelos mais precisos na previsão da duração das viagens.

4 ANÁLISES DOS DADOS E EXPERIMENTOS

4.1 Aquisição dos Dados

Durante a seleção da base de dados para este trabalho, optou-se pelo uso de dados reais, pois garantem maior relevância e confiabilidade aos experimentos, permitindo que os resultados reflitam cenários do mundo real (GERON, 2019) ⁷. O conjunto de dados utilizado foi obtido no repositório de dados abertos da Universidade Federal do Paraná (UFPR), disponível no seguinte link: <https://dadosabertos.c3sl.ufpr.br/curitiba/urbs/>.

As bases de dados utilizadas nesta pesquisa incluem informações detalhadas sobre as linhas de ônibus, geolocalização, categorias de serviços, entre outras características. Esses dados são fornecidos em arquivos compactados em formato .tar.gz (Tape Archive). No momento em que esses arquivos são extraídos, eles assumem o formato JavaScript Object Notation (JSON), um padrão amplamente utilizado para a troca de dados estruturados. Este repositório disponibiliza dados diariamente, desde 18 de janeiro de 2017 até a data atual, realizando atualizações contínuas. Para obter os dados completos de um único dia, é necessário realizar o download de oito datasets, cujas especificações estão detalhadas no apêndice B.1.

Cada arquivo disponibilizado contém um conjunto de dados organizados em estruturas tabulares, onde cada linha representa a ocorrência de um evento e as colunas correspondem às características que descrevem essa instância (SILVA, 2016) ⁸. Para garantir a relevância das informações e facilitar a análise, foram selecionados apenas os datasets mais adequados ao estudo, os quais estão detalhados no apêndice B.2.

Para obter uma visão inicial das informações contidas nos datasets, foi desenvolvido um diagrama esquemático que representa a estrutura dos datasets selecionados. Esse diagrama serve como referência para a compreensão das relações entre os dados, facilitando a identificação das características mais relevantes para o estudo. Os diagramas estão detalhados nos apêndices B.3, B.4, B.5, B.6, B.7, B.8, B.9.

Por meio de uma análise sistemática dos datasets, observou-se que o atributo `cod`, presente no dataset `linhas`, estabelece uma relação direta com os demais datasets selecionados, possibilitando a integração das informações entre diferentes conjuntos de dados. Essa conexão facilita a análise cruzada e proporciona uma visão mais ampla e detalhada das informações disponíveis, conforme demonstrado no apêndice C.1. Com base nessa relação comum do código identificador da linha (`cod`), foram adotados os seguintes procedimentos:

- Remodelagem dos datasets: Reorganizar a estrutura dos datasets para melhorar a consistência e facilitar interpretações e análises.

- Remover atributos desnecessários: Eliminar atributos que não são relevantes para os objetivos do trabalho, deste modo, reduzindo a complexidade dos dados.
- Alterar nomes dos atributos: Renomear os atributos para padronizá-los e torná-los mais descritivos, facilitando a compreensão e manipulação dos dados.
- Construir um novo diagrama de esquemas: Construir um diagrama atualizado que reflita a estrutura remodelada, destacando as relações e conexões entre os datasets.

Essas etapas visam otimizar o processo de análise, garantindo maior eficiência na fase de pré-processamento. Nos apêndices B.10, B.11, B.12, B.13 e B.14, são apresentados os diagramas atualizados, que refletem as modificações realizadas após a aplicação das etapas descritas anteriormente.

4.2 Pré-processamento dos Dados

Dados provenientes do mundo real frequentemente apresentam falhas ou estão organizados de maneira inadequada para análise e aplicação em algoritmos de aprendizado de máquina. Portanto, a etapa de pré-processamento é essencial para corrigir inconsistências e, quando necessário, agregar valor aos dados, garantindo maior qualidade e confiabilidade para os experimentos (SILVA, 2016) ⁸.

A etapa de pré-processamento tem início com a coleta dos dados, automatizada por um algoritmo desenvolvido para realizar requisições a cada dataset, eliminando a necessidade de intervenção manual e aumentando a eficiência do processo. Para assegurar a precisão e a confiabilidade na captura dos dados brutos, foram definidos parâmetros essenciais, como ano, mês, datas de início e fim do período e o caminho de armazenamento dos datasets. Com base nesses critérios, os dados foram coletados em dois períodos específicos: de 1º a 31 de janeiro de 2023 e de 5 a 30 de junho de 2024.

A partir deste ponto, os datasets foram carregados e submetidos às etapas de pré-processamento, que devem ser executadas em uma ordem específica para garantir a integridade e a qualidade dos dados, conforme destacado por Geron (2019) ⁷. Para assegurar essa sequência, foi desenvolvida uma pipeline que organiza as transformações, conforme demonstrado no apêndice C.2, e descritas a seguir:

1. Renomeação dos atributos visando a padronização;
2. Validação do atributo `line_id`, assegurando que contenha apenas registros numéricos;
3. Tratamento de registros não numéricos encontrados no atributo `line_id`, removendo a string "X" de valores no formato "X número" e preservando apenas o número;
4. Conversão dos registros do atributo `line_id` do tipo `object` para `int64`;

5. Ordenação crescente dos registros com base na *feature line_id*;
6. Identificação e remoção de registros duplicados do atributo *line_id*;
7. Merge dos referentes ao mês de Janeiro de 2023 e Junho de 2024;
8. Carregamento dos dois datasets resultantes dos merges anteriores e concatenação dos mesmos para gerar um único conjunto de dados consolidado.

4.3 Extração de Atributos

O processo de extração de atributos, também conhecido como feature engineering, é essencial para ampliar as características dos dados. Neste trabalho, seu objetivo é obter informações sobre a distância e a duração do tempo de viagem entre os pontos inicial e final de cada rota. Para isso, foram utilizadas as coordenadas de latitude e longitude correspondentes a cada trajeto, previamente adquiridas na etapa de aquisição de dados.

Para atingir esse objetivo, realizamos a integração com a API Distance Matrix do Google Maps, que estima o tempo de viagem com base nas rotas sugeridas. Essas rotas consideram a infraestrutura viária, as condições de tráfego em tempo real, restrições viárias e possíveis desvios temporários. No entanto, é importante destacar que a API não necessariamente considera a rota real percorrida pelo ônibus trecho por trecho, mas sim sugere um caminho baseado nas condições do momento. Isso significa que, em algumas situações, como congestionamentos ou bloqueios de tráfego, a API pode indicar rotas alternativas que não correspondem exatamente ao trajeto habitual do transporte público, o que pode impactar a precisão dos dados extraídos.

Essa abordagem, se não devidamente tratada, pode comprometer a precisão das previsões do estudo, uma vez que a consideração de rotas alternativas pode fazer com que os tempos de viagem estimados pela API não representem com fidelidade o comportamento real do transporte público, que opera com trajetos fixos e pré-definidos. Para mitigar esse problema e garantir estimativas mais precisas, foram adotadas algumas estratégias:

1. Validação com dados históricos do transporte público: A captação e organização dos dados históricos de latitude e longitude inicial e final das rotas de ônibus da cidade de Curitiba-PR, extraídos dos registros de GPS presentes no dataset deste estudo, foram fundamentais para o ajuste dos resultados. Essa abordagem permitiu reduzir discrepâncias e aprimorar a precisão do modelo de previsão.
2. Filtragem e correção dos dados extraídos: Após a obtenção das previsões da API, foram aplicadas técnicas de pós-processamento para analisar, identificar e corrigir possíveis discrepâncias. Quando um tempo de viagem apresentava valores significativamente abaixo ou acima da média histórica, foram realizados ajustes com base em análises estatísticas descritivas, garantindo maior consistência nos dados.

Visando proporcionar uma compreensão intuitiva dos trajetos, foi elaborada uma lista dos pontos iniciais e finais das rotas, incluindo os 10 primeiros registros (apêndice C.3, C.4), a rota com a menor distância e a rota com a maior distância (apêndice C.5, C.6).

4.4 Pós-Processamento

Após a extração de atributos, o dataset passou pela etapa de pós-processamento, cujo objetivo foi identificar ajustes adicionais, realizar limpezas complementares e explorar a possibilidade de extrair novos atributos a partir dos dados disponíveis, visando aprimorar ainda mais a qualidade dos dados para as análises subsequentes. A seguir, são apresentados os tratamentos aplicados ao dataset nessa fase.

1. **Análise e tratamento de valores inconsistentes:** Foram removidos os valores nulos dos atributos `station_type`, `vehicle_plate`, `accessibility_info_electric_ramp`. Os valores nulos dos atributos `station_final_latitude`, `station_final_longitude` foram substituídos com base em análises estatísticas;
2. **Transformação de atributos categóricos em numéricos:** Foi realizada a conversão de atributos categóricos em valores numéricos para aqueles cuja soma total de ocorrências é inferior a 20. Os seguintes atributos passaram por essa transformação: `line_payment_method`, `line_category`, `line_color`, `station_type`, `accessibility_info_electric_ramp`, `itinerary_company_name` e `itinerary_special_line`;
3. **Extração de atributos:** Novos atributos foram gerados a partir de dados de duração e distância das viagens. Esses dados foram processados e combinados com conversões baseadas em princípios da física (PARANA, 2003)⁹, permitindo o cálculo dos atributos desejados. A seguir, apresenta-se a lista de atributos extraídos:
 - Atributo `distance_km`, definido pela conversão do valor da distância para quilômetros. A equação é demonstrada no apêndice (D.1);
 - Atributo `duration_min`, definido pela conversão do valor da duração de tempo das viagens para minutos. A equação é demonstrada no apêndice (D.2);
 - Atributo `average_speed_ms`, definido pelo cálculo da velocidade média, obtido dividindo o valor do atributo `distance` pelo do atributo `duration`. A equação é demonstrada no apêndice D.3;
4. **Conversão dos dados de tempo (horas para minutos):** Convertendo dados de tempo dos atributos `vehicle_time` e `accessibility_info_time` para minutos.

A partir dessas melhorias, o dataset é submetido à análise exploratória de dados, onde serão realizadas análises mais detalhadas, com o objetivo de identificar padrões,

anomalias e relações entre os atributos. Essa etapa é crucial para entender profundamente o comportamento dos dados.

4.5 Análise Exploratória dos Dados

A análise exploratória de dados é fundamental para compreender o comportamento do conjunto de dados e prepará-lo para as etapas subsequentes[?]. As análises estatísticas desempenham um papel crucial na identificação de imprecisões e desvios, permitindo a detecção de ruídos e a transformação dos dados por meio de medidas como média e desvio-padrão. Além disso, auxiliam na seleção de atributos relevantes e na eliminação de redundâncias. A seguir são apresentadas algumas estatísticas descritivas obtidas a partir do conjunto de dados:

- **Tamanho da amostra:** Cada atributo contém 12.372 registros, demonstrando que o dataset não apresenta valores ausentes.
- **Média dos valores:** A média dos valores oferece uma visão centralizada de cada atributo. A média de 1286,84 segundos para o atributo *duration* indica que as durações das viagens de ônibus estão concentradas em torno desse valor, o que equivale a aproximadamente 21,45 minutos.
- **Desvio padrão:** Identificou-se um desvio padrão elevado em alguns atributos, como *distance* e *duration*, indicando uma variabilidade significativa nos dados. Atributos relacionados à geolocalização apresentaram desvios padrão baixos, refletindo maior consistência dentro dos intervalos geográficos analisados.
- **Valores mínimos:** Alguns atributos apresentaram valores mínimos consistentes com as expectativas. Por exemplo, a latitude mínima de -25.625 e a longitude mínima de -49.378 estão alinhadas ao intervalo geográfico de Curitiba, definido pela latitude mínima de 25°25'40" (sul) ou -25.4278 graus e longitude mínima de 49°16'23" (oeste) ou -49.2731 graus.
- **Valores máximos:** A distância máxima de 47,9 quilômetros sugere a existência de rotas longas no conjunto de dados, enquanto o tempo máximo de 74,32 minutos (aproximadamente 1 hora, 14 minutos e 19 segundos) reforça a ocorrência de viagens mais extensas.

Em seguida são apresentados e analisados histogramas de diversos atributos do conjunto de dados. A visualização gráfica permite identificar padrões de distribuição e possíveis agrupamentos nos dados.

Inicialmente, foram analisados os histogramas dos atributos *duration*, *duration_min*, *station_initial_latitude*, *station_initial_longitude*, *station_final_latitude* e *station_final_longitude*.

A distribuição desses atributos apresenta um comportamento simétrico, com dados aproximadamente equilibrados em torno da mediana.

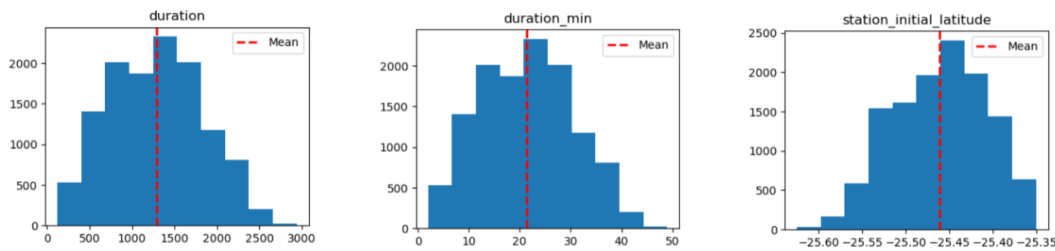


Figura 1 – Atributos: duration, duration_min, station_initial_latitude. (Figura 01)

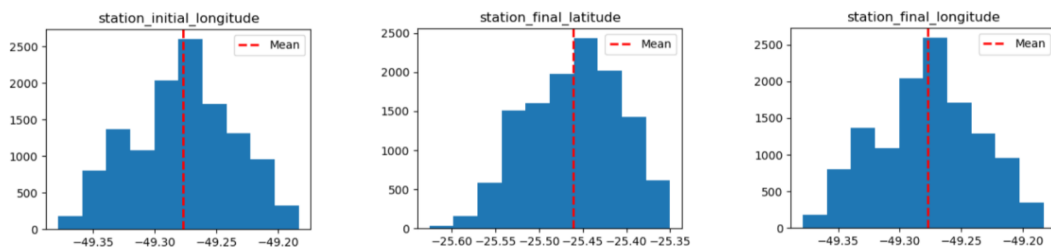


Figura 2 – Atributos: station_initial_longitude, station_final_latitude, station_final_longitude. (Figura 02)

Na sequência, foram analisados os atributos *itinerary_company_name*, *line_category*, *line_color*, *distance*, *distance_km*, *average_speed_ms* e *average_speed_kh*. Esses histogramas apresentam assimetria negativa, indicando que a maioria dos valores está concentrada em faixas superiores, enquanto valores menores são menos frequentes e formam uma cauda à esquerda. Complementando a análise, verificou-se a ausência de assimetria positiva, reforçando a predominância de dados mais elevados.

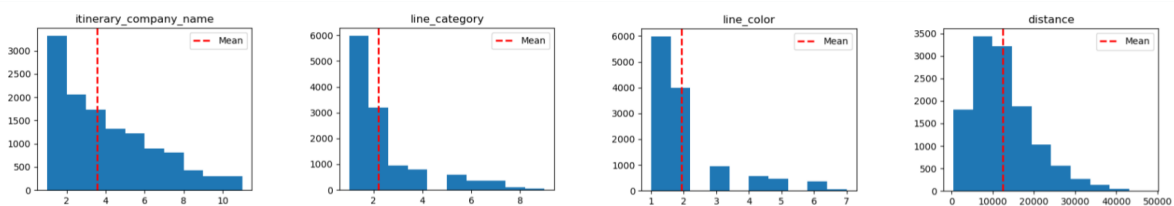


Figura 3 – Atributos: itinerary_company_name, line_category, line_color, distance. (Figura 03)

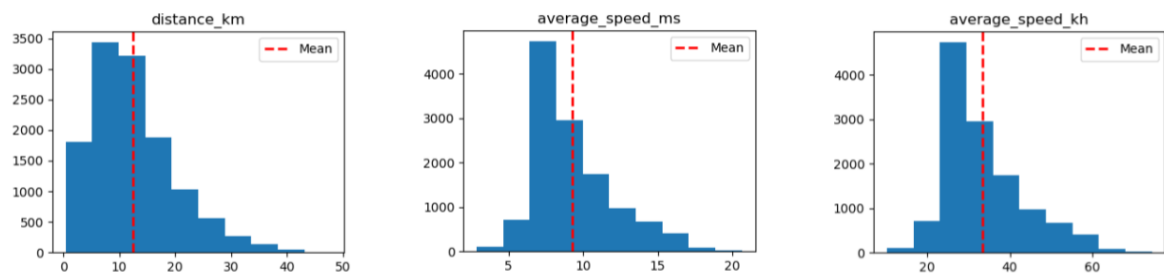


Figura 4 – Atributos: *distance_km*, *average_speed_ms*, *average_speed_kh*. (Figura 04)

Os histogramas dos atributos *vehicle_time* e *accessibility_info_time* apresentam caudas longas e picos acentuados, sugerindo que a maior parte dos dados está concentrada em torno da média. Em alguns casos, observa-se uma distribuição mais achatada, indicando uma maior dispersão dos valores. Os horários de chegada dos ônibus e as informações de acessibilidade mostram uma distribuição relativamente equilibrada ao longo do dia.

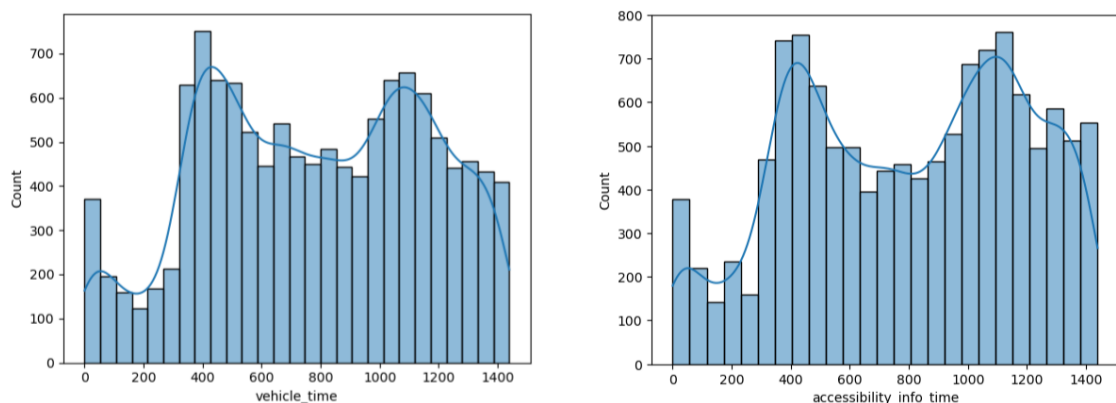


Figura 5 – Atributos: *vehicle_time*, *accessibility_info_time*. (Figura 05)

Os atributos *line_payment_method* e *accessibility_info_electric_ramp* apresentam distribuições bimodais, com dois picos distintos, sugerindo a existência de duas populações diferentes dentro desses dados.

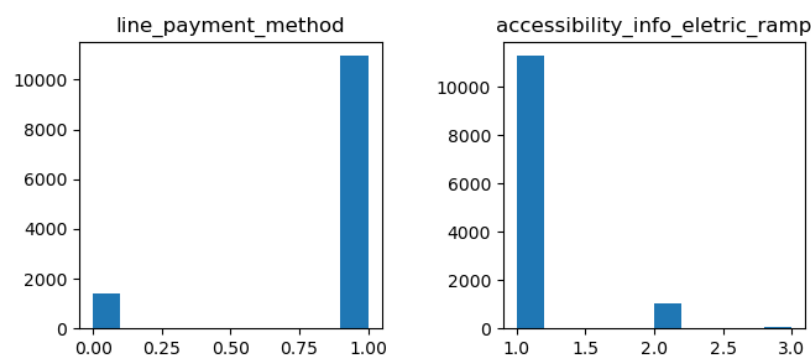


Figura 6 – Atributos: *line_payment_method*, *accessibility_info_electric_ramp*. (Figura 06)

Os atributos *line_color*, *line_category* e *accessibility_info_day* exibem distribuições multimodais, com múltiplos picos, indicando a presença de diferentes grupos ou categorias distintas dentro dos dados.

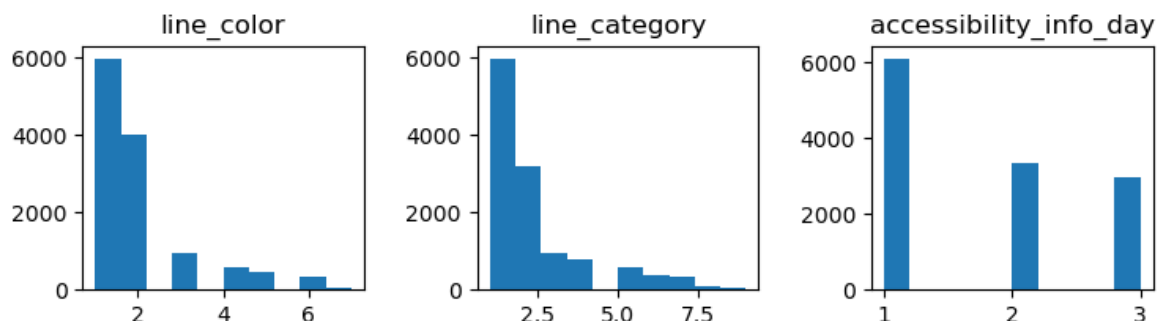


Figura 7 – Atributos: *line_color*, *line_category*, *accessibility_info_day*. (Figura 07)

Foi realizada a análise do atributo *duration*, que será utilizado como variável-alvo nos experimentos posteriores. A análise do histograma revelou que os tempos de viagem variam de valores próximos a 0 até 3000 segundos, sendo que a maioria das viagens ocorre entre 500 e 2000 segundos. Poucas viagens têm duração entre 2500 e 3000 segundos, e menos de 500 registros apresentam tempos de viagem superiores a 2500 segundos.

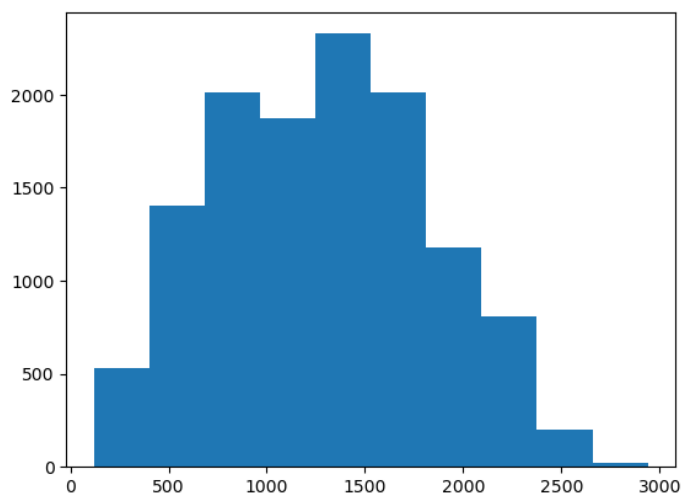


Figura 8 – Atributo: *duration*. (Figura 08)

Em seguida, foram gerados gráficos complementares de agrupamento dos atributos *duration* e *line_category*, permitindo uma análise detalhada dos padrões e tendências na distribuição dos dados. As informações extraídas dessas análises servem de base para estudos mais aprofundados. Inicialmente, são apresentadas as análises e informações referentes ao gráfico de agrupamento do atributo *duration*.

A distribuição dos tempos de viagem revela que 493 trajetos possuem uma duração de 1.140 segundos (equivalente a 19 minutos), enquanto apenas 4 trajetos apresentam um tempo de 2.820 segundos (47 minutos). Essa tendência evidencia a predominância

de trajetos com durações próximas à média, enquanto tempos mais longos são menos frequentes, mas ainda relevantes para a análise. Finalizando esta subseção, são apresentadas as análises do gráfico de agrupamento do atributo *line_category*.

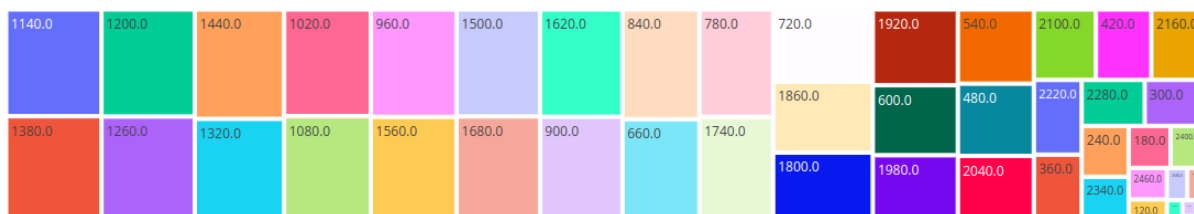


Figura 9 – Gráfico de agrupamento do atributo: *duration*. (Figura 09)



Figura 10 – Gráfico de agrupamento do atributo: *line_category*. (Figura 10)

O atributo *line_category*, ao ser transformado em um atributo numérico, seguiu as seguintes regras conforme a Tabela apresentada no apêndice (E.1). Observamos que a *line_category* com a maior quantidade de linhas é a *ALIMENTADOR*, com 5.985 registros, enquanto a menor é a *JARDINEIRA*, com apenas 54 registros.

Na sequência, realiza-se a etapa de detecção de outliers, com o objetivo de identificar valores anômalos que se distanciam significativamente da média e do comportamento esperado dos dados. Esses valores podem ser resultantes de erros de medição, falhas de digitação, fatores aleatórios ou até mesmo fraudes. A identificação dos outliers é essencial para garantir a qualidade da base de dados, permitindo a aplicação de estratégias adequadas de tratamento, como a remoção, substituição ou manutenção dos registros considerados inconsistentes.

O processo de detecção de *outliers* neste trabalho foi realizado utilizando o método do Intervalo Interquartil (IQR), considerando apenas os atributos numéricos do conjunto de dados. Para cada atributo, foram calculados o primeiro quartil (Q1) e o terceiro quartil (Q3), determinando assim o intervalo interquartil (IQR), conforme a equação apresentada no apêndice (D.4). A partir desse intervalo, os limites inferior e superior foram estabelecidos para identificar valores atípicos, conforme as Equações (D.5) e (D.6). A seguir, no apêndice (E.2), é demonstrada uma tabela contendo os atributos que apresentaram *outliers*.

Após a identificação dos atributos com *outliers*, foram realizadas análises para determinar se os valores extremos representavam erros ou variações legítimas nos dados. Cada *outlier* foi avaliado com base em limites teóricos e contextuais e, como todos os registros estavam dentro de intervalos plausíveis, decidiu-se preservá-los. Essa abordagem assegura a integridade e representatividade do conjunto de dados, evitando a remoção de informações potencialmente relevantes para a análise.

Finalizando a fase de análise exploratória dos dados, foi realizada a análise de correlação, que tem como objetivo investigar a relação entre dois conjuntos de variáveis, quantificando o grau de associação entre eles. Esse método permite identificar tanto a intensidade quanto a direção dessa relação. Neste trabalho, utilizou-se o coeficiente de correlação de Pearson (r), amplamente empregado nesse tipo de análise e descrito no apêndice (D.7). Com base nesse critério, foi gerada uma tabela, apresentada no apêndice (E.3), contendo as 20 combinações de atributos com os maiores coeficientes de correlação, devidamente ranqueadas.

Com o objetivo de simplificar a análise, foram selecionados os atributos que apresentaram as maiores correlações nas combinações e atributos complementares considerados relevantes. Os atributos escolhidos nesta etapa, por apresentarem forte correlação com o atributo-alvo e alto poder discriminativo, serão utilizados na fase de experimentos para a construção dos modelos preditivos. Os atributos selecionados para a análise incluem `line_category`, `line_color`, `distance`, `average_speed_kh`, `duration`, `station_initial_latitude`, `station_initial_longitude`, `station_final_latitude` e `station_final_longitude`.

Para melhorar a interpretação das correlações entre os atributos, optou-se por representá-las graficamente por meio da matriz de correlação, proporcionando uma visualização mais clara de suas interações. A matriz gerada a seguir destaca as correlações mais relevantes.

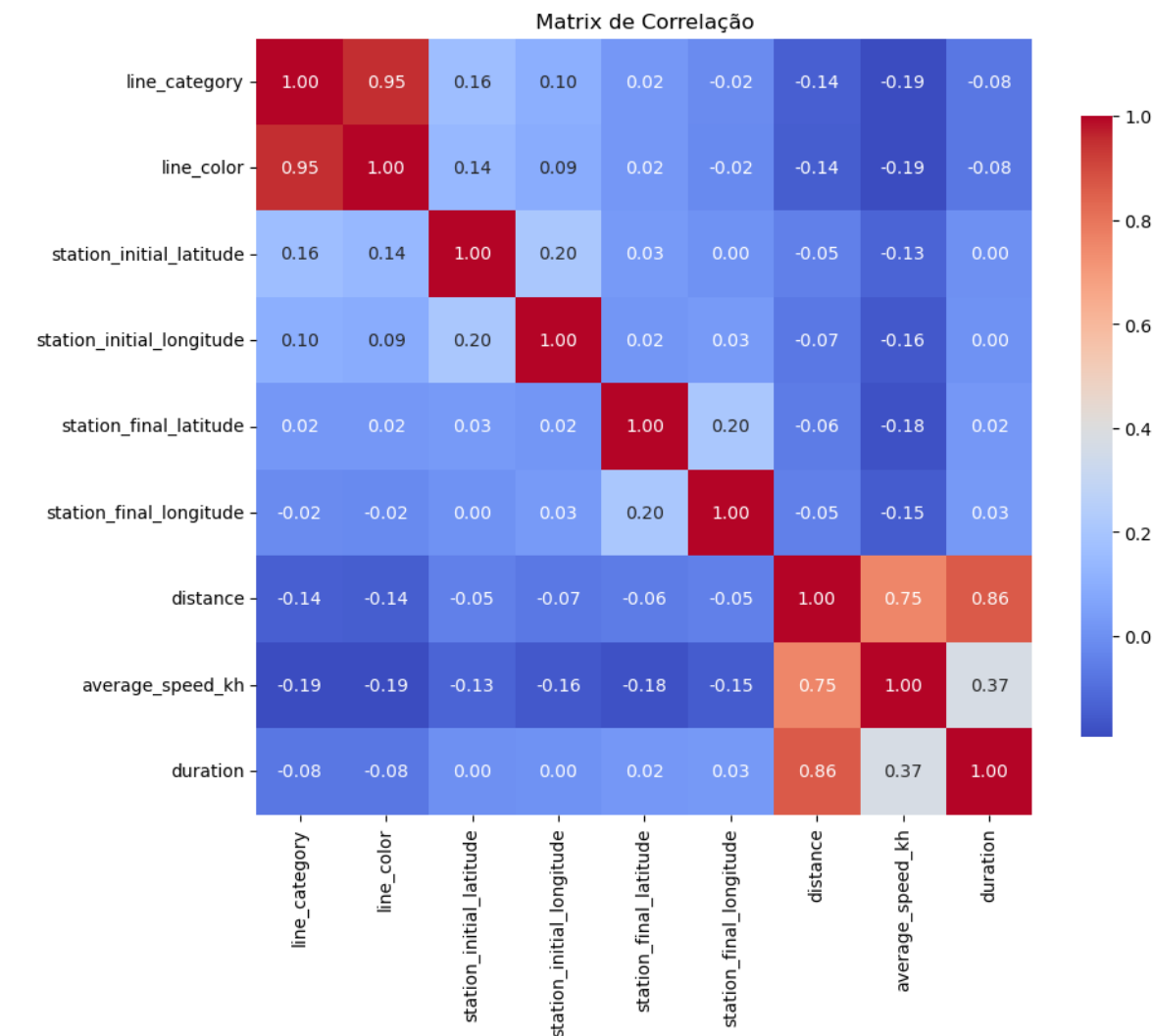


Figura 11 – Gráfico Matrix de Correlação. (Figura 11)

4.6 Algoritmos Tradicionais

Nesta seção, são apresentados os experimentos realizados com o objetivo de prever a duração das viagens de ônibus urbano. Inicialmente, foram conduzidos experimentos utilizando algoritmos tradicionais de aprendizado de máquina. Em uma segunda etapa, foram realizados experimentos explorando técnicas baseadas em redes neurais artificiais.

Os experimentos com ambas as abordagens visam equilibrar eficiência e capacidade de modelar padrões nos dados. Algoritmos tradicionais são eficientes, mas limitados na captura de padrões complexos, enquanto redes neurais artificiais são mais adequadas para modelar relações não lineares, potencialmente melhorando a precisão da predição.

Para a primeira etapa dos experimentos, foram selecionados quatro algoritmos tradicionais de aprendizado de máquina, amplamente utilizados em tarefas de regressão. A seguir, são apresentados de forma objetiva cada um desses algoritmos.

1. **Regressão Linear Simples:** Este algoritmo assume uma relação linear entre as variáveis independentes e a variável dependente. Sua escolha se justifica pela sua simplicidade, eficiência e facilidade de interpretação, permitindo a análise do impacto de cada variável nas previsões. A equação da reta de regressão, apresentada no apêndice (D.8), é utilizada neste algoritmo;
2. **Regressão Linear Múltipla:** Podemos considerar este algoritmo uma extensão da regressão linear simples. O Algoritmo de Regressão Linear Múltipla permite modelar a relação entre uma variável dependente e múltiplas variáveis independentes, tentando ajustar o modelo linear utilizando a equação da regressão linear múltipla apresentada no apêndice (D.9);
3. **Regressão Linear Polinomial:** Este algoritmo estende a regressão linear ao incluir termos polinomiais, permitindo capturar relações não lineares entre as variáveis. Sua flexibilidade torna-o adequado para padrões mais complexos nos dados, mas o aumento no número de parâmetros pode levar ao sobreajuste, especialmente com polinômios de alta ordem (GERON, 2019)⁷. A equação correspondente é apresentada no apêndice (D.10);
4. **Regressão de Vetores de Suporte (SVR):** Este algoritmo mapeia dados de entrada para valores contínuos, minimizando a distância entre os vetores de suporte e a função de regressão. Para lidar com não linearidades, foi utilizado o kernel RBF, que aumenta a capacidade de modelagem ao mapear os dados em um espaço de maior dimensão. A escolha do SVR com RBF justifica-se pela sua capacidade de capturar relações complexas entre os atributos e a duração das viagens, proporcionando maior flexibilidade e precisão nas previsões. A equação do kernel RBF é apresentada no apêndice (D.11).

4.7 Redes Neurais Artificiais

Para a segunda etapa dos experimentos, foram utilizados três algoritmos baseados em redes neurais artificiais, voltados para tarefas de regressão. A seguir, são apresentadas de forma objetiva as características e justificativas para a escolha de cada um desses algoritmos.

1. **Multilayer Perceptron (MLP):** O Perceptron, originalmente um modelo de único neurônio, pode ser expandido para formar redes com múltiplas camadas de neurônios interconectados, resultando na arquitetura de Perceptron Multicamadas (MLP). Esta abordagem foi escolhida para este experimento devido à sua capacidade de modelar relações complexas e não lineares nos dados, utilizando várias camadas para aprender representações mais complexas, o que é essencial para a precisão das

previsões. Sua representação pode ser observada no apêndice (C.7). A arquitetura deste modelo consiste em 8 neurônios na camada de entrada, duas camadas ocultas com 4 neurônios cada e um neurônio na camada de saída. A função de ativação utilizada é a ReLU, conforme apresentada no apêndice (D.12). O treinamento da rede neural foi realizado ao longo de 500 épocas;

2. Regressão com Redes Neurais Artificiais: Neste experimento, foi implementada uma rede neural simples, mantendo a mesma arquitetura do experimento anterior, mas com a conversão dos dados em tensores. A escolha desse modelo se justifica pela necessidade de avaliar seu desempenho inicial antes de explorar arquiteturas mais complexas e otimizações de hiperparâmetros;
3. Regressão com Redes Neurais Artificiais e Tuning de Hiperparâmetros: Neste experimento, foi implementado um modelo de Rede Neural Artificial (RNA) com as melhores combinações de hiperparâmetros, obtidas através da técnica de Grid Search. Essa técnica permite explorar todas as combinações possíveis de hiperparâmetros, avaliando-as por meio de validação cruzada, garantindo uma busca eficiente pelos melhores hiperparâmetros para otimizar o desempenho do modelo.

5 ANÁLISES DOS RESULTADOS

5.1 Critérios de Avaliação

Para os experimentos, utilizou-se uma base de dados com 12.372 registros, dividida em 80% (9.897 registros) para treinamento e 20% (2.475 registros) para teste. A fim de garantir uma avaliação robusta do desempenho dos modelos, foi empregada validação cruzada com 5 folds. A reprodutibilidade da divisão dos dados foi assegurada pela configuração do parâmetro `random_state` com o valor 0.

O experimento de Regressão Linear Simples utilizará o atributo predictor `distance`, enquanto os demais experimentos considerarão um conjunto ampliado de oito atributos previsores, listados a seguir:

1. `line_category`;
2. `line_color`;
3. `distance`;
4. `average_speed_kh`;
5. `station_initial_latitude`;
6. `station_initial_longitude`;
7. `station_final_latitude`;
8. `station_final_longitude`;

O aprendizado adotado é do tipo supervisionado, pois os exemplos de treinamento são rotulados, ou seja, cada instância possui um resultado esperado (GERON, 2019)⁷. A tarefa consiste em um problema de regressão, uma vez que o objetivo é prever um valor numérico referente à duração das viagens, representado pelo atributo `duration`.

A análise de desempenho será realizada através das métricas diagnósticas Erro Médio Absoluto (MAE), Erro Quadrático Médio (MSE), Raiz do Erro Quadrático Médio (RMSE) e Coeficiente de Determinação (R^2). As equações das respectivas métricas são apresentadas nos apêndices (D.13), (D.14), (D.15), (D.16).

A tabela (1) apresenta os resultados das métricas de desempenho dos algoritmos tradicionais de aprendizado de máquina, extraídos dos experimentos. As primeiras observações, antes dos testes estatísticos indicaram que o modelo Regressão Linear Regressão de Vetores de Suporte (SVR) apresentou bom desempenho geral, alcançando os menores valores de MAE, MSE, RMSE, e R^2 próximo de 1. O modelo Regressão Linear Polinomial também demonstrou um desempenho competitivo, com erro relativamente baixo e alto R^2 .

Já a Regressão Linear Múltipla obteve melhorias significativas em relação à Regressão Linear Simples. O modelo Regressão Linear Simples teve o pior desempenho, com altos erros e baixo poder explicativo, sendo o menos indicado para a tarefa.

Método	Conjunto	MAE	MSE	RMSE	R ²
RL. Simples	Treino	225,22 ± 3,02	76540,35 ± 101132,80	276,66 ± 160,85	0,7425
RL. Simples	Teste	221,82 ± 157,32	73952,13 ± 95877,29	271,94 ± 157,32	0,7445
RL. Múltipla	Treino	112,13 ± 1,97	23485,32 ± 50170,81	153,25 ± 104,65	0,9210
RL. Múltipla	Teste	112,72 ± 102,55	23224,25 ± 47085,86	152,40 ± 102,55	0,9198
RL. Polinomial	Treino	30,1397 ± 0,3646	1983,2615 ± 7212,7087	44,5338 ± 32,9757	0,9933
RL. Polinomial	Teste	29,7637 ± 32,9488	1971,5022 ± 8007,0122	44,4016 ± 32,9488	0,9932
RL. SVR	Treino	26,4947 ± 0,6932	946,1328 ± 161065,6673	30,7593 ± 202,8266	0,9968
RL. SVR	Teste	25,8052 ± 193,7452	1194,6055 ± 144564,9470	34,5631 ± 193,7452	0,9959

Tabela 1 – Métricas de Desempenho dos Algoritmos Tradicionais. (Tabela 01)

Os gráficos a seguir, apresentam as curvas de aprendizado dos modelos Regressão Linear Polinomial e SVR, mostrando o MAE em função do tamanho do conjunto de treinamento. A regressão polinomial demonstra o erro de treino aumentando e depois estabiliza, enquanto o erro de validação começa alto e diminui gradualmente, mas permanece maior que o erro de treino. O modelo SVR mantém erros de treino e validação próximos, diminuindo de forma consistente à medida que mais dados são usados.

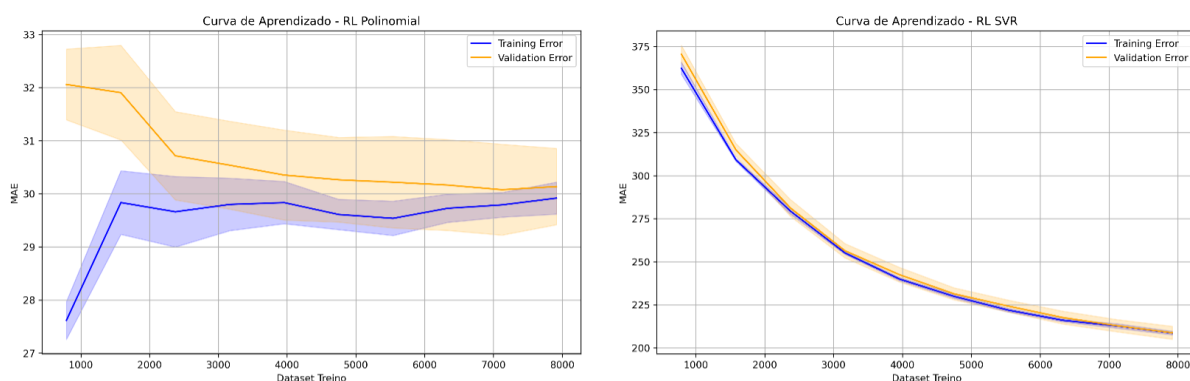


Figura 12 – Gráfico da Curva de Aprendizado dos Modelos RL Polinomial e SVR - Dataset Treino. (Figura 12)

A tabela (2) apresenta os resultados das métricas de desempenho dos modelos baseados em redes neurais artificiais, extraídos dos experimentos. As primeiras observações, antes dos testes estatísticos indicam que o modelo Perceptron Multicamadas (MLP) obteve bom desempenho geral, apresentando os menores valores de erro MAE, MSE, RMSE e um R² alto 0,9870 no teste. O modelo Regressão Linear Redes Neurais Artificiais (RNA) também demonstrou um desempenho competitivo, com R² de 0,9952, o mais alto entre os modelos, e erros relativamente baixos. O modelo Regressão Linear RNA Tuning apre-

sentou o pior desempenho, com os maiores valores de erro e um ajuste menos eficaz, sugerindo que o processo de tuning não foi bem-sucedido na otimização do modelo.

Método	Conjunto	MAE	MSE	RMSE	R ²
MLP	Treino	41,6081 ± 1,4635	4033,1907 ± 240,0610	63,5074 ± 1,9154	0,9864
MLP	Teste	43,5734 ± 1,4892	3763,8852 ± 500,0358	61,3505 ± 4,0013	0,9870
RL. RNA	Treino	38,5265 ± 7,8699	4906,7874 ± 9810,5219	10,8915 ± 5,2857	0,9834
RL. RNA	Teste	39,1328 ± 9,2648	3381,2432 ± 1521,7089	56,4508 ± 13,9483	0,9952
RL. RNA Tuning	Treino	67,2278 ± 22,9072	10921,8915 ± 9198,1032	97,7204 ± 37,0488	0,9656
RL. RNA Tuning	Teste	61,4123 ± 26,7711	10566,5648 ± 11470,7217	92,1914 ± 45,4678	0,9649

Tabela 2 – Métricas de Desempenho dos Algoritmos Baseados em Redes Neurais Artificiais. (Tabela 02)

Os gráficos a seguir analisam o MAE dos modelos MLP e RL. RNA. A curva de aprendizado do modelo MLP indica que os erros de treino e validação diminuem progressivamente com o aumento do dataset, mantendo-se próximos, o que sugere boa generalização. No gráfico do MAE por fold do modelo RL. RNA, observa-se uma variação inicial entre os erros de treino e teste, especialmente no fold 2, mas essa diferença reduz ao longo dos folds, evidenciando uma adaptação progressiva do modelo. No último fold, a convergência dos erros estabiliza o equilíbrio do aprendizado.

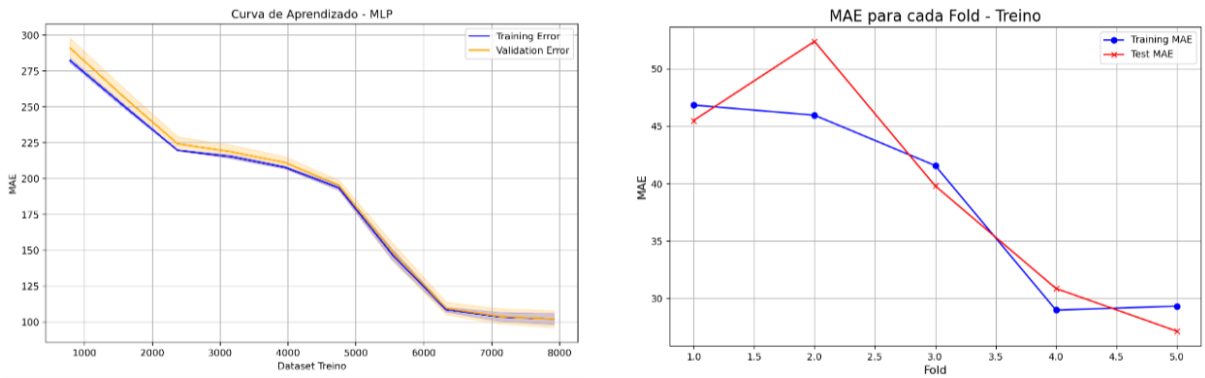


Figura 13 – Gráfico da Curva de Aprendizado do Modelo MLP e Evolução do MAE ao longo dos folds do modelo RL. RNA - Dataset Treino. (Figura 13)

5.2 Análise Comparativa dos Experimentos

Para comparar o desempenho dos modelos tradicionais de aprendizado de máquina com os baseados em redes neurais artificiais, foram coletados inicialmente os valores de MAE de cada fold de ambos os grupos de experimentos conforme apresentado nos apêndices (E.10), (E.11). Para determinar se há diferenças estatisticamente significativas entre os desempenhos dos modelos, aplicou-se o teste de Análise de Variância (ANOVA) de uma

via aos valores de MAE obtidos. O teste de ANOVA compara múltiplas médias, verificando se há variação significativa entre elas. As equações utilizadas nesse teste são detalhadas nos apêndices (D.17) e (D.18).

Ao detalhar a análise estatística dos modelos tradicionais, os resultados do teste ANOVA aplicado às métricas de MAE indicam que a estatística F foi calculada em 1125,8228. Esse valor representa a razão entre a variabilidade entre os grupos e a variabilidade dentro dos grupos, sugerindo uma diferença estatisticamente significativa entre os modelos de aprendizado de máquina analisados.

O valor- p obtido foi menor que 0.0001, muito abaixo do nível de significância de 0.05. Isso indica forte evidência de que existem diferenças reais entre os modelos, e não apenas variações ao acaso. Assim, rejeita-se a hipótese nula, concluindo que pelo menos um dos modelos tem um desempenho significativamente diferente dos demais, com base no MAE.

Como o valor- p indicou diferenças significativas entre os modelos, foi aplicado o teste post-hoc de Tukey para identificar quais deles diferem estatisticamente entre si. Esse teste ajusta os valores- p para múltiplas comparações, utilizando a correção Family-Wise Error Rate (FWER) para controlar o erro tipo I acumulado e reduzir a chance de falsos positivos ao rejeitar indevidamente a hipótese nula. A equação estatística do teste está apresentada no apêndice (D.19).

A análise do Teste de Tukey, apresentada no apêndice (E.9), mostrou que a Regressão Linear Múltipla tem diferenças estatisticamente significativas em relação aos outros modelos, enquanto a Regressão Simples teve o pior desempenho. A Regressão Polinomial obteve o menor MAE médio, indicando o melhor desempenho, mas sua diferença em relação à Regressão SVR não foi estatisticamente significativa ($p\text{-adj} = 0.2979$). Assim, apesar do menor MAE, o Teste de Tukey indica que a Regressão Polinomial e a Regressão SVR têm desempenhos estatisticamente equivalentes.

Na sequência, detalhando a análise estatística dos modelos baseados em Redes Neurais Artificiais, a aplicação do teste ANOVA às métricas de MAE resultou em uma estatística F de 2,0735 e um valor- p de 0,1685. Como o valor- p é superior ao nível de significância adotado ($\alpha = 0,05$), não há evidências estatísticas suficientes para afirmar que existem diferenças significativas entre os modelos analisados. Esse resultado sugere que nenhum dos modelos de RNA apresenta desempenho significativamente superior aos demais, indicando uma performance comparável entre eles.

Continuando, à análise estatística, foi aplicado o teste de Shapiro-Wilk para verificar a normalidade dos dados do MAE. Esse teste avalia se um conjunto de dados segue uma distribuição normal, calculando a estatística W , que compara a distribuição observada com a esperada para uma amostra normalmente distribuída. Os resultados indicaram que os dados dos modelos MLP e RNA seguem uma distribuição normal ($p\text{-valores} > 0.05$), enquanto os dados do modelo RNA Tuning não seguem uma distribuição normal ($p\text{-valor} < 0.05$). Essa informação é essencial para a escolha dos próximos testes estatísticos,

pois testes paramétricos, como o ANOVA, pressupõem normalidade dos dados, tornando os testes não paramétricos mais apropriados nesse caso. Os valores deste teste são detalhados no apêndice (E.4).

Em seguida, foram realizados testes não paramétricos para comparar os modelos. O teste de Friedman (apêndice (E.5)) indicou diferenças gerais entre os modelos, mas os testes de Wilcoxon e Nemenyi (apêndices (E.6),(E.7)) não identificaram diferenças estatisticamente significativas entre pares específicos. A análise descritiva (apêndice (E.8)) mostrou que o modelo RNA obteve a menor média de MAE (39,1328), enquanto o MLP apresentou o menor desvio padrão. Já o modelo RNA Tuning teve a maior média e desvio padrão de MAE, indicando o pior desempenho. Embora não haja diferenças estatisticamente significativas entre os modelos RNA e MLP, o modelo RNA é recomendado por apresentar a menor média de erro.

5.3 Análise Comparativa dos Modelos com Melhor Desempenho

Considerando que os modelos de Regressão Polinomial e RNA apresentaram os melhores desempenhos nos testes estatísticos, tornou-se necessário compará-los para determinar qual modelo se sobressai entre as abordagens tradicionais de aprendizado de máquina e as redes neurais artificiais. Para isso, foi realizado um teste t-pareado utilizando os valores do MAE obtidos em cada fold, permitindo uma avaliação mais precisa das diferenças entre os dois modelos.

Com base nos resultados demonstrados no apêndice (E.12), a análise estatística não encontrou evidências suficientes para rejeitar a hipótese nula, pois o valor-p (0.1399) é maior que o nível de significância de 0.05. Isso indica que não há diferença estatisticamente significativa entre os modelos comparados. No entanto, ao analisar as médias do MAE, a Regressão Polinomial apresentou um valor menor (29,9729) em relação à RNA (39,1328), sugerindo que, embora a diferença não seja estatisticamente significativa, a Regressão Polinomial demonstrou melhor desempenho, com menor erro absoluto médio.

5.4 Análise Comparativa com Trabalho Relacionado

Nesta seção, são apresentadas análises comparativas entre os resultados obtidos no presente trabalho e os do estudo de Gustavo Henrique Larsen, intitulado *“Predição de tempos de viagem em linhas de ônibus baseado em dados de tráfego e redes neurais artificiais”*², referenciado na seção (1.4).

Ao comparar a métrica RMSE obtida pelo modelo *Regressão Linear com Redes Neurais Artificiais* desenvolvido neste trabalho (56,4508) com o valor reportado por (LARSEN, 2020)² para o modelo de *Redes Neurais Artificiais* (297), observa-se que o modelo proposto neste estudo apresenta desempenho significativamente superior. Esse resultado evidencia a eficácia da abordagem adotada e sua competitividade em relação aos mode-

los encontrados na literatura. Complementando a análise, o apêndice (E.15) apresentada uma tabela com comparações adicionais entre os modelos desenvolvidos em ambos os trabalhos.

6 CONCLUSÃO

Em conclusão, os modelos de Regressão Polinomial e Redes Neurais Artificiais (RNA) foram os mais eficazes, com a Regressão Polinomial apresentando um Erro Médio Absoluto (MAE) de 29,9729 segundos, enquanto o modelo RNA obteve 39,1328 segundos. Embora a diferença entre eles não seja estatisticamente significativa, a Regressão Polinomial demonstrou menor erro absoluto médio, sugerindo ser a alternativa mais eficiente para prever a duração das viagens de ônibus urbanos em Curitiba.

A escolha do modelo mais adequado considerou não apenas o desempenho, mas também fatores como interpretabilidade e custo operacional. O modelo de Regressão Polinomial se destaca por apresentar menor MAE e demandar menos recursos computacionais, tornando-se uma opção mais eficiente. Por outro lado, o RNA pode ser mais apropriado em cenários que exigem maior flexibilidade ou lidam com padrões mais complexos nos dados, apesar de seu maior custo computacional. Assim, a decisão do melhor modelo equilibrou precisão, complexidade e viabilidade operacional.

Futuras pesquisas podem aprimorar este estudo por meio do refinamento dos modelos, com foco na otimização dos hiperparâmetros do modelo RNA e na experimentação com diferentes arquiteturas de redes neurais. A ampliação da base de dados e a inclusão de novas variáveis, como dados climáticos e de tráfego, podem proporcionar uma análise mais abrangente e aprimorar a precisão das previsões. Além disso, melhorias no algoritmo de captação de dados, como explorar novas possibilidades e rotas alternativas no Google Maps, podem aumentar a precisão das previsões. Outra linha interessante seria testar a robustez dos modelos utilizando diferentes conjuntos de dados ou variações no pré-processamento, avaliando seu impacto na previsão da duração das viagens de ônibus urbanos.

REFERÊNCIAS

- 1 GRUS, J. **Data Science do Zero: Primeiras regras com o python**. Rua Viúva Cláudio, 291 - Bairro Industrial do Jacaré, CEP: 20.970-031, Rio de Janeiro (RJ): Alta Books Editora, 2016.
- 2 LARSEN, G. H. Predição de tempos de viagem em linhas de Ônibus baseado em dados de tráfego e redes neurais artificiais. **POLI-USP**, Escola Politécnica da Universidade de São Paulo, 2020. Disponível em: <https://www.teses.usp.br/teses/disponiveis/3/3142/tde-20012021-163526/publico/GustavoHenriqueLarsenCorr20.pdf>. Acesso em: 27 jul. 2020.
- 3 MACHADO, S. d. S. B. C. Influências da inteligência artificial (ia) na otimização do tráfego urbano: um referencial teórico. **PUC**, Editora Científica Digital, v. 2, 2022. Disponível em: <https://downloads.editoracientifica.com.br/articles/220207766.pdf>. Acesso em: 2022.
- 4 CIVALI, V. de O. Sistema algorítmico de rede neural artificial para previsão da localização de ônibus de linha em quedas de sinal. **UNICAMP**, XXIX Congresso de Iniciação Científica UNICAMP, 2021. Disponível em: <https://prp.unicamp.br/inscricao-congresso/resumos/2021P17874A33759O2972.pdf>. Acesso em: 2021.
- 5 LUÍS, M. O. B. L. Desenvolvimento de sistema de inteligência artificial (ia) baseado em reconhecimento de padrões para análise de rotas veiculares. **Journal Iberamia**, IBERAMIA, v. 23, 2020. Disponível em: <https://journal.iberamia.org/index.php/intartif/article/view/339/108>. Acesso em: 2020.
- 6 FAYYAD PIATETSKY SHAPIRO, G. S. P. . U. R. U. M. **Advances in Knowledge Discovery and Data Mining**. United States (CA): AAAI Press, 1996. The Mit Press.
- 7 GÉRON, A. **Mãos à Obra: Aprendizado de Máquina com Scikit-Learn TensorFlow**: Conceitos, ferramentas e técnicas para a construção de sistemas inteligentes. Rio de Janeiro (RJ): Alta Books, 2019.
- 8 SILVA SARAJANE MARQUES PERES, C. B. Leandro Augusto da. **Introdução à Mineração de Dados**: Com aplicações em r. Rua Sete de Setembro, 111 – 16o andar 20050-006 – Centro – Rio de Janeiro (RJ) - Brasil: Elsevier Editora Ltda., 2016.
- 9 PARANÁ, D. N. da S. **Física para o Ensino Médio**: Volume Único. Rua Barão de Iguape, 110 - CEP 01507-900 - São Paulo (SP): Editora Ática, 2003.
- 10 LAROSE, C. D. L. D. T. **Discovering Knowledge in Data: An Introduction to Data Mining**. [S.l.]: John Wiley Sons, 2014. Inc. All rights reserved.
- 11 LUÍS, M. O. B. L. Desenvolvimento de sistema de inteligência artificial (ia) baseado em reconhecimento de padrões para análise de rotas veiculares. **Journal Iberamia**, Universidade Regional do Noroeste do Estado do Rio Grande do Sul, 2020. Disponível

em: <https://journal.iberamia.org/index.php/intartif/article/view/339/108>. Acesso em: 2020.

1 7 8 9 4 2 3 6 10 11

APÊNDICE A – CÓDIGOS

A.1 Código Python - Coleta de Dados e Pré-processamento

```
import pandas as pd
import requests
import lzma
import json

"""
    Função reutilizável para carregamento dos dados.
"""
def load_dataset(path, file_type):
    if file_type == 'json':
        return pd.read_json(path)
    elif file_type == 'csv':
        return pd.read_csv(path)
    else:
        return print('Erro ao carregar os dados')

def rename_features(dataset, renamed_columns, inplace, logs):
    if logs:
        print('-'*100)
        print('Renomeando features')
    """
    Parâmetro inplace modifica o dataframe ou series diretamente, sem criar um novo objeto.
    As alterações são efetivadas diretamente no dataframe ou series em memória.
    """
    return dataset.rename(columns=renamed_columns, inplace=inplace)

def remove_features(dataset, features, inplace, logs):
    if logs:
        print('-'*100)
        print('Removendo features do dataset')

    return dataset.drop(columns=features, inplace=inplace)

def validate_numeric_registers(dataset, feature, logs):
    if logs:
        print('-'*100)
        print('Verificando se existem registros não numéricos')
        print(f'Total de registros: {len(dataset[feature])}')

    """
    Parâmetro errors especifica como tratar erros que ocorrem durante a conversão.
    Quando errors='coerce' é definido, qualquer valor que não possa ser convertido em um valor numérico
    o tipo será substituído por NaN (não é um número).
    """
```

```

"""
numeric_data = pd.to_numeric(dataset[feature], errors='coerce')
non_integers = dataset[feature][numeric_data.isna()]

if logs:
    print(f'Total de registros não numéricos: {len(non_integers)}')
    print(f'Total de registros numéricos: {len(numeric_data) - len(non_integers)}')

    if len(non_integers) > 0:
        if logs:
            print('Lista registros não numéricos')
            print(non_integers)
        else:
            if logs:
                print('Não existem registros não numéricos')

def clean_non_numeric_registers(dataset, feature, logs):
    if logs:
        print('-'*100)
        print('Limando string X de registros não numéricos')
    return [str(item)[1:] if str(item).startswith('X') else item for item in dataset[feature]]

def convert_object_to_int(dataset, feature, logs):
    if logs:
        print('-'*100)
        print('Convertendo tipo de object para int')
    return pd.to_numeric(dataset[feature], errors='coerce').astype(int)

def checks_duplicate_registers(dataset, feature, logs):
    if logs:
        print('-'*100)
        print('Verificando se existem registros duplicados na feature line_id')
    duplicates = dataset[dataset[feature].duplicated()]

    if logs:
        print(f'Existem {len(duplicates)} registros duplicados')

def show_duplicate_registers(dataset, feature, logs):
    if logs:
        print('-'*100)
        print('Mostrando registros duplicados na feature line_id')
    duplicates = dataset[dataset[feature].duplicated()]

    if logs:
        if len(duplicates) > 0:
            print(duplicates['line_id'])
        else:
            print('Não existem registros duplicados a serem exibidos')

def sequencing_registers(dataset, feature, logs):
    if logs:
        print('-'*100)
        print('Sequenciando registros pela feature line_id')
    return dataset.sort_values(by=feature).reset_index(drop=True)

def remove_duplicate_registers(dataset, feature, logs):
    if logs:
        print('-'*100)
        print('Removendo registros duplicados')
    return dataset.drop_duplicates(subset=[feature])

```

```

# Coleta dos Dados

# 1-31 2023 01
# 5-30 2024 06
INITIAL_DATE = 5
FINAL_DATE = 30
YEAR = '2024'
MONTH = '06'
days_list = [f"{num:02}" for num in list(range(INITIAL_DATE, FINAL_DATE + 1))]
PATH_ORIGINALS_DATASETS = '/media/marcos/500GB/00_datasets/originals/'
PATH_INITIAL_PREPROCESSING_DATASETS = '/media/marcos/500GB/00_datasets/initial_preprocessing/'
PATH_CONCATENATED_DATASETS = '/media/marcos/500GB/00_datasets/concatenated/'
PATH_JOINED_DATASETS = '/media/marcos/500GB/00_datasets/joined/'

for day in days_list:
    print(f'Iniciando download, extração e salvamento dos datasets - data: {YEAR}/{MONTH}/{day}')
    list_datasets_names = [
        f'{YEAR}_{MONTH}_{day}_linhas',
        f'{YEAR}_{MONTH}_{day}_pontosLinha',
        f'{YEAR}_{MONTH}_{day}_tabelaLinha',
        f'{YEAR}_{MONTH}_{day}_trechosItinerarios',
        f'{YEAR}_{MONTH}_{day}_tabelaVeiculo'
    ]

    for dataset_name in list_datasets_names:
        url = f"https://dadosabertos.c3sl.ufpr.br/curitiba/urbs/{dataset_name}.json.xz"
        response = requests.get(url)

        if response.status_code == 200:
            # Extraindo (descomprimindo) conteúdo do response
            decompressed_data = lzma.decompress(response.content)

            # Carregando conteúdo extraído em formato JSON
            data = json.loads(decompressed_data.decode('utf-8'))

            # Salvando arquivo do dataset, formato JSON, em diretório local
            with open(f'{PATH_ORIGINALS_DATASETS}{dataset_name}.json', 'w', encoding='utf-8') as json_file:
                json.dump(data, json_file, indent=4)
            # print(f"Dataset: {dataset_name} extraído e salvo com sucesso!")
        else:
            print(f"Falha ao extrair e salvar dataset. Status code: {response.status_code}")
    print(f'Finalização de download, extração e salvamento dos datasets - data: {YEAR}/{MONTH}/{day}')
    print('-'*100)

# Etapas de pré-processamento

for day in days_list:
    CAPTURE_DATE = f'{YEAR}_{MONTH}_{day}'

    print('Carregando datasets')

    path = f"{PATH_ORIGINALS_DATASETS}{CAPTURE_DATE}_"

    linhas_df = load_dataset(f'{path}linhas.json', 'json')
    pontos_linha_df = load_dataset(f'{path}pontosLinha.json', 'json')
    veiculos_df = load_dataset(f'{path}tabelaVeiculo.json', 'json')
    tabela_linha_df = load_dataset(f'{path}tabelaLinha.json', 'json')

```

```

trechos_itinerarios_df = load_dataset(f'{path}trechosItinerarios.json', 'json')

print('Datasets carregados com sucesso!')

print('--'*60)

line_df = linhas_df.copy()

line_df_renamed_columns = {
    'COD': 'line_id',
    'NOME': 'line_name',
    'NOME_COR': 'line_color',
    'CATEGORIA_SERVICO': 'line_category',
    'SOMENTE_CARTAO': 'line_payment_method'
}
print(f'Iniciando pré-processamento dataset line_df - data: {CAPTURE_DATE}')
rename_features(line_df, line_df_renamed_columns, True, False)
validate_numeric_registers(line_df, 'line_id', False)
line_df['line_id'] = clean_non_numeric_registers(line_df, 'line_id', False)
validate_numeric_registers(line_df, 'line_id', False)
line_df['line_id'] = convert_object_to_int(line_df, 'line_id', False)
line_df = sequencing_registers(line_df, 'line_id', False)
checks_duplicate_registers(line_df, 'line_id', False)
show_duplicate_registers(line_df, 'line_id', False)
line_df = remove_duplicate_registers(line_df, 'line_id', False)
checks_duplicate_registers(line_df, 'line_id', False)
print(f'Finalizando pré-processamento dataset line_df - data: {CAPTURE_DATE}')

print('--'*60)

station_df = pontos_linha_df.copy()

station_df_renamed_columns = {
    'NOME': 'station_name',
    'NUM': 'station_code',
    'LAT': 'station_initial_latitude',
    'LON': 'station_initial_longitude',
    'SEQ': 'station_sequence',
    'SENTIDO': 'station_direction',
    'TIPO': 'station_type',
    'COD': 'line_id',
}
print(f'Iniciando pré-processamento dataset station_df - data: {CAPTURE_DATE}')
rename_features(station_df, station_df_renamed_columns, True, False)
remove_features(station_df, ['GRUPO', 'ITINERARY_ID'], True, False)
validate_numeric_registers(station_df, 'line_id', False)
station_df['line_id'] = clean_non_numeric_registers(station_df, 'line_id', False)
validate_numeric_registers(station_df, 'line_id', False)
station_df['line_id'] = convert_object_to_int(station_df, 'line_id', False)
station_df = sequencing_registers(station_df, 'line_id', False)
checks_duplicate_registers(station_df, 'line_id', False)
show_duplicate_registers(station_df, 'line_id', False)
station_df = remove_duplicate_registers(station_df, 'line_id', False)
checks_duplicate_registers(station_df, 'line_id', False)
print(f'Finalizando pré-processamento dataset station_df - data: {CAPTURE_DATE}')

print('--'*60)

vehicle_df = veiculos_df.copy()

```

```

vehicle_df_renamed_columns = {
    'VEICULO': 'vehicle_plate',
    'HORARIO': 'vehicle_time',
    'COD_LINHA': 'line_id'
}

vehicle_df_columns_to_remove = ['NOME_LINHA', 'TABELA', 'COD_PONTO']

print(f'Iniciando pré-processamento dataset vehicle_df - data: {CAPTURE_DATE}')
rename_features(vehicle_df, vehicle_df_renamed_columns, True, False)
remove_features(vehicle_df, vehicle_df_columns_to_remove, True, False)
validate_numeric_registers(vehicle_df, 'line_id', False)
vehicle_df['line_id'] = clean_non_numeric_registers(vehicle_df, 'line_id', False)
validate_numeric_registers(vehicle_df, 'line_id', False)
vehicle_df['line_id'] = convert_object_to_int(vehicle_df, 'line_id', False)
vehicle_df = sequencing_registers(vehicle_df, 'line_id', False)
checks_duplicate_registers(vehicle_df, 'line_id', False)
show_duplicate_registers(vehicle_df, 'line_id', False)
vehicle_df = remove_duplicate_registers(vehicle_df, 'line_id', False)
checks_duplicate_registers(vehicle_df, 'line_id', False)
print(f'Finalizando pré-processamento dataset vehicle_df - data: {CAPTURE_DATE}')

print('--'*60)

accessibility_info_df = tabela_linha_df.copy()

accessibility_info_df_renamed_columns = {
    'ADAPT': 'accessibility_info_eletric_ramp',
    'HORA': 'accessibility_info_time',
    'DIA': 'accessibility_info_day',
    'COD': 'line_id'
}

accessibility_info_df_columns_to_remove = ['PONTO', 'NUM', 'TABELA']

print(f'Iniciando pré-processamento dataset accessibility_info_df - data: {CAPTURE_DATE}')
rename_features(accessibility_info_df, accessibility_info_df_renamed_columns, True, False)
remove_features(accessibility_info_df, accessibility_info_df_columns_to_remove, True, False)
validate_numeric_registers(accessibility_info_df, 'line_id', False)
accessibility_info_df['line_id'] = clean_non_numeric_registers(accessibility_info_df, 'line_id', False)
validate_numeric_registers(accessibility_info_df, 'line_id', False)
accessibility_info_df['line_id'] = convert_object_to_int(accessibility_info_df, 'line_id', False)
accessibility_info_df = sequencing_registers(accessibility_info_df, 'line_id', False)
checks_duplicate_registers(accessibility_info_df, 'line_id', False)
show_duplicate_registers(accessibility_info_df, 'line_id', False)
accessibility_info_df = remove_duplicate_registers(accessibility_info_df, 'line_id', False)
checks_duplicate_registers(accessibility_info_df, 'line_id', False)
print(f'Finalizando pré-processamento dataset accessibility_info_df - data: {CAPTURE_DATE}')

print('--'*60)

itinerary_df = trechos_itinerarios_df.copy()

itinerary_df_renamed_columns = {
    'NOME_EMPRESA': 'itinerary_company_name',
    'COD_ITINERARIO': 'itinerary_code',
    'PTO_ESPECIAL': 'itinerary_special_line',
    'COD_LINHA': 'line_id',
}

```

```

itinerary_df_columns_to_remove = [
    'NOME_LINHA',
    'COD_CATEGORIA',
    'NOME_CATEGORIA',
    'COD_EMPRESA',
    'COD_PTO_PARADA_TH',
    'NOME_PTO_PARADA_TH',
    'NOME_PTO_ABREVIADO',
    'SEQ_PTO_ITI_TH',
    'NOME_ITINERARIO',
    'COD_PTO_TRECHO_A',
    'SEQ_PONTO_TRECHO_A',
    'COD_PTO_TRECHO_B',
    'SEQ_PONTO_TRECHO_B',
    'EXTENSAO_TRECHO_A_ATE_B',
    'TIPO_TRECHO',
    'STOP_CODE',
    'STOP_NAME',
    'CODIGO_URBS'
]

print(f'Iniciando pre-processamento dataset itinerary_df - data: {CAPTURE_DATE}')
rename_features(itinerary_df, itinerary_df_renamed_columns, True, False)
remove_features(itinerary_df, itinerary_df_columns_to_remove, True, False)
validate_numeric_registers(itinerary_df, 'line_id', False)
itinerary_df['line_id'] = clean_non_numeric_registers(itinerary_df, 'line_id', False)
validate_numeric_registers(itinerary_df, 'line_id', False)
itinerary_df['line_id'] = convert_object_to_int(itinerary_df, 'line_id', False)
itinerary_df = sequencing_registers(itinerary_df, 'line_id', False)
checks_duplicate_registers(itinerary_df, 'line_id', False)
show_duplicate_registers(itinerary_df, 'line_id', False)
itinerary_df = remove_duplicate_registers(itinerary_df, 'line_id', False)
checks_duplicate_registers(itinerary_df, 'line_id', False)
print(f'Finalizando pre-processamento dataset itinerary_df - data: {CAPTURE_DATE}')

print('--'*60)

print(f'Iniciando merge de todos datasets inicialmente pre-processados - data: {CAPTURE_DATE}')
line_station_df = pd.merge(line_df, station_df, on='line_id')
line_station_vehicle_df = pd.merge(line_station_df, vehicle_df, on='line_id')
line_station_vehicle_accessibility_info_df = pd.merge(line_station_vehicle_df,
accessibility_info_df, on='line_id')
line_station_vehicle_accessibility_info_itinerary_df = pd.merge(
line_station_vehicle_accessibility_info_df, itinerary_df, on='line_id')
df = line_station_vehicle_accessibility_info_itinerary_df
print(f'Merge finalizado - data: {CAPTURE_DATE}')

print('--'*60)

print(f'Salvando dataset - data: {YEAR}_{MONTH}_{day}')
dataset_file_name = f'merged_dataset_{YEAR}_{MONTH}_{day}.csv'
df.to_csv(f'{PATH_INITIAL_PREPROCESSING_DATASETS}{dataset_file_name}', index=False)
print('Dataset salvo com sucesso')

# Concatenando datasets mergiados

df_list = []
for day in days_list:
    df = pd.read_csv(f'{PATH_INITIAL_PREPROCESSING_DATASETS}merged_dataset_{YEAR}_{MONTH}_{day}_

```

```

csv')
df_list.append(df)

df_concat = pd.concat(df_list, ignore_index=True)
df_concat.to_csv(f'{PATH_CONCATENATED_DATASETS}concatenated_dataset_{YEAR}_{MONTH}.csv', index=False)

# Reunindo datasets pré-processados, mergiados e concatenados visando a geração de um único dataset
.

df_concatenated_dataset_list = []
month_list = [
    {'year': '2023', 'month': '01'},
    {'year': '2024', 'month': '06'}
]

for month in month_list:
    df_concatenated = pd.read_csv(f'{PATH_CONCATENATED_DATASETS}concatenated_dataset_{month["year"]}_{month["month"]}.csv')
    df_concatenated_dataset_list.append(df_concatenated)

df_joined = pd.concat(df_concatenated_dataset_list, ignore_index=True)
df_joined.to_csv(f'{PATH_JOINED_DATASETS}joined_dataset.csv', index=False)

df = load_dataset(f'{PATH_JOINED_DATASETS}joined_dataset.csv', 'csv')

```

A.2 Código Python - Integração com Google Maps

```

# Importação de bibliotecas relacionadas ao tratamento dos dados
import pandas as pd
import numpy as np
import requests

# Necessária a instalação desta bibliotecas relacionadas a api google maps
# !pip install gmaps
# !pip install -U googlemaps

# Importação de bibliotecas relacionadas ao google maps
from google.colab import output
output.enable_custom_widget_manager()
import gmaps
import gmaps.datasets
import gmaps.geojson_geometries
from googlemaps import Client
import ipywidgets as widgets
from IPython.display import display

# Função de carregamento do dataset
def load_dataset(file):
    path = f'/content/drive/MyDrive/projeto_final_ia/datasets/{file}.csv'
    return pd.read_csv(path)

def save_dataset(dataset, file):
    path = f'/content/drive/MyDrive/projeto_final_ia/datasets/{file}.csv'
    return dataset.to_csv(path, index=False)

# Carregando dataset
df = load_dataset('joined_dataset')

```

```

# Sequenciando dataset pela feature station_sequence
df.sort_values('station_sequence', ascending=True, inplace=True)

# Convertendo tipo object de features relacionadas a geolocalização para float
df['station_initial_latitude'] = df['station_initial_latitude'].str.replace(',', '.')
df['station_initial_latitude'] = df['station_initial_latitude'].astype(float)

df['station_initial_longitude'] = df['station_initial_longitude'].str.replace(',', '.')
df['station_initial_longitude'] = df['station_initial_longitude'].astype(float)

# Geração das features station_final_latitude e station_final_longitude
df['station_final_latitude'] = df['station_initial_latitude'].shift(-1)
df['station_final_longitude'] = df['station_initial_longitude'].shift(-1)

# Integração com api google maps, visando gerar duas novas features: distance e duration
google_api_key = ''

initial_latitude = df['station_initial_latitude'].tolist()
initial_longitude = df['station_initial_longitude'].tolist()
final_latitude = df['station_final_latitude'].tolist()
final_longitude = df['station_final_longitude'].tolist()

distance = []
duration = []
path_api = 'https://maps.googleapis.com/maps/api/distancematrix/json?units=metric&origins='

for final_lon in range(len(final_longitude)):
    url = f'{path_api}{initial_latitude[final_lon]},{initial_longitude[final_lon]}&destinations={final_latitude[final_lon]}%2C{final_longitude[final_lon]}&key={google_api_key}'
    response = requests.get(url)
    data = response.json()
    try:
        distance.append(data['rows'][0]['elements'][0]['distance']['text'])
        duration.append(data['rows'][0]['elements'][0]['duration']['text'])
    except KeyError as e:
        print(f"KeyError: Missing key {e} in the API response for index {final_lon}")
    except Exception as e:
        print(f"An error occurred: {e}")

# Adicionando valor de distance e duration no último registro
distance.append('0 km')
duration.append('0 mins')

# Removendo string 'm' e 'km' dos valores de distância e convertendo os valores que utilizam
# unidade de medida km para metros
formatted_distance = []
for i in range(len(distance)):
    if distance[i].endswith(' m'):
        modified_item = float(distance[i].rstrip(' m'))
        formatted_distance.append(modified_item)
    else:
        modified_item_km = float(distance[i].rstrip(' km')) * 1000
        formatted_distance.append(modified_item_km)

df['distance'] = formatted_distance

# Removendo string 'mins' e min dos valores de duration e convertendo valores de unidade de medida
# minuto para segundos
formatted_duration = []

```

```
for i in range(len(duration)):
    if (duration[i] != '1 min'):
        formatted_duration.append(float((duration[i].replace(' mins', ''))) * 60)
    else:
        formatted_duration.append(float((duration[i].replace(' min', ''))) * 60)
df['duration'] = formatted_duration

# Salvando dataset
save_dataset(df, 'dataset_integrated_gmaps')

# Testando carregamento do dataset
df_integrated = load_dataset('dataset_integrated_gmaps')

df_integrated.head()

# Montando uma lista com alguns registros relacionados a geolocalização
features = [
    'station_initial_latitude',
    'station_initial_longitude',
    'station_final_latitude',
    'station_final_longitude'
]

features_values = df_integrated[features].iloc[1:10]

list_features_values = features_values.values.tolist()

# Testando visualização com google maps

# Configurando cliente da API do Google Maps
gmaps.configure(api_key=google_api_key)
gmaps_url = "https://maps.googleapis.com/maps/api/js?key=" + google_api_key
maps = []
for index, item in enumerate(list_features_values):

    # Coordenadas iniciais e finais para o ponto A e o ponto B
    INITIAL_LATITUDE = item[0]
    INITIAL_LONGITUDE = item[1]
    FINAL_LATITUDE = item[2]
    FINAL_LONGITUDE = item[3]

    coordinates_A = (INITIAL_LATITUDE, INITIAL_LONGITUDE)
    coordinates_B = (FINAL_LATITUDE, FINAL_LONGITUDE)

    # Criando mapa centrado entre o ponto A e o ponto B
    fig = gmaps.figure(center=(coordinates_A[0], coordinates_A[1]), zoom_level=16, layout={ 'width':
        '600px', 'height': '400px' })

    # Adicionando camada de trajetória entre os pontos A e B
    fig.add_layer(gmaps.directions.Directions(coordinates_A, coordinates_B, mode='driving'))

    maps.append(fig)

box_layout = widgets.Layout(border='4px', padding='4px')

# Exibindo mapas das rotas
row1 = widgets.HBox(maps[:5], layout=box_layout)
row2 = widgets.HBox(maps[5:], layout=box_layout)

display(widgets.VBox([row1, row2]))
```

```
list_features_values_max_dist = [[-25.358615, -49.241901, -25.596019, -49.274454]]
list_features_values_min_dist = [[-25.435052, -49.275876, -25.436659, -49.273807]]
```

A.3 Código Python - Testes Estatísticos

```
# Testes modelos tradicionais

from scipy.stats import f_oneway
from statsmodels.stats.multicomp import pairwise_tukeyhsd
import numpy as np

# MAE para cada fold de cada modelo
mae_rl_simples = [217.8948, 221.6355, 214.4085, 215.8374, 238.0413]
mae_rl_multipla = [116.9502, 112.8189, 107.9391, 104.8629, 120.2038]
mae_rl_polynomial = [29.8763, 29.1873, 28.9052, 30.8245, 31.0714]
mae_rl_svr = [37.5122, 34.5338, 41.4880, 34.4905, 36.1296]

# Realizando o teste ANOVA
f_stat, p_value = f_oneway(mae_rl_simples, mae_rl_multipla, mae_rl_polynomial, mae_rl_svr)

print(f"Estatística F: {f_stat:.4f}")
print(f"Valor-p: {p_value:.4f}")

# Interpretação do resultado
if p_value < 0.05:
    print("Rejeitamos a hipótese nula. Pelo menos um modelo é significativamente diferente.")
else:
    print("Não rejeitamos a hipótese nula. Não há diferença significativa entre os modelos.")

# Agrupando os dados para o teste de Tukey
mae_values = np.concatenate([mae_rl_simples, mae_rl_multipla, mae_rl_polynomial, mae_rl_svr])
models = ['RL Simples'] * len(mae_rl_simples) + ['RL Múltipla'] * len(mae_rl_multipla) + ['RL Polinomial'] * len(mae_rl_polynomial) + ['RL SVR'] * len(mae_rl_svr)

# Realizando o teste de Tukey
tukey_result = pairwise_tukeyhsd(mae_values, models, alpha=0.05)

# Exibindo os resultados
tukey_result.summary()

# Testes modelos baseados em RNA

# !pip install scikit-posthocs
from scipy.stats import f_oneway, shapiro, kruskal, wilcoxon
import scikit_posthocs as sp
from statsmodels.stats.multicomp import pairwise_tukeyhsd
import numpy as np

# MAE para cada fold de cada modelo
mae_mlp = [42.0543, 45.2809, 45.2541, 43.4433, 41.8344]
mae_rna = [45.5012, 52.3862, 39.7903, 30.8527, 27.1338]
mae_rna_tuning = [49.9603, 114.8611, 47.4836, 49.3284, 45.4279]

# Realizando o teste ANOVA
f_stat, p_value = f_oneway(mae_mlp, mae_rna, mae_rna_tuning)
```

```
print(f"Estatística F: {f_stat:.4f}")
print(f"Valor-p: {p_value:.4f}")

# Interpretação do resultado
if p_value < 0.05:
    print("Rejeitamos a hipótese nula. Pelo menos um modelo é significativamente diferente.")
else:
    print("Não rejeitamos a hipótese nula. Não há diferença significativa entre os modelos.")

# Teste de Shapiro-Wilk para cada modelo
stat_mlp, p_mlp = shapiro(mae_mlp)
stat_rna, p_rna = shapiro(mae_rna)
stat_rna_tuning, p_rna_tuning = shapiro(mae_rna_tuning)

# Exibir resultados
print(f"MLP: Estatística W = {stat_mlp:.4f}, p-valor = {p_mlp:.4f}")
print(f"RNA: Estatística W = {stat_rna:.4f}, p-valor = {p_rna:.4f}")
print(f"RNA Tuning: Estatística W = {stat_rna_tuning:.4f}, p-valor = {p_rna_tuning:.4f}")

# Interpretação
alpha = 0.05 # Nível de significância
if p_mlp < alpha:
    print("MLP: Os dados NÃO seguem uma distribuição normal.")
else:
    print("MLP: Os dados seguem uma distribuição normal.")

if p_rna < alpha:
    print("RNA: Os dados NÃO seguem uma distribuição normal.")
else:
    print("RNA: Os dados seguem uma distribuição normal.")

if p_rna_tuning < alpha:
    print("RNA Tuning: Os dados NÃO seguem uma distribuição normal.")
else:
    print("RNA Tuning: Os dados seguem uma distribuição normal.")

# Teste de Kruskal-Wallis
h_stat, p_kruskal = kruskal(mae_mlp, mae_rna, mae_rna_tuning)
print(f"Estatística H: {h_stat:.4f}")
print(f"Valor-p: {p_kruskal:.4f}")

# Se p < 0.05, há diferença significativa      Aplicar teste de Dunn
if p_kruskal < 0.05:
    print("Há diferença significativa entre os modelos. Aplicando Dunn...")

    # Criar tabela para o teste de Dunn
    all_data = [mae_mlp, mae_rna, mae_rna_tuning]
    dunn_result = sp.posthoc_dunn(all_data, p_adjust="bonferroni")

    print(dunn_result)
else:
    print("Não há diferença significativa entre os modelos.")

stat, p_value = wilcoxon(mae_mlp, mae_rna)

print(f"Estatística de Wilcoxon: {stat:.4f}")
print(f"Valor-p: {p_value:.4f}")

# Interpretação do resultado
if p_value < 0.05:
```

```

    print("Há diferença significativa entre os modelos.")
else:
    print("Não há diferença significativa entre os modelos.")

# Comparação entre os modelos
print("MLP vs RNA:")
print(wilcoxon(mae_mlp, mae_rna))

print("\nMLP vs RNA Tuning:")
print(wilcoxon(mae_mlp, mae_rna_tuning))

print("\nRNA vs RNA Tuning:")
print(wilcoxon(mae_rna, mae_rna_tuning))

import pandas as pd
import scipy.stats as stats
import statsmodels.sandbox.stats.multicomp as multi
import numpy as np

data = {
    'MLP': [42.0543, 45.2809, 45.2541, 43.4433, 41.8344],
    'RNA': [45.5012, 52.3862, 39.7903, 30.8527, 27.1338],
    'RNA Tuning': [49.9603, 114.8611, 47.4836, 49.3284, 45.4279]
}

df = pd.DataFrame(data)

# Teste de Friedman
stat, p = stats.friedmanchisquare(df['MLP'], df['RNA'], df['RNA Tuning'])
print(f"Friedman: stat={stat:.3f}, p={p:.4f}")

# Testes de Wilcoxon Pairwise
print("\nWilcoxon Pairwise:")
for col1 in df.columns:
    for col2 in df.columns:
        if col1 != col2:
            stat, p = stats.wilcoxon(df[col1], df[col2])
            print(f"{col1} vs {col2}: stat={stat:.3f}, p={p:.4f}")

# Análise Descritiva
print("\nAnálise Descritiva:")
print(df.mean())
print(df.std())

# Teste de Nemenyi (post-hoc para Friedman) – Corrigido
print("\nNemenyi Test:")
data_values = df.values.flatten() # Flatten the DataFrame values
groups = np.repeat(df.columns, len(df)) # Create groups for each value
m_comp = multi.MultiComparison(data_values, groups)
result = m_comp.tukeyhsd()
print(result)

# Teste comparando os melhores modelos

from scipy.stats import ttest_rel
import numpy as np

mae_rna = [45.5012, 52.3862, 39.7903, 30.8527, 27.1338]
mae_rl_polynomial = [29.8763, 29.1873, 28.9052, 30.8245, 31.0714]

```

```
t_stat, p_value = ttest_rel(mae_rna, mae_rl_polynomial)

print(f"Estatística t: {t_stat:.4f}")
print(f"Valor-p: {p_value:.4f}")

# Interpretando o resultado
alpha = 0.05 # Nível de significância de 5%
if p_value < alpha:
    print("Rejeitamos a hipótese nula: Os modelos têm desempenhos significativamente diferentes.")
else:
    print("Não rejeitamos a hipótese nula: Não há evidência de diferença significativa entre os modelos.")

# Cálculo da média do MAE para cada modelo
mean_mae_rna = np.mean(mae_rna)
mean_mae_rl_polynomial = np.mean(mae_rl_polynomial)

print(f"Média MAE RNA: {mean_mae_rna:.4f}")
print(f"Média MAE RL Polynomial: {mean_mae_rl_polynomial:.4f}")

if mean_mae_rna < mean_mae_rl_polynomial:
    print("A RNA tem menor MAE médio, portanto, é melhor em termos de erro absoluto médio.")
else:
    print("A Regressão Polinomial tem menor MAE médio, portanto, é melhor em termos de erro absoluto médio.")
```

A.4 Repositórios de Códigos Abertos

Para detalhamento e reprodução dos resultados dos experimentos desse trabalho, os códigos podem ser acessados pelo link: [Projeto TCC IA Aplicada](#).

APÊNDICE B – DICIONÁRIO DE DADOS

B.5 Dicionário de Dados Brutos

Arquivos de Dados Brutos	
linhas.json	shapeLinha.json
pois.json	tabelaLinha.json
pontosLinha.json	tabelaVeiculo.json
trechosItinerarios.json	veiculo.json

Tabela B.1

Arquivos de Dados Brutos Seleccionados
linhas.json
pontosLinha.json
tabelaVeiculo.json
tabelaLinha.json
trechosItinerarios.json

Tabela B.2

linhas	tabela_veiculo	tabela_linha	pontos_linha
cod nome somente_cartao categoria_servico nome_cor	cod_linha nome_linha veiculo horario tabela cod_ponto	hora ponto dia num tabela adapt cod	nome num lat lon seq grupo sentido tipo itinerary_id cod

Tabela B.3

B.6 Dicionário de Dados Remodelado

trechos_itinerarios
cod_linha
nome_linha
cod_categoria
nome_categoria
cod_empresa
nome_empresa
cod_pto_parada_th
nome_pto_parada_th
nome_pto_abreviado
seq_pto_iti_th
cod_itinerario
nome_itinerario
pto_especial
cod_pto_trecho_a
seq_pto_trecho_a
cod_pto_trecho_b
seq_pto_trecho_b
extensao_trecho_a_ate_b
tipo_trecho
stop_code
stop_name
codigo_urbs

Tabela B.4

Dicionário de dados: dataset linhas (292 registros)		
Atributo	Tipo	Descrição
COD	object	Código identificador da linha de ônibus.
NOME	object	Nome identificador da linha de ônibus.
SOMENTE_CARTAO	object	Método de pagamento da linha de ônibus. Identifica como: S: recebe pagamento através de cartão e dinheiro; N: não recebe pagamento através de cartão.
CATEGORIA_SERVICO	object	Identifica categoria de serviço da linha de ônibus.
NOME_COR	object	Identifica a cor do ônibus.

Tabela B.5

Dicionário de dados: dataset tabelaVeiculo (16326 registros)		
Atributo	Tipo	Descrição
COD_LINHA	object	Código identificador da linha de ônibus.
NOME_LINHA	object	Nome identificador da linha de ônibus.
VEICULO	object	Placa de identificação do veículo (Ônibus).
HORARIO	object	Horário de circulação do veículo (Ônibus).
TABELA	object	Número identificador da tabela.
COD_PONTO	object	Código identificador do ponto da linha de ônibus.

Tabela B.6

Dicionário de dados: dataset tabela_linha (62689 registros)		
Atributo	Tipo	Descrição
HORA	object	Identifica o horário da linha de ônibus.
PONTO	object	Nome do ponto relacionado à linha de ônibus.
DIA	int64	Dia da coleta dos dados.
NUM	int64	Número identificador da linha.
TABELA	object	Número identificador da tabela.
ADAPT	object	Tipo de acessibilidade da linha de ônibus (RAMPA ELÉTRICA ou ELEVADOR).
COD	object	Código identificador da linha de ônibus.

Tabela B.7

Dicionário de dados: dataset pontosLinha (18643 registros)		
Atributo	Tipo	Descrição
NOME	object	Nome do ponto da linha de ônibus.
NUM	int64	Número identificador do ponto.
LAT	object	Latitude inicial do ponto.
LON	object	Longitude inicial do ponto.
SEQ	int64	Número identificador da sequência do ponto.
GRUPO	object	Nome identificador do grupo no qual o ponto pertence.
SENTIDO	object	Nome do sentido do ponto ou direção.
TIPO	object	Nome identificador do tipo do ponto.
ITINERARY_ID	int64	Número identificador do itinerário relacionado ao ponto.
COD	object	Código identificador da linha de ônibus.

Tabela B.8

Dicionário de dados: dataset trechosItinerarios (1443 registros)		
Atributo	Tipo	Descrição
COD_LINHA	object	Código identificador da linha de ônibus.
NOME_LINHA	object	Nome identificador da linha de ônibus.
COD_CATEGORIA	int64	Código identificador da categoria da linha de ônibus.
NOME_CATEGORIA	object	Nome identificador da categoria da linha de ônibus.
COD_EMPRESA	int64	Código identificador da empresa da linha de ônibus.
NOME_EMPRESA	object	Nome identificador da empresa da linha de ônibus.
COD_PTO_PARADA_TH	int64	Código identificador ponto de parada.
NOME_PTO_PARADA_TH	object	Nome identificador ponto de parada.
NOME_PTO_ABREVIADO	object	Nome abreviado do ponto.
SEQ_PTO_ITI_TH	int64	Número de sequência do ponto relacionado ao itinerário.
COD_ITINERARIO	int64	Código identificador do itinerário.
NOME_ITINERARIO	object	Nome identificador do itinerário.
PTD_ESPECIAL	object	Identifica se o ponto tem acessibilidade ("S"ou "N").
COD_PTO_TRECHO_A	int64	Código identificador do ponto ou trecho "A".
SEQ_PTO_TRECHO_A	int64	Número de sequência do ponto ou trecho "A".
COD_PTO_TRECHO_B	int64	Código identificador do ponto ou trecho "B".
SEQ_PTO_TRECHO_B	int64	Número de sequência do ponto ou trecho "B".
EXTENSAO_TRECHO_A_ATE_B	object	Valor identificador do trecho "A"até "B".
TIPO_TRECHO	object	Identificador do tipo de trecho: T: Trecho de ida; E: Trecho de volta.
STOP_CODE	int64	Código identificador do ponto de parada.
STOP_NAME	object	Nome identificador do ponto de parada.
CODIGO_URBS	int64	Número de identificação atribuído pela Urbanização de Curitiba S/A (URBS).

Tabela B.9

line	
Atributo	Descrição
line_id	Identificador único da linha
line_name	Nome da linha
line_color	Cor da linha
line_category	Categoria da linha
line_payment_method	Método de pagamento da linha

Tabela B.10

station	
Atributo	Descrição
station_name	Nome da estação
station_code	Código da estação
station_initial_latitude	Latitude inicial da estação
station_initial_longitude	Longitude inicial da estação
station_sequence	Sequência da estação
station_direction	Direção da estação
station_type	Tipo da estação
line_id	Identificador da linha

Tabela B.11

vehicle	
Atributo	Descrição
vehicle_plate	Placa do veículo
vehicle_time	Horário do veículo
line_id	Identificador da linha

Tabela B.12

accessibility_info	
Atributo	Descrição
accessibility_electric_ramp	Disponibilidade de rampa elétrica
accessibility_time	Tempo de acessibilidade
accessibility_day	Dia de acessibilidade
line_id	Identificador da linha

Tabela B.13

itinerary	
Atributo	Descrição
itinerary_company_name	Nome da empresa do itinerário
itinerary_code	Código do itinerário
itinerary_special_line	Linha especial do itinerário
line_id	Identificador da linha

Tabela B.14

APÊNDICE C – FIGURAS

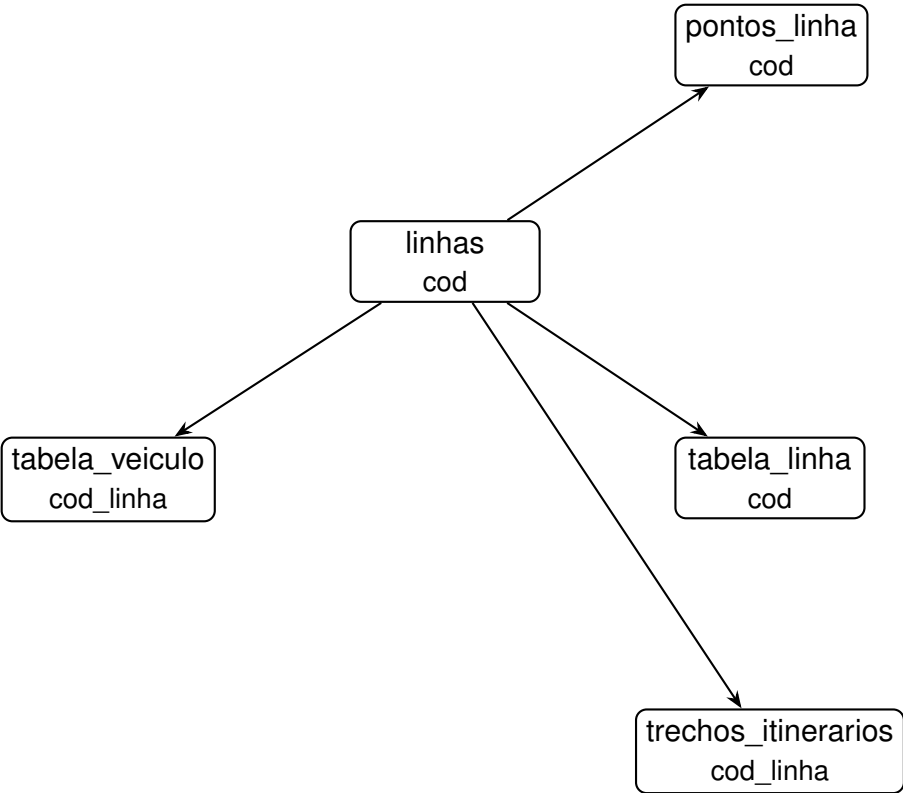


Figura C.1 – Relação atributo cod

C.7 Pré-Processamento de Dados

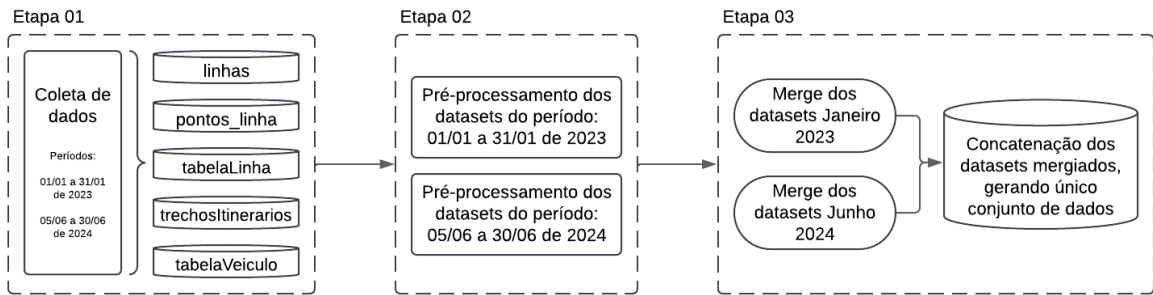


Figura C.2 – Etapas - Pré-Processamento de Dados

C.8 Integração Google Maps

Figura C.3

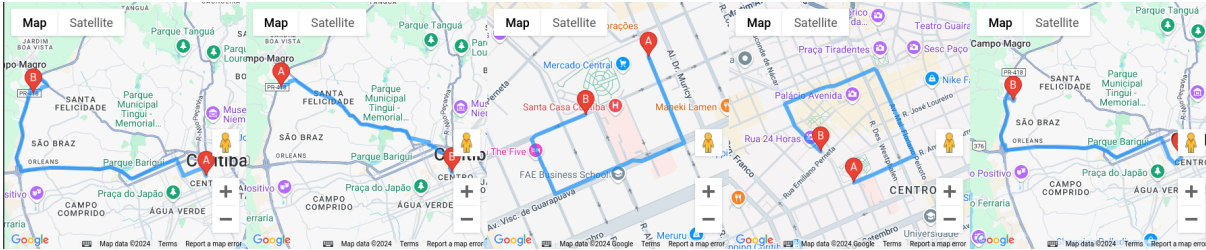


Figura C.4

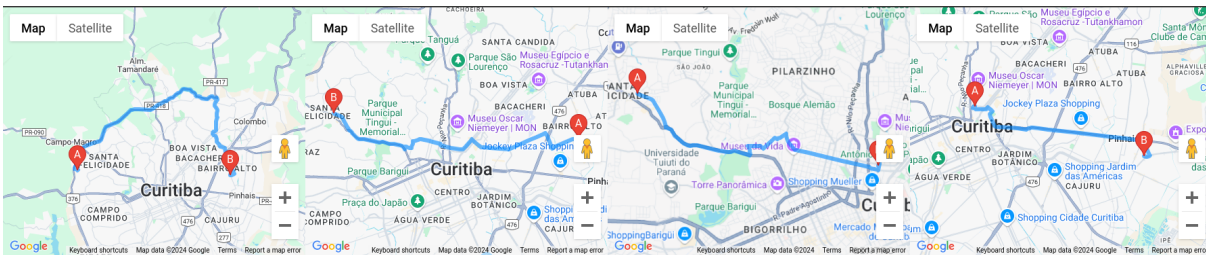
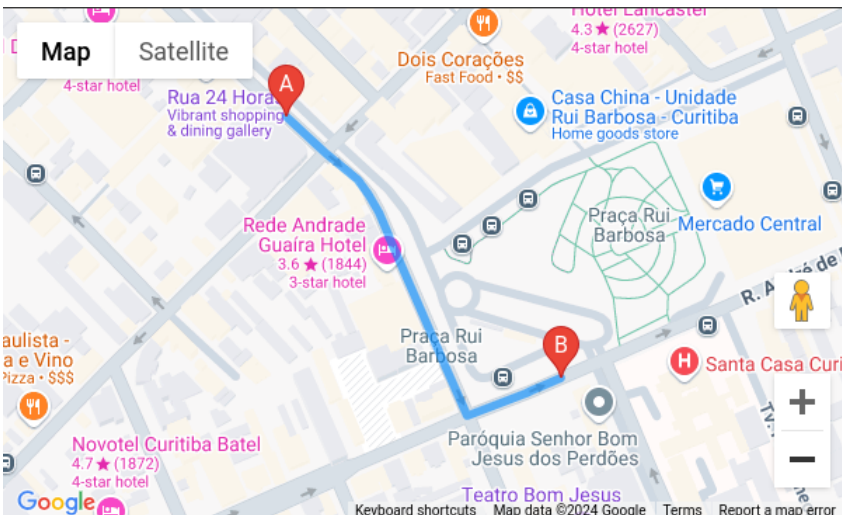


Figura C.5



C.9 Arquitetura da Redes Neurais Artificiais (RNA)

Figura C.6

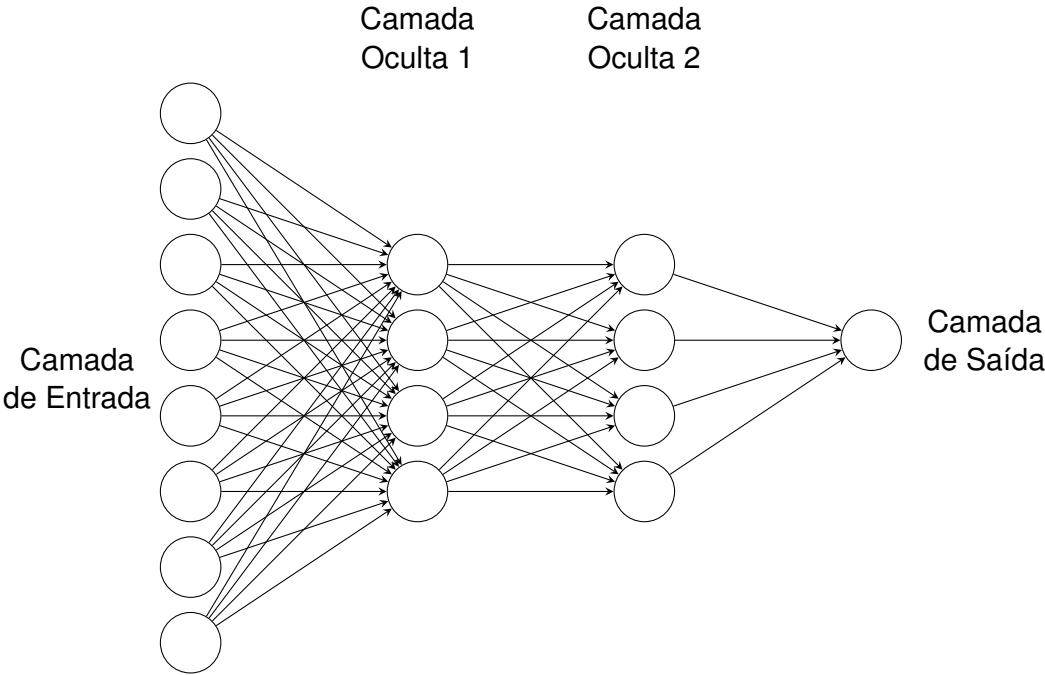
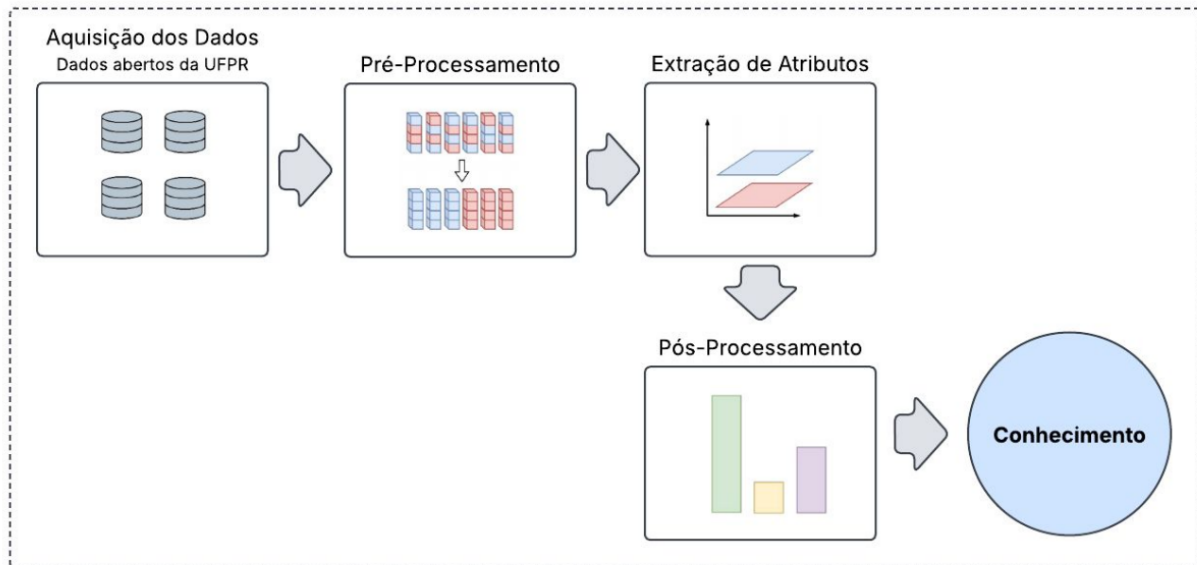


Figura C.7 – Perceptron Multicamadas (MLP)

C.10 Etapas do Processo KDD

Figura C.8



APÊNDICE D – EQUAÇÕES

D.11 Equações Utilizadas

$$\text{km} = \frac{\text{m}}{1000} \quad (\text{D.1})$$

$$m = \frac{s}{60} \quad (\text{D.2})$$

$$v = \frac{d}{t} \quad (\text{D.3})$$

$$IQR = Q3 - Q1 \quad (\text{D.4})$$

$$\text{Limite Inferior} = Q1 - 1.5 \times IQR \quad (\text{D.5})$$

$$\text{Limite Superior} = Q3 + 1.5 \times IQR \quad (\text{D.6})$$

$$r_{v_1, v_2} = \frac{\sum_{i=1}^n (v_{1i} - \overline{v_1})(v_{2i} - \overline{v_2})}{\sqrt{\sum_{i=1}^n (v_{1i} - \overline{v_1})^2} \cdot \sqrt{\sum_{i=1}^n (v_{2i} - \overline{v_2})^2}} \quad (\text{D.7})$$

$$\hat{y} = b_0 + b_1 \cdot x \quad (\text{D.8})$$

$$y = b_0 + b_1 \cdot x_1 + b_2 \cdot x_2 + \cdots + b_n \cdot x_n \quad (\text{D.9})$$

$$y = b_0 + b_1 \cdot x_1 + b_2 \cdot x_1^2 + \cdots + b_n \cdot x_1^n \quad (\text{D.10})$$

$$K(x, y) = \exp(-\gamma \|x - y\|^2) \quad (\text{D.11})$$

$$f(x) = \begin{cases} x, & \text{se } x > 0 \\ 0, & \text{se } x \leq 0 \end{cases} \quad (\text{D.12})$$

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i| \quad (\text{D.13})$$

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad (\text{D.14})$$

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2} \quad (\text{D.15})$$

$$R^2 = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2} \quad (\text{D.16})$$

$$F = \frac{\text{Variância entre os grupos}}{\text{Variância dentro dos grupos}} \quad (\text{D.17})$$

$$F = \frac{\frac{\sum_{i=1}^k n_i (\bar{X}_i - \bar{X})^2}{k-1}}{\frac{\sum_{i=1}^k \sum_{j=1}^{n_i} (X_{ij} - \bar{X}_i)^2}{N-k}} \quad (\text{D.18})$$

$$q = \frac{\bar{X}_i - \bar{X}_j}{\sqrt{\frac{\text{Var}(\text{MAE})}{n}}} \quad (\text{D.19})$$

APÊNDICE E – TABELAS

E.12 Tabelas Relacionadas às Análises Exploratórias de Dados

Categoria	Código
ALIMENTADOR	1
CONVENCIONAL	2
MADRUGUEIRO	3
TRONCAL	4
LINHA DIRETA	5
EXPRESSO	6
INTERBAIRROS	7
LIGEIRÃO	8
JARDINEIRA	9

Tabela E.1 – Atributo line_category e suas regras numéricas. (Tabela 03)

Atributo	Quantidade de Outliers
line_category	502
line_color	1448
line_payment_method	1422
accessibility_info_eletric_ramp	1095
station_initial_latitude	6
station_final_latitude	17
distance	333
average_speed_ms	440
average_speed_kh	440
duration	2

Tabela E.2 – Tabela de outliers. (Tabela 04)

Atributo 1	Atributo 2	Coefficiente
duration_min	duration	1.000000
distance	distance_km	1.000000
average_speed_ms	average_speed_kh	1.000000
line_category	line_color	0.952449
distance	duration	0.861906
distance	duration_min	0.861906
distance_km	duration	0.861906
distance_km	duration_min	0.861906
distance	average_speed_kh	0.754364
distance_km	average_speed_ms	0.754364
distance	average_speed_ms	0.754364
distance_km	average_speed_kh	0.754364
line_category	accessibility_info_eletric_ramp	0.538726
line_color	accessibility_info_eletric_ramp	0.529036
average_speed_ms	duration_min	0.373648
average_speed_ms	duration	0.373648
average_speed_kh	duration	0.373648
average_speed_kh	duration_min	0.373648
vehicle_time	accessibility_info_time	0.209952
station_initial_latitude	station_initial_longitude	0.204075

Tabela E.3 – Tabela de Coeficientes de Correlações entre Atributos. (Tabela 05)

E.13 Tabelas Relacionadas aos Experimentos

Modelo	Estatística W	p-valor
MLP	0.8460	0.1824
RNA	0.9620	0.8219
RNA Tuning	0.6098	0.0008

Tabela E.4 – Resultados do teste de Shapiro-Wilk para normalidade dos dados. (Tabela 06)

Teste	Valor
Estatística de Friedman (<i>stat</i>)	7.600
Valor-p (<i>p</i>)	0.0224

Tabela E.5 – Resultados do teste de Friedman para comparação dos modelos. (Tabela 07)

E.14 Tabelas Relacionadas as Análises de Trabalhos e Artigos

Comparação	Estatística (stat)	Valor-p (p)
MLP vs RNA	4.000	0.4375
MLP vs RNA Tuning	0.000	0.0625
RNA vs MLP	4.000	0.4375
RNA vs RNA Tuning	0.000	0.0625
RNA Tuning vs MLP	0.000	0.0625
RNA Tuning vs RNA	0.000	0.0625

Tabela E.6 – Resultados do teste de Wilcoxon para comparações pareadas entre os modelos. (Tabela 08)

Modelo 1	Modelo 2	Dif. Média	p-ajustado	Lim. Inferior	Lim. Superior	Rejeita H_0 ?
MLP	RNA	11.1299	0.6441	-21.4726	43.7324	Não
MLP	RNA Tuning	-8.1211	0.7879	-40.7237	24.4814	Não
RNA	RNA Tuning	-19.251	0.2931	-51.8536	13.3515	Não

Tabela E.7 – Resultados do teste de Nemenyi para comparações múltiplas entre os modelos. (Tabela 09)

Modelo	Média	Desvio Padrão
MLP	43.5734	1.6650
RNA	39.1328	10.3583
RNA Tuning	61.4123	29.9310

Tabela E.8 – Análise descritiva dos modelos baseados em RNA: média e desvio padrão. (Tabela 10)

group1	group2	meandiff	p-adj	lower	upper	reject
RL Múltipla	RL Polinomial	-82.582	0.0	-93.3292	-71.8348	True
RL Múltipla	RL SVR	-75.7242	0.0	-86.4714	-64.977	True
RL Múltipla	RL Simples	109.0085	0.0	98.2613	119.7557	True
RL Polinomial	RL SVR	6.8579	0.2979	-3.8893	17.6051	False
RL Polinomial	RL Simples	191.5906	0.0	180.8434	202.3378	True
RL SVR	RL Simples	184.7327	0.0	173.9855	195.4799	True

Tabela E.9 – Comparação Múltipla de Médias - Teste de Tukey HSD, FWER=0.05. (Tabela 11)

Modelo	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
RL Simples	217,8948	221,6355	214,4085	215,8374	238,0413
RL Múltipla	116,9502	112,8189	107,9391	104,8629	120,2038
RL Polinomial	29,8763	29,1873	28,9052	30,8245	31,0714
RL SVR	37,5122	34,5338	41,4880	34,4905	36,1296

Tabela E.10 – MAE para cada fold dos modelos tradicionais. (Tabela 12)

Modelo	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
MLP	42,0543	45,2809	45,2541	43,4433	41,8344
RNA	45,5012	52,3862	39,7903	30,8527	27,1338
RNA Tuning	49,9603	114,8611	47,4836	49,3284	45,4279

Tabela E.11 – MAE para cada fold dos modelos baseados em RNA. (Tabela 13)

Métrica	Valor
Estatística t	1.8379
Valor-p	0.1399
Conclusão	Sem diferença significativa entre os modelos
Média MAE RNA	39.1328
Média MAE Regressão Polinomial	29.9729
Melhor modelo	Regressão Polinomial (menor MAE)

Tabela E.12 – Comparação estatística entre Regressão Polinomial e Redes Neurais Artificiais (RNA). (Tabela 14)

Trabalho/Artigo	Pontos Abordados	Presente Trabalho
Influências da Inteligência Artificial na Otimização do Tráfego Urbano: Um Referencial Teórico	- Perda de tempo em ambientes de transporte urbano; - Técnicas que aumentam a eficiência das infraestruturas existentes; - Técnicas de Inteligência Artificial que influenciam na otimização do tráfego urbano; - Pesquisa de natureza descritiva e exploratória.	Pesquisa de natureza descritiva e exploratória.
Sistema Algorítmico de Rede Neural Artificial para Previsão da Localização de Ônibus	- Problemas de comunicação dos ônibus com GPS; - Problemas de intermitência e falhas quando a comunicação em tempo real, prejudicando a confiabilidade dos serviços de transporte; - Proposta de algoritmo baseado em RNA (Redes neurais artificiais) para estimar a posição dos ônibus em casos de falhas na comunicação; - Dados contendo informações do comportamento histórico dos ônibus e dados de GPS mais recentes para realizar as estimativas; - Benefícios esperados para experiência dos usuários e a gestão do transporte, com disponibilidade de estimativas de localização mesmo em momentos de instabilidade e quedas no sistema de comunicação.	Dados contendo informações do comportamento histórico dos ônibus e dados de GPS mais recentes para realizar as estimativas.
Desenvolvimento de Sistema de Inteligência Artificial (IA) Baseado em Reconhecimento de Padrões Para Análise de Rotas Veiculares	- Busca pela modernização do processo; - Clusterização de Rotas; - Treinamento, Validação, Simulações e Testes utilizando RNAs; - Comparação entre clusterização e aprendizado.	Treinamento, Validação, Simulações e Testes utilizando RNAs.

Tabela E.13 – Análise Comparativa de Trabalhos

Trabalho/Artigo	Pontos Abordados	Presente Trabalho
Predição de tempos de viagem em linhas de ônibus baseado em dados de tráfego e redes neurais artificiais	- Opções de rotas; - Técnicas de data mining; - Extração de dados que agregam informações brutas; - Análise de dados geográficos; - Transformação da forma como as pessoas se deslocam nas cidades, proporcionando uma mobilidade mais eficiente e conveniente; - Análises de séries temporais; - Algoritmos de machine learning utilizados: SVN, KNN, Regressão Linear, Algoritmo Genético, Florestas Aleatórias, Rede Neural Artificial; - Diagnóstico do modelo através das métricas: Raiz do Erro Quadrático Médio (RMSE), Erro Médio Absoluto (MAE).	Análise de dados geográficos.

Tabela E.14 – Análise Comparativa de Trabalhos

	Presente Trabalho	Larsen (2020)
Modelos	RL RNA	RNA
Neurônios de Entrada	8	10
Camadas Ocultas	2	1
Função de Ativação	ReLU	Sigmoid
Otimizador	Adam	Nadam
RMSE	56,4508	297
Normalização	Não	Sim

Tabela E.15 – Comparação de Modelos RNA