CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
CAMPUS TIMÓTEO

Calebe Libério Pedroso

ANÁLISE E MODELAGEM DO CONSUMO DE COMBUSTÍVEL EM VEÍCULOS AUTOMOTORES UTILIZANDO ALGORITMOS DE MACHINE LEARNING E DADOS OBD

Timóteo

2024

Calebe Libério Pedroso

ANÁLISE E MODELAGEM DO CONSUMO DE COMBUSTÍVEL EM VEÍCULOS AUTOMOTORES UTILIZANDO ALGORITMOS DE MACHINE LEARNING E DADOS OBD

Monografia apresentada à Coordenação de Engenharia de Computação do Campus Timóteo do Centro Federal de Educação Tecnológica de Minas Gerais para obtenção do grau de Bacharel em Engenharia de Computação.

Orientador: Leonardo Lacerda Alves

Timóteo

2024

Calebe Libério Pedroso

ANÁLISE E MODELAGEM DO CONSUMO DE COMBUSTÍVEL EM VEÍCULOS AUTOMOTORES UTILIZANDO ALGORITMOS DE MACHINE LEARNING E DADOS OBD

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia de Computação do Centro Federal de Educação Tecnológica de Minas Gerais, campus Timóteo, como requisito parcial para obtenção do título de Engenheiro de Computação.

Trabalho aprovado. Timóteo, 14 de Fevereiro de 2025.

Leonardo Lacerda Alves
Orientador

Douglas Nunes de Oliveira
Professor Convidado

Odilon Corrêa da Silva
Professor Convidado

Timóteo
2025

FOLHA DE APROVAÇÃO DE TCC Nº 2/2025 - DECOMTM (11.63.11)

(Nº do Protocolo: NÃO PROTOCOLADO)

(Assinado digitalmente em 20/02/2025 13:17)
DOUGLAS NUNES DE OLIVEIRA
PROFESSOR ENS BASICO TECN TECNOLOGICO
DECOMTM (11.63.11)
Matrícula: ###212#8

(Assinado digitalmente em 20/02/2025 13:51)
LEONARDO LACERDA ALVES
PROFESSOR ENS BASICO TECN TECNOLOGICO
DECOMTM (11.63.11)
Matrícula: ###653#3

(Assinado digitalmente em 17/02/2025 22:46)
ODILON CORREA DA SILVA
PROFESSOR ENS BASICO TECN TECNOLOGICO
DECOMTM (11.63.11)
Matrícula: ###944#5

Visualize o documento original em https://sig.cefetmg.br/documentos/ informando seu número: ***2****, ano:* ***2025****, tipo:* ***FOLHA DE APROVAÇÃO DE TCC****, data de emissão:* ***17/02/2025*** *e o código de verificação:* ***edbefc1d35***

.

.

.

Resumo

A aplicação de técnicas de aprendizado de máquina para análise de dados tem ganhado destaque em diversos cenários, especialmente naqueles que envolvem grandes volumes de dados e problemas complexos.A classificação de dados, um dos pilares do aprendizado de máquina, consiste na construção de modelos capazes de prever a categoria à qual um novo dado pertence. Diversos algoritmos, como Árvores de Decisão, Máquinas de Vetores de Suporte (SVM), Redes Neurais e Naive Bayes, apresentam características e aplicações específicas, sendo a escolha do modelo ideal dependente da natureza dos dados e dos objetivos da análise. Este trabalho investiga a aplicação de algoritmos de aprendizado de máquina na análise de dados de consumo de combustível coletados por meio de varreduras OBD (*On-Board Diagnostics*). O estudo avalia o desempenho de diferentes modelos de classificação, destacando os resultados obtidos em termos de métricas de desempenho e sua aplicabilidade na compreensão de padrões de consumo de combustível. Os resultados deste estudo demonstram que o algoritmo Random Forest obteve melhor desempenho na classificação do consumo de combustível, alcançando uma acurácia de 92%, seguido por Árvore de Decisão. Além disso, verificou-se que atributos como ENGINE_RPM, ENGINE_LOAD, SPEED, MAF, THROTTLE_POS e INTAKE _MANIFOLD_PRESSURE possuem maior relevância na classificação do consumo, conforme análise de importância de atributos. Esses achados reforçam a aplicabilidade do aprendizado de máquina na compreensão dos fatores que influenciam a eficiência do consumo de combustível e podem auxiliar na otimização de estratégias para redução de custos e impacto ambiental.

Palavras-chave: Aprendizado de máquina, Classificação de dados, Algoritmos de classificação, OBD Scan, Consumo de combustível.

Abstract

The application of machine learning techniques for data analysis has gained prominence in several scenarios, especially in those involving large volumes of data and complex problems. Data classification, one of the pillars of machine learning, consists of building models capable of predicting the category to which a new piece of data belongs. Several algorithms, such as Decision Trees, Support Vector Machines (SVM), Neural Networks and Naive Bayes, have specific characteristics and applications, and the choice of the ideal model depends on the nature of the data and the objectives of the analysis. This work investigates the application of machine learning algorithms in the analysis of fuel consumption data collected through OBD (On-Board Diagnostics) scans. The study evaluates the performance of different classification models, highlighting the results obtained in terms of performance metrics and their applicability in understanding fuel consumption patterns. The results of this study demonstrate that the Random Forest algorithm obtained the best performance in the classification of fuel consumption, reaching an accuracy of 92%, followed by Decision Tree. Furthermore, it was found that attributes such as ENGINE_RPM, ENGINE_LOAD, SPEED, MAF, THROTTLE_POS and INTAKE _MANIFOLD_PRESSURE have greater relevance in the consumption classification, according to the attribute importance analysis. These findings reinforce the applicability of machine learning in understanding the factors that influence fuel consumption efficiency and can help optimize strategies to reduce costs and environmental impact.

Keywords: Machine learning, Data classification, Classification algorithms, OBD Scan, Fuel consumption.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

SVM	Support Vector Machine
KNN	k-nearest neighbors
MPL	Multilayer Perceptron
OBD	On-Board Diagnostics
NaN	Not a Number
UFRN	Universidade Federal do Rio Grande do Norte
MAF	Mass Air Flow
MAE	Mean Absolute Error
MSE	Mean Squared Error
SMOTE	Synthetic Minority Over-sampling Technique
LDA	Linear Discriminant Analysis
PCA	Principal Component Analysis
IOT	Internet of Things
IAT	Intake Air Temperature
RPM	Revolutions Per Minute
PID	Parameter ID
VSS	Vehicle Speed Sensor
MAP	Manifold Absolute Pressure
SA	Spark Advance
RFECV	Recursive Feature Elimination with Cross-Validation
RFE	Recursive Feature Elimination

CSV	Comma-Separated Values
VIF	Variance Inflation Factor
INCA	Integrated Network Control Assistant
ECU	Engine Control Unit

Sumário

1 Introdução

1.1 Introdução

A classificação de dados desempenha um papel fundamental no aprendizado de máquina, permitindo a construção de modelos capazes de associar novas instâncias a categorias previamente definidas (HAN; KAMBER; PEI, 2011; DOMINGOS, 2012). Diversas técnicas têm sido amplamente empregadas para essa finalidade, como Árvores de Decisão, Máquinas de Vetores de Suporte (SVM), Redes Neurais e Random Forest, que se destacam por sua capacidade de adaptação a diferentes tipos de dados e complexidades (HASTIE; TIBSHIRANI; FRIEDMAN, 2009; BREIMAN, 2001a; BISHOP, 2006).

No contexto da análise automotiva, essas técnicas são utilizadas para identificar padrões nos dados coletados por meio de varreduras OBD (*On-Board Diagnostics*), fornecendo informações valiosas sobre o comportamento dos veículos e os fatores que impactam seu desempenho (WU et al., 2008; INTERNATIONAL, 2020).

O estudo dessas abordagens contribui para um melhor entendimento do consumo de combustível e sua relação com variáveis operacionais, possibilitando avanços na otimização de recursos e na redução do impacto ambiental.

1.2 Justificativa

A análise de padrões de consumo de combustível por meio de aprendizado de máquina surge como uma área promissora, mas ainda pouco explorada, especialmente quando se trata de dados obtidos via OBD. Apesar de avanços significativos em técnicas de aprendizado de máquina, há uma lacuna na literatura quanto à aplicação dessas metodologias para categorizar e compreender os fatores que influenciam o consumo de combustível.

Entender esses padrões pode gerar impactos significativos. Além de otimizar o uso de combustível e reduzir emissões poluentes, os insights gerados podem ser aplicados para orientar melhorias em veículos e em políticas de gestão de frotas. No entanto, trabalhar com dados OBD apresenta desafios técnicos importantes, como a variabilidade nas condições de condução e a complexidade intrínseca dos dados, que

exigem uma análise cuidadosa.

Portanto, este trabalho busca não apenas contribuir para preencher essa lacuna no conhecimento, mas também explorar o potencial do aprendizado de máquina como uma ferramenta prática e acessível para o setor automotivo.

1.3 Problema

Diante desse contexto, surge a seguinte questão central: **Modelos de classificação aplicados a dados OBD são capazes de inferir com precisão o consumo de combustível?**

A análise de padrões de consumo de combustível apresenta diversos obstáculos. Dados coletados via OBD frequentemente possuem alta dimensionalidade e podem conter ruídos que dificultam a construção de modelos robustos. Além disso, variáveis externas, como condições climáticas, características das vias e comportamento do motorista, influenciam significativamente os resultados, aumentando a dificuldade de identificar padrões consistentes.

Este trabalho propõe investigar a eficácia de diferentes modelos de classificação na análise de consumo de combustível, considerando essas complexidades. A partir de uma abordagem que combina técnicas consolidadas de aprendizado de máquina e um foco prático, espera-se superar essas barreiras, oferecer análises mais precisas e contribuir para a adoção de soluções que otimizem a eficiência energética e reduzam impactos ambientais no setor automotivo.

1.4 Objetivos

O objetivo principal deste trabalho é avaliar a eficácia de metodologias de aprendizado de máquina na classificação de dados obtidos por varreduras OBD, com foco no consumo de combustível de veículos automotores.

1.4.1 Objetivos Específicos

Foram estabelecidos os objetivos específicos abaixo.

1. **Caracterizar a base de dados** para identificar e descrever variáveis relevantes na classificação de tipos de combustível.

2. **Aplicar e avaliar algoritmos de classificação** de IA, como Árvores de Decisão, Random Forest, SVM e Redes Neurais, para determinar a eficácia desses modelos na classificação de combustível.

3. **Comparar o desempenho dos algoritmos** com base em métricas como acurácia, MAE, MSE e F1-Score, identificando o modelo mais eficiente.

4. **Validar a adequação da base de dados** para uso em modelos de classificação, assegurando que os dados são robustos e adequados para os propósitos de classificação.

2 Procedimentos metodológicos

Para cumprir com os objetivos levantados foi executada uma metodologia adequada e consistente. Dessa forma, o capítulo em questão irá detalhar os procedimentos utilizados no trabalho, desde o tratamento da base de dados até o procedimento experimental. Visto isso, a metodologia empregada visou garantir uma robustez e consistência para a pesquisa, tornando os resultados obtidos generalizáveis para o contexto de análise de consumo de combustíveis de veículos automotores.

2.1 Base de Dados

2.1.1 Seleção da Base de Dados

A seleção da base de dados foi um passo crucial para a realização deste trabalho. A base de dados que foi utilizada neste estudo provém do trabalho de Barreto (2018) e foi coletada de um único automóvel, em um percurso específico e repetitivo. Apesar de ser restrita a um veículo e a um único trajeto, essa base de dados ainda possui características relevantes para o objetivo deste trabalho, tais como a consistência dos dados, o detalhamento dos parâmetros do veículo e a repetibilidade das condições.

Primeiramente, o autor assegura que a coleta dos dados foi realizada de maneira consistente, garantindo a homogeneidade das condições de captura ao longo do tempo. Isso facilita a análise dos padrões de consumo, proporcionando dados confiáveis e comparáveis.

Em segundo lugar, a base contém informações detalhadas sobre diversos parâmetros do veículo, como velocidade, RPM e outros dados provenientes da OBD, que são essenciais para a análise do comportamento de condução. Finalmente, o fato de o percurso ser sempre o mesmo permite que a análise se concentre em variações intrínsecas do veículo e do comportamento de condução, minimizando o impacto de variáveis externas como o trajeto.

Por fim, a base escolhida foi construída com o apoio de 19 voluntários, que realizaram o mesmo trajeto. Gerando assim, 19 tipos de amostras para a base perfis, nomeadas s1 a s19.

2.1.2 Veículo Utilizado

O veículo utilizado pelo autor no experimento foi um veículo Chevrolet S10, ano 2015, equipado com motor 2.5 movido a gasolina (Figura 1). O autor também informa que a escolha deste automóvel se deu por conveniência, baseada em sua disponibilidade. O modelo escolhido é comercializado ao público em geral e possui características similares a outros veículos dessa categoria, o que amplia a aplicabilidade dos resultados obtidos para outros automóveis amplamente utilizados. (BARRETO, 2018)

Figura 1 – S10 2015 (BARRETO, 2018)

2.1.3 Trajeto Percorrido Para Gerar a Base de Dados

O trecho definido pelo autor corresponde a uma área urbana na cidade de Natal, Rio Grande do Norte. O percurso abrange aproximadamente 18,8 km, com início na Universidade Federal do Rio Grande do Norte (UFRN) e término no Estádio Maria Lamas Farache (Frasqueirão). De acordo com a estimativa do Google Maps, realizada por volta das 15h, a viagem leva cerca de 34 minutos, considerando o caminho mais rápido. Uma visão geral desse trajeto pode ser visualizada na Figura 8, enquanto uma ilustração do trajeto está disponível na Figura 1 e também pode ser acessada por meio do link: https://goo.gl/A7KViD. (BARRETO, 2018).

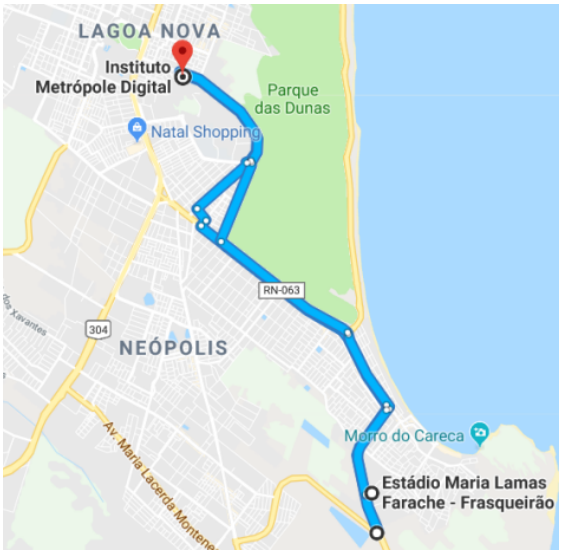


Figura 2 – (BARRETO, 2018)

2.2 Pré-processamento dos Dados

Após a seleção da base de dados, foi realizada uma etapa de pré-processamento ilustrada pela figura 3 para garantir a qualidade e a adequação dos dados para a aplicação de algoritmos de aprendizado de máquina. O pré-processamento envolveu as seguintes etapas:

1. **Limpeza dos Dados:** Foi realizada a identificação e remoção de dados inconsistentes, inválidos ou duplicados, garantindo a integridade e a confiabilidade do conjunto de dados.

2. **Tratamento de Valores Faltantes:** Foram utilizadas técnicas de imputação para lidar com valores faltantes, evitando a perda de informações importantes.

3. **Conversão de Dados:** Efetuou-se a conversão de valores obtidos da base em unidades de medidas mais amigáveis para utilização nos procedimentos. Na prática, as colunas numéricas foram transformadas para tipos apropriados, removendo caracteres não numéricos e garantindo a correta interpretação dos valores. A coluna *ENGINE_RUNTIME*, por exemplo, foi convertida para um formato de tempo (`pd.to_timedelta`), possibilitando cálculos temporais mais precisos. Já as colunas categóricas, como *VEHICLE_ID* e *FUEL_TYPE*, foram convertidas para o tipo `category`, otimizando o armazenamento e processamento dos dados.

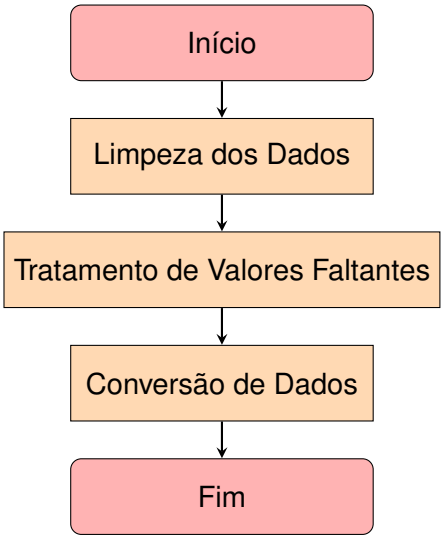


Figura 3 – Fluxograma do Pré-processamento dos Dados

2.3 Caracterização da Base de Dados

Para análise de qual domínio melhor se adéqua à base de dados escolhida, efetuou-se a geração de uma matriz de correlação para identificar relações significativas entre as variáveis.

Com base na matriz de correlação gerada, foram identificados pares de variáveis com correlações fortes (acima de 0.6). O valor de linha base foi definido em módulo, logo relações inversamente proporcionais também foram consideradas na análise.

A partir de uma análise sensível dos valores com maior correlação, foi possível definir o tipo de problema de *machine learning* ao qual a base de dados se adequava. Esta análise preliminar indicou que as variáveis poderiam ser utilizadas para problemas de regressão e classificação, associados ao consumo de combustível.

2.3.1 Seleção de Variáveis

Após a análise da matriz de correlação, foi realizada a seleção de atributos relevantes para a execução dos modelos de aprendizado de máquina. Para essa etapa, foram adotadas técnicas de seleção de atributos, como:

- **Filtro baseado na correlação:** variáveis com correlação muito alta (>0.90) entre si foram avaliadas para evitar redundância, mantendo apenas uma representante por grupo de atributos altamente correlacionados.
- **Importância de atributos com modelos supervisionados:** algoritmos como Random Forest e Regressão Logística foram utilizados para estimar a relevância de cada variável na predição do consumo de combustível.

Com base nos resultados obtidos, foram selecionadas as variáveis que apresentaram maior influência sobre o consumo de combustível, garantindo que os modelos fossem treinados apenas com informações relevantes, reduzindo a dimensionalidade dos dados e melhorando a eficiência computacional.

2.4 Cálculo do Consumo de Combustível

Algumas variáveis escolhidas para serem utilizadas não possuem dados para alguns campos, dentre eles, campos que são cruciais para o processo de classificação, devido a isso, algumas medidas de adequação foram necessárias. Nesse sentido, como o campo **FUEL-ECONOMY** está vazio outras variáveis disponíveis foram utilizadas para calcular o consumo do veículo. Visto isso, foi utilizada uma fórmula comum que relaciona a quantidade de combustível consumido com a distância percorrida e outras variáveis relevantes. Uma abordagem comum é usar a taxa de fluxo de massa de ar (MAF) e a velocidade do veículo para estimar o consumo de combustível.

$$\text{Consumo de Combustível (L/100 km)} = \frac{\text{Velocidade (km/h)} \times \text{Densidade do Combustível (g/L)}}{3600 \times \text{MAF (g/s)}} \tag{2.1}$$

1. **MAF (g/s):** Taxa de fluxo de massa de ar.
2. **Velocidade (km/h):** Velocidade do veículo.
3. **Densidade do Combustível (g/L):** Aproximadamente 720 g/L para gasolina.

2.5 Discretização e Balanceamento dos Dados

Para otimizar a análise e melhorar a performance dos modelos de classificação, executou-se técnicas de discretização e balanceamento dos dados.

Algumas variáveis contínuas foram transformadas em categorias discretas para simplificar a interpretação dos dados e melhorar a eficiência dos algoritmos de aprendizado de máquina. Utilizou-se técnicas como *binning*, que segmenta os valores em intervalos definidos.

Para corrigir possíveis desbalanceamentos entre as classes, aplicou-se métodos como *oversampling* e *undersampling*. O *oversampling* aumenta a quantidade de exemplos da classe minoritária, enquanto o *undersampling* reduz a quantidade de exemplos da classe majoritária, garantindo uma distribuição mais equitativa dos dados.

2.6 Execução de Modelos de Classificação

Os algoritmos escolhidos para este estudo foram selecionados com base em sua capacidade de lidar com dados complexos e variáveis contínuas relacionadas ao consumo de combustível. A seguir, apresentamos uma justificativa para a escolha de cada modelo:

1. **Random Forest:** Como um modelo de ensemble baseado em múltiplas Árvores de Decisão, o Random Forest é capaz de reduzir a variância e minimizar o risco de *overfitting*, tornando-se uma opção robusta para lidar com a complexidade dos dados de consumo de combustível. Sua capacidade de lidar com grandes volumes de dados e identificar a importância de variáveis reforça sua adequação ao estudo.

2. **K-Nearest Neighbors (KNN):** Esse algoritmo é útil para capturar padrões locais nos dados, pois classifica as instâncias com base na similaridade com seus vizinhos mais próximos. Sua simplicidade e eficácia em conjuntos de dados com distribuições bem definidas fazem do KNN uma abordagem complementar na comparação de modelos.

3. **Máquinas de Vetores de Suporte (SVM):** O SVM é indicado para problemas onde há uma separação clara entre classes, especialmente quando os dados possuem alta dimensionalidade. Sua capacidade de encontrar um hiperplano de

separação ótimo permite uma boa generalização, sendo uma escolha interessante para avaliar padrões complexos de consumo.

4. **Naive Bayes:** Esse método probabilístico é eficiente para conjuntos de dados com alta dimensionalidade e é adequado quando há suposições de independência entre as variáveis preditoras. Como o problema envolve múltiplos atributos que influenciam o consumo de combustível, o Naive Bayes pode oferecer uma abordagem alternativa rápida e eficiente.

5. **Multi-Layer Perceptron (MLP):** Como uma rede neural artificial com múltiplas camadas, o MLP é capaz de capturar relações não lineares entre as variáveis, sendo adequado para problemas onde os padrões subjacentes são complexos. Sua capacidade de aprendizado profundo pode melhorar a acurácia da classificação, especialmente quando treinado com um conjunto de dados suficientemente grande.

2.7 Avaliação dos Modelos de Classificação

Após os testes iniciais com os modelos de classificação, foi conduzida uma etapa de validação cruzada para avaliar a robustez e a generalização dos modelos. Essa abordagem divide o conjunto de dados em múltiplas partições (*folds*), utilizando cada uma delas alternadamente como conjunto de teste, enquanto as demais são usadas para treinamento. Dessa forma, é possível obter uma avaliação mais consistente, reduzindo o viés associado à divisão única dos dados.

O código implementado para essa validação agrupou os resultados das métricas de desempenho por modelo, calculando a média de cada métrica ao longo dos folds. As métricas consideradas incluem:

- **Mean Squared Error (MSE):** Erro quadrático médio, que penaliza desvios maiores entre os valores reais e previstos.
- **R-Squared (R^2):** Proporção da variância explicada pelo modelo, com valores próximos a 1 indicando bom ajuste.
- **Mean Absolute Error (MAE):** Erro absoluto médio, uma medida robusta menos sensível a outliers.
- **Accuracy:** Taxa de acertos em classificações discretas.

- **F1-Score:** Métrica que pondera precisão e recall, útil em cenários com classes desbalanceadas.

A função implementada utilizou um agrupamento por modelo (*groupby*), permitindo calcular as médias das métricas para cada modelo avaliado. O resultado da validação cruzada, portanto, fornece uma visão consolidada do desempenho dos modelos, destacando aqueles com maior potencial para a tarefa de classificação do consumo de combustível. Essa abordagem robusta garante que as conclusões tiradas sejam representativas, minimizando os efeitos de variações nos conjuntos de treinamento e teste.

3 Revisão bibliográfica

“As grandes ideias surgem da observação dos pequenos detalhes.”
Augusto Cury

Esta sessão trata sobre conceitos importantes para realização deste trabalho, bem como uma breve visão geral sobre o domínio de estudo. Será abordado também o estado da arte, e trabalhos que foram usados como base teórica.

De início é importante salientar que para a compreensão completa e consistente deste trabalho faz-se necessário um breve conteúdo sobre as técnicas e ferramentas utilizadas para algoritmos de classificação, e conceitos principais da ferramenta OBD.

3.1 Algoritmos de classificação

Algoritmos de classificação são ferramentas para aprendizado de máquina que categorizam dados em diferentes classes, são amplamente utilizados em diferentes campos. Nesta seção, serão discutidos cinco algoritmos de classificação: Árvores de Decisão, Random Forest, K-Nearest Neighbors (KNN), Máquinas de Vetores de Suporte (SVM), Naive Bayes e MPL.

3.1.1 Árvores de Decisão

As Árvores de Decisão são algoritmos de classificação que utilizam uma estrutura em árvore para tomar decisões baseadas nos atributos dos dados. Cada nó interno representa uma "pergunta" sobre um atributo, e cada folha representa uma classificação ou decisão final. "As Árvores de Decisão são conhecidas por sua simplicidade e interpretabilidade, sendo fáceis de visualizar e entender" (COVER; HART, 1967)

3.1.2 Random Forest

O Random Forest é um algoritmo de ensemble que utiliza múltiplas Árvores de Decisão para melhorar a precisão da classificação e reduzir o risco de overfitting. "Ele constrói várias árvores de decisão durante o treinamento e produz a classe que é o

modo das classes (classificação) ou a média das previsões (regressão) de cada árvore individual"(BREIMAN, 2001b).

3.1.2.1 K-Nearest Neighbors (KNN)

O K-Nearest Neighbors (KNN) é um algoritmo de classificação baseado em instâncias, que classifica um novo dado com base nas classes dos k vizinhos mais próximos no espaço de características. "É um algoritmo simples e eficaz, especialmente para problemas onde a fronteira de decisão é complexa"(COVER; HART, 1967).

3.1.3 Máquinas de Vetores de Suporte (SVM)

As Máquinas de Vetores de Suporte (SVM) são algoritmos de classificação que procuram encontrar um hiperplano de separação ótimo entre as classes de dados. "Elas são eficazes em espaços de alta dimensionalidade e são conhecidas por sua capacidade de generalização robusta"(CORTES; VAPNIK, 1995).

3.1.4 Naive Bayes

O Naive Bayes é um conjunto de algoritmos de classificação baseados no Teorema de Bayes, com a suposição "ingênua"de independência entre os preditores. "É um algoritmo simples, mas poderoso, especialmente para grandes conjuntos de dados"(MCCALLUM; NIGAM, 1998).

3.1.5 Multi-Layer Perceptron (MLP)

O Multi-Layer Perceptron (MLP) é uma classe de redes neurais artificiais que utiliza múltiplas camadas de neurônios para modelar relações complexas entre as variáveis. "É particularmente útil em tarefas de classificação e regressão devido à sua capacidade de aprender representações não lineares"(CORTES; VAPNIK, 1995).

3.2 OBD 2

Segundo (SAE) (2021) O OBD(On Board Diagnostics) foi criado em 1990 nos Estados Unidos, para que fosse possível obter informações sobre a saúde de um automóvel. Informações como temperatura, emissão de gases e rotação de motor, entre outras, são possíveis de serem coletadas e identificadas.

A partir dos anos 2000 foi lançada uma evolução do OBD, o OBD 2 que virou uma ferramenta quase obrigatória para todos os carro fabricados desses deste ano. Com ela tornou-se possível fazer uma varredura no sistema do carro, diagnosticar problemas ou saber informações do automóvel, possibilitando o controle ou a realização da manutenção adequada quando necessário.

3.3 Análise Discriminante Linear (LDA) e Análise de Componentes Principais (PCA)

A Análise de Componentes Principais (PCA) e a Análise Discriminante Linear (LDA) são duas técnicas amplamente utilizadas para redução de dimensionalidade em problemas de aprendizado de máquina e análise de dados. Ambas têm o objetivo de transformar o espaço original dos dados em um novo espaço reduzido, preservando informações relevantes para as tarefas de modelagem e classificação.

3.3.1 Análise de Componentes Principais (PCA)

A PCA é um método estatístico não supervisionado que busca encontrar um novo conjunto de eixos ortogonais (componentes principais) que melhor explicam a variabilidade dos dados. Esses componentes são combinações lineares das variáveis originais e são ordenados de acordo com a quantidade de variância que capturam.

O principal objetivo da PCA é eliminar redundâncias, reduzir a complexidade dos dados e facilitar a visualização e análise, ao mesmo tempo que preserva a maior parte da informação original. A transformação ocorre a partir da decomposição da matriz de covariância dos dados, utilizando autovalores e autovetores para determinar os componentes principais mais relevantes (JOLLIFFE; CADIMA, 2016).

A PCA é amplamente utilizada em diversas aplicações, incluindo compressão de dados, remoção de ruído, análise exploratória e pré-processamento para algoritmos de aprendizado de máquina (WOLD; ESBENSEN; GELADI, 1987).

3.3.2 Análise Discriminante Linear (LDA)

Diferentemente da PCA, a LDA é uma técnica supervisionada, projetada para maximizar a separabilidade entre diferentes classes em um conjunto de dados rotulado. O objetivo da LDA é encontrar uma projeção linear dos dados para um espaço de menor dimensão, garantindo que as classes permaneçam bem separadas.

A LDA utiliza duas matrizes principais para determinar os eixos de projeção:

1. **Matriz de dispersão intra-classe** (S_W): Mede a variabilidade dentro de cada classe.
2. **Matriz de dispersão entre classes** (S_B): Mede a variabilidade entre as diferentes classes.

A função objetivo da LDA é encontrar uma transformação linear que maximize a razão entre S_B e S_W, garantindo que as amostras da mesma classe fiquem agrupadas e as classes distintas permaneçam separadas (FISHER, 1936).

A LDA é amplamente utilizada para classificação de padrões e reconhecimento de imagens, sendo aplicada em áreas como reconhecimento facial, detecção de anomalias e bioinformática (DUDA; HART; STORK, 2001).

3.3.3 Comparação entre PCA e LDA

Enquanto a PCA busca uma representação compacta dos dados reduzindo a redundância, a LDA é mais adequada para problemas de classificação, onde a separação entre categorias é essencial. Ambas as técnicas podem ser complementares, sendo utilizadas em conjunto para melhorar o desempenho de modelos preditivos.

3.4 Técnicas de Discretização e Balanceamento

A qualidade dos dados utilizados em algoritmos de aprendizado de máquina desempenha um papel crucial no desempenho dos modelos. Em muitos cenários, os dados precisam ser pré-processados para melhorar a representatividade e a robustez dos modelos, sendo a discretização e o balanceamento duas das técnicas mais empregadas.

3.4.1 Discretização de Variáveis

A discretização consiste na conversão de variáveis contínuas em categorias discretas, facilitando a análise e melhorando o desempenho de alguns algoritmos. Existem dois principais tipos de discretização: supervisionada e não supervisionada. Técnicas populares incluem:

- **Equal-Width Binning**: Divide os dados em intervalos de largura fixa (DOUGHERTY; KOHAVI; SAHAMI, 1995).
- **Equal-Frequency Binning**: Os intervalos são definidos de forma que cada um contenha aproximadamente o mesmo número de amostras (GARCIA; LUENGO; HERRERA, 2013).
- **Entropy-Based Discretization**: Utiliza medidas de entropia para definir os melhores pontos de corte, sendo amplamente usada em aprendizado supervisionado (FAYYAD; IRANI, 1993).

A discretização pode melhorar a interpretabilidade dos modelos, mas também pode levar à perda de informações, caso não seja aplicada corretamente.

3.4.2 Balanceamento de Dados

Problemas de classificação frequentemente enfrentam conjuntos de dados desbalanceados, onde algumas classes possuem muito mais amostras do que outras. Isso pode levar a vieses nos modelos, favorecendo as classes majoritárias. Algumas das principais técnicas de balanceamento incluem:

- **Oversampling**: Aumenta artificialmente o número de amostras da classe minoritária, sendo o *Synthetic Minority Over-sampling Technique* (SMOTE) uma das abordagens mais populares (CHAWLA et al., 2002a).
- **Undersampling**: Reduz o número de amostras da classe majoritária para equilibrar a distribuição (DRUMMOND; HOLTE, 2003).
- **Combinação de Oversampling e Undersampling**: Métodos híbridos buscam equilibrar as vantagens de ambas as abordagens (BUDA; MAKI; MAZUROWSKI, 2018).

O balanceamento adequado dos dados melhora a capacidade dos modelos de aprendizado de identificar padrões em todas as classes, resultando em classificações mais justas e robustas.

3.5 Validação Cruzada

A validação cruzada é uma técnica essencial na avaliação do desempenho de modelos de aprendizado de máquina, garantindo que os resultados não sejam enviesados por uma divisão específica dos dados (KOHAVI, 1995). O objetivo principal dessa abordagem é estimar a capacidade de generalização do modelo para dados não vistos, reduzindo o risco de overfitting e underfitting.

3.5.1 Métodos de Validação Cruzada

Diversos métodos de validação cruzada são utilizados na literatura, cada um com características específicas que os tornam mais adequados para diferentes cenários:

- **Holdout**: Divide o conjunto de dados em duas partes, normalmente 70% para treino e 30% para teste. É simples, mas pode produzir estimativas instáveis devido à dependência da amostra escolhida (RASCHKA, 2020).
- **k-Fold Cross-Validation**: O conjunto de dados é dividido em k partes iguais. O modelo é treinado k vezes, utilizando $k - 1$ partes para treino e a parte restante para teste. Essa abordagem reduz a variância dos resultados e melhora a estimativa do erro do modelo (ARLOT; CELISSE, 2010).
- **Leave-One-Out Cross-Validation (LOOCV)**: Um caso extremo de k-fold onde k é igual ao número total de amostras. Cada instância é usada uma vez para teste, e todas as outras para treino. Apesar de fornecer estimativas imparciais, é computacionalmente custoso (JAMES et al., 2013).
- **Stratified k-Fold Cross-Validation**: Uma variação do k-fold onde a proporção das classes é mantida em cada divisão, sendo particularmente útil para bases de dados desbalanceadas (WITTEN et al., 2016).
- **Repeated k-Fold Cross-Validation**: Executa múltiplas execuções do k-fold com diferentes divisões, proporcionando uma avaliação mais robusta da performance do modelo (RASCHKA, 2020).

3.5.2 Importância da Validação Cruzada

A validação cruzada é fundamental para selecionar modelos e ajustar hiperparâmetros. Métodos como Grid Search e Random Search frequentemente utilizam

validação cruzada para encontrar a melhor combinação de parâmetros (BERGSTRA; BENGIO, 2012). Além disso, ao fornecer uma estimativa mais confiável do erro de generalização, evita que o modelo seja ajustado excessivamente aos dados de treinamento.

3.6 Estado da arte

A técnica de utilização de dados veiculares para uso de algoritmos de aprendizagem de máquina é bem comum na academia. (BARRETO, 2018) realiza a coleta desses dados via OBD-Scan. O autor atribui tal fato a ferramenta oferecer um diagnóstico consistente do veículo, que pode ser obtido de maneira simples. Neste trabalho ele utiliza técnicas de aprendizado de máquina para identificar perfis de uso de automóveis a partir desses dados.

Foi utilizada a plataforma de aprendizagem Weka sobre a base, e, com isso, foram definidos 10 grupos que tiveram os melhores índices Silhoutte e DB. "Foram escolhidos 6 métodos de classificação (preditivos), são eles: Árvore de Decisão (J48); KNN, MLP; NB; RF e SVM. Para uma análise mais aprofundada e posterior validação estatística, tais métodos foram executados utilizando 3 tipos de configuração: split percentage 50% / 50%; split percentage 75% / 25% e 10-fold cross validation. "(BARRETO, 2018).

Para acurácia dos dados foi utilizados o teste de Friedman e Índice Nemenyi, e as melhores acurácias ocorreram com a base definida em 3 grupos.Após a realização destes teste contatou-se que o MLP obteve o melhor resultado. Após isso, foi definidos nomes para os 3 clusters, que contextualizassem conforme a justificativa do trabalho (High, Mid, Low).

Foi criado um software desenvolvido em Java com Android Studio. Ele atua como um cliente que solicita e exibe o perfil do motorista a partir de informações classificadas, armazenando e apresentando graficamente o último perfil identificado e a data do trajeto. A única tela permite a inserção da identificação do veículo para consulta, e a estrutura de pacotes e classes.

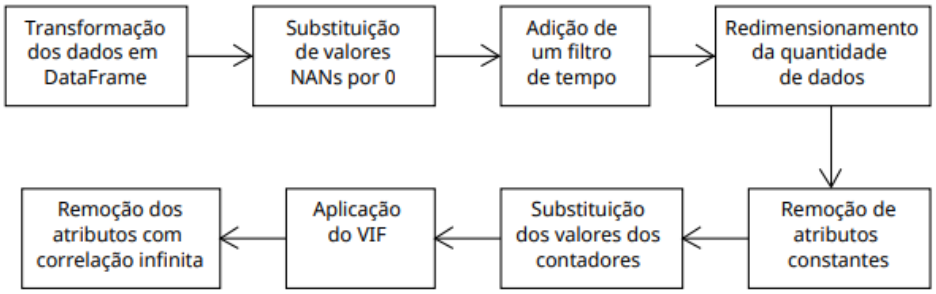
O autor retrata também que para expandir a plataforma, podem ser estudadas algumas variações e extensões, como a inclusão de um módulo para capturar dados de problemas automotivos e sua relação com os perfis identificados, um módulo de mapeamento de trajetos dos condutores, e a predição de problemas automotivos com

a base de dados coletada. Além disso, seria interessante integrar com outras plataformas para agregar dados adicionais e concluir a submissão de um artigo internacional. A interação com serviços e projetos como o Smart Metrópolis destacou a importância dos dados automotivos para futuras pesquisas. As principais dificuldades enfrentadas foram a diversidade de formatos e a integração com plataformas como a Weka, que exigiu desenvolvimento adicional e adaptação às especificidades dos formatos de dados.

Já o trabalho de SILVA (2022) trás uma forma diferente de coletar os dados. "Para obter as variáveis relacionadas na ECU, utilizou-se como base o arquivo A2L da ECU do carro utilizado no Projeto Rota2030, um Renault Sandero ano 2016. O A2L contém inúmeras informações de comunicação, características, sistemas e componentes da unidade que se refere, incluindo as variáveis das causas levantadas."(pag 36). O autor ressalta que os dados fornecidos pelo A2L são de um formato que não facilita muito a utilização, e que foi preciso utilizar uma analisadora sintática para formatação dos dados.

O autor ressalta também que alguns dados obtidos vieram com baixa qualidade, e cita que alguns campos estavam NAN que indica um valor ausente, Tratou também valores que sempre permaneciam constantes, O autor ilustrou as etapas de tratamento com o seguinte diagrama:

Figura 21 – Diagrama do processo de tratamento do dados



Fonte: Autora (2022).

Figura 4 – Tratamento dos dados (SILVA, 2022)

Os arquivos de saída do INCA em formato DAT foram convertidos para DataFrame usando um módulo desenvolvido no projeto. Da mesma forma, dados em formato Parquet foram transformados em DataFrame para uso nos algoritmos do Python e Sklearn. Após a conversão, foi feita a substituição de NaNs por zeros e adicionado um filtro de tempo. Para lidar com a alta quantidade de dados, foi realizado um resample de 10 milissegundos. Variáveis constantes e multicolineares foram removidas usando VIF (Variance Inflation Factor). A seleção de atributos e os algoritmos de identificação de falhas foram validados pela pontuação f1, que combina precisão e recall.

$$VIF = \frac{1}{1 - R^2}$$

Figura 5 – Formula VIF (SILVA, 2022)

$$Pontuação f1 = 2 * \frac{Recall * Precisão}{Recall + Precisão}$$

Figura 6 – Formula F1 (SILVA, 2022)

Como o A2L trás uma vasta quantidade de parâmetros, o autor sentiu a necessidade de selecionar os principais. Para isso foram utilizados alguns algoritmos de seleção. O autor realiza então uma análise de alguns métodos e trás um resumo comparativo deles.

Classificação	Método	Características
Filtro	SelectPercentile	Seleção com base em percentil das pontuações mais altas.
Filtro	SelectKBest	Seleção com base nas K pontuações mais altas.
Wrapper	SFS	Seleção avançando ou retrocedendo com base na pontuação de validação cruzada do estimador com um atributo recursivamente.
Wrapper	RFE	Seleção com base em *ranking* de atributos eliminando o menos importante recursivamente.
Wrapper	RFECV	Seleção recursiva com validação cruzada com base em *ranking* de atributos mais importantes para o treinamento.
Embutida	SelectFromModel	Seleção eliminando recursivamente os valores, classificados por pontuação de importância, menores que o *threshold* estabelecido.

Figura 7 – Resumo comparativo dos métodos utilizados (SILVA, 2022)

SelectPercentile:

Para implementar o método SelectPercentile, iniciou-se com a leitura do dataset em formato CSV e a divisão dos dados em conjuntos de treinamento e teste com 30% dos dados para teste, como mostrado abaixo:

```
X_train, X_test, y_train, y_test = train_test_split(
    df.drop(labels=['contador']), df['contador']
)
```

A função `SelectPercentile` da biblioteca Scikit-learn foi utilizada com o regressor `mutual_info_regression`, ajustando o percentual de atributos a ser mantido para 20%, 15% e 10%. A função foi configurada da seguinte forma:

$$selected\_percent = SelectPercentile(mutual\_info\_regression, percentile = 15) \tag{3.1}$$

O `mutual_info_regression` mede a dependência entre variáveis contínuas, sendo adequado para dados contínuos como o contador de falhas. Esse método foi escolhido para redução de dimensionalidade e melhor adaptação aos dados temporais coletados (SILVA, 2022).

SelectKBest:

Para implementar o algoritmo com o método SelectKBest, seguiu-se o mesmo procedimento de leitura e divisão de dados. Utilizou-se a função `SelectKBest` da biblioteca Scikit-learn com o regressor `r_regression`, configurando para selecionar os melhores 15, 10 e 5 atributos. Esses valores foram escolhidos com base em experimentos anteriores que identificaram 8 e 9 atributos como os mais informativos. A configuração foi:

$$selected\_best = SelectKBest(r\_regression, k = 10) \tag{3.2}$$

O `r_regression` calcula o coeficiente de correlação de Pearson para medir a relação estatística entre cada atributo e o alvo. Este coeficiente, conforme a Equação 3, é adequado para dados contínuos e melhora com o aumento da amostra, tornando-o apropriado para o SelectKBest (SILVA, 2022).

Sequential Feature Selector:

O procedimento inicial para leitura e divisão de dados seguiu métodos anteriores. Para usar a função `SequentialFeatureSelector` do Scikit-learn, foram comparados dois estimadores: `LogisticRegression` e `LassoCV`. O `LogisticRegression` foi configurado com até 1000 iterações e selecionou 5 atributos, usando validação cruzada de 10 pastas e a métrica de erro quadrático médio. O `LassoCV`, com até 5000 iterações, seguiu configurações semelhantes. A escolha desses parâmetros foi ajustada com base em testes para melhorar a seleção de atributos. O `LogisticRegression` aplica regularização L2, enquanto o `LassoCV` usa regularização L1, e ambos foram comparados para avaliar o desempenho na seleção de atributos (SILVA, 2022).

Recursive Feature Elimination:

O algoritmo foi iniciado com a leitura e divisão de dados, conforme o procedimento padrão. Em seguida, foi implementada a função `RFE` da biblioteca Scikit-learn com dois estimadores: `LogisticRegression` e `RandomForestRegressor`. Para o `LogisticRegression`, foram configuradas 1000 iterações, seleção de 15, 10 e 5 atributos, passo de 3 e verbosidade de 5. Já para o `RandomForestRegressor`, foram uti-

lizados os mesmos parâmetros de seleção de atributos, passo e verbosidade, mas o critério de medida da qualidade foi ajustado para poisson após testes iniciais. O `RandomForestRegressor` é eficaz com grandes conjuntos de dados e controla o overfitting, tornando-o adequado para simulações de falha e problemas de predição (SILVA, 2022).

Recursive Feature Elimination with Cross-Validation:

A implementação do `RFECV` seguiu a mesma abordagem do `RFE`, começando com a leitura e divisão dos dados. Foi utilizada a função `RFECV` da biblioteca Scikit-learn com dois estimadores: `RandomForestRegressor` e `LogisticRegression`. O modelo foi configurado para selecionar um mínimo de 15, 10 e 5 atributos, com validação cruzada de 10 pastas. A configuração do `RFECV` com `RandomForestRegressor` foi realizada da seguinte forma:

```
selected_rfecv = RFECV(RandomForestRegressor(n_estimators=100,
    criterion='poisson'),
min_features_to_select=5, cv=10)
```

A escolha dos estimadores para `RFECV`, assim como para o `RFE`, visou comparar os resultados de ambos com os dados das falhas e identificar o melhor desempenho para a seleção de atributos (SILVA, 2022).

SelectFromModel:

Para implementar o algoritmo com o método `SelectFromModel`, seguiu-se o procedimento inicial de leitura e divisão de dados. Utilizou-se o `RandomForestRegressor` com o critério de erro quadrado padrão e configurou-se para selecionar até 15, 10 e 5 atributos, com thresholds de 0.1 e nenhum. A configuração foi realizada da seguinte forma:

```
selected_classifier = SelectFromModel
(RandomForestRegressor(criterion='squared_error'),
    max_features=5, threshold=0.1)
```

`SelectFromModel` usa o estimador para atribuir importância aos atributos, e o `RandomForestRegressor` foi escolhido por evitar overfitting e lidar bem com dados contínuos. A escolha do estimador baseou-se na sua capacidade de fornecer uma avaliação robusta da importância dos atributos sem problemas de overfitting (SILVA, 2022).

Para analisar e comparar os conjuntos de dados gerados pelos algoritmos de Seleção de Atributos (SA) para falha de ignição, foi desenvolvido um MLP com quatro camadas (entrada, duas escondidas e saída). O MLP foi treinado por 10 épocas usando bibliotecas open source em Python, com tempos de treinamento de aproximadamente 10 minutos. Simulações foram realizadas com diferentes conjuntos de variáveis, desde 290 até 37 atributos, e os resultados foram utilizados para treinar e avaliar a rede neural.

Almeida (2017) aborda no seu trabalho um sistema que permita a análise detalhada do consumo de combustível de veículos automotores, proporcionando insights para otimizar o uso do combustível, reduzir custos operacionais e melhorar a eficiência energética dos veículos. O sistema, de forma similar ao trabalho de Barreto (2018) utiliza dados obtidos via OBD para serem trabalhados.

Os dados coletados neste trazem informações como volume de combustível consumido, distâncias percorridas, condições de condução (como aceleração e desaceleração), e características específicas do motor e do veículo. Os resultados gerados com a analise desses dados permitem o usuário do sistema, identificar nível de combustível, perfis de economia, tipo de rota percorrida e comparar o consumo com benchmarks conhecidos.

Meseguer et al. (2015) de forma semelhante analisa a relação entre comportamento na direção e consumo de combustível, criando assim, um sistema nomeado "DrivingStyles", um aplicativo criado em sistema Android, que utiliza dados da Unidade de Controle Eletrônico (ECU) de um veículo para analisar estilos de direção e oferecer feedback em tempo real aos motoristas. Como os autores anteriores Meseguer et al. (2015) utiliza um sistema baseado em OBD-|| para capturar variáveis como: velocidade, aceleração, RPM do motor, posição do acelerador e localização do veículo para serem tratadas, processadas e posteriormente utilizadas no cálculo.

Neste trabalho o autor faz uso de redes neurais que são treinadas para identificar perfis de direção com base em padrões nos dados. A plataforma também calcula o consumo instantâneo de combustível usando vários parâmetros, incluindo fluxo de ar em massa, pressão absoluta do coletor e temperatura do ar de admissão. O autor conclui que a direção agressiva, caracterizada por aceleração rápida, altas rotações do motor e trocas de marcha frequentes, leva consistentemente a um consumo de combustível significativamente maior em comparação à direção suave e silenciosa. Os autores demonstram que adotar um estilo de direção mais eficiente pode resultar

em economia de combustível que varia de 15% a 20%, traduzindo-se em benefícios econômicos e ambientais substanciais.

No trabalho de Oliveira (2022), o consumo de combustível é estimado a partir dos PIDs fornecidos pelo OBD-II. Na ausência do PID 94, o consumo pode ser calculado com base na velocidade do veículo (`VSS`) e na taxa de massa de ar (`MAF`). Quando o sensor `MAF` não está disponível, sua estimativa pode ser feita utilizando a pressão absoluta do coletor de admissão (`MAP`), a temperatura do ar de admissão (`IAT`) e as rotações por minuto (`RPM`), por meio da variável sintética `IMAP`:

$$IMAP = \frac{RPM \times MAP}{IAT/2}$$

Figura 8 – Formula IMAP

O valor de `MAF` pode então ser calculado como:

$$MAF = \frac{\frac{IMAP}{60} \times EV \times VDM \times MM}{R}$$

Figura 9 – Formula MAF

Por fim, o consumo de combustível em km/L é obtido pela fórmula:

$$\text{Consumo} = \frac{VSS}{3600} \times MAF^{-1} \times AC \times \mu g$$

Figura 10 – Formula Consumo

onde `EV` é a eficiência volumétrica, `VDM` é o volume de deslocamento do motor, `MM` é a massa molecular média do ar, `R` é a constante universal dos gases ideais, `AC` é a razão estequiométrica de ar/combustível Ug é a densidade da gasolina.

4 Análise dos Dados e Resultados

Neste capítulo, apresentamos a análise da base de dados utilizada para prever o consumo de combustível, a execução dos modelos de classificação, a análise dos experimentos realizados e, por fim, uma comparação entre os modelos propostos.

4.1 Análise Exploratória dos Dados

Inicialmente, foi realizada uma análise exploratória dos dados para compreender as características da base e identificar possíveis problemas, como valores ausentes e correlações entre variáveis.

4.1.1 Descrição das Variáveis da Base de Dados

A base de dados utilizada no experimento contém as seguintes variáveis, com suas respectivas descrições:

- **TIME**: Representa o timestamp da coleta, indicando o momento exato de cada registro((SAE), 2021)
- **DATA**: Data específica em que o dado foi coletado. ((SAE), 2021)
- **LATITUDE e LONGITUDE**: Coordenadas geográficas do veículo no momento da coleta. ((SAE), 2021)
- **ALTITUDE**: Altura em relação ao nível do mar, medida em metros. ((SAE), 2021)
- **VEHICLE_ID**: Identificação única do veículo monitorado. ((SAE), 2021)
- **BAROMETRIC_PRESSURE**: Pressão barométrica medida no ambiente, em kPa. ((SAE), 2021)
- **ENGINE_COOLANT_TEMP**: Temperatura do líquido de arrefecimento do motor, em graus Celsius. ((SAE), 2021)
- **FUEL_LEVEL**: Nível de combustível no tanque, como uma porcentagem do total. ((SAE), 2021)

- **ENGINE_LOAD**: Carga do motor, representando o esforço relativo que o motor está realizando. ((SAE), 2021)
- **AMBIENT_AIR_TEMP**: Temperatura do ar ambiente, em graus Celsius. ((SAE), 2021)
- **ENGINE_RPM**: Rotação do motor, medida em rotações por minuto (RPM). ((SAE), 2021)
- **INTAKE_MANIFOLD_PRESSURE**: Pressão no coletor de admissão, medida em kPa. ((SAE), 2021)
- **MAF (Mass Air Flow)**: Fluxo de massa de ar, indicando a quantidade de ar que entra no motor, em gramas por segundo. ((SAE), 2021)
- **Term Fuel Trim Bank 1**: Ajuste de combustível de curto prazo para o banco de cilindros 1. ((SAE), 2021)
- **FUEL_ECONOMY**: Economia de combustível, expressa em quilômetros por litro (km/L). ((SAE), 2021)
- **Long Term Fuel Trim Bank 2**: Ajuste de combustível de longo prazo para o banco de cilindros 2. ((SAE), 2021)
- **FUEL_TYPE**: Tipo de combustível utilizado (e.g., gasolina, etanol). ((SAE), 2021)
- **AIR_INTAKE_TEMP**: Temperatura do ar na entrada do sistema de admissão, em graus Celsius. ((SAE), 2021)
- **FUEL_PRESSURE**: Pressão do combustível no sistema, medida em kPa. ((SAE), 2021)
- **SPEED**: Velocidade do veículo, em km/h. ((SAE), 2021)
- **Short Term Fuel Trim Bank 2**: Ajuste de combustível de curto prazo para o banco de cilindros 2. ((SAE), 2021)
- **Short Term Fuel Trim Bank 1**: Ajuste de combustível de curto prazo para o banco de cilindros 1. ((SAE), 2021)
- **ENGINE_RUNTIME**: Tempo total de funcionamento do motor, em segundos. ((SAE), 2021)

- **THROTTLE_POS**: Posição da borboleta do acelerador, medida em percentual. ((SAE), 2021)
- **DTC_NUMBER**: Número de códigos de falha registrados pelo sistema OBD. ((SAE), 2021)
- **TROUBLE_CODES**: Lista dos códigos de falha registrados no veículo. ((SAE), 2021)
- **TIMING_ADVANCE**: Avanço de ignição, em graus. ((SAE), 2021)
- **EQUIV_RATIO**: Razão estequiométrica entre ar e combustível utilizada na combustão. ((SAE), 2021)

4.2 Carregamento e pré-processamento

Para utilização dos modelos é necessário realizar o carregamento e tratamento dos dados. Foi utilizado a biblioteca do pandas para carregar o arquivo CSV contendo a base de dados. De mesmo modo, foi utilizada a biblioteca do pandas para modelar os dados contidos na base. No processo de modelagem dos dados foi necessário atribuir tipos bem definidos para as variáveis da base. Fez-se necessário também preencher dados faltantes com valores padronizados, tratamento de valores ausentes, e com as variáveis numéricos efetuou-se uma conversão dos valores para padrão brasileiro numérico.

Para evidenciar a distribuição e porcentagem de valores nulos, foi gerado um gráfico com a faixa de valores faltantes para cada variável figura 11.

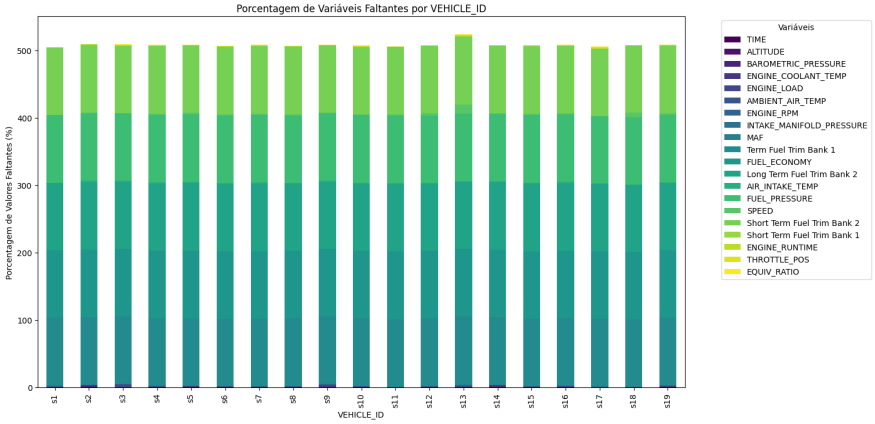


Figura 11 – Gráfico de valores faltantes para cada variável.

4.3 Identificação de valores ausentes

No processo de análise dos dados, o tratamento de valores ausentes foi realizado com o objetivo de garantir a consistência e qualidade dos dados para os modelos de aprendizado de máquina. Para isso, adotou-se a seguinte abordagem:

- **Substituição de valores ausentes:** Algumas colunas categóricas, como `TROUBLE_CODES`, tiveram os valores ausentes preenchidos com valores padrão (*e.g.*, `No Codes`). Este procedimento garante que as variáveis categóricas possam ser processadas adequadamente nos modelos.
- **Conversão de tipos:** Foram aplicadas conversões nas variáveis numéricas que apresentavam valores ausentes ou inconsistências, como formatações não numéricas. Durante este processo, valores inválidos foram convertidos para `NaN`, utilizando a função `pd.to_numeric` com o parâmetro `errors='coerce'`.
- **Remoção de colunas com muitos valores ausentes:** Durante a Análise Exploratória de Dados (EDA), identificou-se que as colunas `Term Fuel Trim Bank 1`, `FUEL_ECONOMY`, `Long Term Fuel Trim Bank 2`, `FUEL_PRESSURE` e `Short Term Fuel Trim Bank 2` apresentavam 100% de valores ausentes. Como essas colunas não fornecem informações relevantes para os modelos, foram removidas do conjunto de dados.
- **Remoção de linhas específicas:** Para algumas variáveis críticas, como `MAF`, foram removidas as linhas em que os valores estavam ausentes ou eram inválidos

(*e.g.*, strings vazias ou `None`). Este tratamento assegura que os dados de entrada dos modelos estejam completos para variáveis essenciais.

- **Imputação de valores ausentes:** Nos modelos de classificação, valores ausentes em variáveis numéricas foram tratados com a técnica de imputação utilizando a média (*mean imputation*). Essa etapa foi implementada por meio de um transformador no pipeline de pré-processamento.

A estratégia combinada de substituição, remoção e imputação assegura que os dados processados sejam consistentes e prontos para análise, minimizando possíveis impactos de valores ausentes no desempenho dos modelos.

4.4 Análise de correlações

Foi calculado a matriz de correlação entre as variáveis numéricas para identificar relações lineares, para cada trajeto presente na base de dados. A Figura 22 apresenta um mapa de calor da matriz de correlação para a amostra s1, permitindo visualizar a força e a direção das correlações. No capítulo 7 estão os mapas de correlação para as outras dezoito amostras.

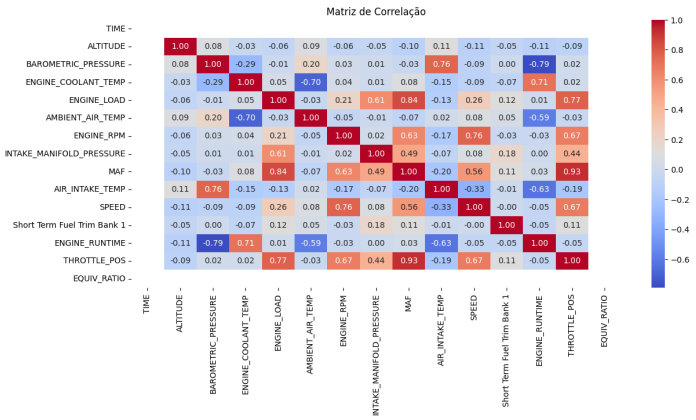


Figura 12 – Matriz de correlação para a amostra s1.

As variáveis com maior correlação foram essas listadas abaixo:

- **MAF** (Fluxo de Ar na Admissão) e **THROTTLE_POS** (Posição do Acelerador), com uma correlação de 0,93, indicando que o fluxo de ar aumenta com a abertura do acelerador.

- **ENGINE_LOAD** (Carga do Motor) e **MAF**, com uma correlação de 0,84, mostrando uma relação direta entre o fluxo de ar e a carga do motor.
- **ENGINE_LOAD** e **THROTTLE_POS**, com uma correlação de 0,77, indicando que a carga do motor é fortemente influenciada pela posição do acelerador.
- **ENGINE_RPM** (Rotação por Minuto) e **SPEED** (Velocidade), com uma correlação de 0,76, demonstrando que a rotação do motor aumenta proporcionalmente à velocidade do veículo.
- **AIR_INTAKE_TEMP** (Temperatura do Ar de Admissão) e **BAROMETRIC_PRESSURE**, com uma correlação de 0,76, evidenciando uma relação entre a temperatura do ar e a pressão atmosférica.
- **ENGINE_COOLANT_TEMP** (Temperatura do Líquido de Arrefecimento) e **ENGINE_RUNTIME**, com uma correlação de 0,71, indicando que o líquido de arrefecimento aquece conforme o motor opera por mais tempo.
- **AMBIENT_AIR_TEMP** (Temperatura do Ar Ambiente) e **ENGINE_COOLANT_TEMP**, com uma correlação negativa de -0,70, sugerindo que temperaturas externas mais baixas podem levar a uma menor temperatura do líquido de arrefecimento.
- **SPEED** e **THROTTLE_POS**, com uma correlação de 0,67, confirmando que a velocidade está diretamente relacionada à abertura do acelerador.
- **ENGINE_RPM** e **THROTTLE_POS**, com uma correlação de 0,67, reforçando a influência da posição do acelerador na rotação do motor.
- **ENGINE_RPM** e **MAF**, com uma correlação de 0,63, demonstrando a relação entre o fluxo de ar e a rotação do motor.
- **AIR_INTAKE_TEMP** e **ENGINE_RUNTIME**, com uma correlação negativa de -0,63, sugerindo que o aumento do tempo de operação reduz a temperatura do ar de admissão.
- **ENGINE_LOAD** e **INTAKE_MANIFOLD_PRESSURE** (Pressão do Coletor de Admissão), com uma correlação de 0,61, indicando que a carga do motor está relacionada à pressão no coletor de admissão.

Essas relações fornecem insights importantes sobre o comportamento dinâmico do veículo, trazendo pistas de como se caracteriza essa base de dados.

4.5 Seleção de Variáveis

Com base na análise de correlação e nos estudos sobre o domínio, selecionaram-se as variáveis mais relevantes para a modelagem: `ENGINE_LOAD`, `SPEED`, `ENGINE_RPM`, `INTAKE_MANIFOLD_PRESSURE`, `MAF` e `THROTTLE_POS`. A análise de correlação revelou fortes relações entre essas variáveis, indicando sua significativa influência na previsão do consumo de combustível. Além disso, essa influência é corroborada tanto pela análise estatística quanto pelo entendimento teórico do funcionamento do motor, justificando sua inclusão no processo de modelagem. Assim, essas variáveis serão utilizadas como elementos-chave na construção do modelo preditivo, dado seu papel central na eficiência energética e no desempenho veicular. Abaixo, na Figura 13, é apresentado um gráfico das correlações entre as variáveis selecionadas.

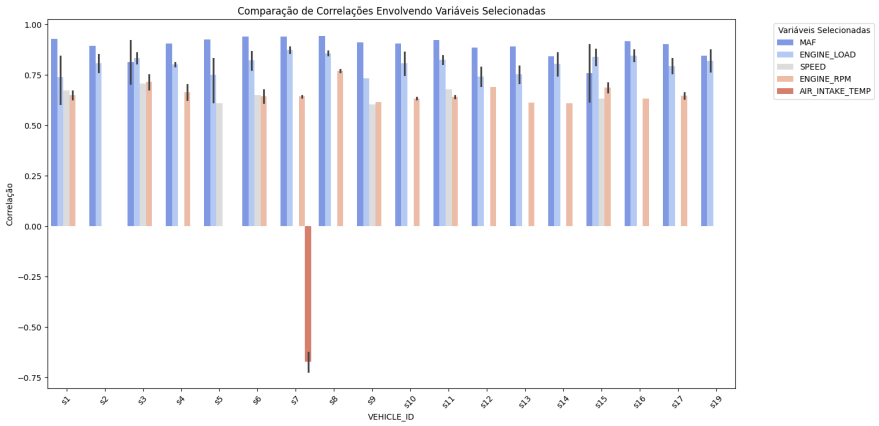


Figura 13 – Correlações entre as variáveis selecionadas para o experimento.

4.6 Análise das Principais Variáveis

As variáveis `ENGINE_LOAD`, `SPEED`, `ENGINE_RPM`, `MAF`, `THROTTLE_POS` e `INTAKE_MANIFOLD_PRESSURE` compartilham características relacionadas ao desempenho e funcionamento do motor de um veículo.

Primeiramente, todas essas variáveis descrevem aspectos críticos do comportamento do motor em operação. A variável `ENGINE_LOAD` representa a carga do motor, ou seja, a quantidade de trabalho que ele está realizando em um dado momento. A `SPEED` reflete a velocidade do veículo, que está diretamente relacionada à potência gerada pelo motor. A variável `ENGINE_RPM` refere-se à velocidade de rotação do motor,

medida em rotações por minuto, sendo um fator essencial para a entrega de potência e o consumo de combustível. Já a MAF (*Mass Air Flow*) mede o fluxo de ar que entra no motor, fundamental para a mistura ar-combustível necessária à combustão.

Além dessas, a variável THROTTLE_POS (*Throttle Position*) indica a abertura da borboleta de aceleração, que regula a quantidade de ar que entra no motor, afetando diretamente a combustão e o consumo de combustível. Por sua vez, a variável INTAKE_MANIFOLD_PRESSURE (*Pressão no Coletor de Admissão*) reflete a pressão do ar que entra na câmara de combustão, influenciando o desempenho do motor, a eficiência da queima do combustível e, consequentemente, o consumo.

Essas variáveis influenciam diretamente o consumo de combustível. A ENGINE_LOAD e a ENGINE_RPM impactam a quantidade de combustível necessária para manter o motor operando sob diferentes condições. A SPEED também reflete o consumo, uma vez que velocidades muito altas ou muito baixas podem aumentar a demanda de combustível. A MAF e a THROTTLE_POS determinam a quantidade de ar que entra no motor, regulando a mistura ar-combustível. Além disso, a INTAKE_MANIFOLD_PRESSURE pode indicar condições de carga e eficiência da combustão, contribuindo para a análise do desempenho do veículo.

Outro ponto em comum entre essas variáveis é sua interdependência no funcionamento do motor. Por exemplo, um aumento na ENGINE_LOAD geralmente leva a um aumento na ENGINE_RPM e na demanda de combustível. O MAF e a THROTTLE_POS influenciam diretamente a quantidade de ar disponível para combustão, enquanto a INTAKE_MANIFOLD_PRESSURE regula a pressão do ar admitido no motor, afetando sua eficiência energética.

Por fim, essas variáveis são amplamente utilizadas como indicadores do estado operacional do veículo. Seu monitoramento possibilita análises preditivas voltadas para a eficiência energética, consumo de combustível e diagnóstico de falhas no sistema do motor, permitindo uma compreensão aprofundada do desempenho e da saúde do veículo.

4.7 Definição da Variável Alvo

A escolha da variável alvo é um dos aspectos mais críticos na execução dos modelos de aprendizado de máquina, pois define o objetivo da análise e a qualidade dos resultados obtidos. Este capítulo discute as possíveis opções de variável alvo para

classificação do consumo de combustível, considerando as alternativas disponíveis nos dados coletados.

4.7.1 Variável *ENGINE_LOAD*

A variável *ENGINE_LOAD*, que representa a carga no motor, foi inicialmente considerada como uma possível variável alvo devido à sua relação indireta com o consumo de combustível. Essa variável fornece informações sobre a quantidade de esforço que o motor está realizando em um determinado momento, o que pode ser correlacionado com o consumo de combustível em situações específicas. No entanto, essa escolha apresenta limitações importantes:

- **Relação indireta com o consumo de combustível**: Embora o *ENGINE_LOAD* esteja relacionado à demanda no motor, ele não mede diretamente o consumo de combustível.
- **Sensibilidade a fatores externos**: A carga no motor pode ser influenciada por variáveis como inclinação da estrada, aceleração e condições ambientais, dificultando sua interpretação isolada.

4.7.2 Variáveis *FUEL_LEVEL* e *ENGINE_RUNTIME*

Outra abordagem considerada foi utilizar a combinação das variáveis *FUEL_LEVEL* e *ENGINE_RUNTIME* para inferir o consumo de combustível. Essa metodologia baseia-se na ideia de que a variação do nível de combustível ao longo do tempo pode ser interpretada como uma medida de consumo. Apesar de sua simplicidade, essa abordagem também possui limitações:

- **Imprecisão em cenários dinâmicos**: Em condições onde o veículo alterna frequentemente entre marcha lenta e aceleração, a relação entre tempo e consumo pode ser distorcida.
- **Dependência de medições precisas**: Pequenas variações ou erros na medição do nível de combustível podem impactar significativamente os resultados.

4.7.3 Estimativa baseada no *MAF*

Uma abordagem mais robusta para a definição da variável alvo envolve o uso da fórmula derivada do *Mass Air Flow* (MAF) para estimar o consumo de combustível em termos de litros por 100 km:

$$\text{Consumo (L/100 km)} = \frac{\text{Velocidade (km/h)} \times \text{Densidade do Combustível (g/L)}}{3600 \times \text{MAF (g/s)}}. \quad (4.1)$$

Essa metodologia apresenta várias vantagens:

- **Relação direta com o consumo de combustível**: O MAF mede diretamente a quantidade de ar que entra no motor, que está proporcionalmente ligada ao combustível consumido.
- **Adaptação a condições dinâmicas**: A inclusão da velocidade no cálculo ajusta a estimativa ao deslocamento real do veículo.

Por outro lado, essa abordagem também apresenta desafios, como a dependência de dados precisos do sensor MAF e a necessidade de ajustar os cálculos para diferentes tipos de combustível.

4.7.4 Impossibilidade de Utilizar *FUEL_ECONOMY*

Embora a variável *FUEL_ECONOMY* representasse a opção ideal para o objetivo de classificar o consumo de combustível, essa variável não possui valores disponíveis nos dados coletados. Isso inviabiliza sua utilização direta e justifica a busca por alternativas descritas neste capítulo.

4.7.5 Utilização do MAF

Após ponderar as alternativas, a estimativa baseada no MAF foi escolhida como a variável alvo mais adequada para este trabalho, devido à sua relação direta com o consumo de combustível e sua capacidade de refletir as condições reais de condução. As limitações das outras abordagens destacam a importância de selecionar uma variável alvo que seja representativa do fenômeno que se deseja modelar, mesmo que essa escolha exija cálculos adicionais para sua definição. Para segmento será gerada apartir desse cálculo uma variável denominada *FUEL_COMSUMPTION*.

4.8 Modelos de Classificação

Foram definidos 6 modelos de predição para o experimento, são eles: Random Forest, SVM, Decision Tree, KNN e Naive Bayes, MLP.

Como a variável alvo definida foi a FUEL_COMSUMPTION, que por sua vez é uma variável calculada, gerada pela fórmula 4.1, e o interesse depositado nesse modelo é que seja feita uma classificação dos dados, faz-se necessário que os dados contidos nela passem por um processo de discretização. Dessa forma, para entender quantos intervalos os valores contínuos dela serão melhor divididos, foi feita uma análise que como primeira etapa foi criado uma gráfico de distribuição dos valores, que pode ser visto abaixo na figura 14.

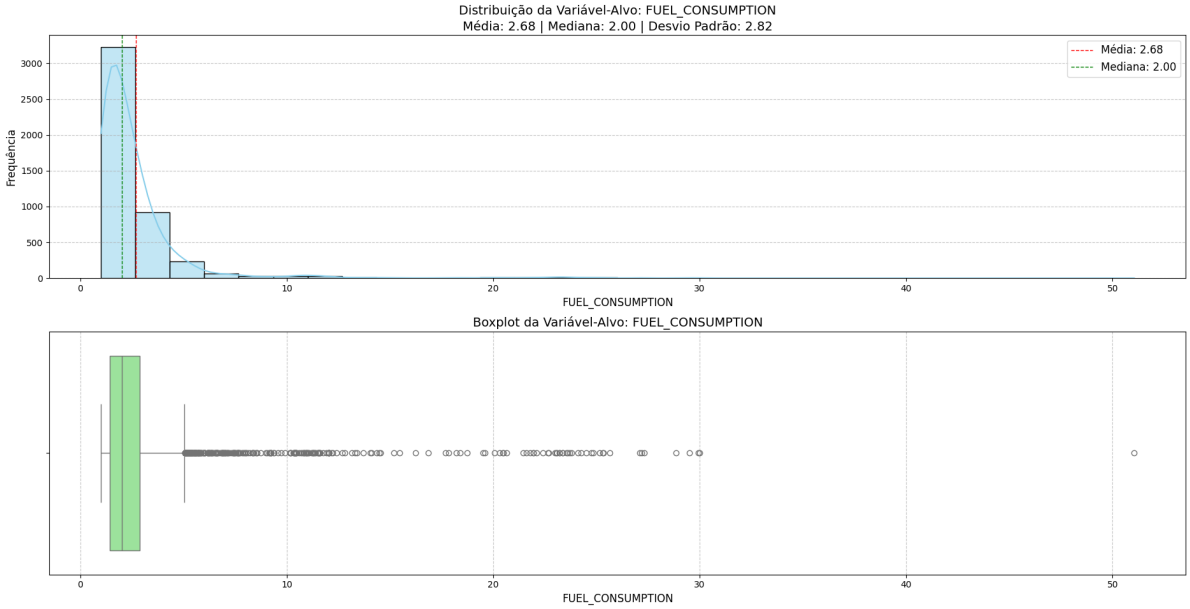


Figura 14 – Distribuição dos valores contínuos da variável FUEL_COMSUMPTION

Analisando o gráfico é perceptível que a distribuicão possui uma curva simples, com características de variação monomodal. Segundo (FASTERCAPITAL, 2024), tal variação trás pistas que a quantidade de intervalos seja próxima à valores comuns, como 2 ou 3. Para confirmar tal suspeita é importante testar as hipóteses, logo, posteriormente, serão executados os modelos de classificação variando a quantidade de núcleos.

4.8.1 Avaliação dos Modelos de Classificação

Após o execução dos modelos de classificação, foi realizada uma etapa de avaliação detalhada para determinar sua eficácia. A análise foi conduzida utilizando as seguintes métricas:

- **Mean Squared Error (MSE):** Mede o erro quadrático médio entre os valores previstos e reais, penalizando erros maiores. Um valor menor indica um melhor desempenho do modelo.
- **R-Squared (R^2):** Avalia a proporção da variância do alvo explicada pelos preditores. Valores próximos de 1 indicam que o modelo explica bem a variabilidade dos dados.
- **Mean Absolute Error (MAE):** Calcula o erro médio absoluto das previsões, fornecendo uma medida mais robusta às outliers em comparação ao MSE.
- **Accuracy:** Mede a proporção de previsões corretas em relação ao total de previsões. É especialmente útil em classificações discretas.
- **F1-Score:** Combina precisão e recall em uma única métrica, sendo particularmente útil em situações de desequilíbrio entre classes, ao ponderar a relevância de falsos positivos e falsos negativos.

A avaliação foi realizada utilizando técnicas de validação cruzada, dividindo o conjunto de dados em múltiplos subconjuntos de treinamento e teste, o que garante maior generalização dos resultados.

4.9 Discretização e Balanceamento da Base de Dados

A análise de consumo de combustível requer técnicas específicas para lidar com as particularidades dos dados automotivos, como a heterogeneidade e a pre-

sença de desbalanceamento nas classes, quando os dados são categorizados. Para este trabalho, foram utilizadas as técnicas de *discretização* e *balanceamento*, que desempenharam papéis importantes na preparação da base de dados.

4.9.1 Discretização dos Dados

A discretização foi realizada utilizando o método *KBinsDiscretizer*, uma técnica que permite dividir uma variável contínua em classes distintas, tornando possível a aplicação de métodos supervisionados que dependem de dados categóricos. O modelo configurado utilizou o parâmetro *strategy* como *uniform*, garantindo que os intervalos fossem igualmente espaçados. O número de classes (*bins*) foi variado entre 2, 3, 5 e 10 para avaliar o impacto dessa granularidade no desempenho dos modelos. Tal abordagem é amplamente reconhecida na literatura como uma estratégia eficaz para melhorar a expressividade de variáveis contínuas em algoritmos baseados em categorias (LIU et al., 1998; KOTSIANTIS; KANELLOPOULOS, 2006).

4.9.2 Balanceamento dos Dados

Outro desafio encontrado foi o desbalanceamento nas classes geradas após a discretização. Para resolver esse problema, utilizou-se a técnica de *Synthetic Minority Oversampling Technique* (SMOTE), que realiza a sobresampling das classes minoritárias por meio da criação de exemplos sintéticos (CHAWLA et al., 2002b). Esta abordagem é particularmente útil em problemas onde há uma disparidade significativa entre as frequências das classes, permitindo melhorar o desempenho dos modelos preditivos em termos de métricas como *accuracy* e *F1-Score*.

O processo de balanceamento foi realizado sobre os dados discretizados, garantindo que todas as classes tivessem a mesma representatividade. Estudos anteriores demonstram que a combinação de discretização e balanceamento pode levar a melhorias substanciais no desempenho de modelos de aprendizado de máquina, especialmente em problemas com variáveis contínuas e desbalanceamento de classes (DING; ZHANG, 2008).

4.9.3 Impacto no Treinamento dos Modelos

A utilização das técnicas de discretização e balanceamento impactou positivamente o treinamento dos modelos, e isto será evidenciado nos resultados das métricas de desempenho, apresentadas na próxima seção. A aplicação dessas técnicas garan-

tiu que os modelos pudessem generalizar melhor, evitando viés em favor das classes majoritárias e melhorando a robustez das predições.

Em suma, a combinação de discretização e balanceamento da base de dados provou ser uma etapa essencial para a construção de modelos mais eficientes e confiáveis, demonstrando sua relevância na análise de dados automotivos.

4.10 Análise dos Resultados de LDA e PCA

Nesta subseção, são apresentados os resultados das análises realizadas utilizando os métodos de redução de dimensionalidade PCA (*Principal Component Analysis*) e LDA (*Linear Discriminant Analysis*). Esses métodos têm como objetivo compreender a estrutura dos dados e avaliar sua separabilidade em diferentes dimensões.

4.10.1 Resultados do PCA

A análise com PCA revelou que a primeira componente principal explica **46%** da variância nos dados, enquanto a segunda componente explica **25%**. Esses valores indicam que, embora a maior parte da variabilidade esteja concentrada nas primeiras componentes, as informações estão distribuídas por múltiplas dimensões. Esse resultado sugere que os dados possuem relações complexas entre as variáveis, dificultando sua separação em espaços de baixa dimensionalidade.

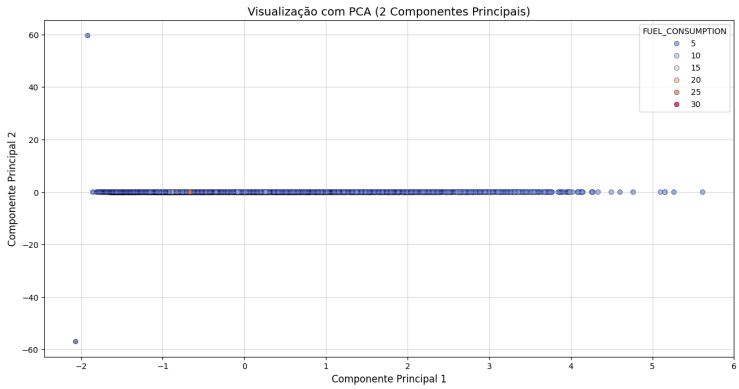


Figura 15 – Gráfico da variância explicada por cada componente principal.

O gráfico da Figura 15 mostra a variância explicada por cada componente principal. Observa-se uma diminuição gradual na contribuição das componentes após a

segunda, indicando que estas têm relevância menor para a descrição dos dados.

4.10.2 Resultados do LDA

Com o LDA, foi possível verificar que a primeira componente discriminante explica **100%** da variância associada às classes, enquanto a segunda componente não possui contribuição relevante (**0%**). Esse resultado indica que os dados são **linearmente separáveis** na projeção discriminante.

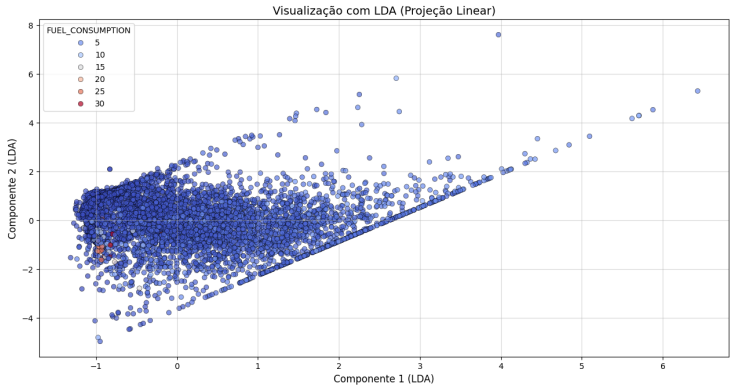


Figura 16 – Projeção dos dados no espaço discriminante definido pelo LDA.

A Figura 16 apresenta a projeção dos dados no espaço definido pela primeira e segunda componentes discriminantes. Nota-se que as classes apresentam uma separação clara, validando a capacidade do LDA de reduzir a dimensionalidade mantendo a separabilidade entre as classes.

Os resultados do PCA e do LDA evidenciam diferenças importantes na estrutura dos dados:

- O PCA revela que os dados possuem múltiplas dimensões significativas, o que pode dificultar a utilização de algoritmos que assumem linearidade simples, como o *Support Vector Machine (SVM)*.
- O LDA, por outro lado, demonstra que os dados são linearmente separáveis no espaço discriminante, indicando que algoritmos baseados em árvores, como *Random Forest* e *Decision Tree*, são mais adequados para explorar essa característica.

4.11 Análise Comparativa de Intervalos e Estratégia de Separação dos Dados

As figuras a seguir apresentam as métricas de desempenho (*Accuracy*, *F1-Score*, *MAE*, *MSE*, e R^2) para diferentes modelos com quatro configurações de intervalos (2, 3, 5 e 10). Foi realizado também variações referentes a distribuição do conjunto treino|teste, é importante garantir que a melhor distribuição seja definida, logo foram aplicadas as variações, *split-percentage* 80% | 20%, *split-percentage* 70% | 30%, *split-percentage* 60% | 40%, *split-percentage* 50% | 50%.

4.11.1 Acurácia (Accuracy)

A figura 1 apresenta os resultados de acurácia dos modelos para diferentes configurações de Bins e Teste. A Random Forest obteve a maior acurácia com valores variando entre 0.8909 e 0.9654, enquanto o Naive Bayes apresentou os menores valores, variando de 0.4986 a 0.7751, conforme ilustrado na Figura 17.

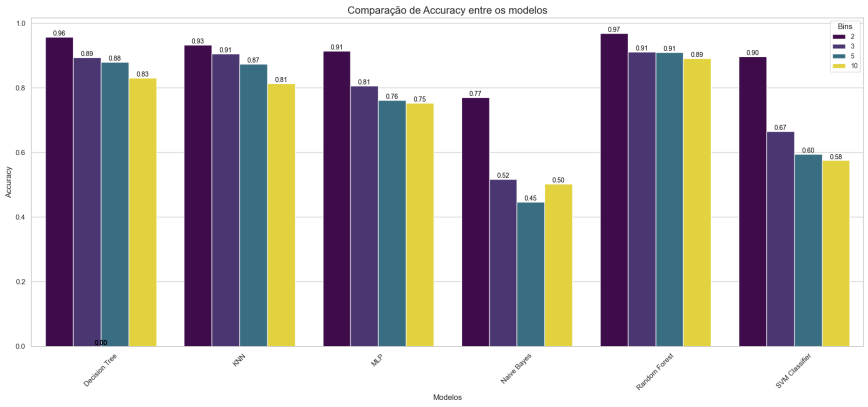


Figura 17 – Comparação de Acurácia entre os Modelos.

4.11.2 F1-Score

Os valores de F1-Score são apresentados na Tabela 2. A Random Forest também se destacou com valores variando entre 0.7505 e 0.9654, enquanto o Naive Bayes teve os menores valores, variando de 0.3508 a 0.7971. A Figura 18 ilustra esses resultados.

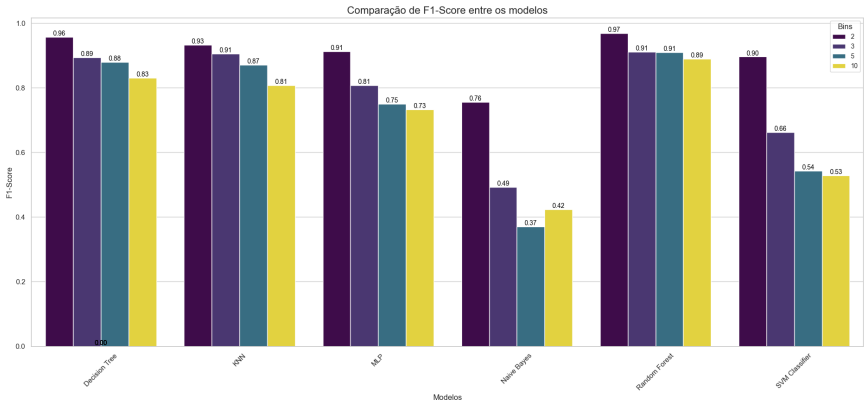


Figura 18 – Comparação de F1-Score entre os Modelos.

4.11.3 Erro Absoluto Médio (MAE)

A Tabela 3 apresenta os valores de MAE dos modelos. O Random Forest obteve os menores valores, variando entre 0.0356 e 0.0530, enquanto o Naive Bayes apresentou os maiores valores, variando de 0.1949 a 0.9351, conforme ilustrado na Figura 19.

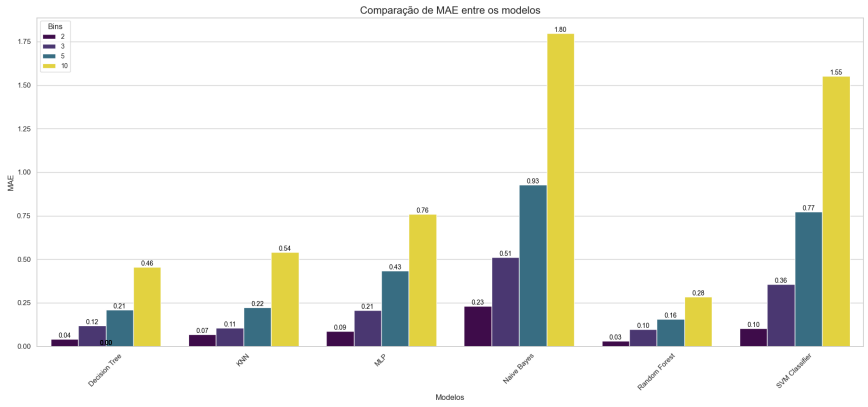


Figura 19 – Comparação de MAE entre os Modelos.

4.11.4 Erro Quadrático Médio (MSE)

Os valores de MSE são apresentados na Tabela 4. O Random Forest obteve os menores valores, variando entre 0.0229 e 0.0286, enquanto o Naive Bayes teve os

maiores valores, variando de 0.1949 a 1.8020. A Figura 20 ilustra esses resultados.

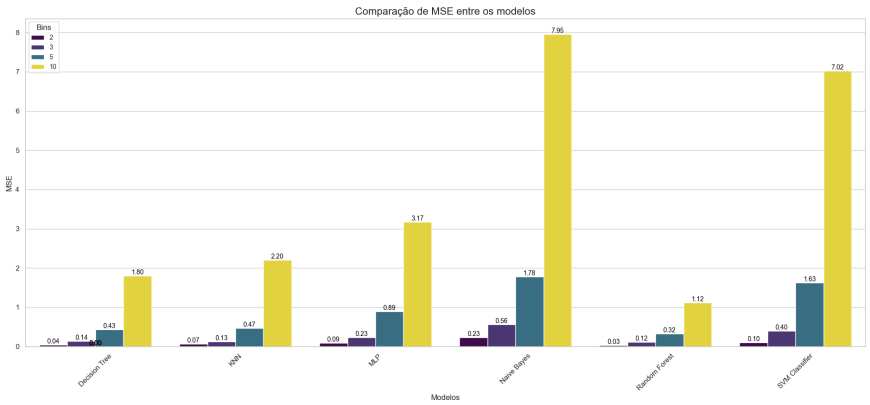


Figura 20 – Comparação de MSE entre os Modelos.

4.11.5 Coeficiente de Determinação (R^2)

A Tabela 5 apresenta os valores de R^2 dos modelos. O Random Forest obteve os melhores valores, variando de 0.8950 a 0.9260, enquanto o Naive Bayes apresentou os piores valores, variando de -0.0999 a 0.2202, conforme ilustrado na Figura 21.

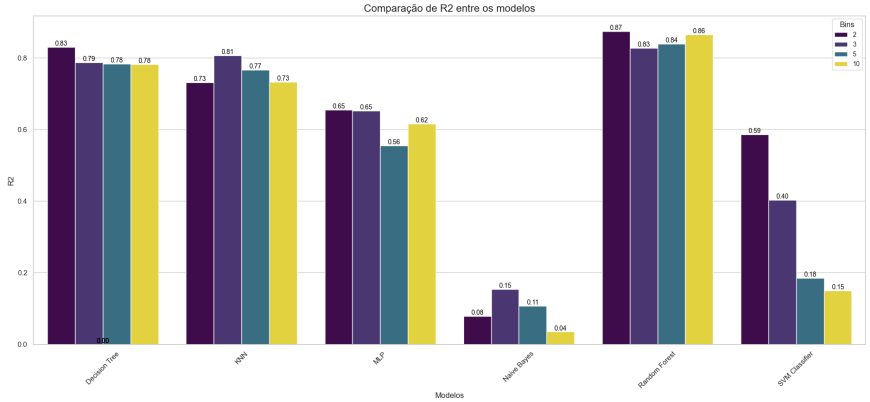


Figura 21 – Comparação de R^2 entre os Modelos.

Tabela 1 – Acurácia dos modelos para diferentes valores de intervalos (Bins)

Bins	Intervalo	Decision Tree	KNN	MLP	Naive Bayes	Random Forest	SVM Classifier
2	0.2	0.954	0.931	0.899	0.732	0.976	0.892
	0.3	0.959	0.921	0.915	0.760	0.974	0.899
	0.4	0.966	0.940	0.908	0.779	0.971	0.896
	0.5	0.951	0.940	0.932	0.808	0.953	0.898
3	0.2	0.890	0.884	0.793	0.508	0.897	0.664
	0.3	0.895	0.887	0.825	0.523	0.910	0.657
	0.4	0.902	0.927	0.783	0.503	0.932	0.678
	0.5	0.887	0.925	0.823	0.532	0.905	0.665
5	0.2	0.888	0.921	0.714	0.430	0.932	0.553
	0.3	0.856	0.848	0.774	0.453	0.904	0.623
	0.4	0.859	0.844	0.743	0.473	0.899	0.552
	0.5	0.918	0.880	0.818	0.431	0.906	0.654
10	0.2	0.855	0.812	0.698	0.508	0.864	0.563
	0.3	0.835	0.824	0.765	0.491	0.845	0.573
	0.4	0.839	0.831	0.752	0.515	0.906	0.566
	0.5	0.795	0.785	0.795	0.500	0.901	0.600

Tabela 2 – F1-Score dos modelos para diferentes valores de intervalos (Bins)

Bins	Intervalo	Decision Tree	KNN	MLP	Naive Bayes	Random Forest	SVM Classifier
2	0.2	0.95376028	0.93080026	0.89921859	0.71122107	0.97591588	0.89216955
	0.3	0.95904863	0.92090999	0.91448966	0.74593852	0.97421705	0.89869629
	0.4	0.96550416	0.93945404	0.90817537	0.76790561	0.97091282	0.89556709
	0.5	0.95135902	0.93973229	0.93145761	0.79976171	0.95277245	0.89796806
3	0.2	0.88947873	0.88401826	0.79519552	0.48540514	0.89733836	0.65916968
	0.3	0.89453387	0.88724102	0.82598985	0.49685574	0.91039182	0.65373423
	0.4	0.90208642	0.92684682	0.78392082	0.4792569	0.93157505	0.67573846
	0.5	0.88720876	0.92544639	0.82486407	0.50659237	0.9047105	0.6609991
5	0.2	0.88723058	0.91958159	0.70123104	0.35201939	0.93177142	0.4861731
	0.3	0.85487717	0.84616864	0.76009288	0.37812311	0.90252626	0.57920562
	0.4	0.85860328	0.84221812	0.73009687	0.399968	0.89819634	0.48492074
	0.5	0.91745021	0.87771896	0.81097339	0.35299353	0.90501241	0.61934295
10	0.2	0.85410009	0.80732717	0.66932289	0.42735894	0.86103574	0.51267253
	0.3	0.83388	0.81894435	0.74608789	0.40966026	0.84143329	0.52441381
	0.4	0.83807264	0.82604932	0.73257729	0.43923418	0.90369448	0.50887111
	0.5	0.79420827	0.77916346	0.78175975	0.4210192	0.95010905	0.56701234

Tabela 3 – MAE dos modelos para diferentes valores de intervalos (Bins)

Bins	Intervalo	Decision Tree	KNN	MLP	Naive Bayes	Random Forest	SVM Classifier
2	0.2	0.04623935	0.06913672	0.10055576	0.26772879	0.02408299	0.1075213
	0.3	0.04094838	0.07903186	0.08530501	0.23991109	0.02578414	0.1010126
	0.4	0.03449426	0.06046684	0.09155243	0.22108188	0.02908485	0.10414969
	0.5	0.04864384	0.06017489	0.06835631	0.19244108	0.04722099	0.10170446
3	0.2	0.12205333	0.12773486	0.22028408	0.51906305	0.11188637	0.35923249
	0.3	0.11669713	0.12606745	0.18710749	0.50483469	0.09789001	0.36637315
	0.4	0.10883489	0.08391277	0.23145171	0.52498442	0.07671028	0.3397134
	0.5	0.12575017	0.08597348	0.18811684	0.49319111	0.10443625	0.35687369
5	0.2	0.19654493	0.13408092	0.52517048	0.93753599	0.11495681	0.83939991
	0.3	0.25578341	0.27081523	0.41226387	0.92740681	0.17056268	0.72819477
	0.4	0.24865889	0.28046674	0.47246553	0.91007728	0.17590544	0.84415821
	0.5	0.13947145	0.21072857	0.32949448	0.93153109	0.16370469	0.67843375
10	0.2	0.38200302	0.54547135	0.99164846	1.78916149	0.36474542	1.60737691
	0.3	0.44605388	0.49953509	0.7004313	1.8306055	0.41890999	1.56582087
	0.4	0.42690304	0.48931272	0.76562763	1.77050916	0.2431104	1.60478071
	0.5	0.57074105	0.63005881	0.58198622	1.79969753	0.11181986	1.43110738

Tabela 4 – MSE dos modelos para diferentes valores de intervalos (Bins)

Bins	Intervalo	Decision Tree	KNN	MLP	Naive Bayes	Random Forest	SVM Classifier
2	0.2	0.0462	0.0691	0.1006	0.2677	0.0241	0.1075
	0.3	0.0409	0.0790	0.0853	0.2399	0.0258	0.1010
	0.4	0.0345	0.0605	0.0916	0.2211	0.0291	0.1041
	0.5	0.0486	0.0602	0.0684	0.1924	0.0472	0.1017
3	0.2	0.1458	0.1506	0.2475	0.5729	0.1304	0.4048
	0.3	0.1393	0.1521	0.2121	0.5612	0.1144	0.4134
	0.4	0.1307	0.1052	0.2600	0.5813	0.0933	0.3754
	0.5	0.1516	0.1087	0.2113	0.5434	0.1227	0.4002
5	0.2	0.4049	0.2763	1.0780	1.7772	0.2336	1.7538
	0.3	0.5343	0.5658	0.8405	1.7955	0.3538	1.5402
	0.4	0.5163	0.5896	0.9718	1.7815	0.3613	1.7752
	0.5	0.2840	0.4378	0.6720	1.7699	0.3382	1.4474
10	0.2	1.4906	2.2149	4.2728	7.9458	1.4432	7.3037
	0.3	1.7702	2.0062	2.8904	8.0834	1.6543	7.1121
	0.4	1.6776	1.9926	3.1989	7.8446	0.9511	7.3180
	0.5	2.2778	2.5795	2.3158	7.9444	0.4116	6.3630

Tabela 5 – (R^2) dos modelos para diferentes valores de intervalos (Bins)

Bins	Intervalo	Decision Tree	KNN	MLP	Naive Bayes	Random Forest	SVM Classifier
2	0.2	0.81500889	0.72326717	0.59759906	-0.07124629	0.90364201	0.56967477
	0.3	0.83619724	0.68376915	0.6587301	0.04025574	0.89683523	0.59575376
	0.4	0.86201558	0.75810555	0.63375643	0.1156521	0.8836458	0.58333661
	0.5	0.80536099	0.75928581	0.72656005	0.23021488	0.81110357	0.59311887
3	0.2	0.78196867	0.77276377	0.63094938	0.1478382	0.80533651	0.39460919
	0.3	0.79081438	0.77197966	0.68124733	0.15524652	0.8274547	0.38161751
	0.4	0.80429641	0.84233704	0.61200078	0.13053939	0.86103138	0.43698343
	0.5	0.77217141	0.83709216	0.68303046	0.1830249	0.81573378	0.39846135
5	0.2	0.79780949	0.86300475	0.46420867	0.10597702	0.88392732	0.11774706
	0.3	0.733826	0.71788109	0.57962173	0.09990008	0.82222314	0.22955486
	0.4	0.74226206	0.70497951	0.51387124	0.10771305	0.81971923	0.11585292
	0.5	0.85820489	0.7800416	0.66399947	0.11600356	0.83071789	0.27660197
10	0.2	0.81916691	0.73090404	0.48114528	0.03694809	0.82461162	0.11779872
	0.3	0.78476976	0.75696518	0.64872882	0.01679408	0.79946038	0.14201553
	0.4	0.79759601	0.7579103	0.61294035	0.04882406	0.88448925	0.11059069
	0.5	0.7246196	0.6870078	0.71876584	0.03789974	0.9501608	0.23114525

- **Intervalo 2:**
  - Melhores resultados foram observados com 20% dos dados no conjunto de teste:
    - **Random Forest**: Accuracy entre 0,961 e 0,970, F1-Score alto, MAE em torno de 0,05, MSE baixo e R^2 alto.
    - **Decision Tree**: Accuracy em torno de 0,953 a 0,964, MAE em torno de 0,046 a 0,042, MSE baixo e R^2 alto.
    - **Naive Bayes**: Resultados variáveis, com menores precisões e MAEs mais elevados.
- **Intervalo 3:**
  - Melhores resultados foram observados com 20% e 40% dos dados no conjunto de teste:
    - **Random Forest**: Accuracy até 0,952, F1-Score consistente, MAE e MSE baixos, R^2 alto (até 0,926).
    - **Decision Tree**: Accuracy em torno de 0,903 a 0,896, MAE e MSE moderados.
    - **Naive Bayes**: Desempenho inferior, especialmente com porcentagens maiores no conjunto de teste.
- **Intervalo 5:**
  - Melhores resultados foram observados com 40% dos dados no conjunto de teste:
    - **Random Forest**: Accuracy de 0,872, F1-Score alto, MAE em torno de 0,198, MSE relativamente baixo, R^2 positivo (até 0,926).
    - **Decision Tree**: Accuracy até 0,896, com MAE em torno de 0,179, MSE moderado.
    - **Naive Bayes**: Precisão e R^2 baixos, com MAE e MSE mais altos.
- **Intervalo 10:**
  - Melhores resultados foram observados com 30% dos dados no conjunto de teste:
    - **Random Forest**: Accuracy em torno de 0,834, F1-Score alto, MAE em torno de 0,308, MSE relativamente baixo, R^2 positivo.

- **Decision Tree**: Accuracy até 0,853, com MAE e MSE moderados.
- **Naive Bayes**: Precisão e R^2 mais baixos, com maiores MAE e MSE.

A priori a melhor combinação observada foi utilizando o modelo Random Forest no Intervalo 2 com 20% dos dados no conjunto de teste, apresentando os melhores resultados em termos de acurácia, F1-Score, MAE, MSE e R^2.

4.12 Validação Cruzada

A validação cruzada foi utilizada como técnica principal para avaliar o desempenho dos modelos empregados neste estudo. O método adotado consistiu na validação cruzada com repetição (*repeated cross-validation*), configurada para cinco repetições, cada uma com divisões distintas dos dados em conjuntos de treinamento e teste. Este procedimento foi essencial para garantir maior confiabilidade e robustez nos resultados, minimizando vieses decorrentes de particularidades da divisão inicial dos dados.

Tabela 6 – Resultados da validação cruzada para os modelos testados.

Modelo	**MSE**	**R^2**	**MAE**	**Acurácia**	**F1-Score**
Decision Tree	0.605843	0.795381	0.206864	0.890432	0.890088
KNN	0.715500	0.759206	0.234561	0.881132	0.879476
MLP	1.094809	0.619197	0.371989	0.808622	0.800966
Naive Bayes	2.632648	0.093849	0.866235	0.559079	0.510832
Random Forest	0.395877	0.851256	0.142551	0.920262	0.919476
SVM Classifier	2.288846	0.330929	0.695991	0.683260	0.657291

Conforme os resultados apresentados, o modelo *Random Forest* destacou-se como o de melhor desempenho global. Este modelo apresentou os melhores resultados em todas as métricas. Adicionalmente, o modelo *Decision Tree* apresentou métricas próximas, destacando-se como uma alternativa eficiente, especialmente em termos de Acurácia e F1-Score.

A validação cruzada permitiu avaliar de forma robusta os modelos propostos, garantindo que o desempenho observado reflita sua capacidade de generalização em novos dados.

5 Análise dos Resultados

Os experimentos realizados neste trabalho tiveram como principal objetivo avaliar o desempenho de algoritmos de *Machine Learning* na tarefa de classificação do consumo de combustível em veículos automotores. A meta foi explorar como essas técnicas podem identificar padrões associados a diferentes níveis de consumo, utilizando dados automotivos coletados por meio de varreduras OBD (*On-Board Diagnostics*).

5.1 Análise dos Resultados dos Modelos

Os resultados indicam que a melhor divisão entre treinamento e teste foi de 80% para treinamento e 20% para teste, utilizando o modelo Random Forest no Intervalo 2. Essa configuração apresentou a maior acurácia (até 0,970), F1-Score elevado, além de baixos valores de MAE e MSE, garantindo um modelo robusto e com alta capacidade preditiva para o consumo de combustível.

As métricas também evidenciaram que o modelo *Random Forest* apresentou o melhor desempenho nos experimentos realizados. Essa performance pode ser atribuída à sua capacidade de lidar com bases de dados multidimensionais e à robustez no tratamento de dados com características não lineares. Apesar da análise realizada com o LDA indicar que os dados são linearmente separáveis, o *Random Forest* foi capaz de capturar interações mais complexas entre as variáveis independentes, como `MAF`, `ENGINE_LOAD` e `THROTTLE_POS`, e a variável dependente `FUEL_CONSUMPTION`. Além disso, o modelo obteve métricas superiores, como alta acurácia e *F1-Score*, que são essenciais para problemas de classificação com dados desbalanceados (SOKOLOVA; LAPALME, 2009).

Por outro lado, o modelo *Naive Bayes* apresentou o pior desempenho, o que pode ser explicado pelas simplificações inerentes ao algoritmo. Conforme Domingos e Pazzani (1997), o *Naive Bayes* assume independência condicional entre as variáveis preditoras e geralmente utiliza distribuições probabilísticas gaussianas. No entanto, essas premissas raramente correspondem à realidade, especialmente em bases de dados com variáveis correlacionadas, como a utilizada neste trabalho, onde variáveis como `ENGINE_LOAD` e `INTAKE_MANIFOLD_PRESSURE` apresentam uma relação clara. Além disso, o método de balanceamento de classes com o Schölkopf e Smola (2002)

introduziu exemplos sintéticos que o *Naive Bayes* não conseguiu incorporar de forma eficiente, o que comprometeu ainda mais o seu desempenho.

A análise com o LDA revelou que a primeira componente discriminante explica 100% da variância entre as classes, indicando que os dados são linearmente separáveis na projeção discriminante. Embora essa característica favoreça algoritmos lineares, como o LDA, ela não exclui a vantagem de modelos como o *Random Forest*, que exploram a dimensionalidade dos dados e capturam padrões mais complexos. De acordo com Hastie, Tibshirani e Friedman (2009), o *Random Forest* tem uma vantagem inerente ao modelar relações não lineares e interações de alta ordem, características frequentemente presentes em bases de dados reais.

As métricas utilizadas, como *F1-Score* e acurácia, são fundamentais para avaliar o desempenho dos modelos em bases de dados desbalanceadas. O *F1-Score*, por exemplo, é particularmente relevante porque combina precisão e sensibilidade em uma única métrica, sendo menos suscetível a resultados distorcidos em cenários onde uma classe domina (SOKOLOVA; LAPALME, 2009). No entanto, o balanceamento realizado com o SMOTE teve um impacto significativo nos resultados. Embora tenha corrigido o desbalanceamento inicial das classes, igualando a quantidade de instâncias em cada classe, essa técnica também introduziu dados sintéticos que podem não refletir completamente a realidade, afetando a generalização dos modelos, especialmente os mais sensíveis, como o *Naive Bayes*.

Em resumo, o *Random Forest* mostrou-se altamente eficaz para a tarefa proposta, aproveitando tanto a separabilidade linear dos dados quanto a sua dimensionalidade. Por outro lado, o desempenho limitado do *Naive Bayes* evidencia as limitações de algoritmos baseados em pressupostos simplificados, especialmente em bases complexas e balanceadas artificialmente. Esses resultados ressaltam a importância de alinhar a escolha do modelo às características da base de dados e às especificidades do problema.

5.2 Relação da Discretização e Balanceamento com os Resultados

Outra etapa fundamental desse processo foi a discretização da variável contínua `FUEL_CONSUMPTION`, medida em litros por 100 km. Essa transformação foi necessária para converter o problema de regressão em um problema de classificação,

permitindo a divisão do consumo de combustível em categorias ou classes que representassem diferentes níveis de consumo, como “baixo”, e “alto”. Essa abordagem foi importante, pois possibilitou uma análise mais interpretável e facilitou a identificação de padrões específicos dentro de cada faixa de consumo.

Além de identificar os modelos mais adequados para a classificação, os experimentos buscaram também avaliar técnicas de balanceamento de classes, como o *SMOTE* (*Synthetic Minority Oversampling Technique*). Esse balanceamento foi crucial devido à distribuição desbalanceada das classes da variável-alvo após a discretização. A Tabela 7 ilustra como as classes estavam distribuídas antes e depois do balanceamento:

Tabela 7 – Distribuição das classes antes e após o balanceamento.

Classe	**Distribuição Antes do Balanceamento**	**Distribuição Após o Balanceamento**
Baixo	6.747	6.747
Alto	52	6.747

Como pode ser observado, antes do balanceamento havia uma prevalência significativa da classe “baixo consumo”, enquanto a classe “alto consumo” era fortemente sub-representada. Essa disparidade poderia levar os modelos a um comportamento enviesado, classificando a maioria dos exemplos como pertencentes à classe majoritária, o que comprometeria sua eficácia.

Ao aplicar o *SMOTE*, foi possível gerar exemplos sintéticos para as classes minoritárias, equilibrando a distribuição das classes e permitindo que os algoritmos de classificação aprendessem padrões de consumo de forma mais consistente e eficaz. Essa etapa foi determinante para melhorar a precisão dos modelos e aumentar sua capacidade de generalizar para dados reais, particularmente para cenários associados ao consumo elevado, que são de maior interesse prático.

No entanto, é importante ressaltar que o objetivo inicial do trabalho, de explorar o uso de algoritmos para fornecer insights sobre o consumo de combustível, foi parcialmente alcançado. Pois, o conjunto de variáveis utilizadas no experimento não contemplou fatores externos que podem influenciar significativamente o consumo de combustível, como a topografia do terreno, o tipo de combustível utilizado, o peso do veículo ou o perfil de condução do motorista. A ausência dessas variáveis limita a generalização dos resultados para contextos reais, onde o consumo de combustível é afetado por um conjunto muito mais amplo e complexo de fatores.

Além disso, a escolha de discretizar uma variável originalmente contínua também possui implicações. Embora essa abordagem tenha sido necessária para transformar o problema em um contexto de classificação, ela resultou na perda de detalhes que poderiam fornecer informações mais ricas sobre o consumo de combustível. Uma análise futura poderia considerar a aplicação de modelos de regressão diretamente na variável contínua para avaliar se melhores insights podem ser extraídos sem a necessidade de discretização.

Por fim, a distribuição da variável-alvo `FUEL_CONSUMPTION`, após a discretização, apresentou um desbalanceamento considerável entre as classes. Conforme mostrado na Tabela 7, a classe “baixo consumo” foi amplamente dominante, enquanto a classe “alto consumo” estava sub-representada. Esse desbalanceamento trouxe desafios significativos, exigindo a aplicação de técnicas de balanceamento de dados, como o *SMOTE*.

Embora o *SMOTE* tenha sido eficaz para equilibrar as classes e melhorar a capacidade dos modelos de aprender padrões das classes minoritárias, a geração de exemplos sintéticos pode introduzir vieses nos dados. Isso ocorre porque os exemplos criados são baseados em combinações de dados existentes, o que pode levar a um aprendizado que não reflete completamente a complexidade dos cenários reais. Além disso, o balanceamento forçado dos dados pode causar uma simplificação excessiva do problema, mascarando as características que tornam as classes de consumo mais elevadas mais raras e, consequentemente, mais complexas.

Esses fatores indicam que, embora os experimentos tenham demonstrado a viabilidade de se utilizar *Machine Learning* para classificar o consumo de combustível, o trabalho ainda possui limitações que precisam ser abordadas para se obter uma compreensão mais ampla e precisa do problema.

6 Considerações Finais

Este trabalho investigou a aplicação de algoritmos de aprendizado de máquina para a classificação e análise do consumo de combustível em veículos automotores, utilizando dados coletados via OBD (*On-Board Diagnostics*).

6.1 Resultados

Os resultados deste trabalho demonstraram a viabilidade de aplicar algoritmos de aprendizado de máquina para a análise de padrões de consumo de combustível, com destaque para *Random Forest* e Árvores de Decisão. Estes algoritmos não apenas atingiram os objetivos propostos, mas também forneceram insights significativos sobre a classificação de tipos de combustível e os fatores que influenciam o consumo.

Em relação ao primeiro objetivo específico, **explorar a base de dados para identificar características e variáveis relevantes**, a análise revelou que variáveis como ENGINE_RPM, ENGINE_LOAD, SPEED, MAF, THROTTLE_POS e INTAKE_MANIFOLD_PRESSURE apresentam uma forte correlação com os padrões de consumo de combustível. Essas variáveis destacaram-se como fatores-chave para a construção de modelos de classificação, demonstrando sua relevância na representação dos dados e no entendimento dos mecanismos que influenciam o consumo em diferentes condições de condução. A influência combinada dessas variáveis reflete a complexidade da dinâmica do motor e sua interação com a eficiência energética, justificando sua inclusão no processo de modelagem e análise preditiva.

O segundo objetivo, **aplicar e avaliar algoritmos de classificação**, foi atingido por meio da implementação e análise de métodos como Árvores de Decisão, *Random Forest*, SVM, Redes Neurais e outros. Entre esses, o *Random Forest* destacou-se como o modelo mais robusto, apresentando métricas superiores, como o menor MSE (0.395877), maior R^2 (0.851256), menor MAE (0.142551), acurácia de 0.920262 e F1-Score de 0.919476, confirmando sua eficácia para os dados analisados.

O terceiro objetivo, **analisar o desempenho dos algoritmos**, foi alcançado com a comparação sistemática entre as métricas de desempenho. Os modelos foram avaliados em termos de acurácia, recall e F1-Score, e o *Random Forest* superou os demais métodos. Em contrapartida, algoritmos como *Naive Bayes* e *SVM Classifier*

apresentaram limitações, evidenciando que nem todos os métodos são igualmente eficazes para os dados coletados.

Por fim, o quarto objetivo, **validar a adequação da base de dados**, foi atendido ao demonstrar que os dados eram suficientemente robustos para alimentar os modelos de classificação e gerar resultados consistentes. Ainda assim, observou-se que a inclusão de variáveis externas, como condições climáticas e padrões de tráfego, poderia fortalecer os modelos utilizados e ampliar a abrangência dos resultados.

A relevância deste estudo é amplificada pela escassez de bases de dados veiculares disponíveis para pesquisas. Os resultados obtidos contribuem para o avanço do aprendizado de máquina aplicado ao consumo de combustível, especialmente em contextos onde o monitoramento de eficiência energética é essencial. Além disso, a validação de metodologias com dados reais reforça o potencial dessas abordagens para o desenvolvimento de soluções práticas e sustentáveis no setor automotivo.

6.2 Limitações

Apesar dos resultados promissores, o estudo apresentou algumas limitações que podem restringir a generalização e a aplicabilidade dos modelos.

Primeiramente, o conjunto de dados utilizado, embora rico em variáveis internas do veículo, não contemplou fatores externos que influenciam diretamente o consumo de combustível, como condições climáticas, padrões de tráfego, qualidade do combustível e características das vias (inclinação, tipo de pavimento, temperatura externa, entre outros). A ausência dessas informações pode ter impactado a precisão dos modelos, uma vez que determinados padrões de consumo podem estar associados a variáveis contextuais que não foram consideradas.

Além disso, o desbalanceamento de classes nos dados exigiu o uso de técnicas de balanceamento, como o *SMOTE* (Synthetic Minority Over-sampling Technique). Embora esse método seja amplamente utilizado para mitigar o impacto da desproporção entre classes, ele pode introduzir vieses artificiais ao gerar novas instâncias sintéticas, alterando a distribuição original dos dados e, potencialmente, influenciando a performance dos modelos de forma não realista. Esse fator pode ter impactado a capacidade dos algoritmos de generalizar para novos conjuntos de dados não vistos.

Adicionalmente, os experimentos foram conduzidos com dados coletados em condições específicas, como trajetos, períodos do dia e estilos de condução particula-

res, o que pode limitar a aplicabilidade dos resultados a outros cenários. Essa restrição compromete a robustez dos modelos ao serem expostos a diferentes contextos de condução, como variações climáticas, altitudes distintas e diferenças no tipo de combustível utilizado.

Também foi necessário considerar as limitações de recursos computacionais da máquina utilizada neste estudo visto que as mesmas impuseram desafios na execução dos modelos. Embora algoritmos mais complexos, como XGBoost, LightGBM e Logistic Regression, pudessem ser testados, o tempo de processamento elevado tornou sua inclusão inviável dentro do escopo do trabalho. Da mesma forma, a quantidade de iterações na validação cruzada foi ajustada para equilibrar a viabilidade computacional e a qualidade das análises, evitando um custo computacional excessivo.

Por fim, a modelagem do problema como uma tarefa de classificação, em vez de uma predição contínua por regressão, pode ter limitado o nível de granularidade das previsões. Essa abordagem impede a obtenção de estimativas precisas de consumo para diferentes condições operacionais do veículo, reduzindo a capacidade dos modelos de capturar variações sutis no consumo de combustível.

6.3 Trabalhos Futuros

Diante das limitações identificadas neste estudo, diversas possibilidades de pesquisa podem ser exploradas para aprimorar a qualidade e a aplicabilidade dos modelos utilizados.

Para melhorar a robustez dos modelos, poderia-se ampliar o conjunto de dados com variáveis adicionais, como condições climáticas, características das vias e padrões de tráfego. A inclusão desses fatores contribuiria para aumentar a precisão e a confiabilidade das previsões, possibilitando uma análise mais abrangente do consumo de combustível em diferentes cenários.

De mesmo modo, a utilização dos modelos utilizados neste trabalho para comparação com *benchmarks* disponíveis na literatura permitiria avaliar possíveis vieses introduzidos pela metodologia adotada. Essa análise comparativa ajudaria a validar os resultados obtidos e a identificar oportunidades de aprimoramento nos métodos aplicados.

Adicionalmente, considerando as limitações computacionais observadas, a utilização de máquinas com maior capacidade de processamento ou a adoção de so-

luções baseadas em computação em nuvem possibilitaria a execução eficiente de modelos mais complexos. Esse avanço reduziria significativamente o tempo de processamento e permitiria análises mais sofisticadas, ampliando as possibilidades de investigação no tema.

Outra possibilidade seria a utilização de modelos otimizados para análise em tempo real, utilizando sensores integrados aos sistemas OBD dos veículos. Com essa abordagem, seria possível realizar classificações dinâmicas e instantâneas, permitindo a identificação imediata de padrões de consumo e auxiliando na tomada de decisões para otimização do desempenho do veículo.

Por fim, um estudo futuro poderia explorar modelos preditivos baseados em regressão, permitindo a estimativa contínua do consumo de combustível em diferentes condições operacionais. Essa abordagem viabilizaria análises mais detalhadas e poderia ser aplicada para simulações de economia de combustível, além de fornecer recomendações personalizadas para motoristas, auxiliando na adoção de estratégias mais eficientes de condução.

7 Anexos

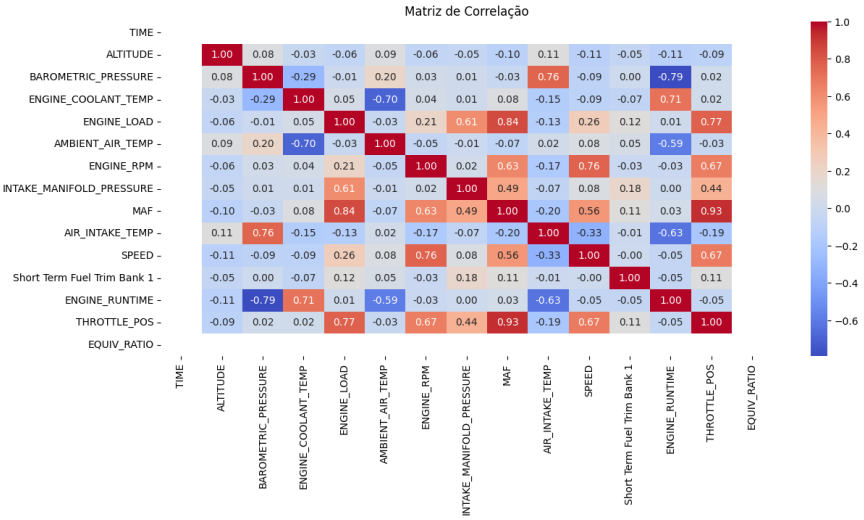


Figura 22 – Matriz de correlação para a amostra s1.

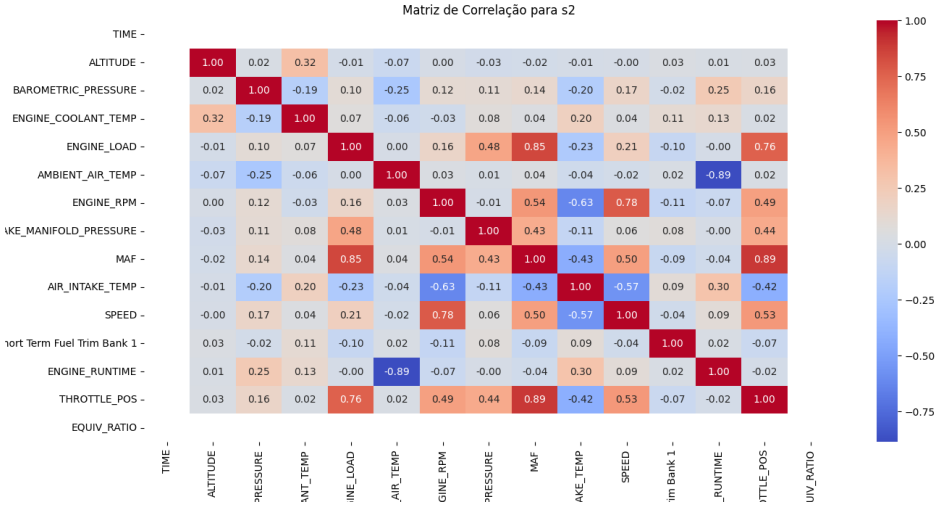


Figura 23 – Matriz de correlação para a amostra s2.

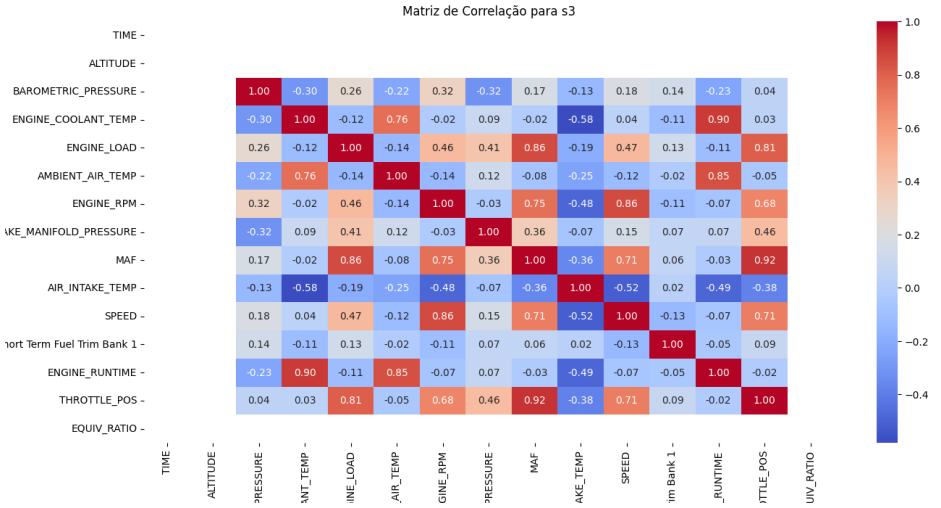


Figura 24 – Matriz de correlação para a amostra s3.

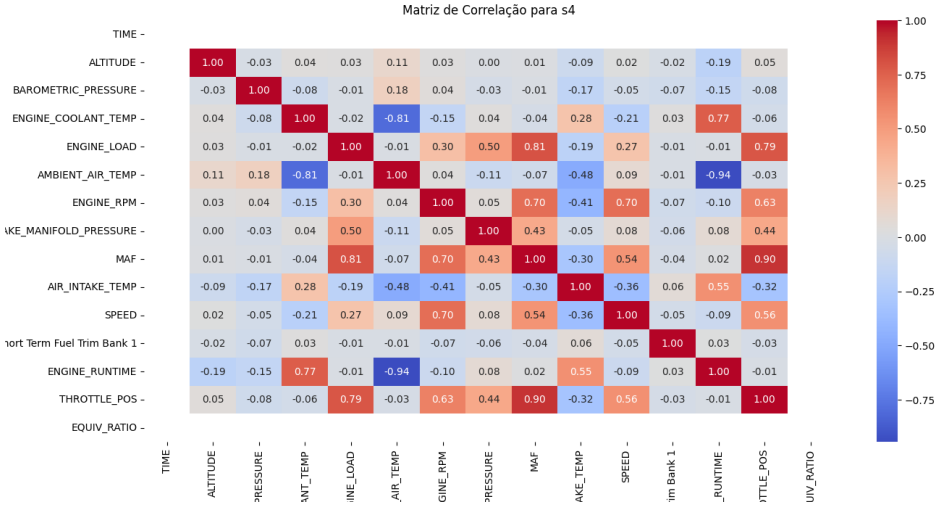


Figura 25 – Matriz de correlação para a amostra s4.

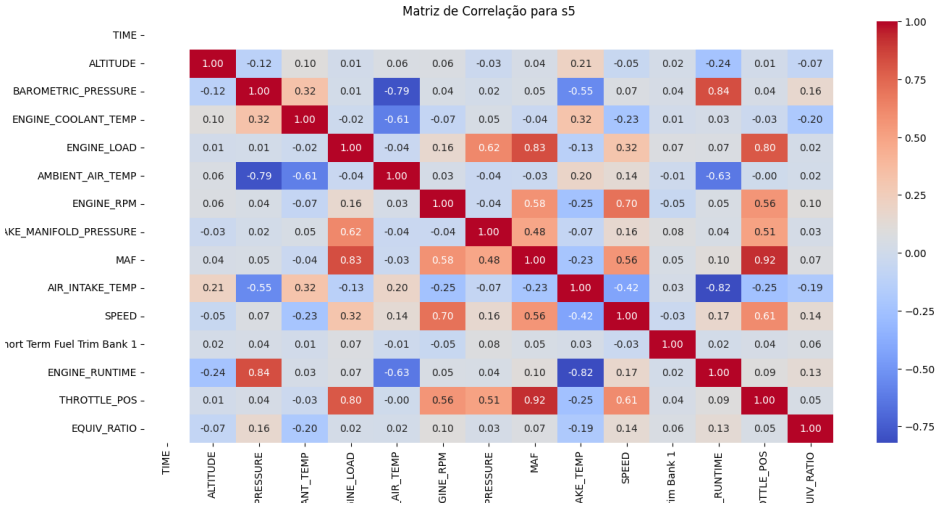


Figura 26 – Matriz de correlação para a amostra s5.

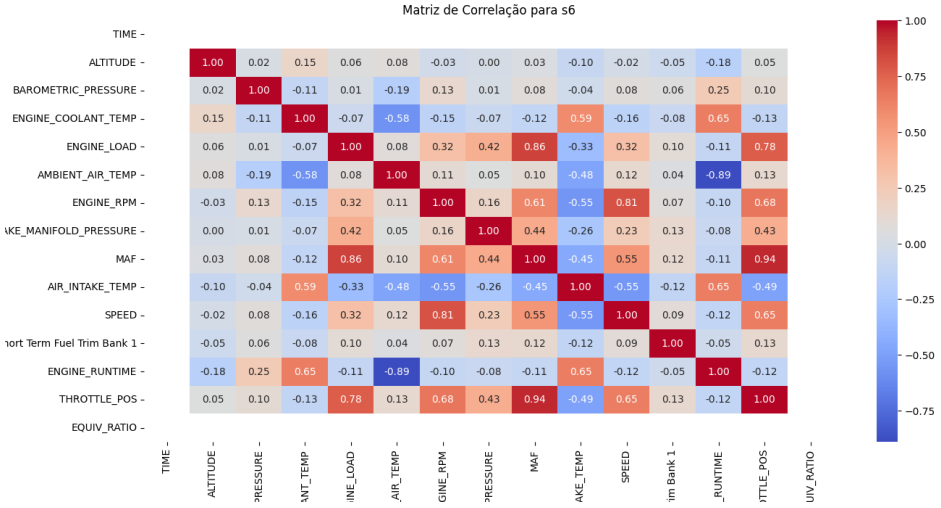


Figura 27 – Matriz de correlação para a amostra s6.

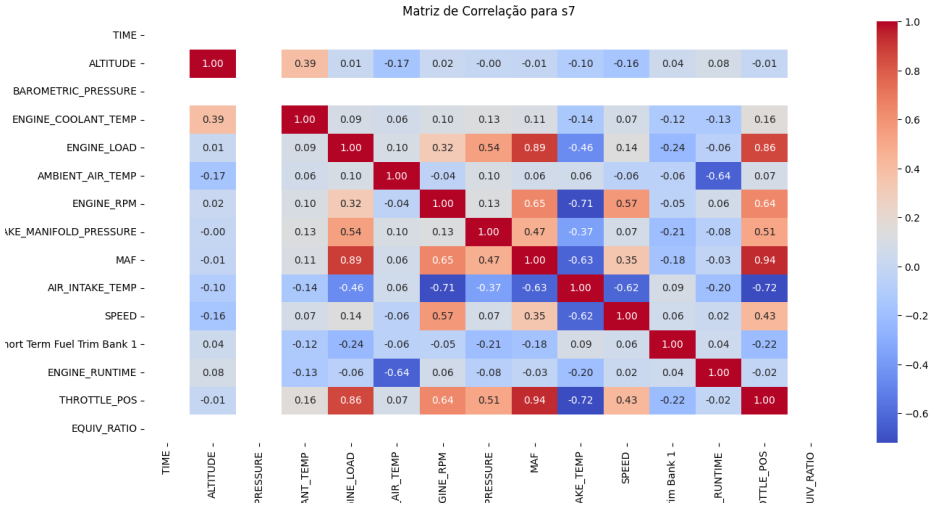


Figura 28 – Matriz de correlação para a amostra s7.

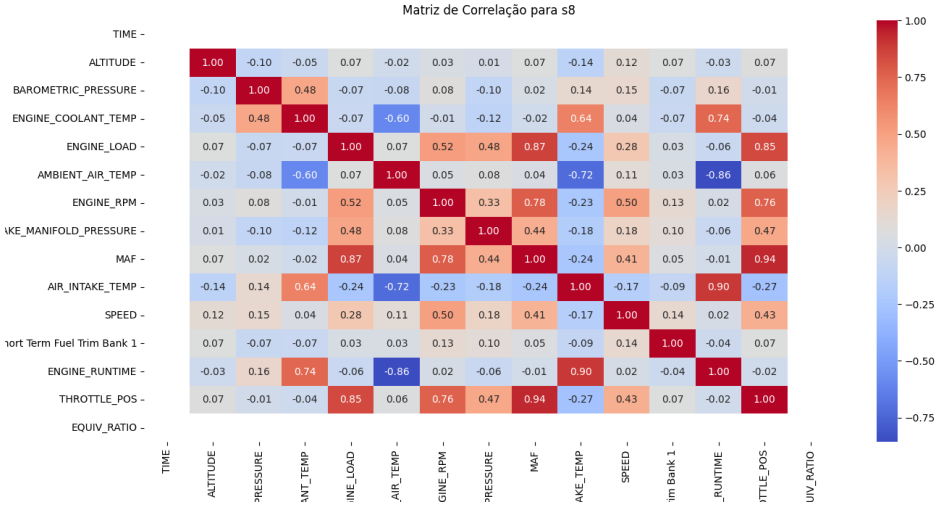


Figura 29 – Matriz de correlação para a amostra s8.

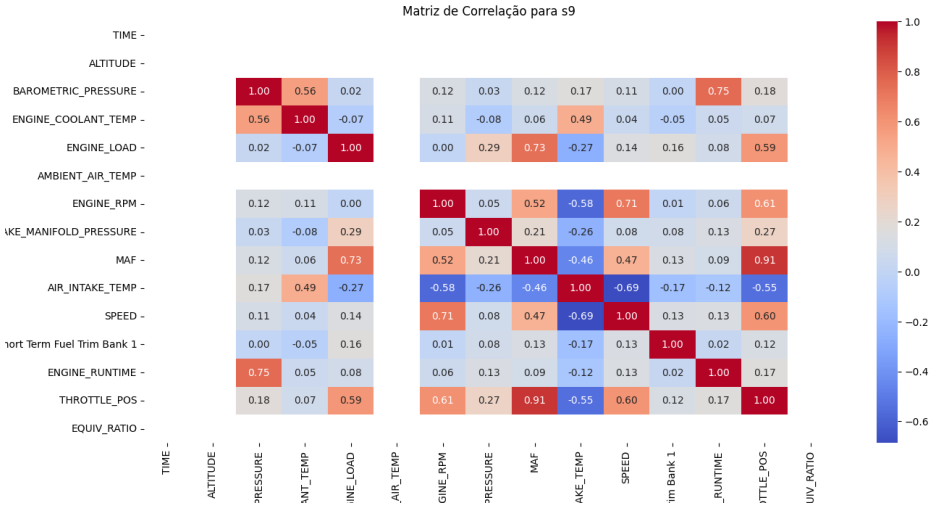


Figura 30 – Matriz de correlação para a amostra s9.

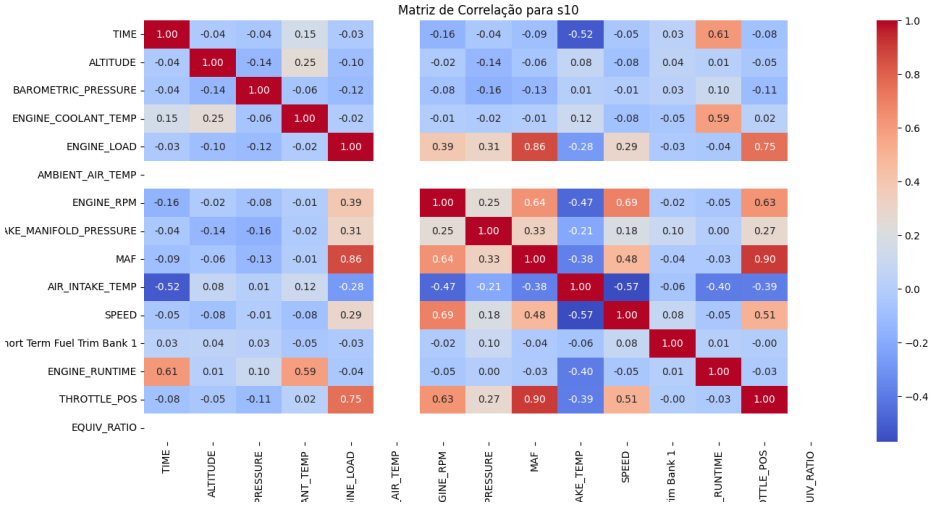


Figura 31 – Matriz de correlação para a amostra s10.

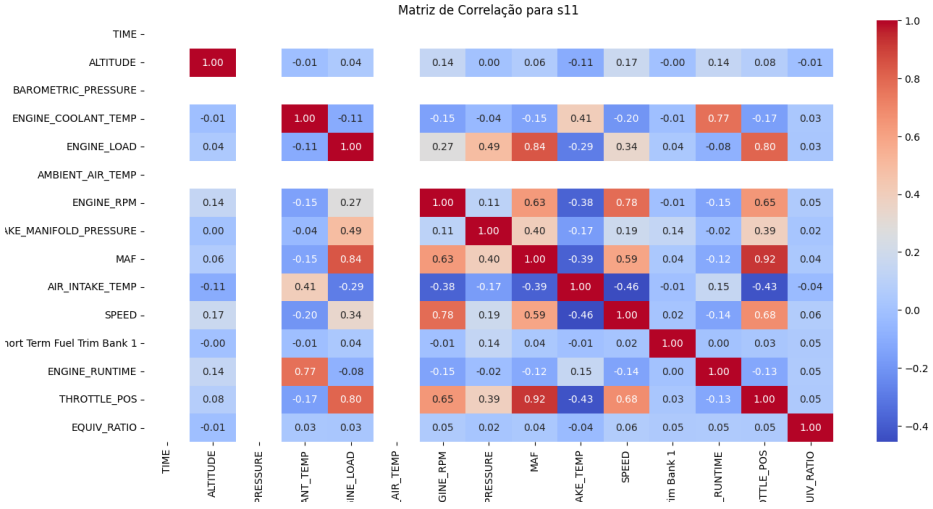


Figura 32 – Matriz de correlação para a amostra s11.

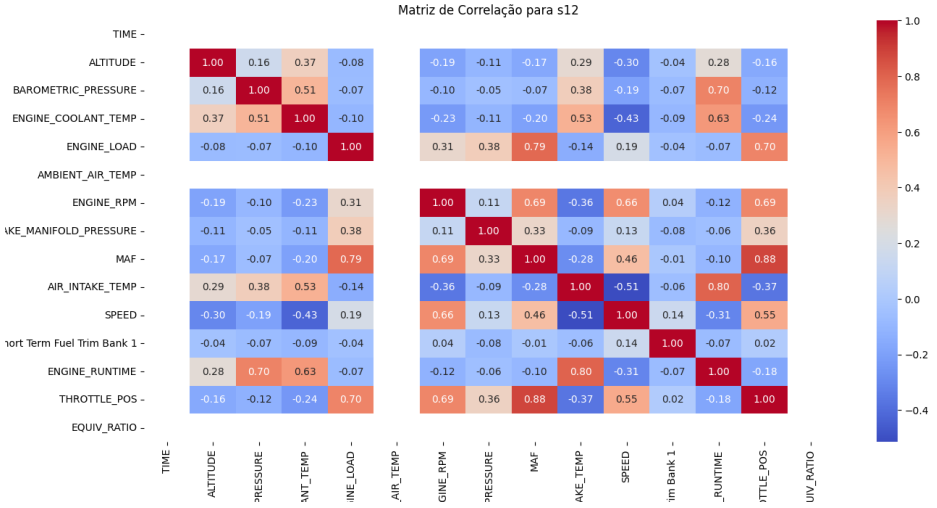


Figura 33 – Matriz de correlação para a amostra s12.

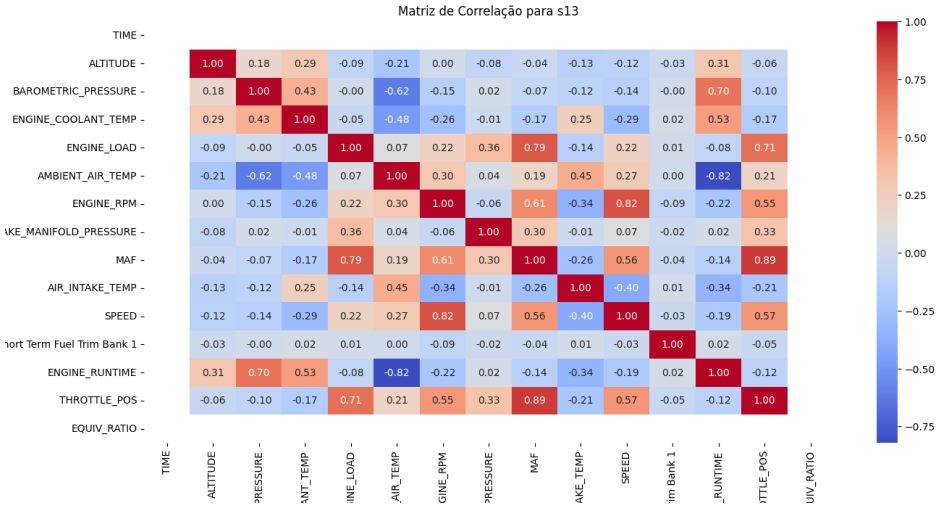


Figura 34 – Matriz de correlação para a amostra s13.

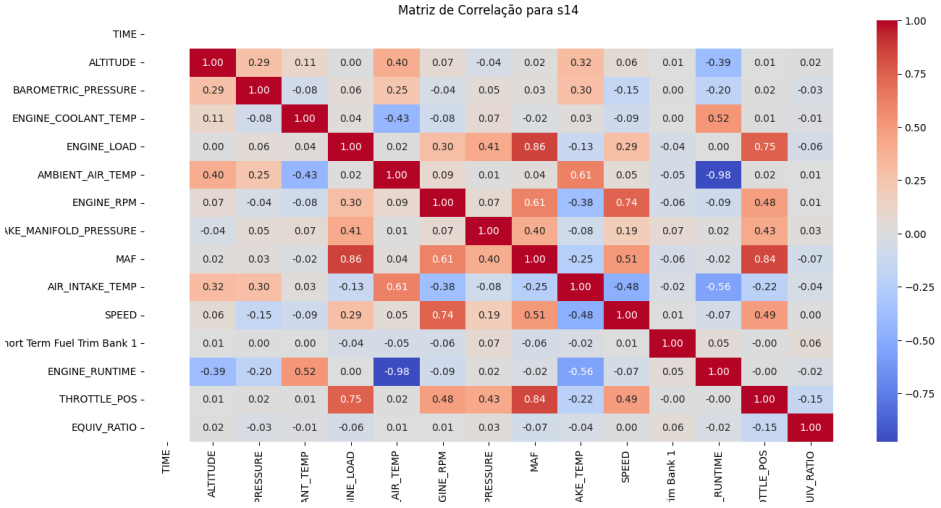


Figura 35 – Matriz de correlação para a amostra s14.

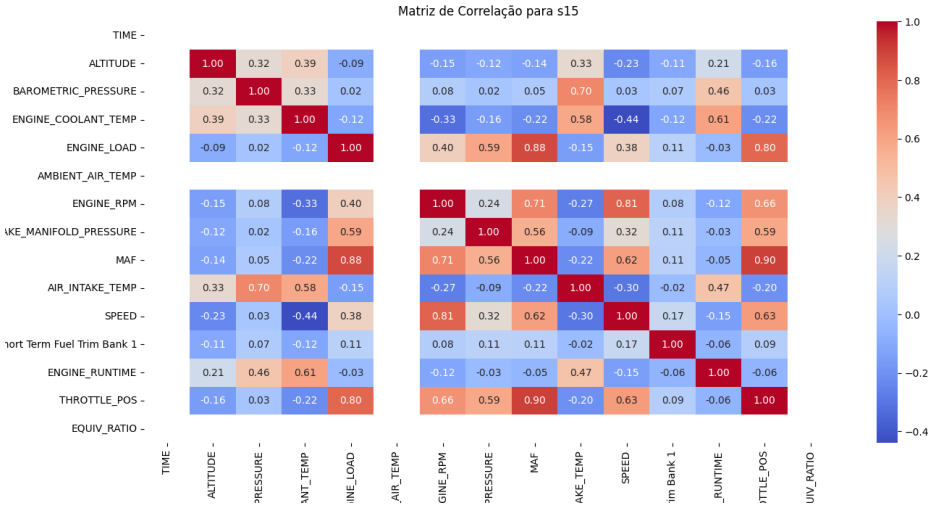


Figura 36 – Matriz de correlação para a amostra s15.

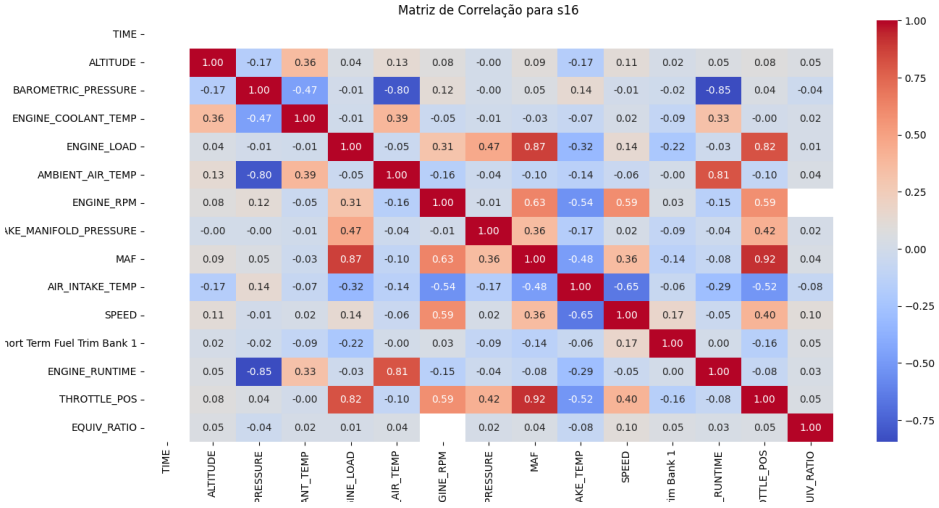


Figura 37 – Matriz de correlação para a amostra s16.

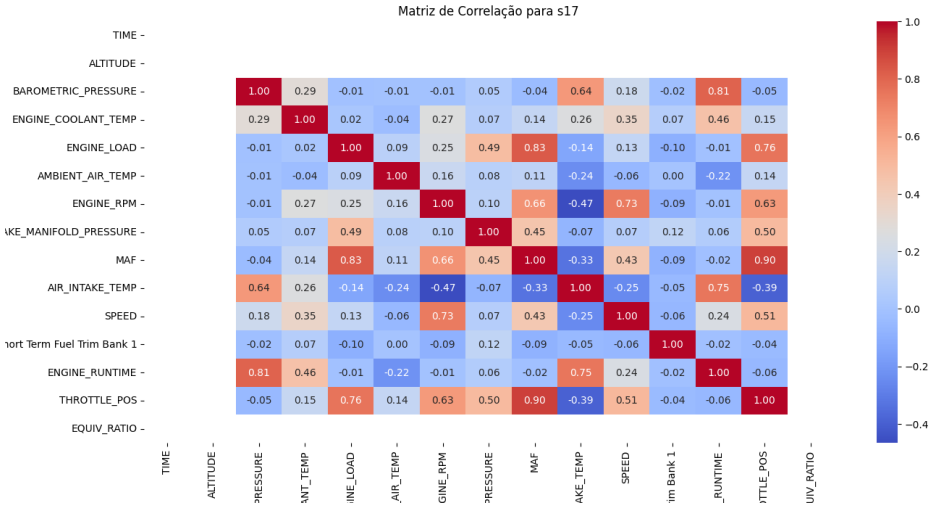


Figura 38 – Matriz de correlação para a amostra s17.

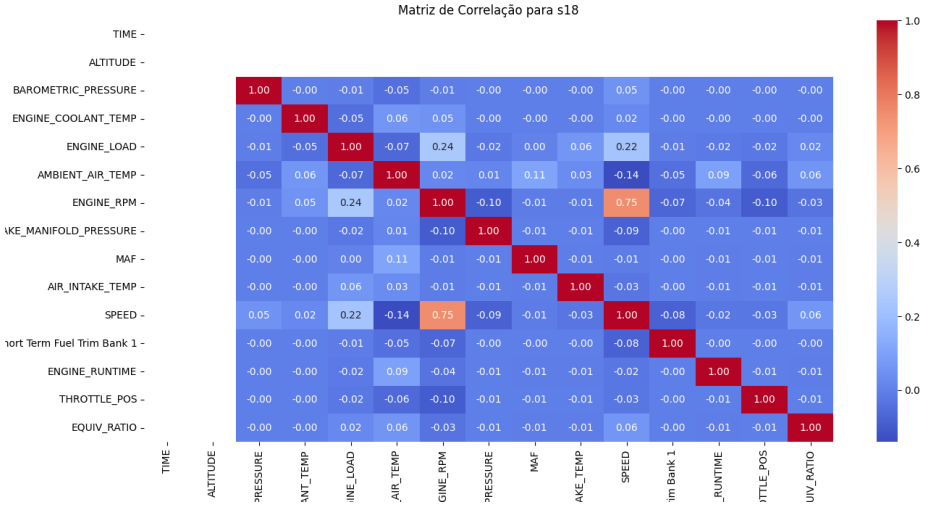


Figura 39 – Matriz de correlação para a amostra s18.

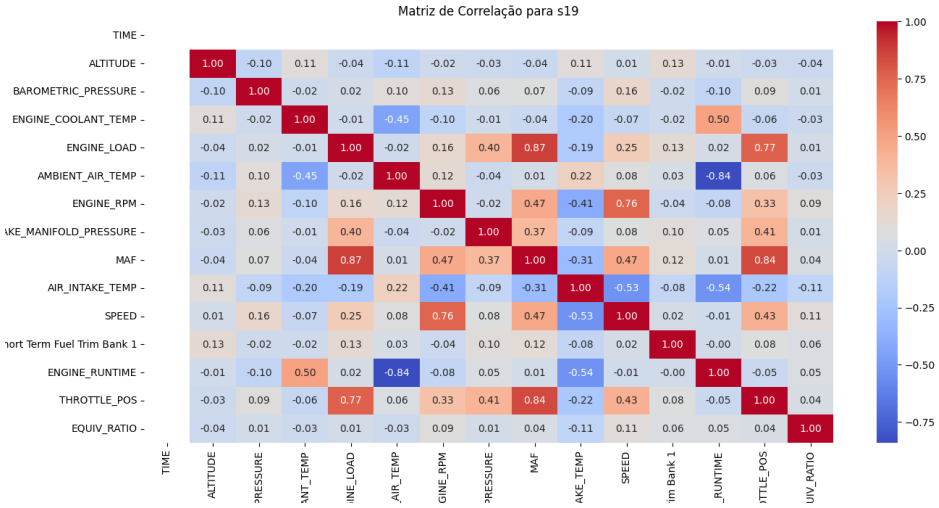


Figura 40 – Matriz de correlação para a amostra s19.

Código Python - EDA

```python
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

class EDA:
    """Classe para análise exploratória de dados (EDA)."""
    @staticmethod
    def perform_eda(data, vehicle_id, selected_vars=None):
        """Realiza EDA para um VEHICLE_ID específico."""
        numeric_data = data.select_dtypes(include=np.number)
        nan_percentages = numeric_data.isnull().sum() / len(
            numeric_data) * 100

        print(f"\nPorcentagem de NaNs em cada coluna para {
            vehicle_id}:\n", nan_percentages)

        # Remover colunas com mais de 80% de NaNs
        numeric_data = numeric_data.dropna(thresh=int(len(
            numeric_data) * 0.2), axis=1)

        # Matriz de correlação
        correlation_matrix = numeric_data.corr()
        plt.figure(figsize=(10, 8))
        sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm
            ', fmt=".2f")
        plt.title(f'Matriz de Correlação para {vehicle_id}')
        plt.show()

        # Encontrar correlações fortes
        correlation_threshold = 0.6
        correlation_df = correlation_matrix.unstack().reset_index
            ()
        correlation_df.columns = ['Variável1', 'Variável2', '
            Correlação']
        strong_correlations = correlation_df[np.abs(correlation_df
            ['Correlação']) > correlation_threshold]
```

```
31          strong_correlations = strong_correlations[
                strong_correlations['Variável1'] != strong_correlations
                ['Variável2']]
32          strong_correlations = strong_correlations[
                strong_correlations['Variável1'] < strong_correlations[
                'Variável2']]
33          strong_correlations = strong_correlations.sort_values(by='
                Correlação', key=lambda x: np.abs(x), ascending=False)
34          print(f"\nPares de Variáveis com Correlação Forte (acima
                de 0.6) para {vehicle_id}:\n", strong_correlations)
35
36          # Filtrar correlações com as variáveis selecionadas
37          if selected_vars:
38              selected_corr = strong_correlations[
39                  (strong_correlations['Variável1'].isin(
                        selected_vars)) |
40                  (strong_correlations['Variável2'].isin(
                        selected_vars))
41              ]
42          else:
43              selected_corr = pd.DataFrame()
44
45          return numeric_data, nan_percentages, strong_correlations,
                selected_corr
46
47      @staticmethod
48      def plot_missing_data(nan_percentages):
49          """Gera um gráfico de barras mostrando a porcentagem de
                valores faltantes por coluna."""
50          missing_data = nan_percentages[nan_percentages > 0]  #
                Filtrar apenas colunas com valores ausentes
51          missing_data = missing_data.sort_values(ascending=False)
                # Ordenar por porcentagem decrescente
52
53          plt.figure(figsize=(12, 6))
54          plt.bar(missing_data.index, missing_data.values, color='
                orange', edgecolor='black')
55          plt.xlabel('Colunas', fontsize=12)
56          plt.ylabel('Porcentagem de Valores Faltantes (%)',
                fontsize=12)
```

```
57          plt.title('Porcentagem de Valores Faltantes por Coluna',
                fontsize=14)
58          plt.xticks(rotation=45, ha='right', fontsize=10)
59          plt.tight_layout()
60          plt.show()
61
62      @staticmethod
63      def plot_distribuiction_target_variable(data, variable):
64          """
65          Exibe a distribuição da variável-alvo com histograma e
                boxplot,
66          ajustando os eixos para refletir corretamente a faixa dos
                valores.
67          """
68
69          # Estatísticas descritivas
70          print(data[variable].describe())
71
72          # Ver valores extremos
73          print("Valores máximos:", data[variable].nlargest(5))
74          print("Valores mínimos:", data[variable].nsmallest(5))
75          plt.figure(figsize=(14, 10))
76
77          # Subplot 1: Histograma com KDE
78          plt.subplot(2, 1, 1)
79          sns.histplot(data[variable], bins=30, kde=True, color='
                skyblue', edgecolor='black')
80
81          mean_val = data[variable].mean()
82          median_val = data[variable].median()
83          std_val = data[variable].std()
84
85          plt.title(
86              f'Distribuição da Variável-Alvo: {variable}\n'
87              f'Média: {mean_val:.2f} | Mediana: {median_val:.2f} |
                    Desvio Padrão: {std_val:.2f}',
88              fontsize=14
89          )
90          plt.xlabel(variable, fontsize=12)
91          plt.ylabel('Frequência', fontsize=12)
```

```
92          plt.axvline(mean_val, color='red', linestyle='--',
                linewidth=1, label=f'Média: {mean_val:.2f}')
93          plt.axvline(median_val, color='green', linestyle='--',
                linewidth=1, label=f'Mediana: {median_val:.2f}')
94          plt.legend(fontsize=12)
95          plt.grid(axis='y', linestyle='--', alpha=0.7)
96
97          # Subplot 2: Boxplot
98          plt.subplot(2, 1, 2)
99          sns.boxplot(data=data, x=variable, color='lightgreen')
100         plt.title(f'Boxplot da Variável-Alvo: {variable}',
                fontsize=14)
101         plt.xlabel(variable, fontsize=12)
102         plt.grid(axis='x', linestyle='--', alpha=0.7)
103
104         # Ajustar layout
105         plt.tight_layout()
106         plt.show()
```

Listing 7.1 – Classe EDA

Código Python - ModelBuilder

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsRegressor
from sklearn.preprocessing import StandardScaler
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.feature_selection import RFE
from sklearn.ensemble import RandomForestRegressor
from sklearn.impute import SimpleImputer
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt
import numpy as np
from sklearn.svm import SVR
import seaborn as sns
import pandas as pd
from datetime import datetime
from sklearn.preprocessing import KBinsDiscretizer
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error, r2_score,
    mean_absolute_error, accuracy_score, f1_score
from sklearn.neural_network import MLPClassifier
from sklearn.naive_bayes import GaussianNB
from imblearn.over_sampling import SMOTE
from data_output import DataOutput
from eda import EDA
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.naive_bayes import GaussianNB


class ModelBuilder:
    """Classe para construção e avaliação de modelos."""

    @staticmethod
    def clean_data(data):
```

```
38          """Limpa o DataFrame removendo linhas com ENGINE_LOAD
                faltando."""
39          data_cleaned = data.dropna(subset=['ENGINE_LOAD'])
40          return data_cleaned
41
42
43      @staticmethod
44      def run_models(data, target_variable='FUEL_CONSUMPTION',
            num_repeats=5):
45          """Executa múltiplos modelos de regressão e avalia o
                desempenho."""
46          # Remover linhas com MAF ou SPEED nulos ou iguais a zero
47          data = data.dropna(subset=['MAF', 'SPEED', 'ENGINE_LOAD',
                'INTAKE_MANIFOLD_PRESSURE', 'THROTTLE_POS'])
48          data = data[(data['MAF'] > 0) & (data['SPEED'] > 0) & (
                data['ENGINE_LOAD'] > 0) &
49                      (data['INTAKE_MANIFOLD_PRESSURE'] > 0) & (data
                            ['THROTTLE_POS'] > 0)]
50
51          density_fuel = 720  # Densidade média do combustível em g/
                L
52          min_value = 0.3
53          max_value = 50
54
55          # Calculando a variável alvo (L/100 km)
56          data[target_variable] = (3600 * data['MAF']) / (data['
                SPEED'] * density_fuel)
57          data = data[(data[target_variable] >= min_value) & (data[
                target_variable] <= max_value)]
58
59          X = data[['MAF', 'THROTTLE_POS', 'INTAKE_MANIFOLD_PRESSURE
                ', 'ENGINE_LOAD']]
60          y = data[target_variable]
61
62          EDA.plot_distribuiction_target_variable(data,
                target_variable)
63
64          metrics_results = []
65
66          for test in [0.2, 0.3, 0.4, 0.5]:
```

```
67              for bins in [2, 3, 5, 10]:
68                  smote = SMOTE(random_state=42)
69                  discretizer = KBinsDiscretizer(n_bins=bins, encode
                        ='ordinal', strategy='uniform')
70                  y_class = discretizer.fit_transform(y.values.
                        reshape(-1, 1)).ravel()
71
72                  preprocessor = ColumnTransformer(
73                      transformers=[('num', Pipeline([
74                          ('imputer', SimpleImputer(strategy='mean')
                                ),
75                          ('scaler', StandardScaler())
76                      ]), X.columns)]
77                  )
78
79                  X_balanced, y_balanced = smote.fit_resample(X,
                        y_class)
80
81                  models = {
82                      'Random Forest': RandomForestClassifier(),
83                      'SVM Classifier': SVC(kernel='rbf'),  # SVM
                            com kernel RBF para maior flexibilidade
84                      'Decision Tree': DecisionTreeClassifier(),
85                      'KNN': KNeighborsClassifier(),
86                      'MLP': MLPClassifier(max_iter=1000),
87                      'Naive Bayes': GaussianNB()
88                  }
89
90
91                  # Repetir as avaliações para consistência
92                  for name, model in models.items():
93                      mse_values, r2_values, mae_values,
                            accuracy_values, f1_values = [], [], [],
                            [], []
94
95                      for _ in range(num_repeats):
96                          X_train, X_test, y_train, y_test =
                                train_test_split(
97                              X_balanced, y_balanced, test_size=test
                                    , random_state=np.random.randint(0,
```

```
                                10000)
98                          )
99
100                         pipeline = Pipeline([
101                             ('preprocessor', preprocessor),
102                             ('model', model)
103                         ])
104
105                         pipeline.fit(X_train, y_train)
106                         y_pred = pipeline.predict(X_test)
107                         y_pred_class = np.round(y_pred) if name
                                not in ['MLP', 'Naive Bayes'] else
                                y_pred
108
109                         mse_values.append(mean_squared_error(
                                y_test, y_pred))
110                         r2_values.append(r2_score(y_test, y_pred))
111                         mae_values.append(mean_absolute_error(
                                y_test, y_pred))
112                         accuracy_values.append(accuracy_score(
                                y_test, y_pred_class))
113                         f1_values.append(f1_score(y_test,
                                y_pred_class, average='weighted'))
114
115                     metrics_results.append({
116                         'Model': name,
117                         'Teste': test,
118                         'Bins': bins,
119                         'MSE': np.mean(mse_values),
120                         'R2': np.mean(r2_values),
121                         'MAE': np.mean(mae_values),
122                         'Accuracy': np.mean(accuracy_values),
123                         'F1-Score': np.mean(f1_values)
124                     })
125
126         results_df = pd.DataFrame(metrics_results)
127         DataOutput.plot_heatmap(results_df)
128
129         DataOutput.save_results_model(results_df, 'C:\\Users\\
              Calebe\\Documents\\Projeto_TCC\\files\\model_results')
```

```
130         print(results_df.pivot_table(index='Bins', columns='Model'
                , values=['MSE', 'R2', 'MAE', 'Accuracy', 'F1-Score']))
131         return results_df
132
133
134     def cross_validation_evaluation(metrics_results):
135         """
136         Realiza a validação cruzada nos dados de métricas.
137
138         Args:
139             metrics_results (pd.DataFrame): DataFrame com colunas
                    ['Model', 'Teste', 'Bins', 'MSE', 'R2', 'MAE', '
                    Accuracy', 'F1-Score'].
140
141         Returns:
142             pd.DataFrame: Resultados médios por modelo.
143         """
144         grouped = metrics_results.groupby("Model").mean()
145         return grouped.reset_index()
```

Listing 7.2 – Classe ModelBuilder

Referências

ALMEIDA, R. de D. *Sistema de Análise de Consumo de Combustível de Veículos Automotores*. 2017 — Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais - Campus Formiga, Formiga, MG, 2017. Trabalho de Conclusão de Curso apresentado como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação. Disponível em: <https://www.formiga.ifmg.edu.br/documents/2017/PublicacoesTCCsBiblioteca/MonografiaRODRIGODEDEUS_FINAL.pdf>. Citado na página 36.

ARLOT, S.; CELISSE, A. A survey of cross-validation procedures for model selection. *Statistics Surveys*, v. 4, p. 40–79, 2010. Citado na página 29.

BARRETO, C. A. da S. Uso de técnicas de aprendizado de máquina para identificação de perfis de uso de automóveis baseado em dados automotivos. *Revista de Engenharia Automotiva*, Universidade de Tecnologia Automotiva, v. 15, n. 2, p. 123–134, Julho 2018. Disponível em: <https://repositorio.ufrn.br/jspui/bitstream/123456789/26017/1/Usot%c3%a9cnicasaprendizado_Barreto_2018.pdf>. Citado nas páginas 5, 16, 17, 18, 30 e 36.

BERGSTRA, J.; BENGIO, Y. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, v. 13, p. 281–305, 2012. Citado na página 30.

BISHOP, C. M. *Pattern Recognition and Machine Learning*. [S.l.]: Springer, 2006. Citado na página 13.

BREIMAN, L. Random forests. *Machine Learning*, Springer, v. 45, n. 1, p. 5–32, 2001. Citado na página 13.

BREIMAN, L. Random forests. *Machine Learning*, v. 45, n. 1, p. 5–32, 2001. Citado na página 25.

BUDA, M.; MAKI, A.; MAZUROWSKI, M. A. A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks*, v. 106, p. 249–259, 2018. Citado na página 28.

CHAWLA, N. V. et al. Smote: synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, v. 16, p. 321–357, 2002. Citado na página 28.

CHAWLA, N. V. et al. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, v. 16, p. 321–357, 2002. Citado na página 50.

CORTES, C.; VAPNIK, V. Support-vector networks. *Machine Learning*, v. 20, n. 3, p. 273–297, 1995. Citado na página 25.

COVER, T.; HART, P. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, v. 13, n. 1, p. 21–27, 1967. Citado nas páginas 24 e 25.

DING, S.; ZHANG, H. Improving the performance of random forest on imbalanced datasets. In: IEEE. *2008 world congress on intelligent control and automation*. [S.l.], 2008. p. 2645–2649. Citado na página 50.

DOMINGOS, P. A few useful things to know about machine learning. *Communications of the ACM*, v. 55, n. 10, p. 78–87, 2012. Citado na página 13.

DOMINGOS, P.; PAZZANI, M. On the optimality of the simple bayesian classifier under zero-one loss. *Machine Learning*, Springer, v. 29, n. 2, p. 103–130, 1997. Citado na página 61.

DOUGHERTY, J.; KOHAVI, R.; SAHAMI, M. Supervised and unsupervised discretization of continuous features. *Machine Learning Proceedings*, p. 194–202, 1995. Citado na página 28.

DRUMMOND, C.; HOLTE, R. C. C4.5, class imbalance, and cost sensitivity: why under-sampling beats over-sampling. *Workshop on Learning from Imbalanced Datasets II, ICML*, 2003. Citado na página 28.

DUDA, R. O.; HART, P. E.; STORK, D. G. *Pattern Classification*. [S.l.]: John Wiley & Sons, 2001. Citado na página 27.

FASTERCAPITAL. *Distribuição Multimodal: Desvendando Vários Modos em Conjuntos de Dados*. 2024. Acessado em: 22 de dezembro de 2024. Disponível em: <https://fastercapital.com/pt/contente/Distribuicao-multimodal--Desvendando-varios-modos-em-conjuntos-de-dados.html>. Citado na página 49.

FAYYAD, U. M.; IRANI, K. B. Multi-interval discretization of continuous-valued attributes for classification learning. *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI)*, p. 1022–1027, 1993. Citado na página 28.

FISHER, R. A. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, v. 7, n. 2, p. 179–188, 1936. Citado na página 27.

GARCIA, S.; LUENGO, J.; HERRERA, F. Data preprocessing in data mining. *Springer*, 2013. Citado na página 28.

HAN, J.; KAMBER, M.; PEI, J. *Data Mining: Concepts and Techniques*. 3rd. ed. [S.l.]: Elsevier, 2011. Citado na página 13.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2nd. ed. [S.l.]: Springer, 2009. Citado nas páginas 13 e 62.

INTERNATIONAL, S. *On-Board Diagnostics (OBD) Standards*. 2020. Available at: <https://www.sae.org/standards/>. Citado na página 13.

JAMES, G. et al. *An Introduction to Statistical Learning*. [S.l.]: Springer, 2013. Citado na página 29.

JOLLIFFE, I. T.; CADIMA, J. Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, v. 374, n. 2065, p. 20150202, 2016. Citado na página 26.

KOHAVI, R. A study of cross-validation and bootstrap for accuracy estimation and model selection. *International Joint Conference on Artificial Intelligence (IJCAI)*, p. 1137–1145, 1995. Citado na página 29.

KOTSIANTIS, S.; KANELLOPOULOS, D. Discretization techniques: A recent survey. *GESTS international transactions on computer science and engineering*, v. 32, n. 1, p. 47–58, 2006. Citado na página 50.

LIU, H. et al. Discretization: An enabling technique. *Data mining and knowledge discovery*, Springer, v. 6, n. 4, p. 393–423, 1998. Citado na página 50.

MCCALLUM, A.; NIGAM, K. A comparison of event models for naive bayes text classification. In: *AAAI-98 Workshop on Learning for Text Categorization*. [S.l.: s.n.], 1998. p. 41–48. Citado na página 25.

MESEGUER, J. E. et al. Assessing the impact of driving behavior on instantaneous fuel consumption. In: IEEE. *2015 12th Annual IEEE Consumer Communications and Networking Conference (CCNC)*. Las Vegas, NV, USA, 2015. Conference Paper. Disponível em: <https://www.researchgate.net/profile/Javier-Meseguer/publication/285614280_Assessing_the_impact_of_driving_behavior_on_instantaneous_fuel_consumption/links/5694e36608ae820ff074585a/Assessing-the-impact-of-driving-behavior-on-instantaneous-fuel-consumption.pdf?origin=publication_detail>. Citado na página 36.

OLIVEIRA, R. M. de. *Desenvolvimento de um Sistema para Monitoramento Automotivo Baseado em OBD-II*. 2022. Dissertação (Mestrado) — CEFET-MG - Campus Araxá, 2022. Disponível em: <https://www.eng-automacao.araxa.cefetmg.br/wp-content/uploads/sites/152/2022/08/TCC2-Rubens-Moreno-de-Oliveira.pdf>. Citado na página 37.

RASCHKA, S. Machine learning in python: Main developments and technology trends in data science, machine learning, and artificial intelligence. *Information*, v. 11, n. 4, p. 193, 2020. Citado na página 29.

(SAE), S. of A. E. *SAE J1979: E/E Diagnostic Test Modes*. SAE International, 2021. Padrão OBD-II para diagnóstico automotivo. Disponível em: <https://www.sae.org>. Citado nas páginas 25, 38, 39 e 40.

SCHöLKOPF, B.; SMOLA, A. J. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. [S.l.]: MIT Press, 2002. Citado na página 61.

SILVA, M. E. R. D. *SELEÇÃO DE ATRIBUTOS EM APRENDIZADO DE MÁQUINA PARA IDENTIFICAÇÃO DE FALHAS EM MOTORES DE COMBUSTÃO INTERNA*. 2022 — Universidade Federal de Santa Catarina, Florianópolis, SC, 2022. Trabalho de Conclusão de Curso apresentado como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação. Disponível em: <https://repositorio.ufsc.br/bitstream/handle/123456789/237549/TCC_Feature_selection.pdf?sequence=1&isAllowed=y>. Citado nas páginas 5, 31, 32, 33, 34 e 35.

SOKOLOVA, M.; LAPALME, G. A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, Elsevier, v. 45, n. 4, p. 427–437, 2009. Citado nas páginas 61 e 62.

WITTEN, I. H. et al. *Data Mining: Practical Machine Learning Tools and Techniques*. 4th. ed. [S.l.]: Morgan Kaufmann, 2016. Citado na página 29.

WOLD, S.; ESBENSEN, K.; GELADI, P. Principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, v. 2, n. 1-3, p. 37–52, 1987. Citado na página 26.

WU, X. et al. Top 10 algorithms in data mining. *Knowledge and Information Systems*, v. 14, p. 1–37, 2008. Citado na página 13.