



Trabalho de Conclusão de Curso

Identificação do motorista agressivo analisando dados de sensores utilizando Aprendizagem de Máquina.

de Matheus Vieira Faria

orientado por

Prof. Dr. Bruno Almeida Pimentel

Universidade Federal de Alagoas
Instituto de Computação
Maceió, Alagoas
02 de Junho de 2025

UNIVERSIDADE FEDERAL DE ALAGOAS
Instituto de Computação

**IDENTIFICAÇÃO DO MOTORISTA AGRESSIVO
ANALISANDO DADOS DE SENSORES UTILIZANDO
APRENDIZAGEM DE MÁQUINA.**

Trabalho de Conclusão de Curso submetido
ao Instituto de Computação da Universidade
Federal de Alagoas como requisito parcial
para a obtenção do grau de Engenharia de
Computação.

Matheus Vieira Faria

Orientador: Prof. Dr. Bruno Almeida Pimentel

Banca Avaliadora:

Glauber Rodrigues Leite	Prof. Dr., IC-UFAL
Jobson de Araújo Nascimento	Prof. M.e., IC-UFAL

Maceió, Alagoas
02 de Junho de 2025

Catálogo na Fonte
Universidade Federal de Alagoas
Biblioteca Central
Divisão de Tratamento Técnico

Bibliotecário: Marcelino de Carvalho Freitas Neto – CRB-4 - 1767

F221i Faria, Matheus Vieira.

Identificação do motorista agressivo analisando dados de sensores utilizando aprendizagem de máquina / Matheus Vieira Faria. – 2025.
69 f. : il.

Orientador: Bruno Almeida Pimentel.

Monografia (Trabalho de conclusão de curso em Engenharia de Computação) - Universidade Federal de Alagoas, Instituto de Computação. Maceió, 2025.

Bibliografia: f. 67-69.

1. Motoristas - Comportamento. 2. Direção agressiva (Trânsito - comportamento). 3. Aprendizado do computador. 4. Redes neurais (Computação). 5. Análise de séries temporais. 6. Acelerômetro. 7. Giroscópios. I. Título.

CDU: 004.032.26

UNIVERSIDADE FEDERAL DE ALAGOAS

Instituto de Computação

IDENTIFICAÇÃO DO MOTORISTA AGRESSIVO ANALISANDO DADOS DE SENSORES UTILIZANDO APRENDIZAGEM DE MÁQUINA.

Trabalho de Conclusão de Curso submetido ao Instituto de Computação da Universidade Federal de Alagoas como requisito parcial para a obtenção do grau de Engenheiro de Computação.

Aprovado em 02 de Junho de 2025:

Documento assinado digitalmente
 **BRUNO ALMEIDA PIMENTEL**
Data: 03/06/2025 12:59:56-0300
Verifique em <https://validar.iti.gov.br>

Bruno Almeida Pimentel,
Prof. Dr., Orientador

Documento assinado digitalmente
 **GLAUBER RODRIGUES LEITE**
Data: 03/06/2025 18:04:50-0300
Verifique em <https://validar.iti.gov.br>

Glauber Rodrigues Leite,
Prof. Dr., IC-UFAL

Documento assinado digitalmente
 **JOBSON DE ARAUJO NASCIMENTO**
Data: 03/06/2025 13:32:38-0300
Verifique em <https://validar.iti.gov.br>

Jobson de Araújo Nascimento,
Prof. M.e., IC-UFAL

Agradecimentos

Agradeço primeiramente à minha família pelo apoio incondicional em todos os momentos, aos meus colegas e amigos que pude compartilhar meu tempo e aprendizado durante toda jornada na universidade, ao meu orientador Bruno Pimentel, pela orientação e paciência durante a realização desse trabalho, e a todos professores e funcionários do IC.

15 de Maio de 2025, Maceió - AL

Matheus Vieira

Resumo

A análise do comportamento do motorista tornou-se um tópico com maior relevância atualmente devido à sua aplicação na melhoria da experiência de direção e da segurança no transporte. Este estudo propõe a identificação do comportamento agressivo do motorista ao dirigir utilizando dados de acelerômetro e giroscópio de smartphones. Foram utilizados métodos de análise temporais e análises no domínio da frequência e espectograma, além da criação e comparação de modelos de Machine Learning e Deep Learning.

Palavras-chave: *Comportamento do motorista; Direção agressiva; Aprendizado de máquina; Redes neurais; Séries temporais; Acelerômetro; Giroscópio.*

Abstract

Driver behavior analysis has become a topic of increasing relevance today due to its application in improving the driving experience and safety in transportation. This study proposes the identification of aggressive driver behavior while driving using accelerometer and gyroscope data from smartphones. Temporal analysis methods and frequency and spectrum domain analyses were used, in addition to the creation and comparison of Machine Learning and Deep Learning models.

Keywords: *Driving behavior; Aggressive driving; Machine learning; Neural networks; Time series; Accelerometer; Gyroscope.*

Lista de Figuras

1	Curva logística representando a função sigmoide utilizada na regressão logística.	20
2	Exemplo de separação de classes utilizando SVM com diferentes kernels. .	21
3	Exemplo ilustrativo do funcionamento do Random Forest. (Fonte: Brital (2021)).	22
4	Ilustração do funcionamento do classificador Naive Bayes. (Fonte: Faleiros (2024)).	23
5	Arquitetura da Rede MPL. (Fonte: Peixoto (2020)).	25
6	Arquitetura de uma CNN. (Fonte: Haque (2023)).	26
7	Arquitetura de uma CNN. (Fonte: Stryker (2021)).	27
8	Exemplo de funcionamento do Transfer Learning. (Fonte: GeeksforGeeks (2025)).	28
9	Gráficos das distribuições das classes.	48
10	A imagem (a) é referente Gráfico de Bland-Altman para os eixos do Acelerômetro e a imagem (b) para os eixos do Giroscópio	48
11	Gráficos dos sinais dos sensores ao longo do tempo para as classes AGGRESSIVE (vermelho), NORMAL (azul) e SLOW (verde). (a) mostra os dados do acelerômetro, (b) os dados do giroscópio e (c) o módulo do acelerômetro (Acc_{mod}).	50
12	Análise de Fourier (FFT e PSD) dos sinais dos sensores. (a) mostra a transformada de Fourier para os eixos do acelerômetro, (b) para os eixos do giroscópio e (c) para o módulo do acelerômetro (Acc_{mod}).	52
13	Análise de Fourier (FFT e PSD) separada por classe no Acelerômetro. . . .	52
14	Análise de Fourier (FFT e PSD) separada por classe no Giroscópio. . . .	53
15	Análise de Fourier (FFT e PSD) separada por classe no Módulo da Aceleração.	53
16	Espectrogramas dos sinais dos sensores (Acc, Gyro e Acc_{mod}) para todas as classes.	55
17	Filtros aplicados nos dados do Acelerômetro	56
18	Filtros aplicados nos dados do Giroscópio	57
19	Filtros aplicados nos dados do Módulo da Aceleração	57

20	Matriz de Confusão - Regressão Logística	58
21	Matriz de Confusão - Regressão Logística (com Normalização)	58
22	Matriz de Confusão - SVM	59
23	Matriz de Confusão - Random Forest	59
24	Matriz de Confusão - Naive Bayes	59
25	Matriz de Confusão - XGBoost	60
26	Matriz de Confusão - MLP (Modelo 2)	62
27	Matriz de Confusão - MLP (Modelo 5)	63
28	Matriz de Confusão - CNN (Modelo 1)	63
29	Matriz de Confusão - CNN (Modelo 3)	63
30	Matriz de Confusão - CNN (Transfer Learning)	64
31	Matriz de Confusão - RNN (Simple RNN)	64
32	Matriz de Confusão - RNN (GRU)	64
33	Matriz de Confusão - RNN (LSTM)	65

Lista de Tabelas

1	Distribuição das Classes após a Conversão Binária	47
2	Porcentagem de pontos dentro e fora dos limites de concordância para os eixos dos sinais.	49
3	Frequências dominantes para cada variável e classe.	54
4	Potência na frequência dominante (PSD) para cada variável e classe. . . .	54
5	Frequências máximas médias dos espectrogramas (Hz) para cada variável e classe.	56
6	Resultados dos Modelos Clássicos de ML (Métricas da Classe 1)	58
7	Resultados dos Modelos MLP	61
8	Resultados dos Modelos CNN	61
9	Resultados dos Modelos RNN	62
10	Melhores resultados dos modelos de Deep Learning em cada arquitetura . .	62

Lista de Símbolos

$AccX$ Valor da aceleração no eixo x do acelerômetro.

$AccY$ Valor da aceleração no eixo y do acelerômetro.

$AccZ$ Valor da aceleração no eixo z do acelerômetro.

Acc_{mod} Módulo da aceleração de $AccX$, $AccY$ e $AccZ$

$X[k]$ Transformada de Fourier no domínio da frequência, índice k .

$PSD[k]$ Densidade espectral de potência na frequência k .

$P(Y = 1|X)$ Probabilidade da classe positiva dada as variáveis de entrada.

$S(t, f)$ Intensidade espectral na frequência f no tempo t (espectrograma).

Lista de Abreviaturas

IC Instituto de Computação.

UFAL Universidade Federal de Alagoas.

ML *Machine Learning*.

DL *Deep Learning*.

MLP *Multilayer Perceptron*.

CNN *Convolutional Neural Network*.

RNN *Recurrent Neural Network*.

LSTM *Long Short-Term Memory*.

GRU *Gated Recurrent Unit*.

SVM *Support Vector Machine*.

FFT *Fast Fourier Transform*.

PSD *Power Spectral Density*.

Sumário

1	Introdução	15
1.1	Objetivos	16
1.1.1	Objetivos Específicos	16
1.2	Estrutura do Documento	16
2	Fundamentação Teórica	17
2.1	Método de Bland-Altman	17
2.2	Transformada Rápida de Fourier (FFT)	18
2.3	Densidade Espectral de Potência (PSD)	18
2.4	Espectrogramas e STFT	19
2.5	Modelos de Aprendizagem de Máquina (Machine Learning)	20
2.5.1	Regressão Logística	20
2.5.2	Support Vector Machines (SVM)	21
2.5.3	Random Forest	22
2.5.4	Naive Bayes	22
2.5.5	XGBoost	23
2.6	Modelos de Aprendizado Profundo (Deep Learning)	24
2.6.1	Redes MLP (Multilayer Perceptron)	24
2.6.2	Redes Neurais Convolucionais (CNN)	25
2.6.3	Redes Neurais Recorrentes (RNN)	26
2.7	Aprendizado por Transferência (Transfer Learning)	27
3	Metodologia	29
3.1	Dataset e Coleta de Dados	29
3.2	Análise Inicial	30
3.3	Análise de Diferenças Temporais	30
3.3.1	Verificação de Continuidade Temporal	30
3.3.2	Análise das Diferenças de Medições	31
3.3.3	Visualização e Interpretação	31
3.4	Criação de Features	31
3.4.1	Nova Feature	32
3.4.2	Seleção das Features	32
3.5	Análise de Sinais no Domínio da Frequência	32

3.5.1	Transformada Rápida de Fourier (FFT)	33
3.5.2	Densidade Espectral de Potência (PSD)	33
3.5.3	Implementação da Análise Espectral	33
3.6	Análise com Espectrogramas	34
3.6.1	Aplicação e Visualização dos Espectrogramas	34
3.6.2	Visualização por Classes e Variáveis	34
3.7	Processamento dos Dados para ML/DL	34
3.7.1	Eliminação de Ruído dos Dados	34
3.7.2	Conversão para Classificação Binária	35
3.7.3	Estruturação dos Dados para Modelagem	35
3.7.4	Distribuição das Classes nos Conjuntos	36
3.7.5	Conclusão do Processamento	36
3.8	Modelos de Machine Learning	37
3.8.1	Implementação da Classe Base para Modelos de Classifi- cação	37
3.8.2	Modelos Utilizados	37
3.8.3	Procedimento de Treinamento e Avaliação	38
3.9	Modelos de Aprendizado Profundo (Deep Learning)	39
3.9.1	Implementação da Classe Base para Modelos de DL	39
3.9.2	Modelos MLP (Multilayer Perceptron)	40
3.9.3	Estratégias de Treinamento e Otimização dos modelos MLP	40
3.9.4	Modelos de Redes Neurais Convolucionais (CNN)	41
3.9.5	Modelos CNN com Duas Camadas Convolucionais	41
3.9.6	Modelos CNN com Três Camadas Convolucionais	42
3.9.7	Modelo CNN com Transfer Learning	42
3.9.8	Modelos de Redes Neurais Recorrentes (RNN)	42
3.9.9	Modelos RNN Simples	43
3.9.10	Modelos GRU (Gated Recurrent Units)	43
3.9.11	Modelos LSTM (Long Short-Term Memory)	44
3.10	Avaliação Experimental	44
3.11	Métricas de Avaliação Utilizadas	45
4	Resultados e discussões	47
4.1	Análise Exploratória dos Dados	47
4.1.1	Distribuição das Classes	47
4.1.2	Visualização da Distribuição	48
4.1.3	Análise das Diferenças de Medições (Concordância)	48
4.2	Análise Temporal dos Sinais	49
4.2.1	Verificação de Continuidade Temporal	49
4.2.2	Visualização dos Sinais Temporais	50

4.3	Análise no Domínio da Frequência	51
4.4	Análise com Espectrogramas	54
4.5	Eliminação de Ruído dos Dados	56
4.6	Avaliação dos Modelos de Machine Learning	57
4.7	Avaliação dos Modelos de Deep Learning	61
4.8	Comparação Geral dos Modelos	65
5	Conclusões	67
Bibliografia		68

1 Introdução

A segurança no trânsito é um desafio global, com aproximadamente 1,19 milhão de mortes anuais por acidentes de trânsito, segundo a Organização Mundial da Saúde (World Health Organization, 2023). Comportamentos agressivos ao volante, como excesso de velocidade e frenagens bruscas, estão entre as principais causas de acidentes graves, sendo responsáveis por 56% dos acidentes fatais em estudos norte-americanos (Paleti et al., 2010).

A popularização de smartphones com sensores embutidos (acelerômetros e giroscópios) oferece uma oportunidade para monitorar padrões de direção de forma acessível. Trabalhos recentes demonstram que dados de sensores de smartphones podem classificar comportamentos agressivos com até 94,07% de acurácia utilizando algoritmos de *Random Forest*, superando outras técnicas como SVM (86,16%) e outras redes neurais (86,14%) em cenários reais (Bose et al., 2018). Além disso, a integração dessas tecnologias em *smart cities* tem se mostrado promissora para reduzir acidentes e otimizar o fluxo de tráfego, especialmente considerando que fatores humanos respondem por 90-95% dos acidentes (Antić et al., 2018).

A análise do comportamento do motorista por meio de sensores busca identificar manobras perigosas como acelerações súbitas e curvas fechadas, que elevam o risco de colisões em até 30% quando associadas a direção agressiva (Paleti et al., 2010). Esses dados podem ser processados por algoritmos de aprendizado de máquina e aprendizado profundo em busca de identificar comportamentos em categorias como SLOW, NORMAL ou AGGRESSIVE. Esta abordagem é particularmente vantajosa por sua escalabilidade e baixo custo, alinhando-se aos princípios de cidades inteligentes (*smart cities*) que visam ambientes urbanos mais seguros (Algherbal e Ratrou, 2025).

Além disso, sistemas baseados em sensores de smartphone podem ser integrados a plataformas de feedback em tempo real, como o *DERSense*, que gera alertas imediatos ao detectar frenagens bruscas ou acelerações súbitas (Bose et al., 2018). Estudos mostram que essa abordagem participativa, combinada com aprendizado de máquina, permite identificar áreas críticas para direção agressiva e segmentos de via com más condições, facilitando intervenções direcionadas (Bose et al., 2018).

Dessa forma, a combinação de sensores acessíveis e técnicas de análise de dados abre caminho para soluções inovadoras em segurança viária, alinhando-se aos objetivos de *smart cities* de criar ambientes urbanos mais seguros e eficientes. Este trabalho explora essa abordagem, porém focando na identificação do comportamento de direção agressivo, usando dados de sensores e algoritmos de aprendizado de máquina e aprendizado profundo.

1.1 Objetivos

Este estudo tem como objetivo principal desenvolver e avaliar modelos de aprendizado de máquina (ML) e aprendizado profundo (DL) capazes de identificar comportamentos agressivos de motoristas com base em sinais multivariados coletados por sensores de smartphones.

1.1.1 Objetivos Específicos

- Realizar o pré-processamento e organização dos dados do dataset “Drive Behavior”, incluindo suavização dos sinais e conversão para classificação binária.
- Analisar os sinais no domínio do tempo e da frequência, empregando técnicas como FFT, PSD e espectrogramas para explorar padrões de direção.
- Desenvolver e treinar diferentes modelos de ML (como Regressão Logística, SVM, Random Forest, Naive Bayes e XGBoost) e avaliar seu desempenho na detecção de comportamento agressivo.
- Projetar e avaliar modelos de DL (MLP, CNN, RNN), considerando variações de arquitetura e estratégias de regularização.
- Aplicar Transfer Learning utilizando modelos pré-treinados para análise de sinais transformados em imagens, comparando sua eficácia com abordagens tradicionais.
- Comparar o desempenho entre os modelos de ML e DL com base em métricas como acurácia, recall, F1-score e acurácia balanceada, destacando os melhores resultados obtidos.

1.2 Estrutura do Documento

Com o objetivo de proporcionar uma leitura clara e progressiva, este trabalho está dividido em capítulos que abordam os conteúdos de forma lógica e gradual. A seção 2 traz a fundamentação teórica necessária para compreender os principais conceitos relacionados à análise de sinais, técnicas de processamento e modelos de aprendizado de máquina e aprendizado profundo utilizados ao longo do trabalho.

Na sequência, a seção 3 descreve em detalhes os procedimentos metodológicos adotados, desde a escolha e preparação do dataset até a construção e aplicação dos modelos. A seção 4 apresenta os resultados obtidos com base nas análises realizadas, tanto no domínio do tempo quanto no domínio da frequência, além do desempenho dos modelos implementados. Por fim, a seção 5 conclui o trabalho, destacando as contribuições alcançadas, as limitações encontradas e sugestões para possíveis estudos futuros.

2 Fundamentação Teórica

Esta seção apresenta os principais conceitos teóricos que fundamentam o desenvolvimento metodológico deste trabalho. Serão abordadas técnicas de análise de dados, análise temporal e análise espectral de sinais, bem como os modelos de aprendizado de máquina e aprendizado profundo utilizados para a identificação do motorista agressivo.

2.1 Método de Bland-Altman

O método de Bland-Altman é uma técnica estatística amplamente utilizada para avaliar a concordância entre dois métodos de medição. Este método é especialmente útil para identificar vieses sistemáticos e a presença de erros aleatórios entre duas medições que, teoricamente, deveriam fornecer resultados semelhantes.

De acordo com Martin Bland e Altman (1986), a simples correlação entre duas variáveis não garante concordância, pois mesmo quando os valores medidos por dois métodos são linearmente relacionados, eles podem apresentar diferenças significativas. O método de Bland-Altman propõe a construção de um gráfico da diferença entre as duas medições em função da média das mesmas, identificando possíveis vieses e variabilidade dos dados.

A abordagem consiste em calcular, para cada par de medições, a diferença e a média, conforme as fórmulas:

$$d_i = x_i - y_i \quad (1)$$

$$m_i = \frac{x_i + y_i}{2} \quad (2)$$

Onde x_i e y_i são as duas medições realizadas no mesmo instante de tempo. A diferença d_i representa o erro entre as medições, e m_i é a melhor estimativa disponível do valor “verdadeiro” desconhecido.

A partir dessas diferenças, calcula-se:

- A média das diferenças (viés), que indica o deslocamento sistemático entre os métodos:

$$\bar{d} = \frac{1}{n} \sum_{i=1}^n d_i \quad (3)$$

- O desvio padrão das diferenças:

$$s = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (d_i - \bar{d})^2} \quad (4)$$

Assumindo distribuição normal dos erros, estima-se os limites de concordância (também chamados de *limits of agreement*), onde espera-se que aproximadamente 95% das

diferenças estejam contidas:

$$\text{Limite Superior} = \bar{d} + 1,96 \times s \quad (5)$$

$$\text{Limite Inferior} = \bar{d} - 1,96 \times s \quad (6)$$

Esses limites são representados graficamente em um diagrama de Bland-Altman, onde se plota d_i versus m_i . A análise visual permite observar se há tendência nos erros (por exemplo, erros maiores em altas medições) ou presença de outliers.

2.2 Transformada Rápida de Fourier (FFT)

A Transformada Rápida de Fourier (FFT) é uma técnica matemática que permite decompor um sinal temporal em suas componentes de frequência. Ela é uma versão eficiente da Transformada de Fourier Discreta (DFT), calculando a amplitude e a fase de cada componente senoidal presente no sinal. Conforme destacado por Rao et al. (2010), a FFT desempenha um papel no processamento digital de sinais, sendo amplamente aplicada em sistemas de comunicação, processamento de áudio e vídeo, e análise espectral em geral.

A FFT transforma um sinal no domínio do tempo $x[n]$ em um sinal no domínio da frequência $X[k]$, de acordo com a expressão:

$$X[k] = \sum_{n=0}^{N-1} x[n] \cdot e^{-j\frac{2\pi}{N}kn} \quad (7)$$

Nesta expressão:

- $x[n]$ representa o sinal no tempo discreto.
- $X[k]$ é o sinal transformado no domínio da frequência.
- N é o número total de amostras.
- j é a unidade imaginária ($\sqrt{-1}$).

2.3 Densidade Espectral de Potência (PSD)

A Densidade Espectral de Potência (PSD) é uma função que descreve como a potência do sinal se distribui ao longo das frequências. Howard (2004) apresenta uma fundamentação teórica abrangente para a definição da PSD baseada na Transformada de Fourier direta, ressaltando sua utilidade em sistemas de comunicação e eletrônica, especialmente na análise de ruído e sinais aleatórios.

A estimativa da densidade espectral de potência (PSD) por meio da Transformada Rápida de Fourier (FFT), segue a abordagem clássica descrita por Rao et al. (2010), onde

a magnitude ao quadrado da FFT normalizada fornece uma representação da energia distribuída nas frequências, conforme a seguinte expressão:

$$PSD[k] = \frac{1}{N} \cdot |X[k]|^2 \quad (8)$$

Nesta expressão:

- $X[k]$ representa o valor da Transformada Rápida de Fourier (FFT) no índice k .
- $|X[k]|^2$ representa a potência na frequência k (magnitude ao quadrado).
- N é o número total de amostras do sinal, usado para normalizar a potência total distribuída nas frequências.

Essa função fornece uma representação clara da energia presente em cada frequência do sinal.

2.4 Espectrogramas e STFT

Um espectrograma é uma representação visual que mostra como o conteúdo espectral de um sinal varia com o tempo. Ele é construído a partir de janelas de tempo sobrepostas, aplicando-se a Transformada de Fourier a cada janela para obter a distribuição de frequências local. A intensidade das frequências ao longo do tempo é então representada por meio de um mapa de cores, onde cada cor indica a amplitude da componente espectral em determinado instante.

Matematicamente, o espectrograma de um sinal $x(t)$ é obtido pela aplicação da Transformada de Fourier de Curta Duração (STFT), que representa a análise espectral local do sinal por meio da aplicação de uma janela deslizante. A definição formal da STFT é dada por Boashash (2016):

$$S(t, f) = \left| \int_{-\infty}^{\infty} x(\tau) \cdot w(\tau - t) \cdot e^{-j2\pi f\tau} d\tau \right|^2 \quad (9)$$

Nesta expressão:

- $x(\tau)$: sinal no tempo.
- $w(\tau - t)$: janela que delimita o trecho do sinal analisado.
- f : frequência.
- $S(t, f)$: intensidade da frequência f no instante t .

O resultado da STFT é um espectro de frequência para cada intervalo de tempo, formando uma matriz cujas dimensões correspondem ao tempo e à frequência. A magnitude do valor em cada posição da matriz indica a intensidade da frequência naquele instante.

2.5 Modelos de Aprendizagem de Máquina (Machine Learning)

2.5.1 Regressão Logística

A regressão logística é um modelo estatístico amplamente utilizado para problemas de classificação binária, no qual a variável dependente é categórica e assume dois possíveis resultados. A função logística, também chamada de função sigmoide, foi originalmente formulada pelo matemático belga Pierre François Verhulst no século XIX para modelar o crescimento populacional, introduzindo o conceito da função logística (Cramer, 2002). Décadas depois, essa função foi adaptada para aplicações estatísticas e biomédicas, levando ao desenvolvimento do modelo logit por Joseph Berkson na década de 1940 (Cramer, 2004).

A principal característica desse modelo é a utilização da função logística (sigmoide) para mapear uma combinação linear das variáveis independentes em uma probabilidade entre 0 e 1, permitindo a modelagem de variáveis dependentes categóricas.

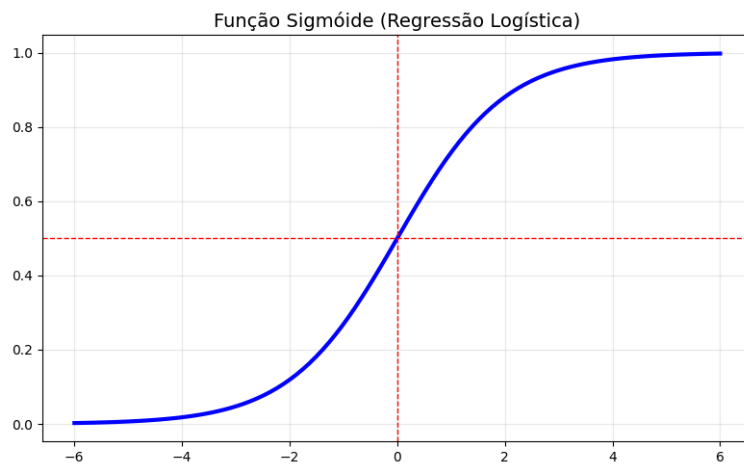


Figura 1: Curva logística representando a função sigmoide utilizada na regressão logística.

Matematicamente, a regressão logística estima a probabilidade de uma observação pertencer a uma determinada classe com base na transformação da combinação linear de variáveis independentes por meio da função logística, definida como:

$$P(Y = 1|X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_n X_n)}} \quad (10)$$

Onde:

- Y : variável dependente binária (ex.: 1 para "agressivo", 0 para "não agressivo").
- X_i : variáveis independentes (features).
- β_i : coeficientes estimados pelo modelo.

Além de sua robustez estatística, esse modelo é frequentemente utilizado em estudos sobre comportamento de direção, como no trabalho de Alomari et al. (2023), que utilizaram regressão logística para prever ações de motoristas em cruzamentos sinalizados com base em variáveis como velocidade e distância até o semáforo. De maneira semelhante, em Ghandour et al. (2021), aplicaram o modelo na classificação de comportamentos de motoristas em situações reais, observando sua eficácia em contextos onde a variável-alvo é binária, como distinguir entre direção agressiva e normal.

2.5.2 Support Vector Machines (SVM)

As Máquinas de Vetores de Suporte (SVM) são algoritmos de aprendizado supervisionado amplamente utilizados para tarefas de classificação e regressão. O conceito fundamental do SVM é encontrar o hiperplano que melhor separa as classes no espaço de características, maximizando a margem entre os pontos de dados mais próximos de cada classe, conhecidos como vetores de suporte (Burgess, 1998).

O SVM foi inicialmente proposto por Vladimir Vapnik e Alexey Chervonenkis em 1963, com aprimoramentos posteriores por Bernhard Boser, Isabelle Guyon e Vapnik em 1992, que introduziram o uso de funções kernel para lidar com dados não linearmente separáveis (Cristianini e Shawe-Taylor, 2000).

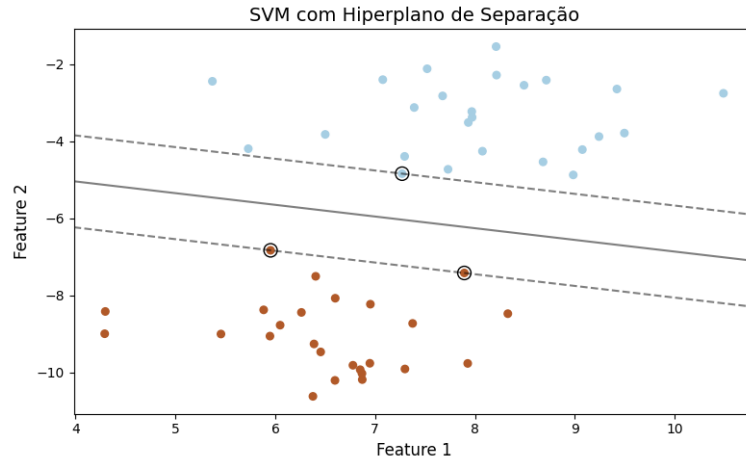


Figura 2: Exemplo de separação de classes utilizando SVM com diferentes kernels.

Matematicamente, o objetivo do SVM é resolver o seguinte problema de otimização:

$$\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 \quad \text{sujeito a} \quad y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1, \quad \forall i \quad (11)$$

Onde:

- $\mathbf{w}$: vetor de pesos.
- b : termo de bias.

- y_i : rótulo da classe da amostra i (+1 ou -1).
- $\mathbf{x}_i$: vetor de características da amostra i .

Para casos em que os dados não são linearmente separáveis, o SVM utiliza o truque do kernel, que mapeia os dados para um espaço de características de maior dimensão, permitindo a separação linear nesse novo espaço (Burges, 1998).

2.5.3 Random Forest

A Floresta Aleatória (Random Forest) é um algoritmo de aprendizado supervisionado baseado em *ensemble*, que combina múltiplas árvores de decisão para melhorar a precisão e controlar o *overfitting*. O método foi proposto por Leo Breiman em 2001, como uma extensão do *bagging*, incorporando a seleção aleatória de subconjuntos de atributos durante a construção de cada árvore, o que resulta em árvores menos correlacionadas e, conseqüentemente, em um modelo mais robusto (Breiman, 2001).

O algoritmo constrói diversas árvores de decisão utilizando diferentes subconjuntos aleatórios dos dados de treinamento e dos atributos. A predição final é obtida por meio da agregação das predições individuais das árvores, geralmente por votação majoritária no caso de classificação ou média no caso de regressão. Essa abordagem reduz a variância do modelo e melhora sua capacidade de generalização (Breiman, 2001).

Random Forest

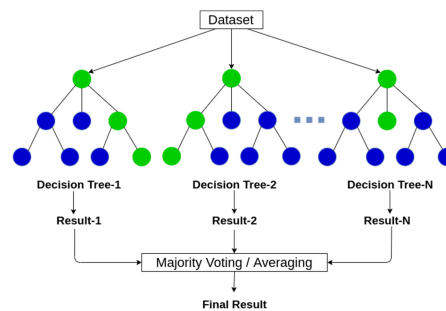


Figura 3: Exemplo ilustrativo do funcionamento do Random Forest.
(Fonte: Brital (2021)).

2.5.4 Naive Bayes

O Naive Bayes é um classificador probabilístico baseado no teorema de Bayes, que assume independência condicional entre as variáveis preditoras, dado o valor da variável

de classe. Apesar dessa suposição simplificadora, o Naive Bayes tem demonstrado desempenho competitivo em diversas aplicações práticas, especialmente em cenários com alta dimensionalidade e conjuntos de dados de tamanho limitado (Zhang, 2004).

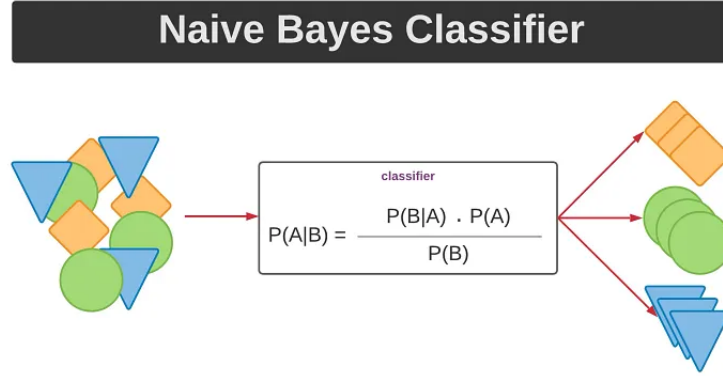


Figura 4: Ilustração do funcionamento do classificador Naive Bayes. (Fonte: Faleiros (2024)).

Para lidar com variáveis contínuas, utiliza-se a variante Gaussian Naive Bayes, que assume que os dados seguem uma distribuição normal dentro de cada classe. Essa abordagem permite calcular a probabilidade de uma observação pertencer a uma classe específica com base na média e variância dos atributos contínuos (John e Langley, 1995).

O modelo baseia-se na seguinte equação, derivada do teorema de Bayes com a suposição de independência:

$$P(C_k | x_1, x_2, \dots, x_n) = \frac{P(C_k) \cdot \prod_{i=1}^n P(x_i | C_k)}{P(x_1, x_2, \dots, x_n)} \quad (12)$$

Onde:

- $P(C_k | x_1, x_2, \dots, x_n)$: probabilidade posterior da classe C_k dado o vetor de características.
- $P(C_k)$: probabilidade a priori da classe C_k .
- $P(x_i | C_k)$: probabilidade da feature x_i dado que a classe é C_k .
- $P(x_1, x_2, \dots, x_n)$: probabilidade conjunta das features (constante para todas as classes).

2.5.5 XGBoost

O XGBoost é um algoritmo de aprendizado de máquina baseado em árvores de decisão que utiliza a técnica de *boosting* para melhorar a performance preditiva. Desenvolvido por Chen e Guestrin (2016), o XGBoost é conhecido por sua eficiência computacional

e capacidade de lidar com grandes volumes de dados, sendo amplamente utilizado em competições de ciência de dados e aplicações do mundo real.

O algoritmo constrói modelos de forma aditiva, onde cada novo modelo é ajustado para corrigir os erros dos modelos anteriores. A função objetivo do XGBoost incorpora uma aproximação de segunda ordem (Taylor) da função de perda, permitindo uma otimização mais precisa:

$$\mathcal{L}^{(t)} = \sum_{i=1}^n \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t) \quad (13)$$

Onde:

- $\mathcal{L}^{(t)}$: função de perda na iteração t .
- g_i : gradiente da função de perda em relação à predição anterior para a amostra i .
- h_i : o valor da segunda derivada (hessiana, no caso unidimensional aqui) da função de perda em relação à predição anterior para a amostra i .
- $f_t(x_i)$: predição do novo modelo para a amostra i .
- $\Omega(f_t)$: termo de regularização que penaliza a complexidade do modelo, promovendo a generalização.

A inclusão do termo de regularização $\Omega(f_t)$ ajuda a evitar o *overfitting*, tornando o modelo mais robusto. Além disso, o XGBoost implementa técnicas como poda de árvores e manipulação eficiente de dados esparsos, contribuindo para sua escalabilidade e desempenho superior (Chen e Guestrin, 2016).

2.6 Modelos de Aprendizado Profundo (Deep Learning)

2.6.1 Redes MLP (Multilayer Perceptron)

As redes MLP (Multilayer Perceptron) são um tipo de rede neural artificial do tipo *feedforward* composta por múltiplas camadas de neurônios totalmente conectados. Cada neurônio em uma camada está ligado a todos os neurônios da camada subsequente, permitindo a modelagem de relações complexas entre as variáveis de entrada e saída. A estrutura básica de uma MLP inclui uma camada de entrada, uma ou mais camadas ocultas e uma camada de saída, como visto na Figura 5.

Uma característica fundamental das MLPs é a utilização de funções de ativação não lineares, como a sigmoide, tangente hiperbólica (tanh) ou ReLU, que introduzem não linearidades necessárias para que a rede aprenda representações complexas dos dados. O treinamento dessas redes é realizado por meio do algoritmo de retropropagação do erro

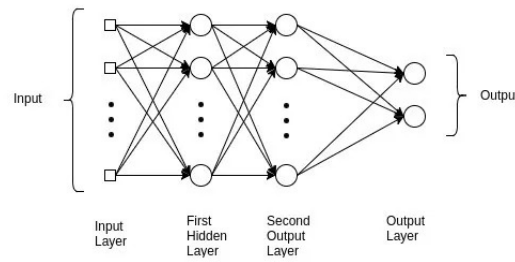


Figura 5: Arquitetura da Rede MPL. (Fonte: Peixoto (2020)).

(backpropagation), que ajusta os pesos com base no gradiente do erro em relação aos parâmetros da rede (Rumelhart et al., 1986).

Teoricamente, uma MLP com pelo menos uma camada oculta com número suficiente de neurônios e funções de ativação adequadas é capaz de aproximar qualquer função contínua definida em um intervalo compacto, conforme demonstrado por Cybenko no seu teorema do aproximador universal (Cybenko, 1989). Essa propriedade torna as MLPs altamente versáteis para tarefas de classificação e regressão.

As MLPs são aplicadas amplamente em contextos como reconhecimento de padrões, previsão de séries temporais, análise de sinais e tomada de decisão em sistemas inteligentes. Haykin destaca que sua capacidade de generalização, combinada com técnicas modernas de regularização e otimização, permite que esses modelos sejam aplicados com eficiência mesmo em cenários de alta dimensionalidade (Haykin, 1999).

2.6.2 Redes Neurais Convolucionais (CNN)

As Redes Neurais Convolucionais (CNNs) são uma classe de redes neurais profundas projetadas para processar dados com estrutura de grade, como imagens. Inspiradas na organização do córtex visual animal, onde neurônios respondem a estímulos em regiões específicas do campo visual, as CNNs utilizam camadas convolucionais para capturar padrões locais nos dados de entrada (LeCun et al., 2015).

A arquitetura típica de uma CNN é composta por camadas convolucionais, camadas de pooling e camadas totalmente conectadas. As camadas convolucionais aplicam filtros (ou kernels) que percorrem a entrada, permitindo a detecção de características como bordas e texturas, assim como mostrado na Figura 6. As camadas de pooling reduzem a dimensionalidade espacial, mantendo as informações mais relevantes, enquanto as camadas totalmente conectadas realizam a classificação final (Goodfellow et al., 2016).

Uma característica fundamental das CNNs é a capacidade de aprender representações hierárquicas dos dados. As primeiras camadas capturam características simples, como bordas e cores, enquanto as camadas mais profundas combinam essas informações para detectar padrões mais complexos, como formas e objetos completos (Zeiler e Fergus, 2014).

O treinamento das CNNs é realizado por meio do algoritmo de retropropagação, ajus-

tando os pesos dos filtros para minimizar a função de perda. Técnicas como normalização por lotes (batch normalization) e dropout são frequentemente empregadas para melhorar a generalização e prevenir o overfitting (Ioffe e Szegedy, 2015).

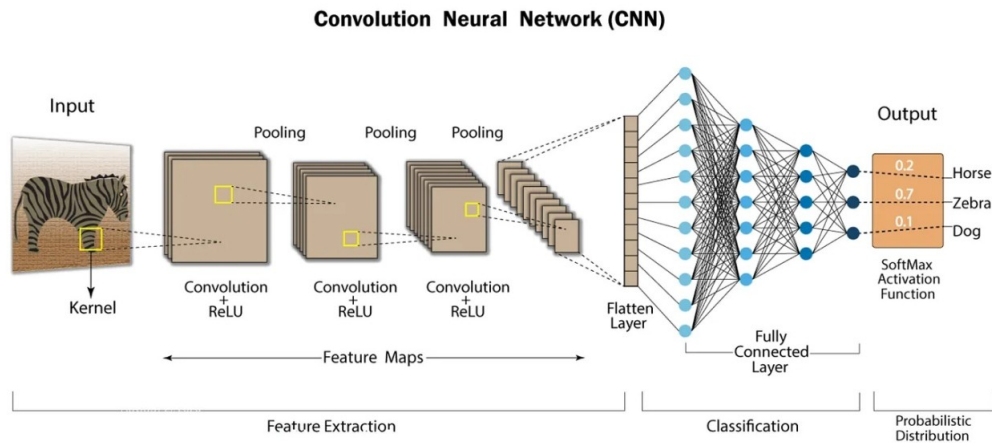


Figura 6: Arquitetura de uma CNN. (Fonte: Haque (2023)).

As CNNs têm sido amplamente aplicadas em diversas tarefas, incluindo reconhecimento de imagens, detecção de objetos, análise de vídeos e processamento de linguagem natural. Sua eficácia em capturar padrões espaciais e hierárquicos as torna uma ferramenta poderosa no campo do aprendizado profundo (Krizhevsky et al., 2012).

2.6.3 Redes Neurais Recorrentes (RNN)

As Redes Neurais Recorrentes (RNNs) são uma classe de redes neurais artificiais projetadas para processar dados sequenciais, como texto, fala e séries temporais, onde a ordem dos elementos é importante. Diferentemente das redes neurais feedforward, que processam entradas de forma independente, as RNNs utilizam conexões recorrentes, onde a saída de um neurônio em um instante de tempo é realimentada como entrada para a rede no próximo instante, como visto na Figura 7. Isso permite que as RNNs capturem dependências temporais e padrões dentro das sequências (Schmidhuber, 2015).

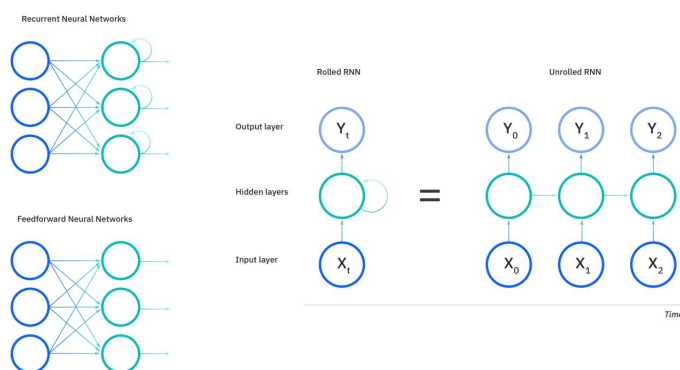


Figura 7: Arquitetura de uma CNN. (Fonte: Stryker (2021)).

O bloco fundamental das RNNs é a unidade recorrente, que mantém um estado oculto tendo uma forma de memória que é atualizada a cada passo de tempo com base na entrada atual e no estado oculto anterior. Esse mecanismo de feedback permite que a rede aprenda com entradas passadas e incorpore esse conhecimento no processamento atual. As RNNs têm sido aplicadas com sucesso em tarefas como reconhecimento de escrita manual conectada, reconhecimento de fala, processamento de linguagem natural e tradução automática neural (Graves et al., 2013).

No entanto, RNNs tradicionais sofrem com o problema do desvanecimento do gradiente, o que limita sua capacidade de aprender dependências de longo prazo. Esse problema foi abordado com o desenvolvimento da arquitetura Long Short-Term Memory (LSTM) em 1997, tornando-se a variante padrão de RNN para lidar com dependências de longo prazo. Posteriormente, as Unidades Recorrentes com Portas (GRUs) foram introduzidas como uma alternativa computacionalmente mais eficiente (Hochreiter e Schmidhuber, 1997).

2.7 Aprendizado por Transferência (Transfer Learning)

O *Transfer Learning* (Aprendizado por Transferência) é uma técnica de aprendizado de máquina que visa aproveitar o conhecimento adquirido em uma tarefa-fonte para melhorar o desempenho em uma tarefa-alvo relacionada, especialmente quando há escassez de dados rotulados na tarefa-alvo. Diferentemente dos métodos tradicionais de aprendizado supervisionado, que assumem que o treinamento e o teste ocorrem no mesmo domínio e distribuição, o Transfer Learning permite a adaptação entre domínios distintos (Pan e Yang, 2009). É possível ver um exemplo desse funcionamento na Figura 8 onde aproveitou o conhecimento de um modelo pré-treinado que identifica cachorros para sua nova tarefa de identificar gatos.

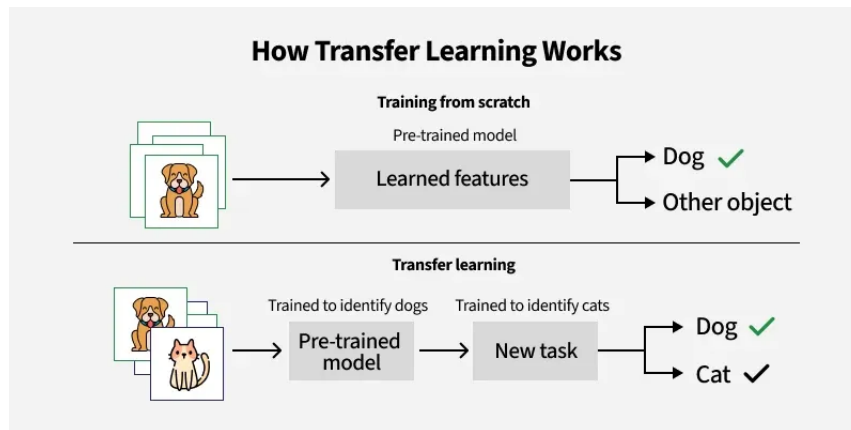


Figura 8: Exemplo de funcionamento do Transfer Learning. (Fonte: GeeksforGeeks (2025)).

No contexto do *Deep Learning*, o Transfer Learning tem se mostrado particularmente eficaz. Modelos profundos pré-treinados em grandes conjuntos de dados, como o ImageNet, podem ser ajustados para tarefas específicas com conjuntos de dados menores, reduzindo significativamente o tempo de treinamento e os recursos computacionais necessários (Zhuang et al., 2021). Essa abordagem é amplamente utilizada em aplicações como visão computacional e processamento de linguagem natural.

Existem diversas estratégias de Transfer Learning, incluindo a adaptação de domínios, onde o objetivo é minimizar a discrepância entre as distribuições dos dados da tarefa-fonte e da tarefa-alvo, e o aprendizado multitarefa, que busca aprender múltiplas tarefas simultaneamente, compartilhando representações comuns (Tan et al., 2018). A escolha da estratégia adequada depende das características específicas das tarefas envolvidas.

3 Metodologia

Nesta seção, serão apresentados os métodos utilizados para alcançar os objetivos deste trabalho. A metodologia está abordando as etapas principais: Dataset, ambiente experimental, pré-processamento, análise dos dados, criação dos modelos e avaliação de desempenho. Todas essas etapas serão descritas detalhadamente nas subseções a seguir.

3.1 Dataset e Coleta de Dados

Para a realização deste trabalho, utilizou-se o dataset intitulado "Drive Behavior", obtido diretamente da plataforma Kaggle. É importante destacar que os dados não foram coletados pelo autor deste trabalho, mas sim disponibilizados publicamente na referida plataforma por seus criadores originais. Todos os códigos realizados durante o trabalho foram feitos dentro do Google Colab.

O dataset contém dados capturados através de sensores de smartphones, especificamente um dispositivo Samsung Galaxy S21, coletando informações a uma taxa de 2 amostras por segundo. As medições incluem dados de acelerômetro e giroscópio ao longo dos eixos X, Y e Z, sendo estes os principais atributos utilizados para caracterizar o comportamento de direção dos motoristas. Os sinais registrados representam a aceleração (m/s^2) e a rotação ($^\circ/\text{s}$) nos respectivos eixos.

Os dados são rotulados em três classes distintas: **SLOW**, **NORMAL** e **AGGRESSIVE**. No entanto, para fins deste trabalho, realizou-se uma conversão para uma classificação binária, na qual a classe **AGGRESSIVE** foi mantida, enquanto as classes **SLOW** e **NORMAL** foram agrupadas na categoria **NÃO AGRESSIVE**. Essa simplificação visa facilitar a identificação de comportamentos agressivos no trânsito, o que é o foco principal da pesquisa.

Após o pré-processamento, os dados foram divididos em conjuntos de treino, validação e teste, mantendo-se a proporção de 50% para o conjunto de treino e 50% divididos igualmente entre validação e teste. Essa divisão foi realizada com o objetivo de garantir que os modelos treinados possuam uma boa generalização e sejam capazes de lidar com dados novos, evitando problemas de overfitting.

Além disso, foi realizada uma análise de continuidade temporal para verificar a existência de falhas ou interrupções no registro dos dados. Não foram encontradas descontinuidades significativas, permitindo que o sinal fosse tratado como uma sequência contínua.

Por fim, os dados brutos passaram por um processo de suavização utilizando um filtro gaussiano, com uma janela de tamanho 5 e um desvio padrão de 3. Este procedimento teve como objetivo reduzir ruídos indesejados e melhorar a qualidade das sequências temporais utilizadas na etapa de treinamento.

3.2 Análise Inicial

Para compreender a estrutura dos dados e identificar possíveis problemas que pudessem interferir no processamento e modelagem, foi realizada uma análise inicial do dataset. Essa etapa teve como principal objetivo garantir a consistência e qualidade dos dados utilizados ao longo do trabalho.

Inicialmente, foi realizada a inspeção das características principais do dataset, verificando o cabeçalho e as estatísticas descritivas para cada variável. Essa inspeção tinha como objetivo tentar identificar possíveis valores ausentes, discrepantes ou fora do intervalo esperado, que poderiam indicar problemas de coleta ou armazenamento dos dados.

Na sequência, foi verificada a presença de valores faltantes (missing values), o que poderia comprometer a eficácia dos modelos de aprendizado de máquina. Porém, não foram encontrados valores faltantes significativos, garantindo que o conjunto de dados estivesse completo para a análise.

Outro ponto relevante foi a verificação da continuidade temporal dos dados. Por serem capturados a partir de sensores de smartphones, havia a possibilidade de descontinuidades no registro dos sinais, como interrupções ou perdas de amostragem. Para isso, foi realizada a ordenação dos registros pelo campo de timestamp e a verificação da diferença de tempo entre amostras consecutivas.

Além disso, foi calculada a distribuição das classes (SLOW, NORMAL, AGGRESSIVE) para identificar possíveis desbalanceamentos. A análise revelou que a classe AGGRESSIVE era consideravelmente menos representada em comparação às demais. Para lidar com esse problema, optou-se pela aplicação de pesos diferenciados nas classes durante o treinamento dos modelos, aumentando a importância das amostras da classe minoritária. Essa mudança obteve um impacto positivo do balanceamento na representação da classe AGGRESSIVE.

3.3 Análise de Diferenças Temporais

Nesta seção, serão abordadas as diferenças temporais observadas nos dados brutos provenientes dos sensores do smartphone. A análise das diferenças temporais é essencial para garantir que as sequências coletadas possam ser tratadas como contínuas, permitindo a aplicação de técnicas de aprendizado profundo que capturam padrões temporais.

3.3.1 Verificação de Continuidade Temporal

Os sinais coletados pelos sensores do dispositivo móvel foram organizados em função do timestamp, a fim de garantir a ordem cronológica correta das amostras. Inicialmente, foi realizada uma verificação da diferença de tempo entre amostras consecutivas para identificar possíveis interrupções no registro dos dados, como falhas de coleta ou atrasos

no processamento. Após a ordenação pelo campo de timestamp, foi constatado que as diferenças temporais entre amostras consecutivas estavam dentro do intervalo esperado, confirmando que os dados coletados podem ser considerados contínuos.

3.3.2 Análise das Diferenças de Medições

Para verificar a concordância entre medições consecutivas dos sinais dos sensores, foi aplicado o método de Bland-Altman que foi discutido anteriormente na Seção 2.

No presente trabalho, ele foi aplicado utilizando as 2 amostras por segundo presentes no mesmo timestamp do dataset para estimar o nível de concordância dos dados.

A partir da aplicação desse método nos dados dos sinais do acelerômetro e do giroscópio, foram observadas as seguintes características:

- Embora os dados sejam bastante estranhos e apresentem diferenças significativas. Além disso, parece não haver um intervalo específico em que os dados discordem mais.
- Os limites de concordância (média ± 2 vezes o desvio padrão) foram estabelecidos para cada eixo dos sinais, garantindo que a maioria das diferenças estivesse dentro desse intervalo. Dessa forma, podemos dizer que a maioria dos dados “concordam” no método.

3.3.3 Visualização e Interpretação

Os gráficos de Bland-Altman foram construídos para cada variável (aceleração e giroscópio nos eixos X, Y, Z), permitindo identificar visualmente os pontos fora dos limites de concordância. Estes pontos indicam possíveis inconsistências de medição ou comportamentos atípicos que podem estar relacionados a ações agressivas do motorista.

Como resultado da análise de diferenças temporais, verificou-se que a maioria das medições consecutivas apresenta alta concordância, indicando que as sequências temporais podem ser utilizadas com confiança na modelagem preditiva. Apenas em situações extremas, típicas de direção agressiva, observou-se aumento da variabilidade entre as medições.

Essa abordagem permitiu garantir a integridade dos dados temporais, essencial para modelos que dependem de padrões sequenciais, como RNN e LSTM.

3.4 Criação de Features

A criação de features é uma etapa fundamental para garantir que os modelos de aprendizado de máquina e aprendizado profundo consigam capturar de forma eficaz os padrões associados ao comportamento agressivo do motorista. Para este trabalho, foi criada uma nova variável derivada dos sinais brutos dos sensores, além de realizadas transformações nos dados originais para aprimorar a qualidade das entradas dos modelos.

3.4.1 Nova Feature

Durante a análise exploratória dos sinais, observou-se que os valores brutos das componentes de aceleração nos eixos X, Y e Z, isoladamente, não representavam adequadamente a intensidade total do movimento. Para capturar de maneira mais robusta a magnitude da aceleração, foi calculado o módulo da aceleração, definido como:

$$Acc_{mod} = \sqrt{AccX^2 + AccY^2 + AccZ^2} \quad (14)$$

Essa nova feature possibilita identificar com mais precisão picos de intensidade que podem estar associados a comportamentos agressivos, como frenagens bruscas, acelerações rápidas ou curvas fechadas.

Além do módulo da aceleração, todas variáveis também foram analisadas individualmente para verificar sua relevância na distinção dos perfis de condução. Para garantir que os dados fossem uniformemente escalados, todas as variáveis passaram por um processo de normalização, de modo a reduzir o impacto de diferenças de escala entre os sinais coletados.

3.4.2 Seleção das Features

Após ter sido criado a feature Acc_{mod} , foram realizadas análises de todas as features no domínio da frequência. A partir das análises realizadas, as seguintes features foram selecionadas para a construção dos modelos:

- $AccX$, $AccY$, $AccZ$: componentes brutas da aceleração.
- $GyroX$, $GyroY$, $GyroZ$: componentes brutas da rotação.
- Acc_{mod} : módulo da aceleração, criado para representar a intensidade total do movimento.

A escolha dessas features levou em consideração tanto sua relevância estatística quanto seu potencial para capturar variações relacionadas ao comportamento agressivo do motorista. Essa abordagem visa aumentar a capacidade dos modelos em identificar padrões específicos de direção, principalmente os associados a eventos rápidos e intensos, como aqueles característicos da classe AGGRESSIVE.

3.5 Análise de Sinais no Domínio da Frequência

A análise de sinais no domínio da frequência é uma etapa essencial para identificar padrões ocultos nos dados temporais, especialmente no contexto da detecção de comportamento agressivo de motoristas. Por meio de técnicas espectrais, como a Transformada Rápida de Fourier (FFT) e a Densidade Espectral de Potência (PSD), é possível investigar

como a energia dos sinais se distribui ao longo das frequências, proporcionando uma interpretação mais profunda dos movimentos capturados pelos sensores. Ambas as técnicas foram explicadas previamente na Seção 2.

3.5.1 Transformada Rápida de Fourier (FFT)

A FFT permite identificar as frequências dominantes no sinal, sendo particularmente útil para detectar variações rápidas características de comportamentos agressivos. No contexto deste trabalho, a FFT foi aplicada às variáveis de aceleração e giroscópio, utilizando as amostras ao longo do tempo para cada eixo (X, Y, Z). Os gráficos da FFT foram gerados para visualizar a amplitude da componente real (em vermelho) e imaginária (em azul) ao longo das frequências.

3.5.2 Densidade Espectral de Potência (PSD)

No contexto da análise de direção agressiva, a PSD possibilita identificar quais frequências carregam mais energia, auxiliando na caracterização dos movimentos típicos de cada perfil de direção.

A PSD dos sinais foi plotada para cada variável (aceleração e giroscópio), tanto para os sinais brutos quanto para o módulo da aceleração Acc_{mod} .

3.5.3 Implementação da Análise Espectral

Para implementar a FFT e a PSD, utilizou-se a biblioteca `numpy` para realizar as transformadas e a biblioteca `matplotlib` para visualização dos resultados. Inicialmente, os sinais foram agrupados por classe (SLOW, NORMAL, AGGRESSIVE) e, em seguida, aplicou-se a análise espectral a cada variável, plotando os resultados para comparação.

O primeiro conjunto de gráficos exibe os sinais brutos ao longo do tempo, diferenciados por classe, utilizando cores para facilitar a visualização: verde para SLOW, azul para NORMAL e vermelho para AGGRESSIVE.

Em seguida, foram gerados gráficos específicos para cada variável, aplicando a FFT e a PSD para identificar as frequências predominantes. Cada gráfico contém tanto a parte real quanto a imaginária da FFT, bem como a PSD, permitindo identificar não apenas a frequência dominante, mas também a intensidade relativa das componentes de alta e baixa frequência.

Por fim, para garantir uma análise completa, também foram gerados gráficos que comparam as classes individualmente, destacando como o comportamento espectral varia entre SLOW, NORMAL e AGGRESSIVE para cada variável analisada.

3.6 Análise com Espectrogramas

Na tentativa de identificar mais padrões dinâmicos que ocorrem ao longo da sequência temporal, foi utilizada a técnica de espectrograma nos dados, que permite visualizar como a energia de um sinal se distribui ao longo do tempo e das frequências simultaneamente.

3.6.1 Aplicação e Visualização dos Espectrogramas

Para realizar a análise dos espectrogramas, foram utilizados os sinais brutos das variáveis coletadas (acelerômetro e giroscópio nos eixos X, Y e Z), assim como a variável derivada Acc_{mod} , que representa a magnitude da aceleração.

O cálculo dos espectrogramas foi realizado utilizando a função `spectrogram` da biblioteca `scipy.signal`, que retorna as frequências (f), os tempos (t) e a matriz de densidade espectral (S_{xx}). A visualização foi feita com o método `pcolormesh`, onde as cores representam a intensidade da densidade espectral.

3.6.2 Visualização por Classes e Variáveis

Foram gerados dois conjuntos de gráficos espectrais para comparação:

- Todos os dados combinados, representando o comportamento médio de todas as classes.
- Dados separados por classe (**SLOW**, **NORMAL**, **AGGRESSIVE**) para cada variável, facilitando a comparação dos padrões específicos de cada perfil de direção.

Os gráficos foram organizados de maneira que, para cada variável (como $AccX$, $AccY$, $AccZ$, $GyroX$, $GyroY$, $GyroZ$ e Acc_{mod}), houvesse um gráfico mostrando o comportamento global e três gráficos individuais para cada classe.

3.7 Processamento dos Dados para ML/DL

Após a realização das análises exploratórias e espectrais dos dados, tornou-se necessário realizar um processamento adequado para que os modelos de aprendizado de máquina (ML) e aprendizado profundo (DL) pudessem ser aplicados de maneira eficiente e robusta. O processamento dos dados incluiu etapas de eliminação de ruído, criação de sequências temporais e preparação para a classificação binária, considerando as características específicas do problema.

3.7.1 Eliminação de Ruído dos Dados

Os dados coletados por sensores de smartphones frequentemente apresentam ruídos devido a vibrações, interferências externas ou variações não relacionadas ao comportamento do motorista. Para garantir a qualidade dos dados utilizados nos modelos de

aprendizado de máquina, foram aplicadas técnicas de suavização. Essas técnicas são fundamentais para reduzir ruídos e melhorar a qualidade dos dados, conforme destacado por Han et al. (2006), que afirma que o pré-processamento é uma etapa crítica no processo de mineração de dados.

As seguintes técnicas de suavização foram utilizadas:

- **Média Móvel (Rolling Average):** Utiliza uma janela deslizante para calcular a média dos valores, suavizando variações abruptas. Foi utilizada uma janela de tamanho 5 para obter suavização adequada.
- **Suavização Exponencial (Exponential Smoothing):** Atribui maior peso às observações mais recentes, mantendo a capacidade de adaptação a mudanças rápidas. Utilizou-se um fator de suavização $\alpha=0.1$.
- **Filtro Gaussiano (Gaussian Smoothing):** Aplica um filtro com uma distribuição gaussiana, atribuindo pesos maiores aos valores centrais da janela. Utilizou-se uma janela de tamanho 5 e desvio padrão igual a 3.

Para avaliar a eficiência das técnicas de suavização, os dados de treino foram visualizados comparando os sinais originais (Raw) com os sinais suavizados pelas três técnicas mencionadas.

3.7.2 Conversão para Classificação Binária

O objetivo principal deste trabalho é identificar comportamentos de direção agressiva. Portanto, as classes originais (SLOW, NORMAL, AGGRESSIVE) foram convertidas para uma classificação binária, conforme descrito a seguir:

- Classe AGGRESSIVE (0) foi mapeada para a classe positiva (1).
- Classes SLOW (2) e NORMAL (1) foram agrupadas na classe negativa (0).

Essa conversão visa simplificar o problema, focando na identificação de comportamentos agressivos, considerados os mais relevantes para a aplicação prática no contexto de segurança no trânsito.

3.7.3 Estruturação dos Dados para Modelagem

Os modelos propostos visam identificar comportamentos agressivos com base em sequências temporais, uma vez que a aceleração de um único instante não fornece informações suficientes para capturar o padrão de direção. Dessa forma, os dados de entrada foram organizados em janelas temporais de 10 segundos, garantindo que cada sequência contenha informações suficientes para modelagem.

Para garantir a integridade dos conjuntos de treino, validação e teste, os dados foram divididos em 50% para treino e 50% para validação e teste (divididos igualmente). As divisões foram feitas de maneira estratificada para manter a proporção das classes.

Os dados foram então estruturados da seguinte forma:

- **Dados tradicionais (sem sequência):** Conjunto de treino, validação e teste no formato tradicional para modelos clássicos de ML.
- **Dados sequenciais:** Conjuntos de treino, validação e teste organizados em janelas de 10 timesteps para modelos DL como RNN e LSTM.

A criação das sequências foi feita respeitando a continuidade temporal, garantindo que os modelos possam capturar a evolução dos sinais ao longo do tempo. Cada sequência é composta por 10 amostras consecutivas para cada uma das variáveis (AccX, AccY, AccZ, GyroX, GyroY, GyroZ e Acc_{mod}), mantendo a coerência temporal dos dados.

3.7.4 Distribuição das Classes nos Conjuntos

Após a divisão dos conjuntos de dados, verificou-se a distribuição das classes para garantir que não houvesse desbalanceamento acentuado entre treino, validação e teste. As distribuições obtidas foram as seguintes:

- **Treinamento:** 2268 amostras da classe NÃO AGGRESSIVE (0) e 812 amostras da classe AGGRESSIVE (1).
- **Validação:** 1264 amostras da classe NÃO AGGRESSIVE (0) e 556 amostras da classe AGGRESSIVE (1).
- **Teste:** 1263 amostras da classe NÃO AGGRESSIVE (0) e 557 amostras da classe AGGRESSIVE (1).

Essa verificação garante que os conjuntos possuam proporções semelhantes, evitando que o modelo seja tendencioso para a classe majoritária.

3.7.5 Conclusão do Processamento

A etapa de processamento dos dados foi fundamental para garantir que os sinais estivessem adequadamente filtrados e estruturados para os modelos propostos. A combinação de técnicas de suavização e organização sequencial dos dados garantiu a integridade e a qualidade das amostras, maximizando as chances de identificar comportamentos agressivos com precisão.

3.8 Modelos de Machine Learning

Nesta seção, serão abordados os modelos de aprendizado de máquina (ML) utilizados neste trabalho para a identificação de comportamentos agressivos de motoristas. A escolha dos modelos foi motivada pela necessidade de testar diferentes abordagens, incluindo métodos lineares, probabilísticos e baseados em árvores de decisão, visando identificar qual técnica apresenta melhor desempenho na classificação binária do comportamento de direção.

3.8.1 Implementação da Classe Base para Modelos de Classificação

Para facilitar a criação, treinamento e avaliação dos modelos de ML, foi desenvolvida uma classe genérica chamada `MLClassificationModels`. Essa classe funciona como um wrapper que encapsula a criação e avaliação dos modelos, padronizando os procedimentos e facilitando a comparação dos resultados.

A classe possui funcionalidades para:

- Treinamento dos modelos, com suporte para balanceamento de classes via pesos.
- Avaliação dos modelos utilizando métricas padrão, como acurácia, acurácia balanceada e relatório de classificação.
- Visualização de matrizes de confusão para melhor interpretação dos resultados.

Essa abordagem modular permitiu que diferentes algoritmos fossem facilmente integrados ao pipeline de testes, garantindo padronização e consistência nos procedimentos de avaliação.

3.8.2 Modelos Utilizados

Os modelos testados no estudo foram escolhidos com base na diversidade de abordagens, desde métodos lineares até técnicas mais robustas baseadas em árvores. A seguir, são apresentados os modelos utilizados e suas configurações.

1. Regressão Logística (Logistic Regression)

Foi utilizado o modelo `LogisticRegression` da biblioteca `scikit-learn` para realizar a tarefa de classificação binária. Inicialmente, o modelo foi treinado sem qualquer forma de normalização, permitindo avaliar seu desempenho bruto sobre os dados de entrada. Em seguida, o mesmo modelo foi testado com a aplicação da padronização dos dados por meio do `StandardScaler`, incorporado via pipeline. O objetivo dessa segunda abordagem foi verificar o impacto da normalização das variáveis de entrada na performance do modelo, especialmente considerando a sensibilidade de modelos lineares à escala dos atributos.

2. Máquinas de Vetores de Suporte (Support Vector Machines – SVM)

No caso do SVM, utilizou-se o modelo `SVC` da biblioteca `scikit-learn`, em conjunto com a normalização dos dados, aplicada por meio de `StandardScaler`. Como o conjunto de dados apresentava desbalanceamento entre as classes, foi configurado um dicionário de pesos no parâmetro `class_weight`, atribuindo peso 3 para a classe `AGGRESSIVE` e peso 1 para as demais amostras. Isso permitiu penalizar erros cometidos na classe minoritária de forma proporcional, favorecendo uma aprendizagem mais equilibrada entre os dois grupos.

3. Floresta Aleatória (Random Forest)

Para a Floresta Aleatória, foi utilizado o modelo `RandomForestClassifier` da biblioteca `scikit-learn`, com paralelismo ativado via `n_jobs=-1` para acelerar o treinamento. O classificador foi configurado para lidar com o desbalanceamento das classes por meio do parâmetro `class_weight`, utilizando a mesma estratégia aplicada na SVM — atribuindo peso 3 à classe `AGGRESSIVE` e peso 1 às demais. O número padrão de árvores foi mantido, e a implementação seguiu o fluxo definido pela classe `MLClassificationModels`, com etapas de ajuste e avaliação nos dados de validação.

4. Naive Bayes (GaussianNB)

O modelo `GaussianNB`, disponível na biblioteca `scikit-learn`, foi utilizado por sua adequação a atributos contínuos, assumindo distribuição Gaussiana para cada variável dentro de cada classe. Para refletir a distribuição real das amostras no conjunto de dados, foi configurado o parâmetro `priors`, atribuindo 30% para a classe `SLOW/NORMAL` e 70% para a classe `AGGRESSIVE`. Essa estratégia de especificação explícita das probabilidades a priori permitiu ajustar o modelo à realidade observada no conjunto de treinamento, sem necessidade de reamostragem ou balanceamento externo.

5. XGBoost

Para o modelo XGBoost, foi utilizado o `XGBClassifier`, com configuração padrão de profundidade e número de árvores, e integrado à estrutura da classe `ClassificationModel`. A estratégia de balanceamento de classes foi implementada por meio do parâmetro `sample_weight_map`, atribuindo peso 3 à classe `AGGRESSIVE` e peso 1 às demais amostras durante o treinamento. Essa abordagem permitiu considerar a importância relativa das classes sem alterar a estrutura do conjunto de dados, beneficiando o aprendizado sem perda de informações. O modelo foi treinado sem aplicação de normalização, considerando que o XGBoost é inerentemente robusto à escala dos atributos.

3.8.3 Procedimento de Treinamento e Avaliação

Para cada modelo, foi seguido o seguinte procedimento:

1. Inicialização do modelo utilizando a classe `MLClassificationModels`, especificando os parâmetros necessários.
2. Treinamento com o conjunto de dados balanceado, utilizando pesos quando aplicável.

3. Avaliação do modelo nos dados de validação, calculando acurácia simples e balanceada.
4. Geração da matriz de confusão e relatório de classificação.

Os modelos foram ajustados utilizando os dados previamente processados, e a configuração de cada modelo foi mantida constante para garantir uma comparação justa entre os resultados. Todos os experimentos foram realizados no ambiente Google Colab, utilizando bibliotecas como `scikit-learn` e `xgboost`.

Essa abordagem metodológica garante que os modelos de ML empregados possam ser comparados de maneira clara e objetiva, proporcionando uma análise criteriosa sobre a eficácia de cada técnica para a tarefa de identificação de direção agressiva.

3.9 Modelos de Aprendizado Profundo (Deep Learning)

Nesta seção, serão abordados os modelos de Aprendizado Profundo (Deep Learning) desenvolvidos para identificar comportamentos agressivos de motoristas. Utilizando técnicas mais avançadas e arquiteturas neurais profundas, busquei capturar padrões temporais complexos que métodos tradicionais de ML não conseguem modelar adequadamente.

3.9.1 Implementação da Classe Base para Modelos de DL

Para padronizar a criação, treinamento e avaliação dos modelos de DL, foi desenvolvida a classe base `DeepLearningModels`. Essa classe encapsula as operações comuns a diferentes arquiteturas, incluindo compilação, treinamento, teste e visualização dos resultados, garantindo modularidade e eficiência no desenvolvimento dos modelos.

A classe `DeepLearningModels` possui as seguintes funcionalidades:

- Criação de modelos personalizados através de subclasses específicas.
- Treinamento com dados de entrada, incluindo ajuste de hiperparâmetros e uso de técnicas de regularização.
- Avaliação de desempenho com métricas como acurácia, precisão, recall e AUC.
- Visualização das curvas de loss, acurácia, precisão e recall ao longo das épocas.
- Geração da matriz de confusão para análise dos resultados.

Essa estrutura modular facilita a inclusão de novas arquiteturas, mantendo o pipeline organizado e flexível para diferentes testes e abordagens.

3.9.2 Modelos MLP (Multilayer Perceptron)

Os modelos MLP foram escolhidos por sua simplicidade e eficiência na modelagem de dados tabulares e não sequenciais. A seguir, são apresentados os seis modelos desenvolvidos.

Modelo MLP 1: Este modelo possui uma arquitetura simples com 4 camadas densas e funções de ativação `tanh`. A estratégia de regularização inclui uma camada `Dropout` com taxa de 0,4 para evitar overfitting. A função de perda utilizada foi `BinaryCrossentropy`, com o otimizador `Adam` e uma taxa de aprendizado que decresce exponencialmente. O treinamento foi monitorado com o `EarlyStopping` para evitar treinamento excessivo.

Modelo MLP 2: Esta variante do MLP utiliza camadas adicionais com `BatchNormalization` para melhorar a estabilidade durante o treinamento. A presença de múltiplas camadas com ativação `tanh` visa aumentar a capacidade do modelo de capturar padrões não lineares. Além disso, a inclusão de uma camada `Dropout` com taxa de 0,7 visa mitigar o risco de overfitting.

Modelo MLP 3: O terceiro modelo MLP utiliza uma arquitetura mais profunda, com mais camadas ocultas e menor número de unidades por camada (5, 3 e 2 neurônios, respectivamente). A ativação `tanh` foi mantida, mas a presença de mais camadas proporciona maior capacidade de expressão, embora com risco de overfitting controlado pelo `Dropout`.

Modelo MLP 4: Este modelo se diferencia pelo uso da função de ativação `selu` (Scaled Exponential Linear Unit), que melhora o fluxo de gradientes e estabiliza a variação das ativações. O uso da `BatchNormalization` auxilia na normalização das saídas das camadas, garantindo um treinamento mais eficiente e rápido.

Modelo MLP 5: Este modelo é uma versão mais complexa do MLP 3, com mais camadas densas, mantendo a função de ativação `tanh`. A ideia é aumentar a profundidade da rede para explorar relações não lineares mais complexas entre as variáveis de entrada. A regularização foi mantida com `Dropout` e `BatchNormalization`.

Modelo MLP 6: Por fim, o modelo MLP 6 apresenta uma arquitetura simplificada, com menos camadas e unidades, para servir de comparação com os modelos mais complexos. A função de ativação `tanh` foi utilizada para garantir continuidade e suavidade nas ativações.

3.9.3 Estratégias de Treinamento e Otimização dos modelos MLP

Os modelos MLP foram treinados utilizando o otimizador `Adam` com taxa de aprendizado adaptativa, definida por um decaimento exponencial para garantir estabilidade no aprendizado. A função de perda utilizada foi a `BinaryCrossentropy`, adequada para problemas de classificação binária.

Para evitar overfitting, foram empregadas as seguintes técnicas de regularização:

- **Dropout:** Para reduzir a correlação entre as unidades ocultas e melhorar a generalização.
- **Batch Normalization:** Para estabilizar o treinamento e acelerar a convergência.
- **Early Stopping:** Para interromper o treinamento quando a perda no conjunto de validação não melhorar após 10 épocas.

Cada modelo foi treinado por até 300 épocas com um batch size de 32, e o treinamento foi conduzido utilizando múltiplas seeds para garantir a reprodutibilidade dos resultados. As curvas de perda e acurácia foram monitoradas para analisar a convergência do modelo.

3.9.4 Modelos de Redes Neurais Convolucionais (CNN)

As Redes Neurais Convolucionais (CNNs) são amplamente utilizadas em problemas que envolvem dados com estrutura espacial, como imagens e séries temporais transformadas em representações visuais. No contexto deste trabalho, as CNNs foram empregadas para identificar padrões recorrentes em séries temporais provenientes de dados de comportamento de direção.

Para aprimorar a capacidade de análise dos sinais temporais, utilizamos abordagens baseadas em gráficos de recorrência (Recurrence Plots), conforme sugerido por Spiegel et al. (2014), que apresentaram uma métrica de distância baseada em recorrência para séries temporais multivariadas. Essa abordagem permite identificar segmentos recorrentes de trajetórias similares em diferentes posições temporais, sendo útil para análise de padrões comportamentais em dados de direção.

3.9.5 Modelos CNN com Duas Camadas Convolucionais

Inicialmente, foi desenvolvido o modelo CNN composto por duas camadas convolucionais (Modelo 1). A primeira camada realiza a extração de características por meio de filtros convolucionais com tamanho ajustável, seguida por uma função de ativação Leaky ReLU. Em seguida, uma segunda camada convolucional expande o número de filtros, aprimorando a capacidade de representação do modelo. Após a extração de características, utiliza-se uma camada de pooling para redução da dimensionalidade e uma camada de dropout para evitar overfitting. Por fim, uma camada totalmente conectada (dense) é utilizada para classificação binária com ativação sigmoid.

Os hiperparâmetros principais incluem:

- **Número de filtros:** 16 e 32
- **Função de ativação:** Leaky ReLU com $\alpha=0.2$
- **Função de perda:** Binary Crossentropy

- **Otimizador:** Adadelta
- **Taxa de dropout:** 0.25

3.9.6 Modelos CNN com Três Camadas Convolucionais

Com o objetivo de explorar uma capacidade de extração de características mais profunda, desenvolvi modelos com três camadas convolucionais (Modelo 2 e 3). A primeira e a segunda camadas utilizam filtros convolucionais semelhantes ao modelo anterior, enquanto a terceira camada convolucional promove uma maior abstração dos padrões identificados. A estrutura subsequente inclui pooling, dropout e camadas densas para garantir uma robusta classificação.

3.9.7 Modelo CNN com Transfer Learning

A utilização de Transfer Learning foi explorada através da arquitetura MobileNetV2, previamente treinada no conjunto de dados ImageNet. Para adequação ao problema em estudo, os gráficos de recorrência foram replicados para formar imagens de tamanho (40,40,3). Apenas as camadas finais foram ajustadas, mantendo as primeiras camadas congeladas para aproveitar o aprendizado prévio. O modelo final consiste na estrutura do MobileNetV2 conectada a camadas densas adicionais, otimizadas para o conjunto de dados utilizado.

De acordo com Spiegel et al. (2014), o uso de gráficos de recorrência pode melhorar significativamente a identificação de padrões recorrentes, especialmente em conjuntos de dados com comportamento temporal complexo. Isso se alinha com o propósito do uso das CNNs no presente trabalho, uma vez que a do motorista agressivo depende de padrões recorrentes em dados multivariados.

A estrutura dos modelos foi ajustada para garantir um equilíbrio entre complexidade e capacidade de generalização, utilizando técnicas de regularização como dropout e inicialização Glorot Uniform para os pesos das camadas convolucionais.

3.9.8 Modelos de Redes Neurais Recorrentes (RNN)

As Redes Neurais Recorrentes (RNNs) são especialmente adequadas para modelar dados sequenciais, uma vez que permitem capturar dependências temporais ao longo do tempo. No contexto deste trabalho, as RNNs foram exploradas com o objetivo de identificar comportamentos agressivos de direção a partir de sequências temporais de dados obtidos por sensores.

A principal vantagem das RNNs reside na sua capacidade de manter uma memória interna, o que as torna eficientes para tarefas em que o contexto anterior é relevante para prever o próximo estado. No entanto, RNNs tradicionais podem sofrer com o problema

de desvanecimento do gradiente, especialmente em sequências longas. Por esse motivo, também foram exploradas variantes mais robustas, como as redes GRU (Gated Recurrent Units) e LSTM (Long Short-Term Memory), que mitigam esse problema por meio de mecanismos específicos de controle de memória.

3.9.9 Modelos RNN Simples

O primeiro modelo desenvolvido foi uma RNN simples com diferentes números de camadas ocultas, variando entre 1 e 4 camadas. Cada camada foi composta por unidades recorrentes configuradas para manter a continuidade da informação ao longo da sequência. A função de ativação utilizada foi a `tanh`, com inicialização `Glorot Uniform` dos pesos. O otimizador utilizado foi o `Adam`, combinado com uma taxa de aprendizado decrescente exponencialmente.

Os hiperparâmetros principais incluem:

- **Número de unidades ocultas:** 5 a 20
- **Função de ativação:** `tanh`
- **Função de perda:** Binary Crossentropy
- **Otimizador:** Adam
- **Inicialização de pesos:** Glorot Uniform

3.9.10 Modelos GRU (Gated Recurrent Units)

Para tentar superar as limitações das RNNs simples, foram testadas redes GRU, conhecidas por sua eficiência na manutenção de informações ao longo de sequências mais longas. O modelo GRU foi configurado com diferentes combinações de unidades ocultas e profundidades, utilizando a função de ativação `tanh` e dropout para reduzir o risco de overfitting.

Os hiperparâmetros principais incluem:

- **Número de unidades ocultas:** 5 a 10
- **Função de ativação:** `tanh`
- **Dropout:** 0.4
- **Otimizador:** Adam
- **Função de perda:** Binary Crossentropy

3.9.11 Modelos LSTM (Long Short-Term Memory)

Por último, foram desenvolvidos modelos LSTM, que se destacam por sua capacidade de armazenar informações por períodos mais longos, graças às células de memória e aos mecanismos de esquecimento. O modelo LSTM foi testado com diferentes profundidades e unidades ocultas, configurados para maximizar o aprendizado temporal sem perda significativa de precisão.

Os hiperparâmetros principais incluem:

- **Número de unidades ocultas:** 5 a 10
- **Função de ativação:** tanh
- **Dropout:** 0.4
- **Otimizador:** Adam
- **Função de perda:** Binary Crossentropy

Em todos os modelos, a estratégia de balanceamento foi adotada com a aplicação de pesos diferenciados para as classes, garantindo que a classe AGGRESSIVE tivesse uma representação adequada no processo de aprendizado. A arquitetura final dos modelos RNN, GRU e LSTM foi ajustada para maximizar a eficiência e a capacidade de generalização, mantendo um controle rigoroso do overfitting por meio de técnicas de regularização.

3.10 Avaliação Experimental

Para avaliar o desempenho dos modelos desenvolvidos neste trabalho, foram conduzidos experimentos padronizados com os conjuntos de treino, validação e teste previamente definidos a partir do dataset.

Os modelos de aprendizado de máquina (ML) foram avaliados com base em métricas tradicionais de classificação binária, como acurácia, acurácia balanceada, recall, precisão e F1-score. As avaliações foram feitas com os dados no formato tabular, onde cada amostra representava um instante temporal com múltiplas variáveis.

Para os modelos de aprendizado profundo (DL), os dados foram organizados em janelas temporais de três timesteps consecutivos, preservando a estrutura sequencial dos sinais. O desempenho desses modelos foi avaliado sobre os dados de validação e teste, com monitoramento contínuo das métricas de treino (perda, acurácia, precisão e recall) por época. As curvas de desempenho ao longo das épocas foram plotadas para facilitar a análise de overfitting.

Durante o treinamento, foram aplicadas técnicas de regularização como **Dropout**, **BatchNormalization** e **EarlyStopping**, além do uso de otimizadores adaptativos como Adam e Adadelta, conforme a arquitetura.

Os experimentos foram conduzidos na plataforma Google Colab, utilizando Python e bibliotecas como `scikit-learn`, `xgboost`, `tensorflow` e `keras`. Para garantir reprodutibilidade, todos os testes foram realizados com sementes aleatórias fixadas.

Ao final, os modelos foram comparados entre si com base em seu desempenho na tarefa de classificação binária, destacando-se aqueles que obtiveram maior recall para a classe AGGRESSIVE, considerada mais relevante para o contexto de segurança no trânsito.

3.11 Métricas de Avaliação Utilizadas

A avaliação do desempenho dos modelos de classificação é fundamental para aferir sua capacidade preditiva. Para tal, diversas métricas são calculadas a partir dos componentes da matriz de confusão: Verdadeiros Positivos (VP), Verdadeiros Negativos (VN), Falsos Positivos (FP) e Falsos Negativos (FN). A seguir, são descritas as métricas utilizadas neste trabalho.

Acurácia (Accuracy): A acurácia representa a proporção de previsões corretas (VP e VN) em relação ao total de instâncias analisadas.

$$\text{Acurácia} = \frac{VP + VN}{VP + VN + FP + FN}$$

Embora seja uma medida intuitiva, pode ser enganosa em cenários com classes desbalanceadas, onde uma classe majoritária pode inflacionar o resultado.

Precisão (Precision): Para uma classe específica (considerada como positiva), a precisão indica a proporção de instâncias que o modelo classificou como positivas e que eram, de fato, positivas.

$$\text{Precisão} = \frac{VP}{VP + FP}$$

É particularmente relevante quando o custo de um Falso Positivo (classificar erroneamente uma instância como positiva) é alto.

Recall (Revocação ou Sensibilidade): Também referente a uma classe específica (positiva), o recall mede a proporção de todas as instâncias verdadeiramente positivas que foram corretamente identificadas pelo modelo.

$$\text{Recall} = \frac{VP}{VP + FN}$$

É crucial quando o custo de um Falso Negativo (não identificar uma instância positiva) é significativo.

F1-Score: O F1-Score é a média harmônica da Precisão e do Recall para uma classe específica. Ele busca um equilíbrio entre essas duas métricas.

$$\text{F1-Score} = 2 \times \frac{\text{Precisão} \times \text{Recall}}{\text{Precisão} + \text{Recall}} = \frac{2 \times VP}{2 \times VP + FP + FN}$$

É uma métrica útil, especialmente em situações com classes desbalanceadas ou quando tanto Falsos Positivos quanto Falsos Negativos têm custos importantes.

Acurácia Balanceada (Balanced Accuracy): A acurácia balanceada é calculada como a média aritmética do Recall para cada classe individualmente. Em um contexto binário, é a média do Recall da classe positiva e do Recall da classe negativa (também conhecida como especificidade).

$$\text{Acurácia Balanceada} = \frac{1}{2} \left(\frac{VP}{VP + FN} + \frac{VN}{VN + FP} \right)$$

Esta métrica é mais apropriada do que a acurácia tradicional para avaliar o desempenho em datasets com classes desbalanceadas, pois considera o desempenho em cada classe de forma equitativa.

AUC (Area Under the ROC Curve): A Área Sob a Curva ROC (Receiver Operating Characteristic) é uma medida da capacidade de um modelo em distinguir entre as classes. A curva ROC é construída plotando-se a Taxa de Verdadeiros Positivos (Recall) no eixo Y contra a Taxa de Falsos Positivos ($FP / (FP + VN)$) no eixo X, para diversos limiares de decisão. A AUC representa a área sob esta curva, com valores variando de 0 a 1. Um valor de AUC próximo a 1 indica um excelente poder de discriminação, enquanto 0.5 sugere um desempenho equivalente ao aleatório. É uma métrica robusta à variação do limiar de classificação e ao desbalanceamento de classes.

4 Resultados e discussões

Nesta seção são apresentados os resultados obtidos a partir das análises realizadas nos sinais dos sensores, bem como o desempenho dos modelos de aprendizado de máquina e aprendizado profundo aplicados. Ao final, é feita uma análise comparativa destacando os modelos com melhor desempenho.

4.1 Análise Exploratória dos Dados

Agora será apresentada a análise exploratória dos dados após as etapas de pré-processamento. Esta análise visa compreender a distribuição das classes nos conjuntos de treinamento, validação e teste, bem como verificar a proporção entre as classes para garantir a integridade dos modelos de classificação.

4.1.1 Distribuição das Classes

Primeiramente, foi realizada a verificação da distribuição das classes nos conjuntos de treinamento, validação e teste, tanto na versão original do dataset quanto após a conversão para classificação binária.

Os resultados mostraram que, no conjunto de treinamento original, a classe *AGGRESSIVE* possui 814 amostras, enquanto as classes *NORMAL* e *SLOW* possuem 997 e 1273 amostras, respectivamente. Após a conversão para classificação binária, a nova distribuição ficou da seguinte forma: 812 amostras para a classe positiva (*AGGRESSIVE*) e 2268 amostras para a classe negativa (*NÃO AGGRESSIVE*).

No conjunto de teste, a distribuição original revelou 1113 amostras para a classe *AGGRESSIVE*, 1200 para a classe *NORMAL* e 1331 para a classe *SLOW*. Após a conversão, o conjunto de teste ficou composto por 557 amostras positivas e 1263 amostras negativas.

A Tabela 1 resume a distribuição das classes após a conversão binária para os conjuntos de treinamento, validação e teste.

Conjunto	Classe	Quantidade	Proporção (%)
Treinamento	AGGRESSIVE	812	26
Treinamento	NÃO AGGRESSIVE	2268	74
Teste	AGGRESSIVE	557	31
Teste	NÃO AGGRESSIVE	1263	69
Validação	AGGRESSIVE	556	31
Validação	NÃO AGGRESSIVE	1264	69

Tabela 1: Distribuição das Classes após a Conversão Binária

4.1.2 Visualização da Distribuição

Para facilitar a interpretação dos resultados, foram gerados gráficos de barras mostrando a proporção das classes nos conjuntos de treinamento, validação e teste. A Figura 9 ilustra essa distribuição das classes.

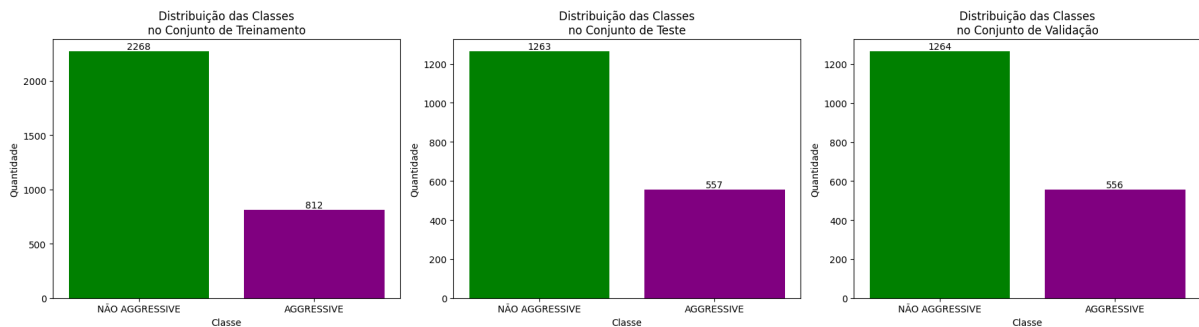


Figura 9: Gráficos das distribuições das classes.

Os gráficos evidenciam que, embora o conjunto de dados apresente certo desbalanceamento entre as classes, a aplicação de técnicas de balanceamento, como a ponderação de classes, foi utilizada para mitigar possíveis problemas de viés no treinamento dos modelos. Essa abordagem é fundamental para garantir que o modelo não se torne tendencioso, priorizando as classes *NORMAL* e *SLOW*.

4.1.3 Análise das Diferenças de Medições (Concordância)

Após a aplicação do método de Bland-Altman, foram gerados gráficos para os eixos dos sinais do acelerômetro e do giroscópio (Figura 10). Observa-se que a maioria dos pontos nos gráficos está dentro dos limites de concordância, indicando que os métodos apresentam boa concordância na maior parte dos casos.

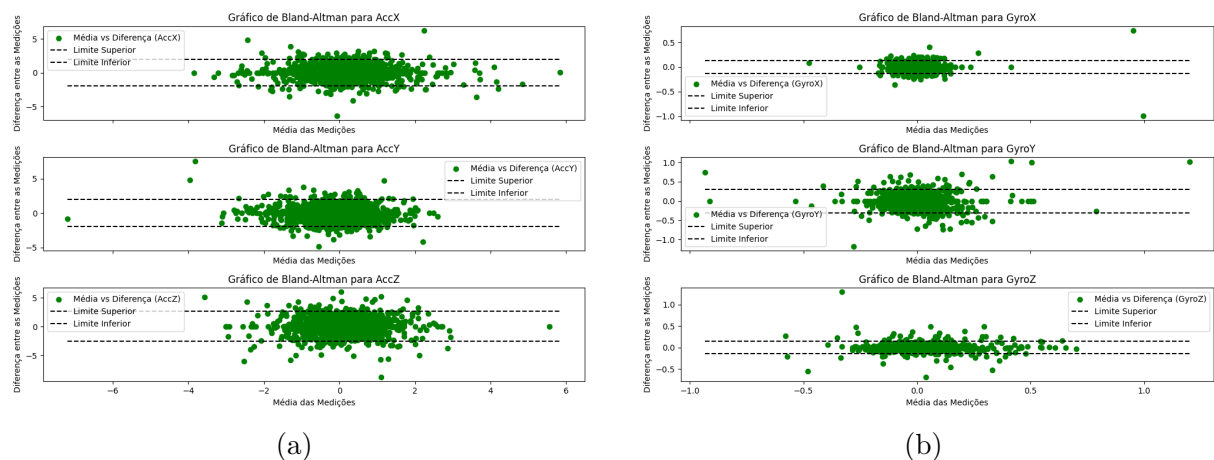


Figura 10: A imagem (a) é referente Gráfico de Bland-Altman para os eixos do Acelerômetro e a imagem (b) para os eixos do Giroscópio

Nos gráficos dos eixos do acelerômetro, observa-se que a variabilidade é um pouco maior quando comparada aos gráficos dos eixos do giroscópio. Isso sugere que as medições de aceleração apresentam maior variabilidade em cenários de direção agressiva. No caso do giroscópio, as medições apresentam diferenças menores, indicando que o comportamento rotacional do veículo é mais consistente do que o de aceleração.

Visualizando os gráficos, uma conclusão possível é que as discrepâncias ocorrem principalmente em situações de direção agressiva, quando há manobras bruscas ou mudanças rápidas de direção. Esse comportamento é esperado, dado que movimentos rápidos e intensos tendem a gerar medições menos precisas devido ao ruído introduzido pelo movimento brusco do dispositivo.

Para melhor quantificar a concordância, foi calculada a proporção de pontos que se encontram dentro e fora dos limites de concordância para cada eixo, conforme apresentado na Tabela 2.

Sinal	Eixo	Dentro dos Limites (%)	Fora dos Limites (%)
Acelerômetro	AccX	94.24	5.76
Acelerômetro	AccY	94.30	5.70
Acelerômetro	AccZ	94.78	5.22
Giroscópio	GyroX	95.50	4.50
Giroscópio	GyroY	94.60	5.40
Giroscópio	GyroZ	96.16	3.84

Tabela 2: Porcentagem de pontos dentro e fora dos limites de concordância para os eixos dos sinais.

A partir da tabela, observa-se que os eixos do giroscópio apresentam uma porcentagem levemente maior de pontos dentro dos limites de concordância quando comparados aos eixos do acelerômetro. Isso confirma a observação de que os sinais de rotação têm menor variabilidade e, portanto, maior concordância entre as medições consecutivas.

Essas análises reforçam a necessidade de técnicas de pré-processamento que tratem a variabilidade dos sinais, para que os modelos preditivos não sejam influenciados por discrepâncias pontuais.

4.2 Análise Temporal dos Sinais

4.2.1 Verificação de Continuidade Temporal

Primeiramente, foi realizada uma verificação para identificar possíveis lacunas nos registros temporais. Após a execução do código de verificação, foi constatado que não há falta de registros no intervalo de timestamps do conjunto de dados, garantindo que

todos os segundos estão devidamente representados. Isso indica que o dataset não possui interrupções significativas.

Ainda assim, foi realizado um procedimento adicional para interpolar valores caso houvesse alguma descontinuidade. Embora essa correção tenha sido implementada, ela não teve impacto prático, pois a continuidade já estava garantida desde o início.

4.2.2 Visualização dos Sinais Temporais

Para visualizar a distribuição temporal dos sinais, foram gerados gráficos dos sinais dos sensores (acelerômetro e giroscópio) ao longo do tempo, discriminando as classes de comportamento de direção: *AGGRESSIVE*, *NORMAL* e *SLOW*. A Figura 11 ilustra os sinais ao longo dos timestamps para cada eixo do acelerômetro e do giroscópio, além da magnitude da aceleração (Acc_{mod}).

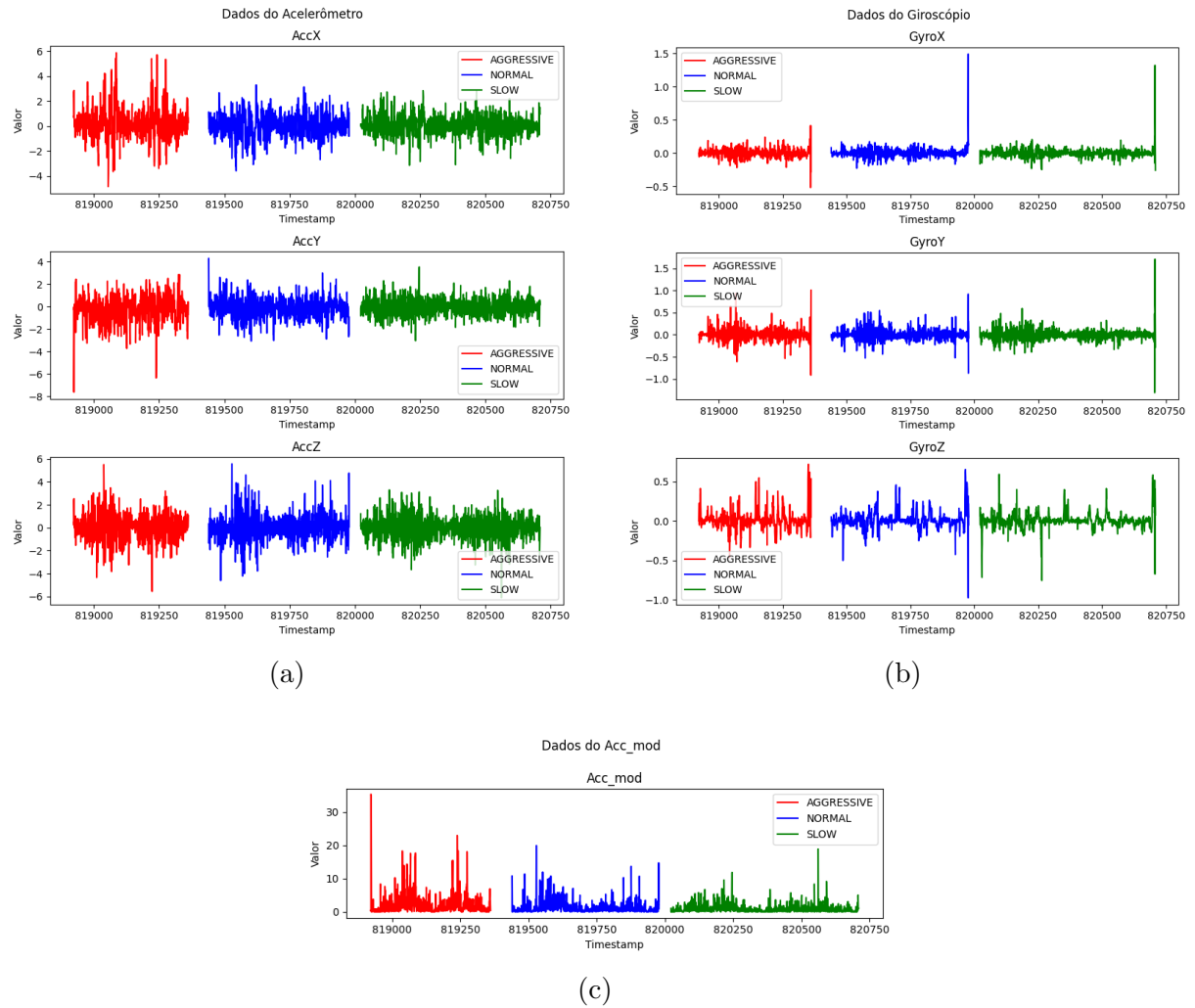


Figura 11: Gráficos dos sinais dos sensores ao longo do tempo para as classes AGGRESSIVE (vermelho), NORMAL (azul) e SLOW (verde). (a) mostra os dados do acelerômetro, (b) os dados do giroscópio e (c) o módulo do acelerômetro (Acc_{mod}).

Os gráficos mostram claramente que os sinais dos sensores variam conforme a classe de direção. Observa-se que os sinais da classe *AGGRESSIVE* apresentam maior variabilidade e amplitude em comparação com as classes *NORMAL* e *SLOW*. Essa diferença é particularmente evidente nos sinais dos eixos do acelerômetro ($AccX$ e $AccY$) e na magnitude da aceleração (Acc_{mod}), o que pode ser interpretado como movimentos mais bruscos e rápidos associados a comportamentos agressivos.

Por outro lado, as classes *NORMAL* e *SLOW* apresentam sinais com menor amplitude e variação, caracterizando uma condução mais estável e controlada. Nos sinais dos giroscópios ($GyroX$, $GyroY$ e $GyroZ$), também é notável algumas distinções entre as classes, mas com menor intensidade se comparada aos sinais de aceleração.

A continuidade dos sinais, aliada à distinção entre as classes, demonstra que os dados possuem a qualidade necessária para serem utilizados nos modelos preditivos, possibilitando que os padrões temporais dos comportamentos de direção sejam capturados.

4.3 Análise no Domínio da Frequência

Inicialmente, os gráficos de FFT e PSD foram gerados para as variáveis do conjunto completo, conforme ilustrado na Figura 12. As variáveis analisadas incluíram os eixos do acelerômetro ($AccX$, $AccY$, $AccZ$), do giroscópio ($GyroX$, $GyroY$, $GyroZ$) e o módulo da aceleração (Acc_{mod}).

Para aprofundar a análise, os gráficos foram gerados separadamente por classe de direção (*AGGRESSIVE*, *NORMAL*, *SLOW*), conforme ilustrado nas Figuras 13, 14 e 15. Essa separação permitiu observar como os padrões de frequência e potência se comportam em diferentes estilos de condução.

Na Tabela 3, são apresentadas as frequências dominantes de cada variável por classe, e na Tabela 4, a potência (PSD) associada à frequência dominante.

A análise espectral revelou que a classe *AGGRESSIVE* apresenta características distintas de intensidade e variação nos sinais. As variáveis $AccX$ e $AccY$, por exemplo, têm frequência dominante em 0 Hz, indicando acelerações constantes e sustentadas, mas com potências significativamente altas (48.3 e 85.6, respectivamente), evidenciando movimentos intensos. O eixo vertical $AccZ$ apresentou uma frequência dominante de 0.8722 Hz, sugerindo oscilações rápidas no plano vertical, possivelmente relacionadas a impactos, lombadas ou dirigibilidade em pisos irregulares.

A variável Acc_{mod} , que representa a intensidade total da aceleração, apresentou potência extremamente elevada (4096.6) em frequência zero para a classe *AGGRESSIVE*, o que indica que motoristas agressivos mantêm níveis de aceleração muito altos, mesmo que com variação frequencial limitada. Isso sugere que a intensidade da aceleração é um dos principais fatores discriminativos nesse perfil de condução.

Para a classe *NORMAL*, observa-se uma maior distribuição de frequência nas variáveis

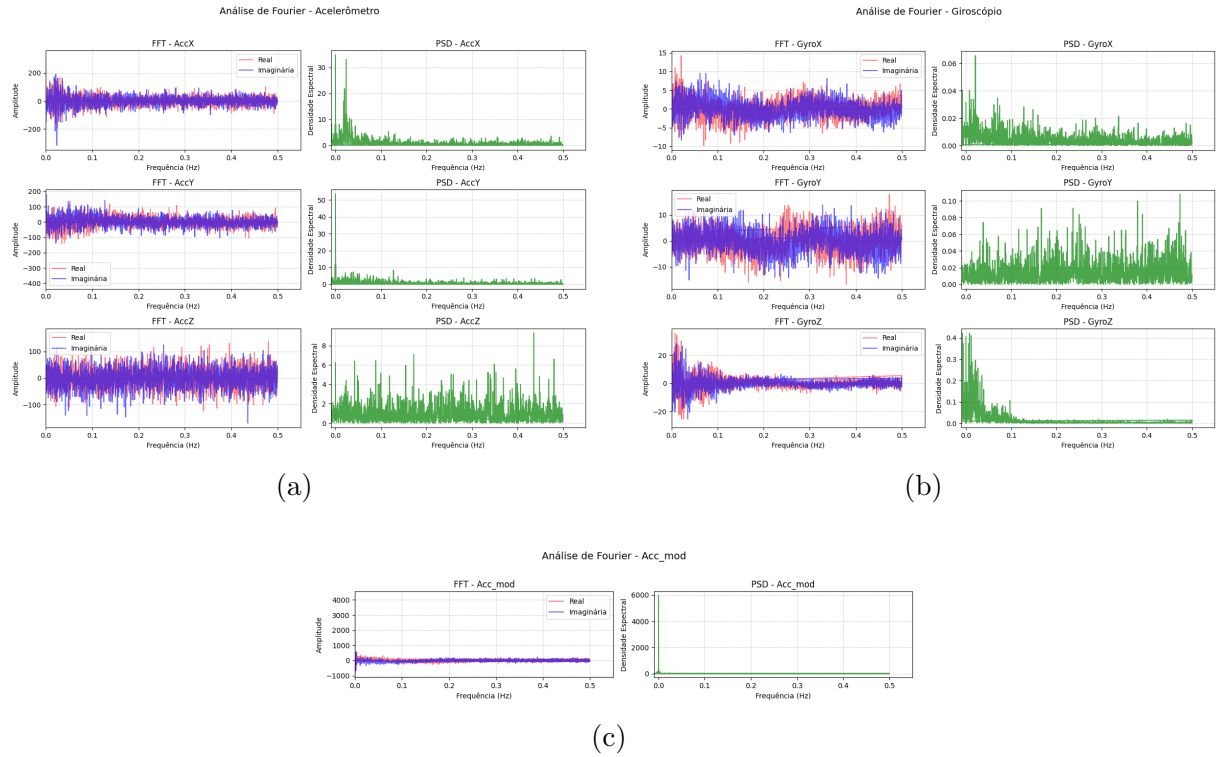


Figura 12: Análise de Fourier (FFT e PSD) dos sinais dos sensores. (a) mostra a transformada de Fourier para os eixos do acelerômetro, (b) para os eixos do giroscópio e (c) para o módulo do acelerômetro (Acc_{mod}).

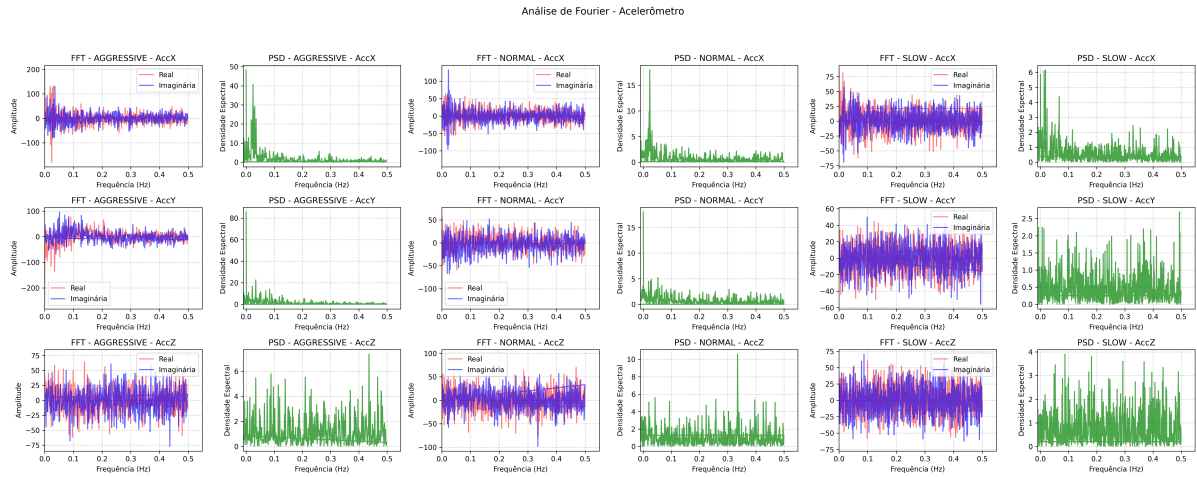


Figura 13: Análise de Fourier (FFT e PSD) separada por classe no Acelerômetro.

do acelerômetro, com frequência dominante em torno de 0.04 a 0.67 Hz, o que aponta para uma condução menos intensa, porém mais variável. Já a classe *SLOW* apresenta as menores potências em quase todas as variáveis, além de um pico inesperado em 0.9882 Hz na variável *AccY*, o que pode estar relacionado a pequenas correções rápidas de direção, como manobras suaves ou ajustes laterais frequentes.

Em relação aos sinais do giroscópio, destaca-se a variável *GyroZ*, que apresentou um

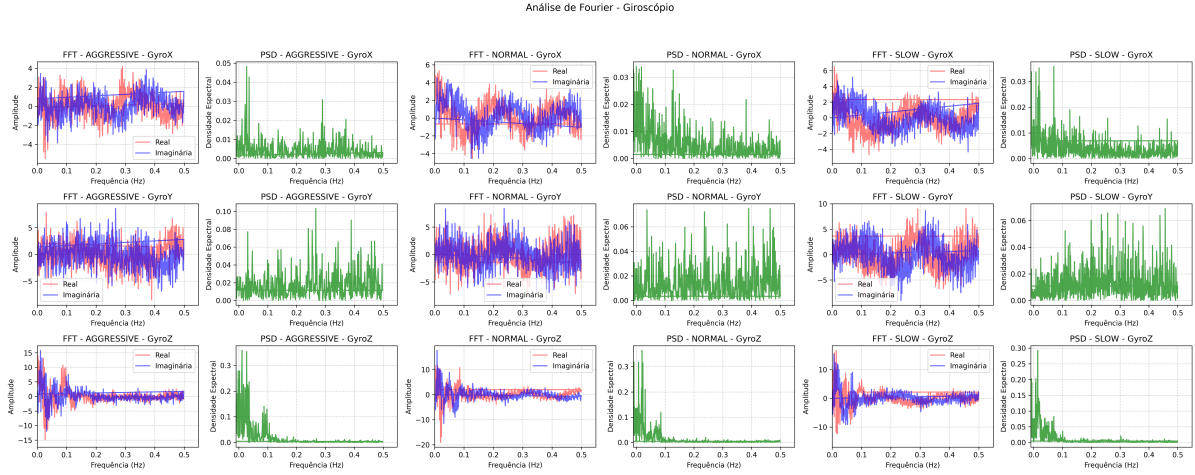


Figura 14: Análise de Fourier (FFT e PSD) separada por classe no Giroscópio.

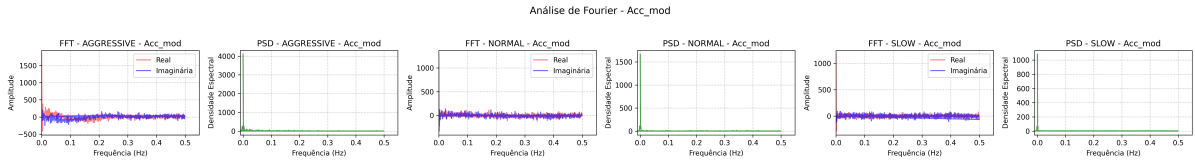


Figura 15: Análise de Fourier (FFT e PSD) separada por classe no Módulo da Aceleração.

comportamento relativamente consistente entre as três classes, com frequência dominante próxima a 0.02–0.04 Hz e potência modesta. Já a variável *GyroY* apresentou aumento gradual na frequência dominante à medida que a direção se torna mais lenta (de 0.5332 Hz para 0.9128 Hz), porém com potência decrescente. Isso pode indicar que condutores mais cautelosos realizam mais microcorreções rotacionais durante a condução, embora com menor intensidade.

De modo geral, a análise espectral mostra que a classe *AGGRESSIVE* se destaca por valores de potência elevados, mesmo quando a frequência dominante é baixa. Isso sugere que condutores agressivos não necessariamente oscilam com frequência, mas mantêm acelerações intensas e constantes. Por outro lado, a classe *SLOW* apresenta menor intensidade geral, porém com sinais mais dispersos no espectro. Tais características indicam que tanto a potência quanto a frequência dominante podem ser utilizadas como atributos relevantes para a diferenciação entre os perfis de direção, sendo especialmente eficazes quando combinadas com o módulo da aceleração (Acc_{mod}).

Esses padrões reforçam o potencial da análise espectral como ferramenta para extração de atributos discriminativos no contexto de classificação de comportamento veicular.

Variável	AGGRESSIVE (Hz)	NORMAL (Hz)	SLOW (Hz)
AccX	0.0000	0.0461	0.0361
AccY	0.0000	0.0000	0.9882
AccZ	0.8722	0.6700	0.1744
Acc_mod	0.0000	0.0000	0.0000
GyroX	0.0541	0.0000	0.1398
GyroY	0.5332	0.7803	0.9128
GyroZ	0.0221	0.0401	0.0283

Tabela 3: Frequências dominantes para cada variável e classe.

Variável	AGGRESSIVE (PSD)	NORMAL (PSD)	SLOW (PSD)
AccX	48.3039	17.9704	6.1630
AccY	85.6697	18.0052	2.6927
AccZ	7.3579	10.5877	3.9010
Acc_mod	4096.6310	1664.3023	1080.4562
GyroX	0.0483	0.0340	0.0359
GyroY	0.1035	0.0750	0.0691
GyroZ	0.3589	0.3646	0.2929

Tabela 4: Potência na frequência dominante (PSD) para cada variável e classe.

4.4 Análise com Espectrogramas

A Figura 16 apresenta os espectrogramas das variáveis dos sensores (*AccX*, *AccY*, *AccZ*, *GyroX*, *GyroY*, *GyroZ*, *Acc_mod*), agrupados por classe de condução (*AGGRESSIVE*, *NORMAL* e *SLOW*). Essa visualização permite observar como a distribuição de energia nas diferentes faixas de frequência evolui ao longo do tempo.

Nas variáveis do acelerômetro, destaca-se a componente vertical (*AccZ*), que apresentou as maiores frequências médias na classe *AGGRESSIVE* (0.36 Hz), refletindo oscilações verticais mais intensas e recorrentes, provavelmente causadas por trepidações, lombadas ou impactos com o solo em alta velocidade. Por outro lado, a classe *SLOW* apresentou uma frequência média significativamente menor (0.17 Hz), compatível com menor atividade vibracional.

Nas componentes *AccX* e *AccY*, as classes *AGGRESSIVE* e *NORMAL* apresentaram energia concentrada em frequências baixas (< 0.06 Hz), indicando acelerações mais sustentadas. A classe *SLOW*, porém, apresentou maior diversidade espectral, com frequência dominante de 0.07 Hz em *AccX* e 0.19 Hz em *AccY*, sugerindo variações rápidas e leves, como microcorreções laterais ou pequenas acelerações.

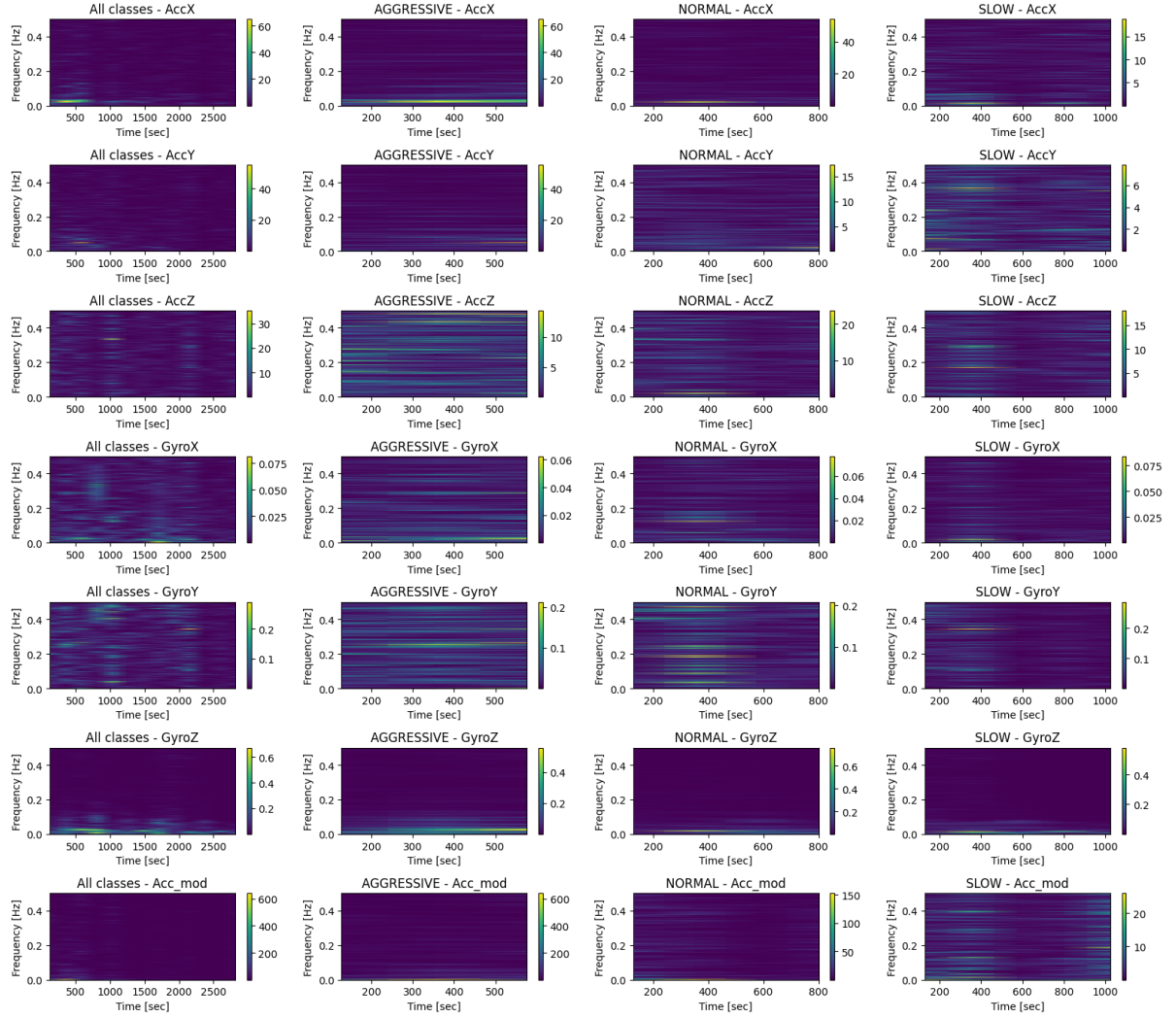


Figura 16: Espectrogramas dos sinais dos sensores (Acc, Gyro e Acc_{mod}) para todas as classes.

As variáveis do giroscópio mostram comportamento complementar. A variável mais expressiva foi *GyroY*, associada à rotação lateral (curvas). A classe *NORMAL* apresentou a frequência média mais alta (0.36 Hz), indicando curvas mais contínuas. Já a classe *AGGRESSIVE* teve picos bem definidos em torno de 0.24 Hz, sugerindo curvas mais abruptas. A classe *SLOW* manteve uma frequência intermediária (0.25 Hz), porém com menor intensidade, associada a microajustes de direção.

Nas demais variáveis do giroscópio (*GyroX* e *GyroZ*), a variação foi limitada, com todas as classes apresentando baixas frequências dominantes (< 0.1 Hz), sem grandes distinções.

Por fim, o espectrograma do módulo da aceleração (Acc_{mod}) reforça os padrões observados nas variáveis separadas. A classe *AGGRESSIVE* teve praticamente toda a sua energia concentrada abaixo de 0.03 Hz, com valores intensos e constantes ao longo do tempo, indicando aceleração total forte e contínua. As classes *NORMAL* e *SLOW* apresentaram maiores variações temporais e espectrais, com destaque para a classe *SLOW*,

que mostrou maior dispersão de energia nas faixas de 0.1 a 0.4 Hz, mesmo com intensidade reduzida.

A Tabela 5 resume as frequências máximas médias obtidas para cada variável e classe.

Variável	AGGRESSIVE	NORMAL	SLOW
AccX	0.03	0.05	0.07
AccY	0.06	0.05	0.19
AccZ	0.36	0.31	0.17
Acc_mod	0.00	0.05	0.04
GyroX	0.04	0.04	0.10
GyroY	0.24	0.36	0.25
GyroZ	0.02	0.02	0.02

Tabela 5: Frequências máximas médias dos espectrogramas (Hz) para cada variável e classe.

4.5 Eliminação de Ruído dos Dados

Como descrito na Seção 3, foram aplicadas 3 técnicas de suavização para reduzir ruídos e melhorar a qualidade dos dados. As Figuras 17, 18 e 19 mostram os resultados obtidos.

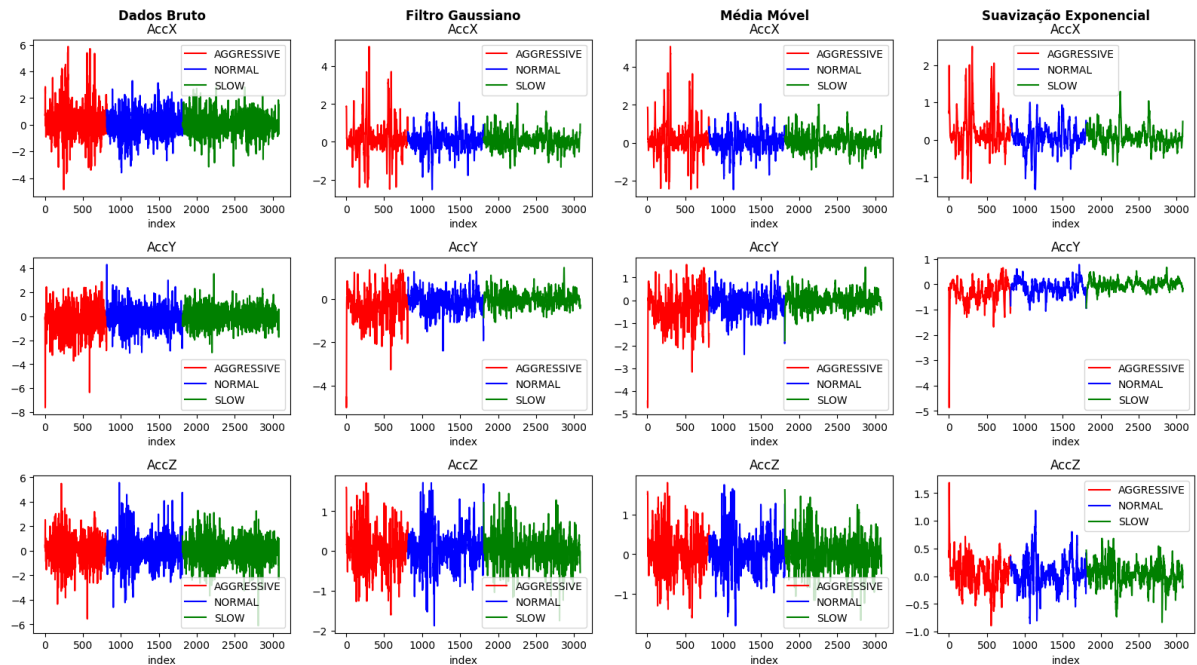


Figura 17: Filtros aplicados nos dados do Acelerômetro

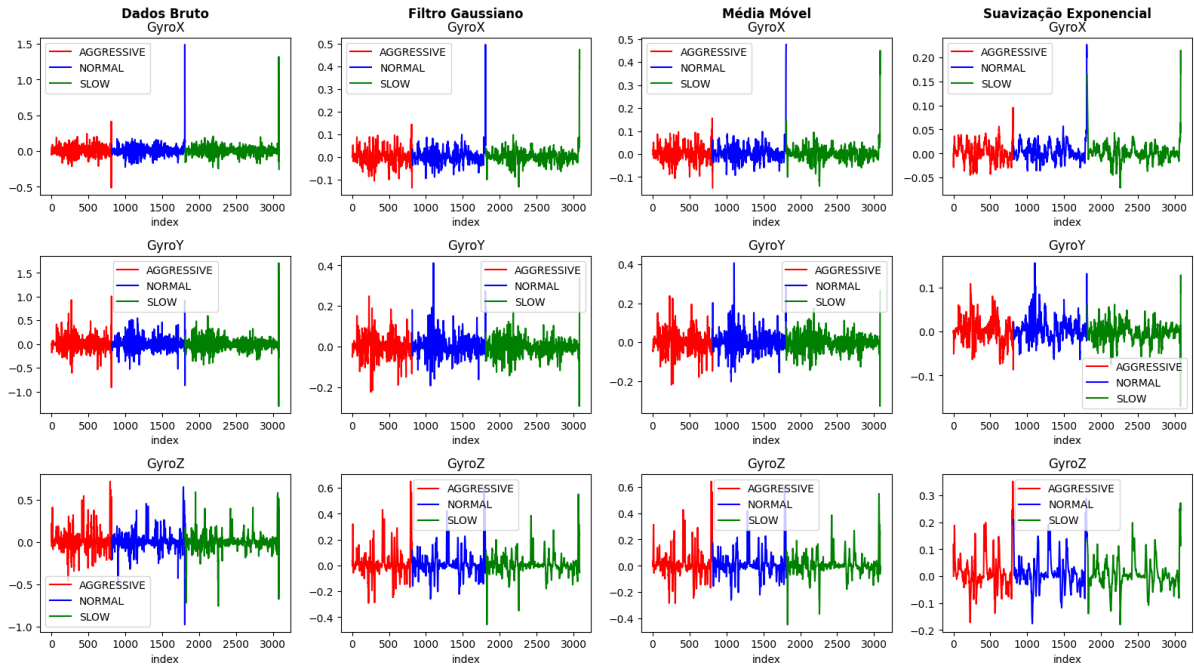


Figura 18: Filtros aplicados nos dados do Giroscópio

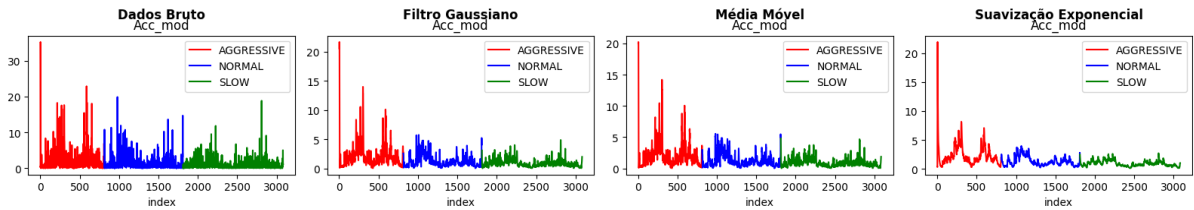


Figura 19: Filtros aplicados nos dados do Módulo da Aceleração

Os gráficos gerados mostraram que o filtro gaussiano apresentou melhor suavização, mantendo as características relevantes do sinal sem distorcer os picos característicos de movimentos agressivos. Dessa forma, ele foi escolhido como técnica de suavização principal que foi utilizada nos dados antes da criação dos modelos de aprendizagem.

4.6 Avaliação dos Modelos de Machine Learning

Nesta seção, apresentamos os resultados obtidos pelos modelos de Machine Learning clássicos aplicados na detecção de comportamentos agressivos de direção. Os modelos foram avaliados com base nas métricas de acurácia, precisão, recall e F1-score, considerando especialmente o desempenho na classe de interesse (classe 1 – direção agressiva).

A Tabela 6 apresenta os resultados quantitativos para os modelos treinados: Regressão Logística (com e sem normalização), SVM, Random Forest, Naive Bayes e XGBoost. Observa-se que, embora a acurácia geral dos modelos seja relativamente próxima (entre 71% e 73%), há variações importantes nas métricas de recall e F1-score, que são mais relevantes para a identificação da classe minoritária (comportamento agressivo).

Modelo	Acurácia (%)	Precisão	Recall	F1-Score
Regressão Logística	73	0.76	0.19	0.31
Reg. Logística (Scaled)	73	0.73	0.20	0.32
SVM	71	0.52	0.55	0.54
Random Forest	73	0.66	0.26	0.37
Naive Bayes	73	0.56	0.53	0.54
XGBoost	71	0.53	0.42	0.47

Tabela 6: Resultados dos Modelos Clássicos de ML (Métricas da Classe 1)

A seguir, apresentamos as matrizes de confusão de cada modelo, que permitem visualizar com mais clareza os acertos e erros em cada classe. As Figuras 20 a 25 mostram a distribuição dos verdadeiros positivos, falsos negativos, verdadeiros negativos e falsos positivos para cada abordagem.

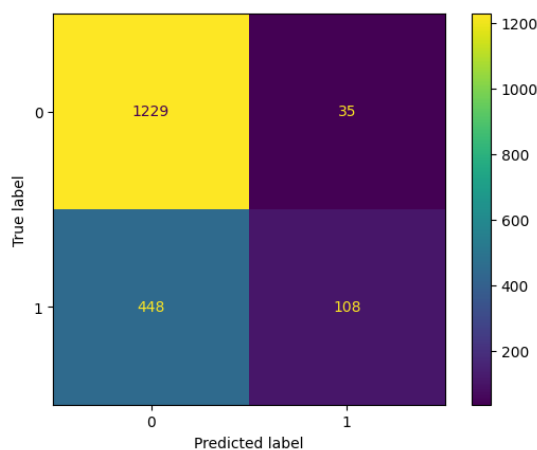


Figura 20: Matriz de Confusão - Regressão Logística

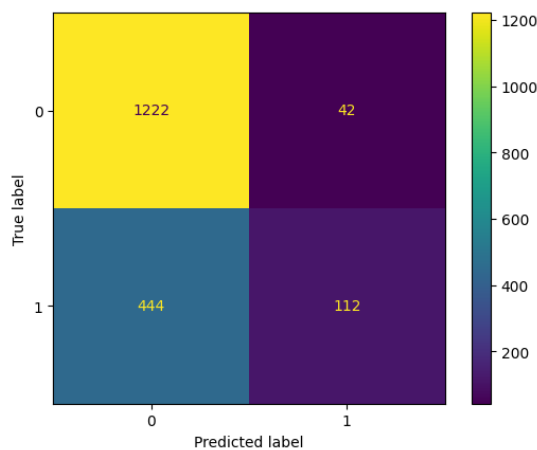


Figura 21: Matriz de Confusão - Regressão Logística (com Normalização)

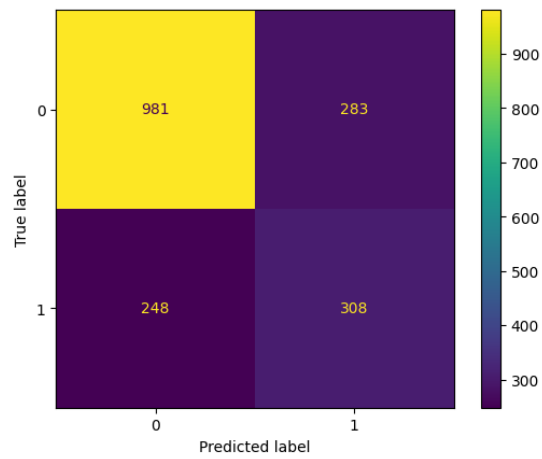


Figura 22: Matriz de Confusão - SVM

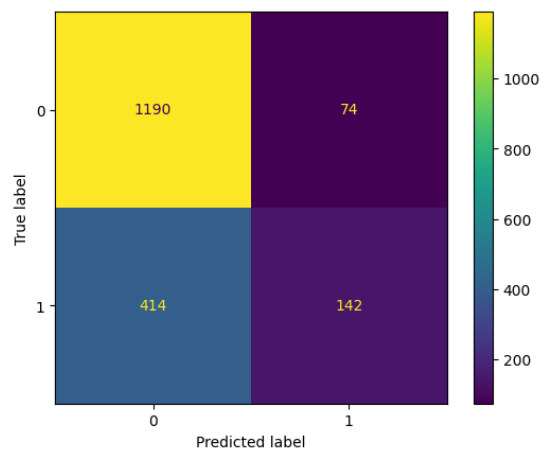


Figura 23: Matriz de Confusão - Random Forest

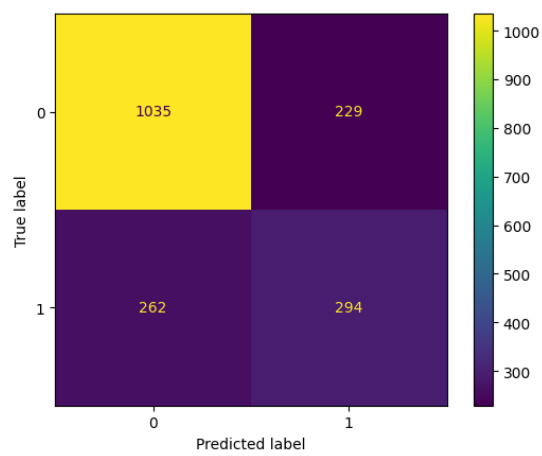


Figura 24: Matriz de Confusão - Naive Bayes

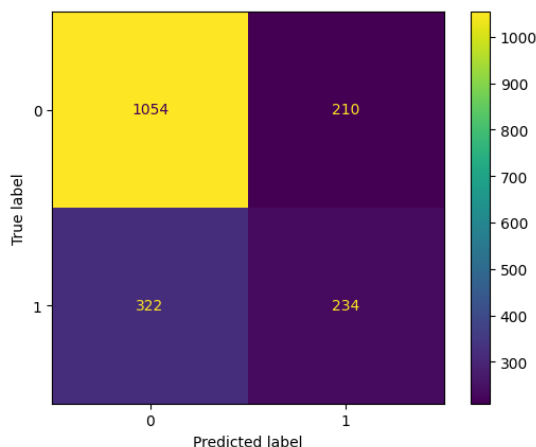


Figura 25: Matriz de Confusão - XGBoost

Observa-se que os modelos de Regressão Logística, mesmo com e sem normalização, obtiveram acurácia de 73%, porém com recall muito baixo (19% e 20%, respectivamente). Isso indica que o modelo teve dificuldades em identificar corretamente os casos de direção agressiva, embora apresentasse alta precisão, ou seja, quando previa como agressiva, normalmente estava correto.

O modelo SVM apresentou um melhor equilíbrio entre precisão (52%) e recall (55%), resultando em um F1-score de 0.54. Isso representa uma melhoria significativa em relação à regressão logística, especialmente na sensibilidade à classe agressiva.

O Random Forest, apesar da acurácia semelhante (73%), também obteve baixo recall (26%), o que compromete seu uso prático em um cenário onde a identificação da direção agressiva é prioridade.

Naive Bayes e XGBoost mostraram desempenhos mais equilibrados, com F1-scores de 0.54 e 0.47, respectivamente. O Naive Bayes, em particular, apresentou o maior valor de acurácia balanceada (67%), sugerindo uma capacidade melhor de tratar o desbalanceamento.

Dentre todos os modelos testados, o SVM e o Naive Bayes demonstraram o melhor desempenho na tarefa de identificar a classe agressiva, apresentando uma boa relação entre sensibilidade (recall) e precisão.

Limitações e Considerações Finais

É importante destacar que, embora os modelos tenham atingido valores satisfatórios de acurácia geral, o principal desafio foi obter uma boa taxa de detecção da classe minoritária. Este fato reforça a necessidade de abordagens que lidem com desbalanceamento, como reamostragem, ajuste de pesos ou uso de métricas específicas durante o treinamento.

Além disso, fatores como a escolha de features, representatividade do dataset e complexidade do comportamento humano ao dirigir influenciam diretamente o desempenho

dos modelos. Nas seções seguintes, são apresentados os resultados com abordagens de aprendizado profundo, buscando superar essas limitações.

4.7 Avaliação dos Modelos de Deep Learning

Os modelos testados incluem: redes neurais densas (MLP), convolucionais (CNN), recorrentes (RNN) e transferência de aprendizado com a MobileNetV2. Todos os modelos foram avaliados com base nas métricas de acurácia, acurácia balanceada, precisão, recall e AUC.

As Tabelas 7, 8 e 9 apresentam os todos resultados obtidos em cada arquitetura. As métricas refletem o desempenho dos modelos especificamente sobre a classe de interesse (direção agressiva).

Modelo MLP	Acurácia (%)	Acur. Bal. (%)	Precisão	Recall	AUC
Modelo 1 (seed 123)	70	68	0.51	0.62	0.68
Modelo 1 (seed 321)	70	68	0.51	0.61	0.68
Modelo 2 (exec. 1)	74	64	0.61	0.40	0.64
Modelo 2 (exec. 2)	73	69	0.56	0.57	0.69
Modelo 3 (exec. 1)	73	68	0.56	0.57	0.68
Modelo 3 (exec. 2)	71	66	0.53	0.54	0.66
Modelo 4	71	68	0.52	0.60	0.68
Modelo 5	65	67	0.45	0.73	0.67
Modelo 6	66	66	0.46	0.68	0.66

Tabela 7: Resultados dos Modelos MLP

Modelo CNN	Acurácia (%)	Acur. Bal. (%)	Precisão	Recall	AUC
Modelo 1 (filter 2)	75	72	0.57	0.65	0.72
Modelo 1 (filter 5)	77	67	0.68	0.43	0.67
Modelo 1 (filter 5, n=32)	75	69	0.59	0.56	0.69
Modelo 2 (filter 3)	30	50	0.30	1.00	0.50
Modelo 2 (filter 2, n=8)	30	50	0.30	1.00	0.50
Modelo 3 (filter 7, n=16)	75	69	0.60	0.53	0.69
Modelo 3 (filter 7, n=32)	68	69	0.48	0.71	0.69
Modelo 4 (Transfer Learning)	64	67	0.45	0.76	0.67

Tabela 8: Resultados dos Modelos CNN

Para melhor visualização, foi criado uma nova tabela com os melhores resultados em cada arquitetura, como visto na Tabela 10.

Modelo RNN	Acurácia (%)	Acur. Bal. (%)	Precisão	Recall	AUC
SimpleRNN (1 camada)	76	69	0.64	0.50	0.69
SimpleRNN (4 camadas)	74	70	0.57	0.59	0.70
GRU (4 camadas)	77	72	0.63	0.59	0.72
LSTM (2 camadas)	77	74	0.60	0.68	0.74
LSTM (3 camadas)	78	72	0.65	0.58	0.72

Tabela 9: Resultados dos Modelos RNN

Modelo	Acurácia (%)	Acur. Bal. (%)	Precisão	Recall	AUC
MLP - Modelo 2	73	69	0.56	0.57	0.69
MLP - Modelo 5	65	67	0.45	0.73	0.67
CNN - Modelo 1	75	72	0.57	0.65	0.72
CNN - Modelo 3	75	69	0.60	0.53	0.69
CNN - T.L.	64	67	0.45	0.76	0.67
RNN - SimpleRNN	74	70	0.57	0.59	0.70
RNN - GRU	77	72	0.63	0.59	0.72
RNN - LSTM	78	72	0.65	0.58	0.72

Tabela 10: Melhores resultados dos modelos de Deep Learning em cada arquitetura

As Figuras 26 até 33 apresentam as matrizes de confusão para os modelos destacados na Tabela 10, revelando o comportamento dos classificadores em termos de acertos e erros por classe.

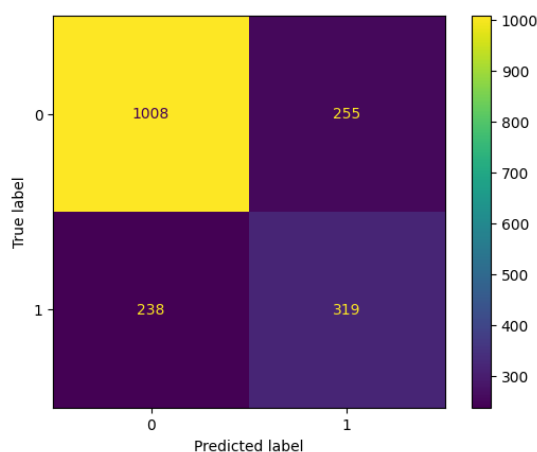


Figura 26: Matriz de Confusão - MLP (Modelo 2)

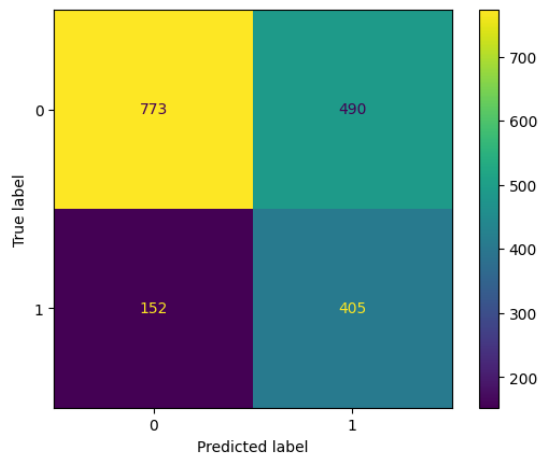


Figura 27: Matriz de Confusão - MLP (Modelo 5)

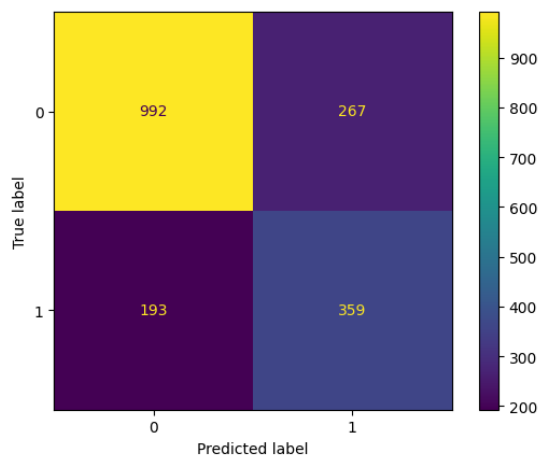


Figura 28: Matriz de Confusão - CNN (Modelo 1)

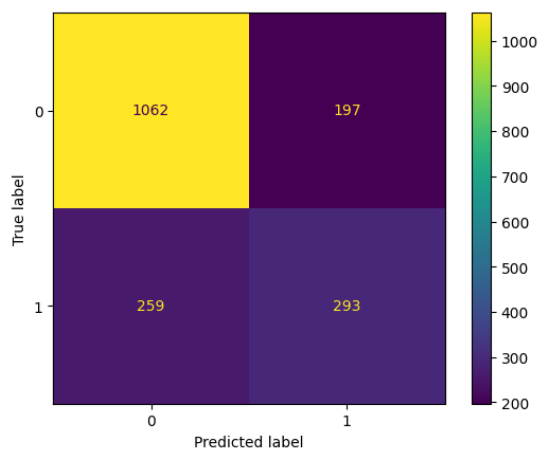


Figura 29: Matriz de Confusão - CNN (Modelo 3)

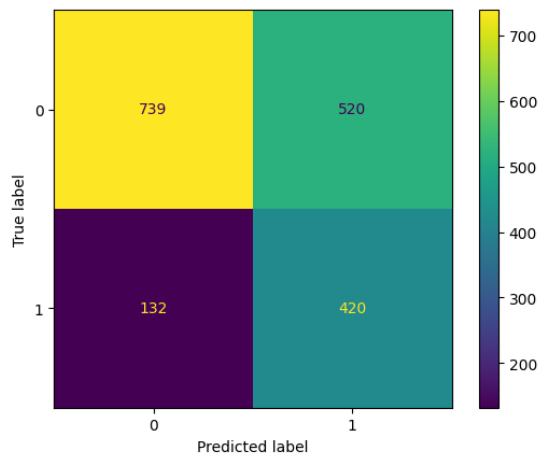


Figura 30: Matriz de Confusão - CNN (Transfer Learning)

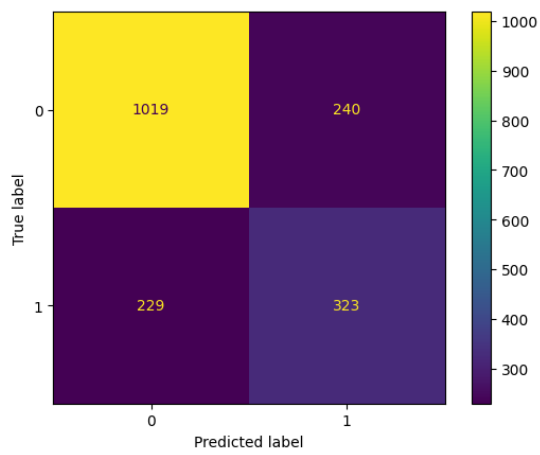


Figura 31: Matriz de Confusão - RNN (Simple RNN)

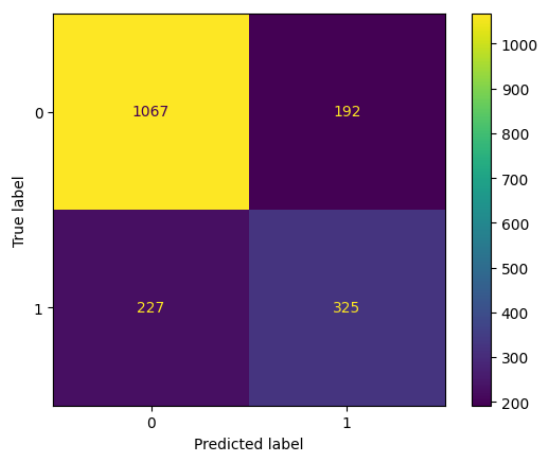


Figura 32: Matriz de Confusão - RNN (GRU)

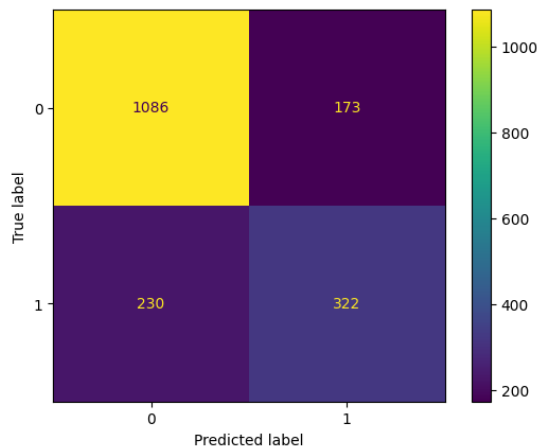


Figura 33: Matriz de Confusão - RNN (LSTM)

Discussão dos Resultados

A arquitetura MLP demonstrou resultados promissores, especialmente no Modelo 2, com F1-score equilibrado e recall razoável. No entanto, o Modelo 5 se destacou pelo maior recall (0.73), ainda que à custa de uma menor precisão (0.45), mostrando tendência a gerar mais falsos positivos.

No caso das CNNs, os melhores desempenhos vieram do Modelo 1 (recall = 0.65) e do Modelo 3 com filtros maiores, demonstrando que a extração de características visuais a partir dos gráficos de recorrência foi útil para a tarefa. O uso de Transfer Learning com MobileNetV2 também mostrou potencial, atingindo o maior recall entre todos os modelos (0.76), apesar de uma menor acurácia geral (64%).

As redes recorrentes (RNNs) superaram os MLPs e CNNs em termos de acurácia e equilíbrio entre precisão e recall. O modelo baseado em LSTM obteve o melhor desempenho geral, com acurácia de 78%, balanced accuracy de 72%, precisão de 0.65 e recall de 0.58. Isso demonstra a capacidade das LSTM de capturar dependências temporais importantes na sequência de sensores.

4.8 Comparação Geral dos Modelos

Os modelos de Machine Learning, como Regressão Logística, SVM, Random Forest, Naive Bayes e XGBoost, apresentaram acurácias próximas entre si (entre 71% e 73%). No entanto, apresentaram dificuldade em alcançar valores satisfatórios de *recall* para a classe de interesse (comportamento agressivo), com destaque para o modelo SVM e o Naive Bayes, que atingiram *recall* de 0.55 e 0.53, respectivamente.

Por outro lado, os modelos de Deep Learning demonstraram, de forma geral, maior capacidade de captura de padrões mais complexos e temporais nos dados. As RNNs (Redes Neurais Recorrentes) obtiveram os melhores desempenhos, com destaque para o modelo LSTM, que alcançou uma acurácia de 78%, *balanced accuracy* de 72%, precisão de

0.65 e *recall* de 0.58. Outro modelo que merece destaque é o GRU, que também apresentou resultados próximos, com *recall* de 0.59 e AUC de 0.72. Além disso, o modelo de Transfer Learning com a MobileNetV2 alcançou o maior *recall* de todos os testes (0.76), ainda que com menor precisão (0.45), evidenciando um comportamento mais sensível para a classe minoritária.

Portanto, pode-se concluir que os modelos baseados em Deep Learning, especialmente as redes recorrentes (LSTM e GRU), foram os mais adequados para a tarefa proposta, superando os modelos clássicos de ML tanto em sensibilidade quanto em equilíbrio geral das métricas. A capacidade dessas arquiteturas de processar sequências temporais mostrou-se essencial para capturar as variações dinâmicas dos sinais dos sensores.

Como conclusão temos que o modelo baseado em LSTM foi identificado como o que apresentou o melhor desempenho geral para a detecção de direção agressiva, considerando o conjunto de métricas analisadas. O modelo GRU também se mostrou altamente competitivo, tornando-se uma alternativa igualmente viável dependendo do foco desejado (sensibilidade, robustez ou desempenho computacional).

5 Conclusões

O presente trabalho teve como objetivo principal o desenvolvimento e comparação de diferentes técnicas de aprendizado de máquina e aprendizado profundo para a detecção de comportamentos agressivos de direção a partir de dados oriundos de sensores. De forma satisfatória, o trabalho cumpriu os objetivos propostos, ao implementar, treinar e avaliar uma variedade de modelos tradicionais (ML) e de Deep Learning (DL), além de realizar uma análise detalhada do desempenho de cada abordagem.

Inicialmente, foram construídos e avaliados modelos clássicos de Machine Learning, como Regressão Logística, SVM, Random Forest, Naive Bayes e XGBoost. Apesar de apresentarem bons resultados em termos de acurácia geral, esses modelos mostraram limitações na identificação da classe minoritária (direção agressiva), especialmente no que diz respeito ao *recall*, métrica fundamental para o problema proposto.

Em seguida, foram desenvolvidos e testados diversos modelos de Deep Learning, incluindo arquiteturas MLP, CNN, RNN (com variantes GRU e LSTM) e até mesmo um modelo baseado em Transfer Learning com MobileNetV2. Os resultados evidenciaram um desempenho superior das arquiteturas de DL em relação aos modelos clássicos, principalmente no tratamento de sequências temporais, onde as RNNs se destacaram. O modelo baseado em LSTM apresentou o melhor desempenho geral, com métricas equilibradas e sensibilidade adequada à detecção de eventos agressivos. O modelo com Transfer Learning obteve o maior *recall*, demonstrando potencial para aplicações mais sensíveis à detecção da classe positiva, mesmo com algumas limitações de precisão.

Para o desenvolvimento deste trabalho, foram abordados os conceitos fundamentais de aprendizado supervisionado, estruturas de dados temporais, engenharia de atributos e normalização. Além disso, foi construída uma arquitetura modular de código que permite a reusabilidade e expansão, abrindo portas para testes com novos algoritmos ou ajustes nas abordagens existentes.

Como possíveis trabalhos futuros, destacam-se: a aplicação de técnicas de reamostragem para balanceamento de dados, como *SMOTE* ou *undersampling*; a investigação de arquiteturas híbridas (como CNN + RNN) que combinem extração espacial e temporal de padrões; a experimentação com métodos de atenção e Transformers, que vêm se mostrando eficazes em séries temporais; o uso de técnicas de explicabilidade (como SHAP ou LIME) para compreender melhor as decisões dos modelos; e a utilização de métodos de *Finetuning* para tentar obter resultados melhores.

Por fim, a estrutura desenvolvida também pode ser aplicada em diferentes contextos de classificação binária com dados de sensores, demonstrando a versatilidade da abordagem proposta. O trabalho contribui, assim, para a evolução de sistemas inteligentes voltados à segurança veicular e análise de comportamentos de direção.

Bibliografia

- Rumelhart, D. E., Hinton, G. E., e Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088):533–536.
- Martin Bland, J. e Altman, D. (1986). Statistical methods for assessing agreement between two methods of clinical measurement. *The Lancet*, 327(8476):307–310. Originally published as Volume 1, Issue 8476.
- Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314.
- John, G. H. e Langley, P. (1995). Estimating continuous distributions in bayesian classifiers. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, pages 338–345. Morgan Kaufmann Publishers Inc.
- Hochreiter, S. e Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.
- Burges, C. J. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167.
- Haykin, S. (1999). *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 2nd edition.
- Cristianini, N. e Shawe-Taylor, J. (2000). *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, Cambridge, UK.
- Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.
- Cramer, J. S. (2002). The origins of logistic regression. *Tinbergen Institute Discussion Papers*.
- Cramer, J. S. (2004). The early origins of the logit model. *Studies in History and Philosophy of Biological and Biomedical Sciences*, 35(4):613–626.
- Zhang, H. (2004). The optimality of naive bayes. *AAAI*, 1:562–567.

- Howard, R. M. (2004). *Principles of Random Signal Analysis and Low Noise Design: The Power Spectral Density and its Applications*. Wiley-IEEE Press.
- Han, J., Kamber, M., e Pei, J. (2006). *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 3 edition.
- Pan, S. J. e Yang, Q. (2009). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359.
- Paleti, R., Eluru, N., e Bhat, C. R. (2010). Examining the influence of aggressive driving behavior on driver injury severity in traffic crashes. *Accident Analysis & Prevention*, 42(6):1839–1854.
- Rao, K. R., Kim, D. N., e Hwang, J.-J. (2010). *Fast Fourier Transform - Algorithms and Applications*. Springer Publishing Company, Incorporated, 1st edition.
- Krizhevsky, A., Sutskever, I., e Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, volume 25, pages 1097–1105.
- Graves, A., Mohamed, A.-r., e Hinton, G. (2013). Speech recognition with deep recurrent neural networks. *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649.
- Spiegel, S., Jain, B., e Albayrak, S. (2014). A recurrence plot-based distance measure. *Translational Recurrences, Springer Proceedings in Mathematics Statistics*, 103:1–15.
- Zeiler, M. D. e Fergus, R. (2014). Visualizing and understanding convolutional networks. In *Computer Vision – ECCV 2014*, pages 818–833, Cham. Springer International Publishing.
- Ioffe, S. e Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456.
- LeCun, Y., Bengio, Y., e Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.
- Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117.
- Goodfellow, I., Bengio, Y., e Courville, A. (2016). *Deep Learning*. MIT Press.
- Boashash, B. (2016). *Time-Frequency Signal Analysis and Processing: A Comprehensive Reference*. Academic Press, 2nd edition.

- Chen, T. e Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 785–794, New York, NY, USA. Association for Computing Machinery.
- Antić, B., Čabarkapa, M., Čubranić Dobrodolac, M., e Čičević, S. (2018). The influence of aggressive driving behavior and impulsiveness on traffic accidents. *International Journal for Traffic and Transport Engineering*, 8(3):372–389.
- Bose, B., Dutta, J., Ghosh, S., Pramanick, P., e Roy, S. (2018). Smartphone based system for real-time aggressive driving detection and marking rash driving-prone areas. In *Proceedings of the Workshop Program of the 19th International Conference on Distributed Computing and Networking*, Workshops ICDCN '18, New York, NY, USA. Association for Computing Machinery.
- Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C., e Liu, C. (2018). A survey on deep transfer learning. In *International Conference on Artificial Neural Networks*, pages 270–279. Springer.
- Peixoto, F. (2020). Mlp-multilayer perceptron-conceito básico.
- Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., e He, Q. (2021). A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76.
- Ghandour, R., Potams, A. J., Boulkaibet, I., Neji, B., e Al Barakeh, Z. (2021). Driver behavior classification system analysis using machine learning methods. *Applied Sciences*, 11(22):10562.
- Stryker, C. (2021). O que é uma rede neural recorrente (rnn).
- Brital, A. (2021). Random forest algorithm explained.
- Haque, K. N. (2023). Convolutional neural network — cnn architecture.
- World Health Organization, W. (2023). Global status report on road safety.
- Alomari, A. H., Al-Mistarehi, B. W., Al-Jammal, A. A., Alhadidi, T. I., e Obeidat, M. S. (2023). Modeling driver behaviour at urban signalised intersections using logistic regression and machine learning. *Promet – Traffic & Transportation*, 35(6):838–854.
- Faleiros, N. (2024). Classificador naive bayes e o teorema de bayes.
- GeeksforGeeks (2025). How-transfer-learning-works.
- Algherbal, E. A. e Ratrout, N. T. (2025). Smart cities and intelligent transportation systems: Integration of autonomous vehicles and their impact on congestion and safety. *Transportation Research Procedia*, 84:504–511.