UNIVERSIDADE FEDERAL DO AMAZONAS
FACULDADE DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Lucas de Góes Muniz de Castro

IDENTIFICAÇÃO DE INCÊNDIOS FLORESTAIS UTILIZANDO SEGMENTAÇÃO DE IMAGENS E APRENDIZADO DE MÁQUINA

Manaus – Amazonas
2023

UNIVERSIDADE FEDERAL DO AMAZONAS
FACULDADE DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Lucas de Góes Muniz de Castro

IDENTIFICAÇÃO DE INCÊNDIOS FLORESTAIS UTILIZANDO SEGMENTAÇÃO DE IMAGENS E APRENDIZADO DE MÁQUINA

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal do Amazonas, como requisito parcial para a obtenção do título de Mestre em Engenharia Elétrica na área de concentração de Controle e Automação.

Orientador: Prof. D.Sc. Celso Barbosa Carvalho

Manaus – Amazonas

2023

Ficha Catalográfica

Ficha catalográfica elaborada automaticamente de acordo com os dados fornecidos pelo(a) autor(a).

Castro, Lucas de Goes Muniz de
C355i Identificação de incêndios florestais utilizando segmentação de imagens e aprendizado de máquina / Lucas de Goes Muniz de Castro . 2023
71 f.: il. color; 31 cm.

Orientador: Celso Barbosa Carvalho
Dissertação (Mestrado em Engenharia Elétrica) - Universidade Federal do Amazonas.

1. Identificação de Incêndios Florestais. 2. Segmentação de imagens. 3. Aprendizado de máquina. 4. Análise de quartis. I. Carvalho, Celso Barbosa. II. Universidade Federal do Amazonas III. Título

LUCAS DE GÓES MUNIZ DE CASTRO

IDENTIFICAÇÃO DE INCÊNDIOS FLORESTAIS UTILIZANDO SEGMENTAÇÃO DE IMAGENS E APRENDIZADO DE MÁQUINA

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal do Amazonas, como requisito parcial para obtenção do título de Mestre em Engenharia Elétrica na área de concentração Controle e Automação de Sistemas.

Aprovada em 18 de agosto de 2023.

BANCA EXAMINADORA

Prof. Dr. Celso Barbosa Carvalho
Presidente
Universidade Federal do Amazonas

Profª. Dra. Marly Guimarães Fernandes Costa, Membro
Universidade Federal do Amazonas

Prof. Dr. Francisco de Assis Pereira Januário, Membro
Universidade Federal do Amazonas

“Um único sonho é mais poderoso que mil realidades.”

(Nathaniel Hawthorne)

Agradecimentos

Agradeço imensamente à minha querida mãe, que mesmo diante dos desafios e limitações deste período difícil ao qual estamos passando, esteve sempre ao meu lado, compartilhando a busca incansável por esse sonho. Sua presença e apoio incondicional foram fundamentais para enfrentar cada obstáculo com coragem e determinação.

Expresso minha gratidão ao Professor Celso, por ter confiado em meu trabalho e acreditado em minha ideia. Sua orientação cuidadosa e paciência dedicada foram essenciais para o desenvolvimento desta jornada.

Ao meu namorado, Hugo, agradeço o apoio inabalável nos momentos difíceis e conturbados.

Ao Instituto de Pesquisas Eldorado, meu sincero agradecimento pelo apoio prestado ao longo dessa trajetória.

À minha querida amiga Marenice, agradeço por todo o companheirismo e força no meio acadêmico. Sua presença e incentivo foram inspiradores, tornando essa jornada mais leve e enriquecedora.

Aos meus amigos, expresso minha gratidão por todo o apoio recebido e palavras de encorajamento e momentos compartilhado.

Resumo da Dissertação apresentada à UFAM como parte dos requisitos necessários para a obtenção do grau de Mestre em Engenharia Elétrica

IDENTIFICAÇÃO DE INCÊNDIOS FLORESTAIS UTILIZANDO SEGMENTAÇÃO DE IMAGENS E APRENDIZADO DE MÁQUINA

Lucas de Góes Muniz de Castro

Orientador: Celso Barbosa Carvalho

Programa: Pós-Graduação em Engenharia Elétrica

Este trabalho propõe o uso de diferentes técnicas de pré-processamento de dados e aprendizado profundo para análise de imagens e detecção de incêndios florestais. As imagens utilizadas para treinamento têm origem em dois diferentes bancos de dados com variação de horário, estação climática e posicionamento. Para o treinamento, optou-se por empregar algoritmos de aprendizagem supervisionada e classificadores probabilísticos, totalizando três origens de treinamento com variações de parâmetros e diferentes técnicas de pré-processamento complementares, como *color perception* e quartis.

A principal métrica de avaliação se refere a acurácia e ao índice de verdadeiros-positivos e falsos-negativos, essenciais para essa aplicação, por se tratar de um sistema de identificação e alerta. Também se considera valores de tempo de processamento e treinamento. Os resultados obtidos foram superiores ao estado-da-arte para identificação de incêndios florestais, com acurácias superiores a 99,6% utilizando a técnica *Random Forest.*

Palavras-chave: Identificação de Incêndios Florestais, Segmentação de imagens, Aprendizado de máquina, Análise de quartis.

Abstract of the Dissertation presented to UFAM as part of the necessary requirements for the degree of Master in Electrical Engineering.

IDENTIFYING FOREST FIRES USING IMAGE SEGMENTATION AND MACHINE LEARNING

Lucas de Góes Muniz de Castro

Master's Advisor: Celso Barbosa Carvalho

Program: Pós-Graduação em Engenharia Elétrica

This work proposes the use of different data preprocessing and deep learning techniques for image analysis and forest fire detection. The images used for training originate from two different databases with variation in time, weather season and positioning. For training, we chose to employ supervised learning algorithms and probabilistic classifiers, totaling three training sources with parameter variations and different complementary pre-processing techniques, such as *color perception* and *quartiles.*

The main evaluation metric refers to accuracy and the rate of true positives and false negatives, essential for this application, as it is an identification and alert system. Processing and training time values are also considered. The results obtained were superior to the state-of-the-art for identifying forest fires, with accuracies greater than 99.6% using the *Random Forest* technique.

Keywords: Forest Fire Identification, Image segmentation, Machine learning, Quartile analysis

Lista de Tabelas

Lista de Figuras

Sumário

1 Introdução

Os últimos quinze anos foram decisivos para a fundação de uma nova era global na tecnologia, sendo especialmente notório o impacto de um conjunto diversificado de forças e tendências associadas à aceleração das descobertas científicas e tecnológicas no campo da informação. A expansão da era da informação promove o ímpeto da *internet das coisas* (IoT), que envolve um ambiente permeado por grandes quantidades de dispositivos inteligentes capazes de detectar, capturar, computar e operar dados do mundo real (Zaslavsky, Perera, & Georgakopoulos, 2013). Todos os dias, esses dispositivos geram fluxos contínuos de dados em tempo real, beneficiando inúmeras aplicações em problemas recorrentes e ainda sem solução, como os incêndios florestais.

Considerado o principal problema ambiental em 2020 (Brandão & Pereira, 2020), os incêndios florestais causaram o prejuízo de mais de US$ 20 bilhões na região oeste dos Estados Unidos e afetaram diretamente mais de 47 mil pessoas no Brasil, devido a poluição causada (Reuters, 2020). Segundo (Giglio et al. 2018), os humanos são responsáveis por iniciar mais de 90% dos incêndios florestais, que podem resultar em impactos significativos para os humanos, diretamente por meio da perda de vidas e destruição de comunidades indígenas ou indiretamente por meio de problemas de saúde causados pela exposição à fumaça.

Tornando-se um problema global, inúmeras empresas e diferentes áreas de pesquisa dedicam esforços para indicar maneiras de conter danos causados pelas queimadas e, na engenharia e computação, as pesquisas para previsão, classificação e identificação de incêndios receberam ampla atenção de cientistas que compreenderam que tais etapas são cruciais na gestão de incêndios florestais, como para respostas a emergência, planejamento do uso da terra e adaptação climática (Giglio et al. 2018).

Um dos principais métodos aplicados se referem a classificação de

imagens para identificação de fumaça ou fogo, o processo é realizado utilizando técnicas de *machine learning* (ML) ou *deep learning* (DL) e tem como um dos principais desafios a classificação da vasta gama de escalas do fogo - desde processos de combustão que ocorrem em uma escala de centímetros, por um período de segundos, até a propagação e crescimento do fogo, que ocorre em uma escala de quilômetros em um período de horas a dias. Além disso, o fato tempo de classificação também desponta como uma barreira para essa aplicação, já que interfere no principal objetivo a ser alcançado (Zaslavsky, Perera, & Georgakopoulos, 2013).

Embora tais complexidades apresentem obstáculos na modelagem de incêndios florestais, a ciência obteve avanços significativos entre 2018 e 2021, sobretudo com a aplicação de técnicas de ML, que podem ser ferramentas valiosas para a detecção de incêndios florestais. Os trabalhos de Chen e Hopkins (2022), Zhang e Qu (2022), Khryashchev e Larionov (2020) e Jadon e Varshney (2020), apresentam metodologias para detecção de incêndios florestais usando aprendizado de máquina, com resultados variando entre 83% e 97,36% de acurácia.

No entanto, essas técnicas também têm suas limitações, incluindo o tempo necessário para o processamento, a complexidade do processo e uma extensa base de dados. Por esse motivo, foi decidido explorar uma abordagem para a detecção de incêndios florestais em imagens, baseado nos estudos propostos por Premal e Vinsley (2014) e Bakri e Adnan (2018), onde a classificação é realizada através da análise pixel-a-pixel, por meio da segmentação de imagens e utilizando valores das componentes RGB das imagens e o cálculo de quartis destas componentes, permitindo o uso de classificadores mais simples e rápidos.

Diante do exposto, este trabalho tem por objetivo contribuir para a melhoria dos valores de acurácia, sensibilidade, especificidade e velocidade de detecção de incêndios florestais em imagens, utilizando técnicas de *machine learning* (ML) aliadas a segmentação de imagens.

1.1 Objetivo geral

O objetivo deste trabalho é definir um método de detecção de incêndios florestais que contribui para o estado-da-arte em termos de complexidade temporal e desempenho.

1.2 Objetivos específicos

- Identificar um algoritmo de ML adequado a classificação de imagens de incêndio.
- Associar técnicas de segmentação de imagens com máquinas de aprendizado para detecção de incêndios florestais;
- Propor a utilização do espaço de cores RGB e dos valores dos quartis das suas componentes como variáveis de entrada em algoritmos de aprendizado de máquina, visando aprimorar a detecção de incêndios florestais.
- Comparar o método proposto a outras técnicas de classificação de incêndios florestais em termos de complexidade temporal e de desempenho.

2 Fundamentação Teórica

Nesta seção, são apresentados conceitos relacionados a algoritmos, técnicas e análise utilizados no desenvolvimento deste trabalho, sendo eles:

i) Segmentação de imagens;
ii) Análise de quartis;
iii) Algoritmo k-Nearest Neighbors (kNN);
iv) Algoritmo Random Forest;
v) Algoritmo Naive Bayes;

2.1 Pré-processamento de imagens

O pré-processamento de imagens é um conjunto de técnicas aplicadas em uma imagem com o objetivo de melhorar a qualidade da imagem ou torná-la mais adequada para análise. Essas técnicas podem incluir a correção de distorções, ajuste de brilho e realce de cor, filtragem para remoção de ruído, remoção de background, entre outras. Esta etapa é importante em diversas aplicações de visão computacional, pois pode ajudar a melhorar a eficiência e precisão de tais algoritmos (Chaki, 2018).

Neste trabalho, foram utilizadas técnicas de pré-processamento de imagens para melhorar a eficiência para identificação de incêndios florestais.

2.1.1 Segmentação de Imagens baseada em Graph Cut: A separação de primeiro plano de uma imagem é uma técnica que pode ser usada para destacar os elementos de um incêndio florestal apresentado em uma, tais como eventos de fumaça e fogo. Isso pode ser útil para o monitoramento e detecção de incêndios florestais, pois define o incêndio ou fumaça como um objeto e possibilita sua separação do restante da imagem.

Existem diferentes métodos para extração de um objeto em primeiro plano (*foreground*) de uma imagem digital, dentre estes, é

possível mencionar o método de segmentação de imagens para extração de objetos através do modelo *Graph Cut* [11].

Proposto por (Boykov; Jolly, 2001), o modelo baseia-se na divisão de um grafo em dois subgrafos. A considerar que um grafo não-orientado G é composto por um conjunto de vértices V e arestas A, temos que $G = (V, A)$, onde A conectam os vértices V e para cada aresta $\alpha \epsilon A$, há um peso positivo p_α associado as arestas. Também existem terminais, que consistem em dois ou mais nós especiais e delimitam como o grafo deve ser dividido. Desta forma, um *graph cut* pode ser definido como um subconjunto C de vértices separados a partir dos terminais, onde $C \subset A$ e $G(C) = (V, A \backslash C)$.

O custo do *graph cut* pode ser definido como:

$$|C| = \sum_{\alpha \in C} p_\alpha \quad (1)$$

Tendo nas arestas A a representação da relação entre um pixel e seus vizinhos e cada vértice V representado por um pixel de uma imagem, é possível avaliar os pesos de cada aresta que são definidos pela probabilidade de um pixel estar em primeiro ou segundo plano. Quando há grande diferença na cor de um pixel, é atribuído um peso baixo àquela aresta. O algoritmo é então aplicado para segmentar o grafo, dividindo-o em dois subgrupos: o vértice de origem e o de destino, a partir da função de custo (*Rother, Kolmogorov e Blake, 2004*).

A Figura 1 ilustra a segmentação de uma imagem de 3x3 pixels. Nos grafos, o custo é mostrado pela espessura da aresta, portanto, as arestas com menores pesos (mais finas) estão aptas a fazerem parte da divisão do *graph cut* a fim de separar/segmentar terminais de objeto (*foreground*) dos terminais de fundo (*background*) [13].

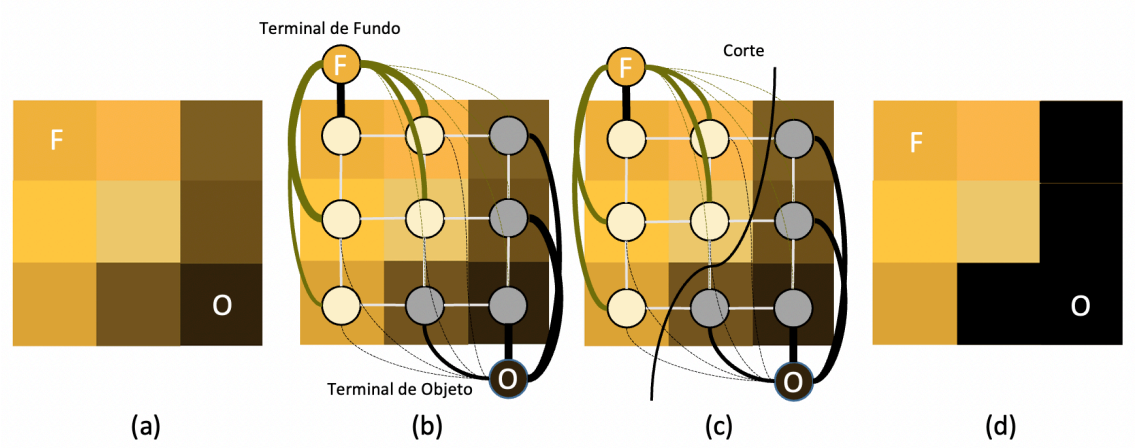

Figura 1 – Segmentação de uma imagem de 3x3 pixels. Em (a) o pixel rotulado como "O" possui um alto contraste em relação a "F". Em (b) são definidos os pesos das arestas A e seus vizinhos. Em (c) é definido o formato de corte a partir dos pesos atribuídos na etapa anterior. Em (d) é concluída a segmentação da imagem. Fonte: Adaptada de Gaspari (2015)

2.1.2 Canais RGB: Modelos matemáticos utilizados para padronizar e representar numericamente as cores são denominados de espaços de cores, definidos por um sequências ordenadas de três ou quatro números para representar as cores de maneira numérica. Esses números podem ser combinados em diferentes ordens e valores para modelar todo o espectro de cor possível para cada espaço de cor (Kakumanu, 2007).

Dentre as formas de representar as cores em diferentes espaços, destaca-se o sistema RGB, onde as cores são construídas a partir da mistura das três cores principais vermelho (R), verde (G) e azul (B). Nesse sistema, cada pixel de uma imagem digital é composto por três componentes de cor, sendo um para cada cor primária. Segundo (Chaves-González, 2010), a combinação dessas cores primárias em diferentes proporções permite a criação de uma ampla gama de cores, incluindo todas as cores visíveis pelo olho humano. A intensidade de cada cor primária é representada por um número inteiro entre 0 e 255, onde 0 representa a ausência da cor e 255 representa a cor mais forte possível. Portanto, ao combinar diferentes valores de intensidade para cada cor

primária, é possível criar uma grande variedade de cores diferentes, como ilustra a Figura 2.

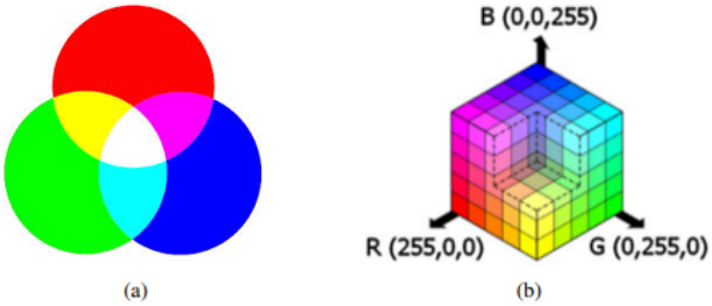

Figura 2 - Cores do sistema RGB (a) e representação do sistema tridimensional RGB (b). Fonte: Adaptado de Feitosa (2015).

Desta maneira, é possível extrair padrões e identificar tons de cores repetidas, a partir da separação desses canais e a comparação de seus valores. Na linguagem Python, a função *split()*, da biblioteca CV2 realiza a separação desses canais em três matrizes diferentes [19]. A figura 3 demonstra a representação gráfica de uma mesma imagem para cada componente de cor RGB.

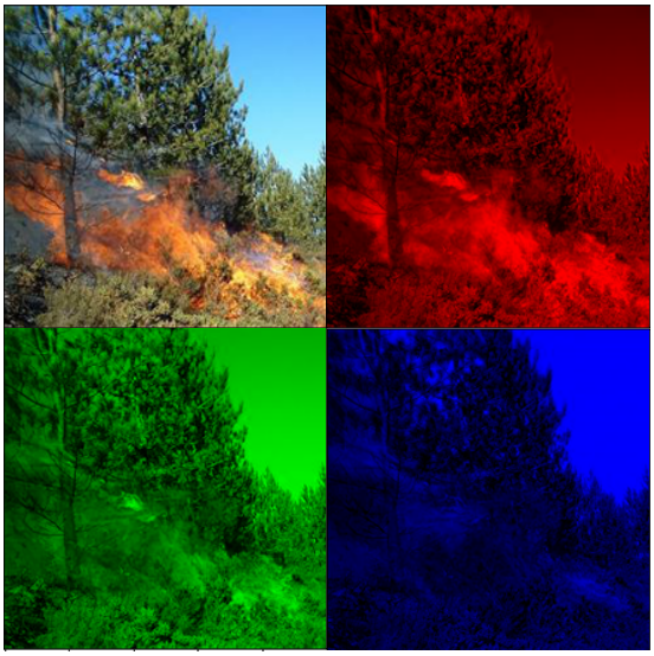

Figura 3 - Representação gráfica da separação dos canais RGB para uma mesma imagem. Fonte: própria.

2.1.3 Análise de Quartis: A análise de quartis é utilizada para dividir um conjunto de dados em quatro subconjuntos com 25% dos elementos do conjunto de dados; O que permite avaliar a tendência central e a dispersão de um conjunto de dados.

De acordo com (Joarder, Firozzaman 2001), o primeiro quartil (Q1) refere-se ao valor das 25% menores observações, o segundo quartil (Q2) corresponde as 50% menores observações e o terceiro quartil (Q3) é o valor que delimita as 25% maiores observações.

O cálculo dos valores dos quartis inferior e superior de uma amostra baseia-se em calcular a mediana do conjunto de dados x_m e depois calcular a mediana da metade inferior x_l e da metade superior x_u dos dados, para um conjunto de n-observações $X = \{x_1 \leq \cdots . \leq x_n\}$. Este cálculo é conhecido como Método de Tukey (Langford, 2006) e é utilizado como definição para a função *quartile ()* da biblioteca *NumPy* [20].

Onde:

$$l = \frac{n + 3}{4} \tag{2}$$

$$m = \frac{n + 1}{2} \tag{3}$$

$$u = \frac{3n + 1}{4} \tag{4}$$

Em caso de valores não inteiros para l, m e u, o método de interpolação linear – uma variação mais simples da interpolação polinomial - é aplicado para instanciar um conjunto de dados visando a construção de um novo ponto de dados baseado nos pontos já conhecidos, demonstrado por:

$$x_i + (x_{i+1} - x_i) \times d \quad (5)$$

Sendo x_i o valor da observação posicionada no índice i, sendo i a parte inteira e d a parte decimal do índice calculado.

Ao aplicar o cálculo dos quartis a um conjunto de dados contendo as componentes R, G ou B de uma imagem, é possível identificar padrões existentes nestes canais, o nível de intensidade de cor e sua variação na imagem, sendo útil para a diferenciação de imagens contendo incêndios de imagens sem incêndios [17].

Além disso, a aplicação da análise de quartis nas componentes RGB de uma imagem também pode ser útil para identificar imagens de outros tipos, como imagens de paisagem ou imagens de vegetação, para detectar padrões nas cores presentes na imagem.

2.2 Classificadores Probabilísticos

Os classificadores probabilísticos podem ser usados para prever a classe c_i de uma nova observação d_i com base em uma série de características ou atributos. Eles são chamados de probabilísticos porque estimam a probabilidade de uma nova observação pertencer a cada uma das classes possíveis e classificam a nova observação com base na classe com a maior probabilidade, (Zhang, 2004).

Um exemplo de classificador probabilístico é o Naive Bayes (NB).

2.2.1 Naive Bayes: Para (Marques, 1999), o NB é um classificador probabilístico baseado na aplicação do teorema de Bayes com fortes suposições de independência. O NB objetiva classificar classes a partir de características dependentes ou independentes.

O termo *naive*, que significa ingênuo traduzindo para o português, se refere a suposição de que os valores dos preditores são independentes entre si, sendo ingênua pois em muitos experimentos do mundo real essa

suposição não ocorre. A vantagem de se assumir isso impacta na redução da complexidade dos cálculos de probabilidade.

De forma abstrata, o modelo de probabilidade do classificador NB é um modelo condicional sobre uma variável de classe dependente com várias características. E o problema que se coloca é: dado uma amostra de um vetor x = $[x_1, \ldots, w_n]$ (características), a classe w deve ser identificada (Ferreira et. al, 2020).

Para eventos discretos, a regra de *Naive-Bayes* é dada por:

Se $P(w_i \mid x) > P(w_j \mid x)$, então, x pertence a classe w_i e a probabilidade condicional em casos de mais de uma característica é dada por:

$$P(w \mid x) = P(w \mid x_1 .., x_n) \tag{6}$$

E a probabilidade condicional é calculada por:

$$P(w \mid x_1, .., x_n) = \frac{P(w)P(x_1,..,x_n|w)}{P(x_1,..,x_n)} \tag{7}$$

Usando a condição de *naive*, de independência, têm-se:

$$P(x_i|w, x_1, .., x_n) = P(x_i \mid w) \tag{8}$$

Logo, temos:

$$P(w \mid x_1, .., x_n) = \frac{P(w)\prod_{j=1}^{n} P(x_j|w)}{P(x_1,..,x_n)} \tag{9}$$

Desde que $P(x_1, .., x_n)$ seja uma constante, podemos usar a seguinte regra de classificação:

$$P(w \mid x_1, .., x_n) \propto P(w)\prod_{j=1}^{n} P(x_j|w) \tag{10}$$

Logo:

$$\widehat{w} = argmax\, P(w) \prod_{j=1}^{n} P(x_j|w) \quad (11)$$

Sendo $\widehat{w}$ a classe estimada. Para eventos contínuos, a regra é dada por:

Se $P(x \mid w_i)P(w_i) > P(w_j \mid x)$, então, x pertence a classe w_i .

Nestes casos, assume-se o cálculo da probabilidade usando o método gaussiano, como é mostrado na equação abaixo:

$$P(w \mid x_i) = \frac{1}{\sqrt{2\pi\sigma_w^2}} exp(-\frac{(x_i-\mu_w)^2}{2\sigma_w^2}) \quad (12)$$

Onde calcula-se μ e σpara cada classe.

Supondo que existem duas classes, $w_1 = \{x_1, x_2, \ldots, x_p\}$ e $w_2 = \{x'_1, x'_2, \ldots, x'_q\}$, cada uma com p e q padrões respectivamente, calcula-se a média e o desvio padrão de cada classe conforme é mostrado nas equações 12, 14 e 15:

$$\mu_{w1} = \frac{x_1 + x_2 + \ldots + x_p}{p} \quad (13)$$

$$\sigma_{w1} = \frac{1}{p}\sum_{i=1}^{p} (x_i - p)^2 \quad (14)$$

$$\mu_{w2} = \frac{x'_1 + x'_2 + \ldots + x'_q}{q} \quad (15)$$

$$\sigma_{w2} = \frac{1}{q}\sum_{i=1}^{q} (x_i - \mu_{w2})^2 \quad (16)$$

Neste trabalho optou-se por implementar o algoritmo na linguagem Python.

2.3 Aprendizagem Supervisionada

De acordo com (Bishop, C. 2006), os classificadores de aprendizagem supervisionada são algoritmos de aprendizado de máquina treinados para classificar entradas em diferentes categorias ou classes. Isso é feito utilizando um conjunto de dados rotulados, onde cada exemplo no conjunto de dados é composto por um conjunto de características e uma saída desejada (ou "rótulo"). O objetivo do classificador é aprender a associar as características de entrada com as saídas desejadas e, assim, ser capaz de classificar novas entradas não vistas durante a fase de treinamento.

Existem vários tipos de classificadores, como Random Forest (RF) e k-Nearest Neighbors (KNN).

2.3.1 k-Nearest Neighbors (KNN): O k-Nearest Neighbors (KNN) é um algoritmo de classificação baseado em instância, ou seja, ele classifica novos exemplos de acordo com o que é semelhante a eles em relação às suas características.

Sendo proposto em 1967 por Cover e Hart e tido como um dos mais simples e eficazes algoritmos de classificação, além de ser não-paramétrico - não necessitando de dados que seguem uma distribuição específica como gaussiana ou exponencial - o kNN insere-se no âmbito da aprendizagem supervisionada, onde ele guarda um conjunto de informações previamente rotuladas, podendo se tornar custoso computacionalmente se o tamanho do conjunto de dados for significativamente grande (Bishop, 2006).

Para aplicações com modelo de representação de dados mais adequado à classificação, como a identificação de incêndios florestais em imagens, reconhecimento facial em imagens fixas ou identificação de padrões genéticos, o algoritmo mostrou-se eficiente e possui resultados muito positivos, segundo Lantz (2015).

De maneira geral, o conjunto de dados já categorizados é chamado de conjunto de aprendizado, onde as linhas são vetores n-dimensionais

que representam instâncias de treinamento com n características. Para prever a classe de uma nova amostra (referida como exemplo de teste), o algoritmo KNN segue os seguintes passos:

1. Calcula a relação de similaridade entre a instância de teste e treinamento.
2. Seleciona as k instâncias de treinamento mais similares (próximas) à instância de teste.
3. Verifica a classe predominante, ou seja, a classe que aparece com maior frequência dentre as k instancias selecionadas.
4. Atribui a instância de teste à classe predominante.

Como exemplificado em (Ferreira e Souza, 2020), a similaridade é calculada mediante uma medida de dissimilaridade que é aplicada entre dois vetores. Várias medidas podem ser utilizadas, tais como as distâncias Euclidiana, Manhattan e Minkowski, sendo geralmente adotada na literatura, a distância Euclidiana definida pela Equação 17:

$$dist(u, v) = \sqrt{\sum_{i=1}^{n} (u_i - v_i)^2} \qquad (17),$$

em que u e v são vetores n-dimensionais; e u_i e v_i representam os valores do atributo i dos respectivos vetores.

O parâmetro ideal k (número de vizinhos mais próximos) é determinado através de experimentos, para cada conjunto de dados, onde o valor de maior precisão é escolhido para definir o classificador. É possível testar diferentes valores de k, variando de um até a raiz quadrada do número de instâncias de treinamento. É recomendado que o valor de k seja ímpar para evitar empates entre duas classes (Lantz, 2015).

2.3.2 Random Forest: A árvore de decisão é um tipo de algoritmo de aprendizado supervisionado que pode ser usado em problemas de regressão e classificação. Este algoritmo para variáveis de entrada e saída categóricas e contínuas, como sugere (Breiman, 2001). Nesta técnica, a população é dividida em dois ou mais grupos homogêneos, baseando-se nos atributos ou variáveis independentes mais significativas.

Conforme ilustrado na Figura 4, cada árvore individual é um modelo bastante simples que possui ramificações, nós e folhas. Os nós contêm os atributos dos quais a função objetivo é dependente. Então os valores da função objetivo vão para as folhas através dos ramos. No processo de classificação de um novo caso, é necessário descer a árvore através de seus ramos até uma folha, passando por todos os valores dos atributos de acordo com o princípio lógico "*IF THEN*" (Breiman, 2001).

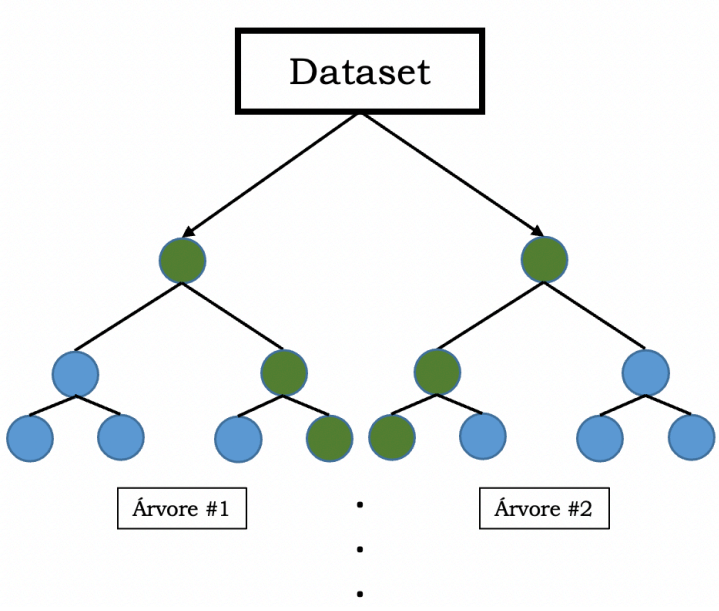

Figura 4 - Modelo de árvore de decisão. Fonte: Adaptado de Breimann (2001).

O algoritmo *Random Forest* (RF) consiste em um *bagging* de Árvores de Decisão (também conhecidas como árvores de classificação ou árvores de regressão), sendo utilizada em estatística, mineração de dados e aprendizado de máquina. Dependendo dos parâmetros estabelecidos, a

variável objetiva receberá um valor específico ou uma classe (a variável objetiva cairá em uma folha específica). O propósito de construir uma árvore de decisão é criar um modelo que prevê o valor da variável objetiva, dependendo de diversas variáveis de entrada (Amit, 1997).

Seu resultado é dado quando todas as árvores treinadas são combinadas em uma composição com uma votação simples, usando o erro médio de todas as amostras a fim de obter um resultado final.

Para demonstrar a capacidade do modelo de generalização, é necessário entender a função margem do modelo, que consiste em medir o nível em que o número médio de votos da classificação para a classe X, excede a classificação para outra classe. Quanto maior a margem, mais confiança se tem na classificação (Amit, 1997).

Dessa forma, dado um conjunto de classificadores $h_1(x), h_2(x), \ldots, h_K(x)$, e com o conjunto de treinamento obtido aleatoriamente a partir da distribuição do vetor aleatório X, Y é definida a função de margem na Equação 18:

$$mg(X,Y) = P_\Theta(h(X,\Theta) = Y) - max_{j \neq Y}(P_\Theta(h(X,\Theta) = j) \quad (18)$$

Onde Θ representa vetores aleatórios distribuídos de forma independente. O erro de generalização pode ser dado por:

$$PE = P_{X,Y}(mg(X,Y)) < 0 \quad (19)$$

É possível demonstrar que pela Lei Forte dos Grandes Números, que o erro de generalização converge para:

$$P_{X,Y}(P_\Theta(h(X,\Theta) = Y) - max_{j \neq Y}(P_\Theta(h(X,\Theta) = j)) < 0 \quad (20)$$

Este resultado explica porque as RFs não se ajustam excessivamente à medida que mais árvores são adicionadas, mas produz um valor limite para o erro de generalização. Por fim, o objetivo principal de *Random Forest* é minimizar o erro de generalização com a menor perda de dados.

2.4 Redes Neurais

Redes Neurais Artificiais (RNA) são modelos inspirados no cérebro, usando unidades de processamento simples chamadas neurônios. Essas redes são interconectadas e armazenam o conhecimento nas conexões entre os neurônios, chamadas pesos. Durante o treinamento, os pesos são ajustados com base em um conjunto de exemplos. As RNA possuem diversas arquiteturas e são usadas em tarefas como reconhecimento de imagens, síntese de fala e modelagem de sistemas dinâmicos. Suas principais características são tolerância a ruídos, processamento paralelo, robustez em caso de falhas e capacidade de generalização (Haykin, 2009).

2.4.1 Redes Neurais Artificiais do tipo Multilayer Perceptron: As redes MLP (Perceptron de Múltiplas Camadas) são caracterizadas pela presença de pelo menos uma camada intermediária (oculta) de neurônios, localizada entre a camada de entrada e a camada de saída. Essa arquitetura é denominada *feedforward*, pois os sinais de entrada são propagados em direção à camada de saída. Dessa forma, a saída dos neurônios da camada anterior é utilizada como entrada para os neurônios da próxima camada, e assim sucessivamente, conforme ilustrado na Figura 5.

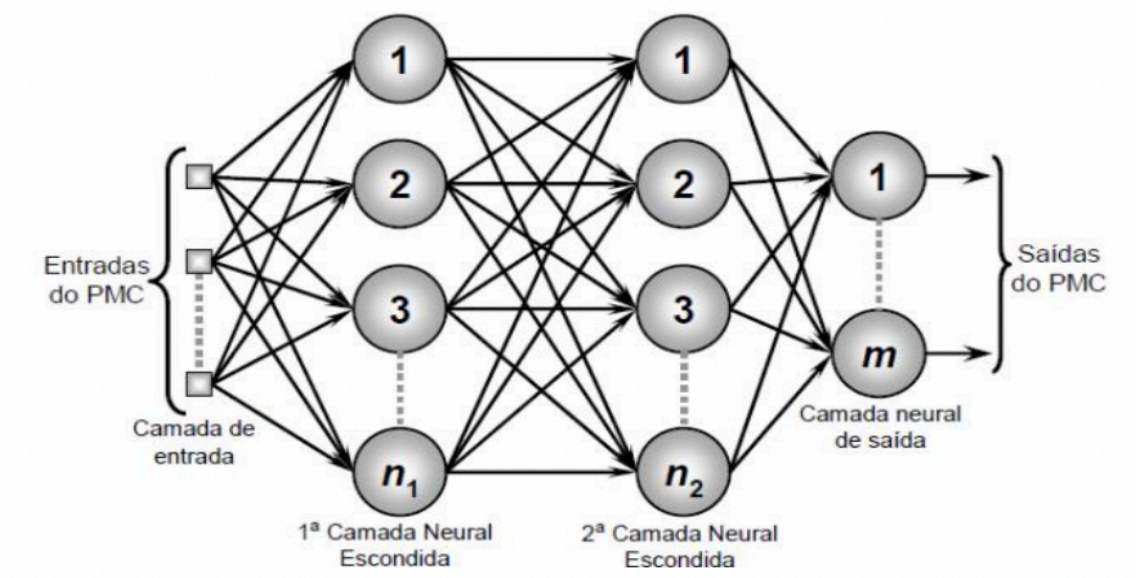

Figura 5 - Rede Neural Perceptron Multicamadas. Fonte: Silva, 2010

Para (Silva, 2010), as redes de múltiplas camadas surgiram como uma solução para superar a restrição das redes de uma única camada, que não podem lidar com problemas não linearmente separáveis. No entanto, o desenvolvimento do algoritmo de treinamento backpropagation em 1986 mostrou que é possível treinar eficientemente redes com camadas intermediárias. Isso resultou no surgimento das Redes Neurais Artificiais mais utilizadas atualmente, conhecidas como *Perceptron Multicamadas* (PMC), treinadas com o algoritmo de *backpropagation*. Esse avanço revolucionou o campo das redes neurais, permitindo abordar problemas complexos de forma eficiente.

O algoritmo de treinamento *backpropagation* consiste em duas fases principais. Na primeira fase, chamada de "propagação adiante", os sinais de entrada do conjunto de treinamento são propagados pela rede, atualizando os pesos sinápticos e limiares dos neurônios e calculando o erro entre a saída da rede e os valores desejados de saída.

Já na segunda fase, denominada "retropropagação", o erro calculado é propagado reversamente, ajustando os pesos sinápticos com uma regra de correção de erro. Essas fases são cruciais para o treinamento eficiente de redes neurais, permitindo que a rede aprenda com os erros e se adapte aos padrões de entrada e saída desejados.

3 Trabalhos relacionados

Inicialmente foram pesquisados trabalhos da literatura relacionados com algoritmos de aprendizado de máquina ou algoritmos de aprendizado profundo ou algoritmos de processamento de imagens, que utilizaram imagens ou conjuntos de dados, realizando análise das componentes RGB das imagens e que pudessem oferecer contribuições relevantes. Dentre os trabalhos pesquisados destacaram-se:

O artigo de Chen e Ma (2014) apresentou o método RBOR (*region-based object recognition*) para identificar objetos de cenas do mundo real, como brinquedos, materiais de escritório ou carros, em imagens de baixa resolução. O método segmenta as imagens coloridas dos modelos de objetos e, também imagens de teste e, utiliza uma SPCNN (*simplified pulse-coupled neural network*) para fazer uma correspondência entre as imagens. O método integra componentes RGB normalizadas em conjunto com espaços de cores oponentes, como vermelho versus verde e azul versus amarelo, o que permite uma representação mais eficiente das diferenças de cor a fim de reduzir efeitos de intensidade de luz, obtendo como melhor resultado a acurácia média de 80% em um base de dados de 1611 imagens, o que demonstra a necessidade de uma melhoria na precisão do modelo, apesar de sua proposta inovadora.

O trabalho de Jiao e Zhang (2019) utilizou o sistema YOLOv3 para propor um algoritmo, baseado em CNN (*Convolutional Neural Network)* de detecção de incêndios florestais em imagens aéreas coletadas por UAVs *(Unmanned Aerial Vehicl).* Os resultados dos testes mostraram que o trabalho alcançou acurácia média de 83% e precisão de 82% para taxas de quadro de até 3,2 *fps* (frame por segundo), um valor baixo quando comparado com outros trabalhos do mesmo segmento, mas com perspectivas interessantes para a aplicação de redes neurais convolucionais na detecção de incêndios florestais, abrindo margem para a aplicação de otimizadores e técnicas de pré-processamento.

O trabalho de Chen e Hopkins (2022) apresentou um conjunto de

dados com imagens RGB/IR *(Red, Green, Blue/Infra-red*) lado a lado coletadas durante um incêndio. As imagens foram rotuladas em fogo/não-fogo e fumaça/não-fumaça. O treinamento do algoritmo do trabalho utilizou método híbrido que colocou em cascata um classificador baseado em DL (*Deep Learning*) e uma técnica de localização de detecção de incêndio, para processar imagens com alimentação dupla, com valores RGB e de imagens termais. O método proposto obteve como melhor resultado a acurácia de 99,91% com precisão de 99,9% para detecção de fogo, utilizando o modelo *fine-tuned* pré-treinado *VGG16*. O fato de terem utilizado uma base de dados com mais de 53 mil imagens RGB/IR e um modelo DL para treinamento tornou o método eficaz, mas de alto custo computacional, algo indicado como uma potencial melhoria pelos autores.

Zhang e Qu (2022) desenvolveram um método de detecção de arco elétrico com base em rede residual (*ResNet*). O método aplicou análise de *wavelets* discretas multicamadas, converteu as wavelets em imagens no espaço de cores RGB, e utilizou as imagens RGB para treinamento da máquina *ResNet*. O método proposto obteve resultados próximos a 97,36% de acurácia com 95,83% de precisão e, dentre os principais resultados, é salientado a correção do problema de *over-fitting* do modelo ao adicionar novos parâmetros, chamadas funções de penalidade, a esta função. No entanto, apesar de ter solucionado o problema, impactou diretamente nos potenciais valores de acurácia e precisão, o que limita seu uso em bases de dados maiores.

O artigo de Khryashchev e Larionov (2020) apresenta uma rede neural convolucional para detecção automatizada de incêndios florestais em fotos aéreas de alta resolução, utilizando diferentes técnicas de aumento de dados, como o *janelamento de dados*, para ampliação dos conjuntos de treinamento, além do uso da segmentação de imagens como técnica de pré-processamento. Para avaliação, foram utilizadas métricas especiais, como coeficiente *Sorensen-Dice*, precisão, recall, *F1-score* e valor *IoU*, que permitiram medir a qualidade do modelo desenvolvido.

Como melhores resultados foram obtidos precisão de 0,45 e IoU de 0,782. Os resultados estão abaixo de outros trabalhos presentes na literatura, certamente devido ao tamanho de sua base de dados e ao modelo utilizado para treinamento, o fato de segmentar as imagens obtidas possibilita a aplicação de diferentes modelos de DL e ML, algo não explorado pelos autores.

Com o objetivo de desenvolver um algoritmo de baixa complexidade e baixo custo computacional, o trabalho desenvolvido por Jadon e Varshney (2020) demonstrou o uso de uma arquitetura MobileNetV2 modificada e uma estratégia simples para composição da base de dados, editando as imagens disponíveis (rotacionando, aproximado, cortando), capaz de superar outras soluções existentes, sendo ainda computacionalmente viável para implantação em hardware de menor capacidade de processamento. Seus resultados demostraram acurácia máxima de 99.17% com precisão de 0.92 para detecção de fogo. Os resultados deste estudo sugerem a possibilidade de extrapolação de dados. Embora não sejam mencionados problemas específicos de o*ver-fitting* ou extrapolação pelos autores, é importante ter em mente que o uso do MobileNetV2 em conjuntos de dados pequenos pode levantar preocupações nesse sentido.

Utilizando a arquitetura *MobileNetV2*, Wu e Li (2020) propôs um algoritmo de baixa complexidade e rápida resposta, utilizando um conjunto de treinamento pequeno e diferentes técnicas de distorção de imagem, como desfoque e ruído, para o aumento e aprimoramento do treinamento. Seus resultados mostraram que o método manteve uma alta precisão de identificação em diferentes situações e teve tem como melhor resultado o valor de 99.7% de acurácia, com acurácia média de 95,2%. O trabalho não reportou dados de sensibilidade ou especificidade e não demonstra sua aplicação em bases de dados diferentes, o que pode sugerir que seus resultados são decorrentes da extrapolação de dados durante o treinamento.

A proposta de Wu e Zhang (2018) apresentou técnicas modernas e

amplamente utilizadas para detecção de incêndios florestais, como *Faster R-CNN, YOLO* e *SSD*, estabelecendo uma comparação entre esses métodos e justificando seus usos com base em características individuais, como precisão, capacidade de detecção de incêndio e tempo de detecção. Utilizou-se uma base de dados de benchmark de incêndio e fumaça, com alterações da classe de fumaça e área de incêndio para minimizar a detecção incorreta. Seus resultados mais relevantes foram obtidos com a rede *R-CNN,* com acurácia de 99.7% para detecção de fogo e 79.7% para detecção de fumaça, um valor baixo quando comparado a outros trabalhos disponíveis na literatura, o que demonstra a necessidade de um tratamento de dados ainda mais robusto ou um aprimoramento da técnica utilizada. O trabalho também não relata métricas de especificidade e sensibilidade, que justificariam dados tão baixos de acurácia para detecção de fumaça.

Almeida e Huang (2022) propuseram um modelo de rede neural convolucional (*CNN*) para detecção de incêndio florestal por meio de imagens RGB, com objetivo de aplicá-lo a veículos aéreos não tripulados (Sigla-UAV) e sistemas de vigilância por vídeo. Como técnica de pré-processamento, é realizado apenas um redimensionamento de imagens na base de dados de 72 mil imagens de fogo, não-fogo e fumaça, alcançando acurácia média de 97,37% e precisão de 97,6%. Seus autores enfatizam a preocupação com o uso da memória de GPU e a velocidade de classificação, utilizando média de 2,3 GB memória e 0,76 segundos para classificação, respectivamente.

O estudo realizado por Bakri e Adnan (2018) concentrou-se no pré-processamento e classificação de imagens de incêndio com base nos valores das componentes RGB de cada pixel. O sistema desenvolvido utilizou técnicas de aumento de contraste para aprimorar as imagens e modelos de cores RGB e YCbCr, com critérios definidos para distinguir os pixels de fogo do plano de fundo e isolar a luminância da crominância da imagem original para detecção de fogo. O sistema obteve uma média de 90% de precisão na detecção de incêndios. No entanto, o modelo não

considera a detecção de imagens de fumaça. Além disso, o estudo utilizou um conjunto de dados com apenas 10 imagens para validar o modelo, sem fornecer informações sobre sensibilidade ou especificidade, que são métricas importantes para avaliação.

Premal e Vinsley (2014) adotaram o modelo de cores *YCbCr* para detecção de incêndios florestais. O espaço de cores *YCbCr* separa efetivamente a luminância da crominância em comparação com outros espaços de cores como RGB e RGB normalizado. O método proposto não apenas separou os pixels da chama do incêndio, mas também separou os pixels do centro do incêndio de alta temperatura, levando em consideração os parâmetros estatísticos da imagem do incêndio no espaço de cores *YCbCr*, como média e desvio padrão. O sistema alcançou acurácia máxima de 99.4%, mas não demonstra sua funcionalidade em bases de dados diferentes, o que pode sugerir que o valor de 99,4% seja advindo da extrapolação de dados.

Os artigos apresentados utilizaram algoritmos de aprendizado de máquina com diferentes técnicas de processamento de imagens, originadas de bancos de dados públicos ou de construção própria. Os melhores resultados foram obtidos utilizando a rede neural *CNN* e a arquitetura *MobileNet*.

Também é possível destacar a técnica de pré-processamento por separação de valores RGB pixel-a-pixel, que obteve resultados satisfatórios com a utilização de um algoritmo de treinamento e classificação mais simples. No entanto, notoriamente alguns trabalhos sofreram com problemas de extrapolação de dados, natural em bases de dados pequenas e sobretudo em arquiteturas *MobileNet*. Além disso, os melhores resultados foram obtidos ao utilizar bases de dados maiores e sistemas de classificação mais complexos, o que eleva o custo computacional e o torna menos viável.

Algumas características dos trabalhos relacionados são apresentadas na Tabela 1.

Tabela 1 - Características dos trabalhos relacionados. Fonte: Própria.

Artigo	**Base de Dados**	**Pré-processamento de Imagens**	**Algoritmo de Treinamento e Classificação**	**Melhor Resultado**
Chen e Ma (2014)	Pública e própria (1611 imagens)	RGB normalizado com espaços de cores oponentes	SPCNN	Acurácia média de 80% para detecção de objetos.
Jiao e Zhang (2019)	Própria (Vídeos de resolução 400x240 a 1920x1080. Média de 25.80fps. Tamanho não divulgado)	Não-aplicável	YOLOv3	Acurácia de 83%, 82% de precisão e 79% de *recall* para detecção de fogo e fumaça.
Chen e Hopkins (2022)	Pública (53.451 frames de vídeos – Imagens RBG/IR)	MSER para overlapping	MobilenetV2	Acurácia de 99.8%, Precisão de 99.78% e Recall de 99.87% para detecção de fogo e fumaça.
Zhang e Qu (2022)	Base Própria (tamanho não divulgado)	Conversão em espaço de cores RGB	Resnet152	Acurácia de 97.36% para detecção de arco-elétrico.
Khryashchev e Larionov	Públicas (1457 e 393	Segmentação	ResNet34, CNN	Precisão de 45%, Recall de

(2020)	imagens)	de Imagens.		47,5%, F1-score de 46% e DSC de 87% para detecção de incêndios em imagens de satélite.
Jadon e Varshney (2020)	Públicas (77 vídeos, 25.656 frames)	Não-aplicável	MobileNetV2	Acurácia máxima de 99.17%, Precisão de 92% e Recall de 92% para detecção de fogo.
Wu e Li (2020)	Própria (2096 imagens)	Função Guassian-blur	MobileNetV2	Acurácia média igual a 95.2% para detecção de fogo e fumaça.
Wu e Zhang (2018)	Própria (1000 imagens)	Não-aplicável	Faster R-CNN, YOLO e SSD	Acurácia média igual a 99.7% para detecção de fogo e 79.7% para detecção de fumaça, com a técnica SSD.
Almeida e Zhang (2022)	Pública (72.000 imagens RGB)	Não-aplicável	CNN	Acurácia média igual a 97,37% e precisão de 97.6% para

				detecção de fogo e fumaça.
Bakri e Adnan (2018)	Pública – RGB e YCbCr (tamanho não divulgado)	Extração de Valores RGB, YCbCr	CNN, SVM	Acurácia média = 90%.
Premal e Vinsley (2014)	Própria – RGB e YCbCr (800 imagens)	Segmentação, Modelo de cores YCbcR	Perceptron Multicamadas	Acurácia média = 99.4%. Sensibilidade = 88%.

O fator decisivo para um método de identificação de incêndios florestais baseado em um algoritmo de baixa complexidade é a aplicação de uma técnica de pré-processamento adequada e que evidencie características comuns nos valores RGBs destas imagens. Para isso, é necessário diminuir ruídos e tratar as imagens para obtenção de apenas valores relevantes para classificação, além de traçar padrões repetidos entre estas. Como solução, este trabalho apresenta proposta da utilização de duas técnicas de pré-processamento, excluindo o fundo das imagens analisadas e tratando os valores das componentes RGB das imagens, tornando os dados adequados para algoritmos de classificação.

4 Material e métodos

Essa seção descreve detalhadamente os procedimentos e recursos utilizados neste estudo científico. O diagrama na Figura 6 demonstra as etapas do método proposto, dividido em três etapas: processamento de dados, treinamento de máquina e classificação, tendo as bases de dados de imagem como entrada e a classificação dessas imagens como saída. Todas as etapas serão detalhadas nas subseções seguintes.

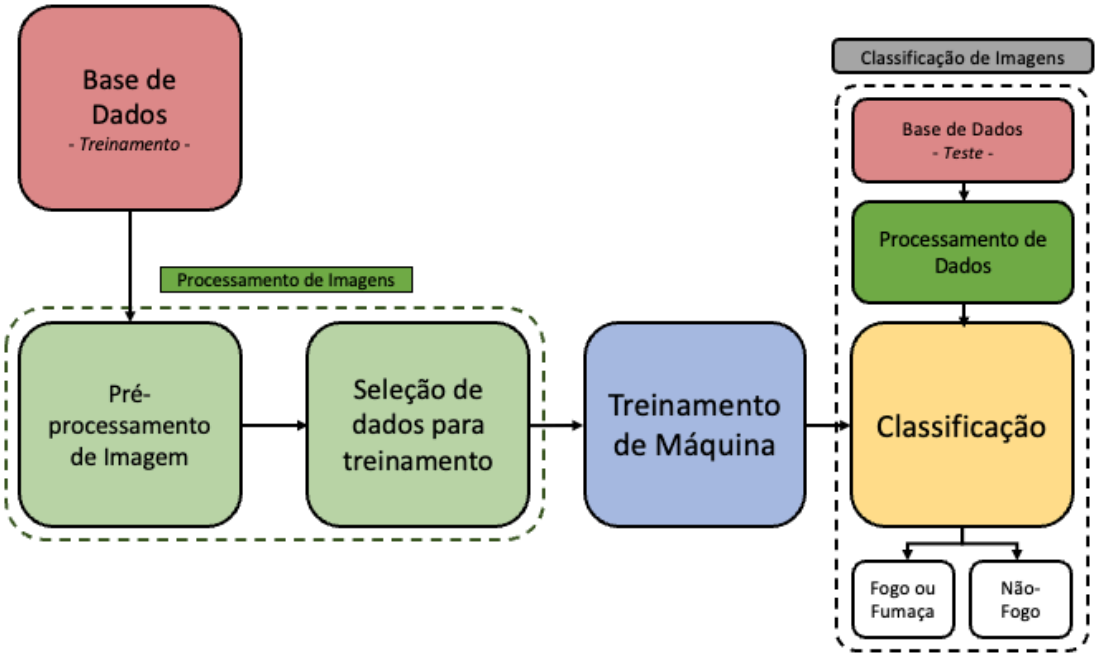

Figura 6 - Diagrama do algoritmo proposto. Fonte: Própria.

4.1 Base de Dados

Os dados utilizados neste trabalho foram prevenientes de duas bases de dados, *CorsicanFire Dataset* [33] e *Deepfire* [34], contendo 1000 e 2500 imagens, respectivamente.

A base CorsicanFire (2017) contém imagens codificadas nos sistemas RGB e NIR (*Near Infra-Red*). Neste conjunto de dados, as imagens NIR são capturadas com maior tempo de integração/exposição, o que aumenta o brilho da área de incêndio e facilita a segmentação com

técnicas simples de processamento de imagem. Estas imagens foram adquiridas no alcance visual, no alcance do infravermelho próximo e sob várias condições de posicionamento, visão, clima, vegetação, distância ao fogo e brilho. A base possui 1000 imagens, divididas igualmente entre imagens de incêndio e imagens sem incêndio, conforme exemplificado na Figura 7.

Figura 7 - Amostras da base de dados CorsicanFire, com imagens contendo incêndio e não-incêndio.

Fonte: CorsicanFire.

A base de dados *DeepFire (A Novel Dataset and Deep Transfer Learning Benchmark for Forest Fire Detection)* (2022) contém um conjunto de imagens criado para ser usado como uma ferramenta de benchmark para o desenvolvimento de sistemas de detecção de incêndios florestais. A base de dados possui 2500 imagens no sistema de cores RGB, com alta resolução, coletadas ao longo de vários anos em diferentes regiões do mundo, e que foram rotuladas manualmente para indicar a presença ou ausência de incêndios florestais.

O conjunto de dados de imagens *"DeepFire"* é considerada valiosa para a comunidade de pesquisa em incêndios florestais, pois fornece uma grande quantidade de imagens rotuladas de alta qualidade que podem ser usadas para treinar e avaliar modelos de aprendizado profundo [32]. Além disso, o conjunto de imagens, exemplificado na Figura 8, inclui um conjunto de teste independente que pode ser usado para avaliar o

desempenho de modelos treinados em outros conjuntos de imagens, o que a torna útil como um benchmark de transferência de aprendizado profundo [32].

Figura 8 - Amostra da base de dados DeepFire. Contendo imagens com incêndio e sem incêndio. Fonte: DeepFire (2022).

Nesta pesquisa, as bases de dados *CorsicanFire* e *DeepFire* foram utilizadas de maneira conjunta para treinamento e validação, com uma porcentagem de divisão, em cada uma das bases, de 70% para treinamento e 30% para validação. Os testes foram realizados utilizando bases de dados externas, indicadas na Tabela 2. As imagens foram selecionadas a fim de obter um valor igual de imagens de fogo/fumaça e não-fogo.

Tabela 2 - Bases de dados utilizadas para realização dos testes. Fonte: Própria.

Base de dados	**Tamanho**	**Tipo**
The flame dataset: aerial imagery pile burn detection using drones [35]	2000 imagens (1500 utilizadas)	Imagens de Fogo e não-fogo.
Kaggle Dataset [36]	500 imagens (500 utilizadas)	Imagens de Fogo, não-fogo e Fumaça.
Center for wildfire research University of Split Faculty of Electrical Engineering et al., 2014 [37]	100 imagens (100 utilizadas)	Imagens com Fumaça

4.2 Diagrama geral da proposta

O diagrama de blocos na Figura 9 apresenta o ciclo de trabalho do método proposto, que inicia com a inserção da base de dados, a qual contém imagens para treinamento e teste, descrito na subseção 4.1. Em seguida, o pré-processamento de imagens é divido em quatro sub-etapas: segmentação, que extrai o primeiro plano da imagem, divisão da imagem em quadrantes, determinação dos valores dos canais RGB e a determinação dos valores de quatro quartis para cada canal de cada quadrante.

Após o pré-processamento, o algoritmo seleciona os valores para treinamento e elimina os dados em que todos os quartis são iguais a 0, essas etapas são descritas detalhadamente na subseção 4.3. Os dados selecionados são utilizados como entrada para o algoritmo de treinamento de máquina, descrito detalhadamente na subseção 4.4. A

classificação de imagens de teste também utiliza as técnicas de processamento utilizadas na fase de treinamento.

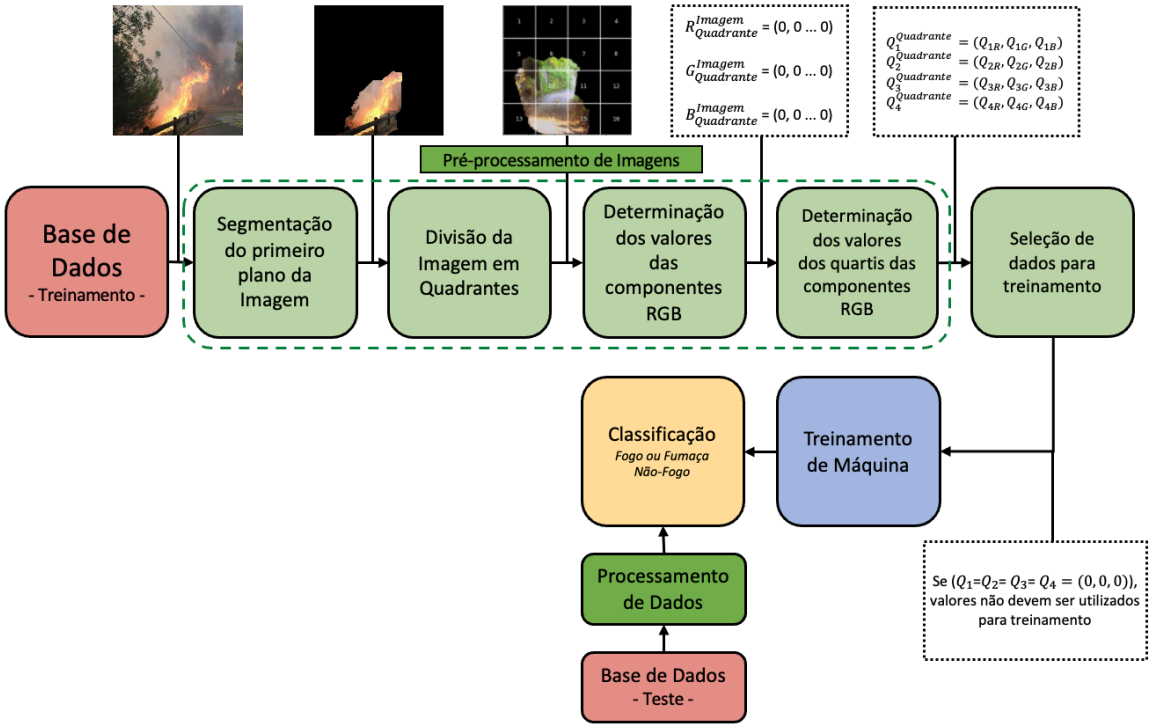

Figura 9 - Diagrama geral da proposta. Fonte: Própria.

4.3 Pré-processamento de Imagens

A proposta se iniciou com a fase de pré-processamento das imagens, dividida em quatro etapas:

1. Segmentação do primeiro plano da imagem.
2. Divisão da imagem em oito quadrantes.
3. Determinação dos valores das componentes RGB.
4. Determinação dos valores dos quartis das componentes RGB.

A segmentação do primeiro plano da imagem foi realizada utilizando a função *grabCut* da biblioteca cv2 [37] utilizando o método de separação de grafos [37].

Para isso, o algoritmo inicia com uma estimativa inicial do primeiro plano e plano de fundo, com base em máscaras de segmentação. Em seguida, ele itera entre as etapas de estimativa e atualização, refinando

os modelos de cor e realizando o cálculo do corte mínimo para atualizar as estimativas das regiões de primeiro plano e plano de fundo. Esse processo é repetido até que a segmentação seja considerada satisfatória.

Ao final da execução do *GrabCut*, obtém-se uma máscara binária indicando quais regiões da imagem são consideradas como primeiro plano (fumaça e fogo) e plano de fundo. Essa máscara é utilizada para isolar e extrair as regiões de interesse da imagem original

Conforme Figura 9 do diagrama geral da proposta, esta fase foi aplicada tanto às imagens das bases de treinamento como para as imagens das bases de testes e teve como objetivo, extrair da imagem o primeiro plano, no qual os fenômenos fogo e fumaça, podem ser evidenciados.

Apenas o primeiro plano da imagem foi mantido, enquanto o background foi descartado. Para as imagens contidas nas bases de dados utilizadas no treinamento, não houve problemas na detecção de presença de incêndio ou fumaça, como ilustra a Figura 10.

Figura 10 - Imagens de incêndio originais (à esquerda) e imagens pós segmentação (à direita). Fonte: própria.

No entanto, é importante ressaltar que, devido ao método de separação e grafos utilizado, a precisão de recorte ainda é baixa, levando o algoritmo a partes da imagem que poderiam ser relevantes.

Além disso, para imagens que não continham fogo, fumaça ou focos de incêndio, o algoritmo destacava a parte mais rugosa ou com maior relevo, considerando partes planas como segundo plano, como ilustra a Figura 11.

Figura 11 - Imagens sem incêndio originais (à esquerda) e imagens pós segmentação (à direita). Fonte: própria.

Uma vez concluída a segmentação de imagens, na proposta realizou-se a divisão da imagem em oito partes de tamanhos iguais (divisão em quadrantes), com intuito de eliminar quadrantes desnecessários pela ausência de 1º plano da imagem, a fim de tornar a fase de treinamento mais simples e veloz. Dessa maneira, a saída deste bloco da proposta do presente trabalho possui oito subpartes das imagens, como ilustra a Figura 12.

Figura 12 - Imagem original (à esquerda), imagem pós segmentação (ao meio) e imagem pós divisão de quadrantes (à direita). Fonte: própria.

Após divisão das imagens em quadrantes, foi realizada a separação das componentes RGB das imagens, utilizando a função *split*() da biblioteca *cv2* [38]. Obtendo os valores dos canais *Red, Green e Blue* para cada um dos pixels presentes na imagem, conforme demonstrado na Figura 13.

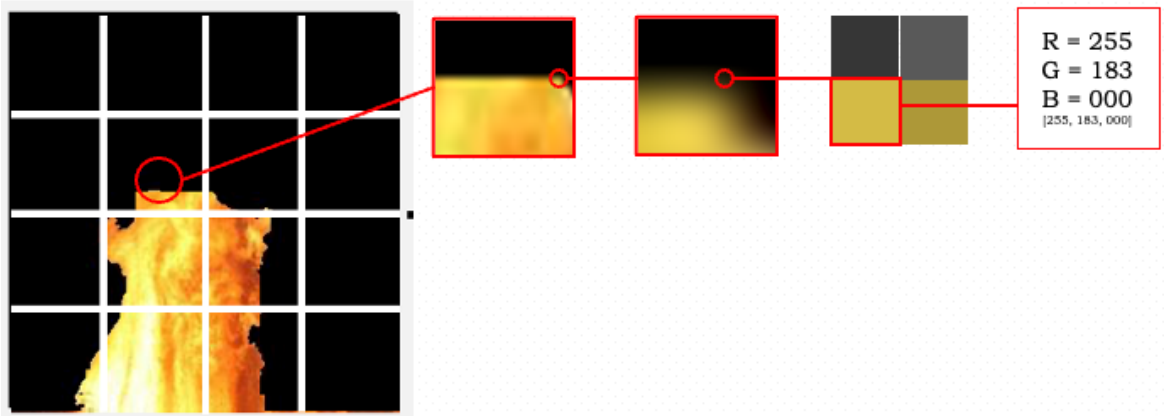

Figura 13 - Representação gráfica dos valores RGB para um pixel. Fonte: própria.

Cada pixel resulta em uma lista contendo os três valores dos canais RGB, conforme ilustrado na Figura 14, no qual a primeira posição da lista corresponde aos valores do canal *Red*, a segunda posição corresponde aos valores do canal Green e a terceira aos valores do canal Blue. A imagem completa é dada por uma matriz de cujo tamanho é dado pelas dimensões da imagem.

$$= \begin{bmatrix} [0, 29, 110] & \cdots & [31, 50, 21] \\ \vdots & \ddots & \vdots \\ [11, 49, 87] & \cdots & [12, 100, 95] \end{bmatrix}$$

(a) (b)

Figura 14 - Representação gráfica dos valores RGB (b), à direita, da imagem original (a). Fonte: própria.

Na etapa seguinte da nossa abordagem, realizamos o cálculo dos quartis para cada componente RGB em cada quadrante da imagem. Para isso, cada pixel é desmembrado em seus componentes individuais de *Red, Green e Blue,* fornecendo assim os valores correspondentes para todos os pixels da imagem, no qual as matrizes correspondentes são convertidas em listas unidimensionais e as ordenadas em ordem crescente. O processo de conversão e ordenação está representado de forma ilustrativa na Figura 15. Essa etapa é fundamental para obtermos informações detalhadas sobre a distribuição dos valores de cor em cada quadrante, permitindo uma análise precisa dos quartis.

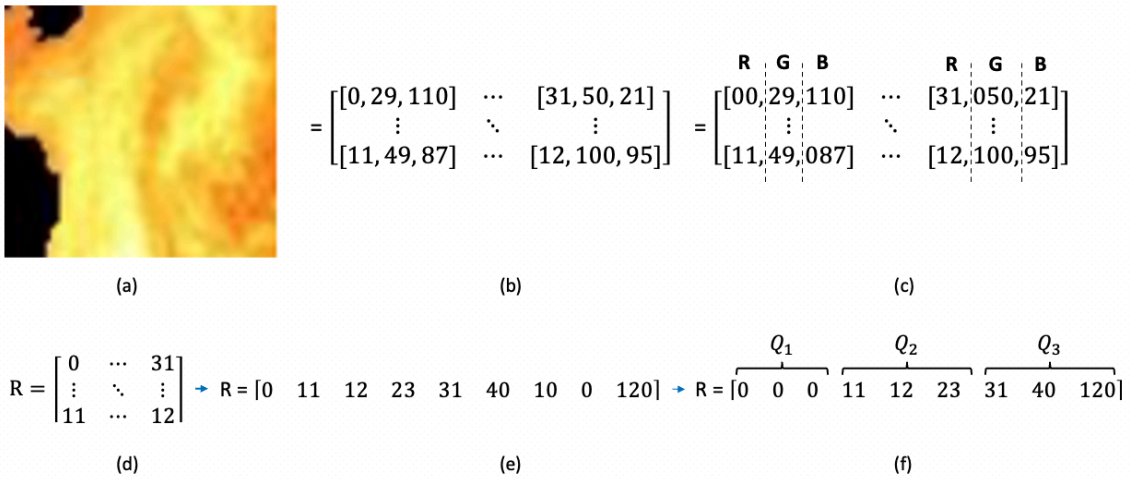

Figura 15 – (a) Quadrante selecionado (b) Representação da matriz numérica dos valores dos canais RGB. (c) Identificação dos canais RGB na matriz numérica. (d) Representação numérica da matriz com valores do canal *Red*. (e) Representação dos valores numéricos de canais RGB no formato de lista. (f) Representação numérica dos quartis dos valores RGB. Fonte: Própria.

A Figura 16 ilustra a determinação dos valores dos quartis (à direita) a partir dos valores das componentes RGB (à esquerda) para uma imagem dividida em 16 quadrantes.

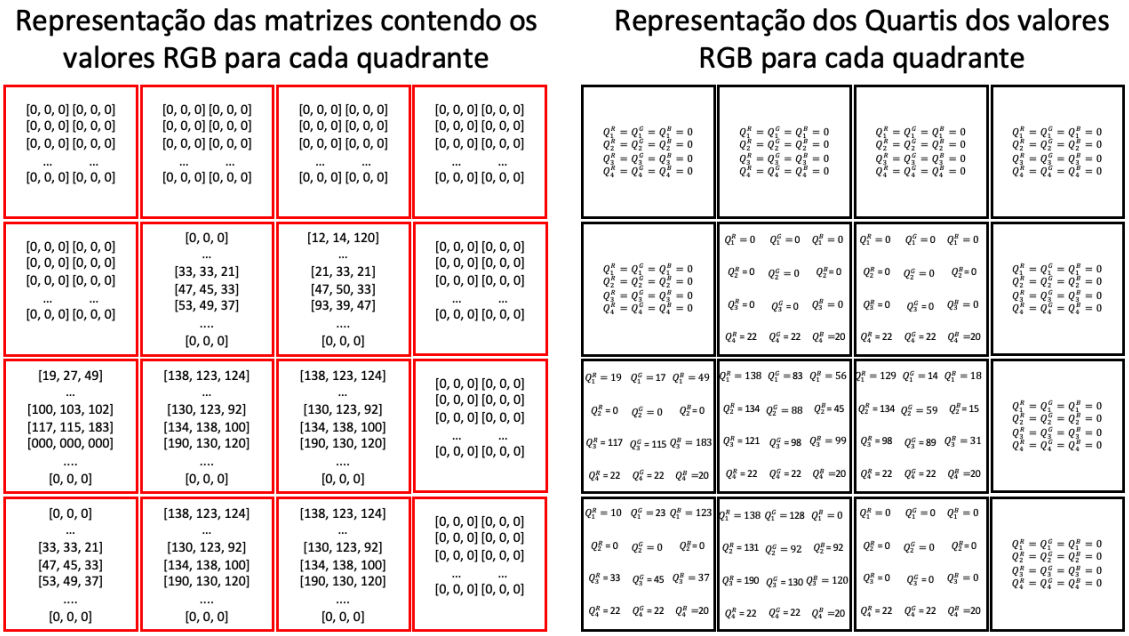

Figura 16 – Valores dos quartis (à direita) determinados a partir dos valores das componentes RGB (à esquerda) dos pixels de cada quadrante da imagem. Fonte: Própria.

A partir da geração dos valores dos quartis, foi possível observar um padrão numérico para a base de treinamento de dados, onde todos os quadrantes possuíam obrigatoriamente doze valores de quartis, além de tornar possível a eliminação de quadrantes da imagem com pouca ou nenhuma informação.

Foi adotado um critério de eliminação dos quadrantes com baixa informação, no qual se estabeleceu que um quadrante seria descartado caso apresentasse valores de quartis iguais a zero para as componentes R, G e B. Essa decisão baseou-se no fato de que valores de (0, 0, 0) para as componentes RGB correspondem à cor preta ou nula, indicando a ausência de informação significativa no quadrante em questão.

As informações dos quadrantes de imagens não eliminados na etapa anterior, nomeada *"Seleção de dados"* na Figura 9, foram inseridas em um arquivo chamado *"input.csv"*. Neste arquivo foram incluídas informações do título da imagem original, quadrante da imagem, 12 valores de quartil (ex. 4 valores para cada uma das componentes RGB do quadrante) e, a informação se a imagem continha fogo, fumaça ou não possuía esses fenômenos.

Para avaliar como as técnicas de pré-processamento afetam o tempo de execução e os valores de acurácia, sensibilidade e especificidade, foram realizados testes incrementais, nos quais as técnicas foram combinadas de acordo com a Tabela 3. Todas as combinações foram executadas sob as mesmas condições e utilizando as mesmas bases de dados.

Tabela 3 - Descrição dos diferentes métodos de pré-processamento utilizados para realização de testes. Fonte: Própria

Método de Pré-processamento	**Etapas Realizadas**	**Dados Utilizados para Treinamento**
M1	Determinação dos valores das componentes RGB e seleção de dados.	Valores das componentes RGB das imagens.
M2	Segmentação de imagens, extração dos valores das componentes RGB e seleção de dados.	Valores das componentes RGB das imagens pós-segmentação.
M3	Segmentação de imagens, divisão em quadrantes e extração dos valores das componentes RGB e seleção de dados	Valores das componentes RGB dos quadrantes das imagens pós-segmentação.
Método Completo (MC)	Segmentação de imagens, divisão em quadrantes, extração dos valores das componentes RGB, cálculo dos quartis e seleção de dados	Valores dos quartis das componentes RGB dos quadrantes das imagens pós-segmentação.

4.4 Desenvolvimento da Máquina de Aprendizado

Utilizando a plataforma Jupyter Notebook, a implementação do modelo foi feita na linguagem Python. Foram empregadas as bibliotecas *pandas*, *numpy* e *matplotlib*, para visualização gráfica dos resultados.

Foram desenvolvidos 04 modelos de aprendizado de máquina. A implementação do modelo Naive Bayes foi realizada utilizando os parâmetros numéricos propostos por John e Langley (1995), enquanto a implementação do modelo kNN foi realizada utilizando como principal parâmetro o valor $1 \leq k \leq 20$ e distância euclidiana = 2. Para o modelo *Random Forest*, a quantidade de árvores foi definida para $100 \leq n \leq 2000$, com incremento de 200 e profundidade máxima da árvore variando nos valores 0, 10, 50 e 100. Para o modelo *Perceptron Multicamadas*,

foram utilizadas como camadas ocultas os valores 100, 50 e 25, taxa de regularização de 0.0001 a 0.01, taxa de aprendizado 'constante', otimizador '*ADAM*', taxa de interações de 100 a 500, tamanho de lote de 32 a 64 e estado aleatório de 42 a 132.

O arquivo input.csv, comentado na Seção 4.2, foi utilizado como entrada para a fase de treinamento dos algoritmos. O arquivo comentado continha os valores de quartis para cada componente RGB de cada quadrante de cada imagem.

Após a execução da fase de treinamento, foi realizada fase de testes com a utilização das bases de dados de testes. Para cada configuração, foram calculadas a acurácia, sensibilidade e especificidade do modelo, a fim de identificar os parâmetros com os melhores resultados.

As imagens das bases de dados de testes passaram pelo mesmo processo que as imagens das bases de treinamento, como mostra o diagrama na Figura 12. Contudo, para as imagens da base de testes, o algoritmo ao final da execução, realizou a classificação, retornou a imagem original inserido como entrada do algoritmo e a possível localização do incêndio.

4.5 Métricas de avaliação de desempenho

Para avaliar o modelo de classificação e a eficiência do teste, foram consideradas a acurácia, sensibilidade e especificidade como medidas de avaliação de desempenho. A acurácia mede a proporção de predições corretas e indica o desempenho geral do modelo e é calculada utilizando o número de todas as predições corretas dividido pelo número total de predições realizadas pelo modelo, conforme indica a Equação 21:

$$Acurácia\ (\%) = \frac{Predições\ Corretas}{Predições\ Realizadas} \tag{21}$$

Em um exemplo hipotético de detecção de incêndio florestal em um conjunto de áreas, o valor de sensibilidade indica a probabilidade de uma área avaliada e com incêndio de ser corretamente identificada. Seu cálculo é dado pelo número de áreas corretamente detectadas com incêndio (verdadeiros positivos) dividido pelo número total de áreas verdadeiramente afetadas por incêndio, conforme indicado pela Equação 22:

$$Sensibilidade\ (\%) = \frac{Verdadeiros\ Positivos}{Verdadeiros\ Positivos + Falsos\ Negativos} \quad (22)$$

O valor de especificidade indica a probabilidade de uma área avaliada e sem incêndio de ser corretamente identificada como livre de incêndio. Seu cálculo é feito utilizando o número de áreas corretamente classificadas como não afetadas por incêndio (verdadeiros negativos) dividido pelo total de áreas não afetadas (verdadeiros negativos e falsos positivos), conforme indicado pela Equação 23:

$$Especificidade\ (\%) = \frac{Verdadeiros\ Negativos}{Verdadeiros\ Negativos + Falsos\ Positivos} \quad (23)$$

5 Resultados Obtidos

5.1 Tempos de pré-processamento e treinamento:

É importante considerar o tempo de processamento de imagens em um algoritmo de identificação de incêndios florestais, uma vez que, quanto menor o tempo de processamento, maior será a rapidez com que as informações são obtidas e, portanto, mais rápida será a capacidade de agir para controlar o incêndio.

Além disso, um tempo de processamento reduzido torna o algoritmo de processamento de imagem mais escalável, permitindo sua aplicação eficiente em conjuntos de dados maiores, o que é particularmente importante em ambientes remotos ou com recursos computacionais limitados.

Neste trabalho, foram considerados os tempos de cada etapa do algoritmo, desde a leitura completa das imagens até o treinamento, sob execução na máquina *Macbook Pro M2 (2022)* com imagens de 500x500 pixels. Ao todo, o treinamento durou, em média 13 minutos, uma vez que o tempo de treinamento das máquinas de aprendizado computacional avaliadas foram similares.

A Figura 17 representa graficamente os tempos associados às subfases do pré-processamento, detalhadas na Tabela 3 da seção 4.3, e da fase de treinamento das máquinas de aprendizado computacional para um conjunto de 3500 imagens, conforme detalhado na Tabela 2 da seção 4.1. É importante ressaltar que não observamos diferenças significativas nos tempos de treinamento entre os três algoritmos avaliados, dessa forma o item “Treinamento de Máquina” se refere a uma média de tempo de execução dentre as 4 máquinas empregadas.

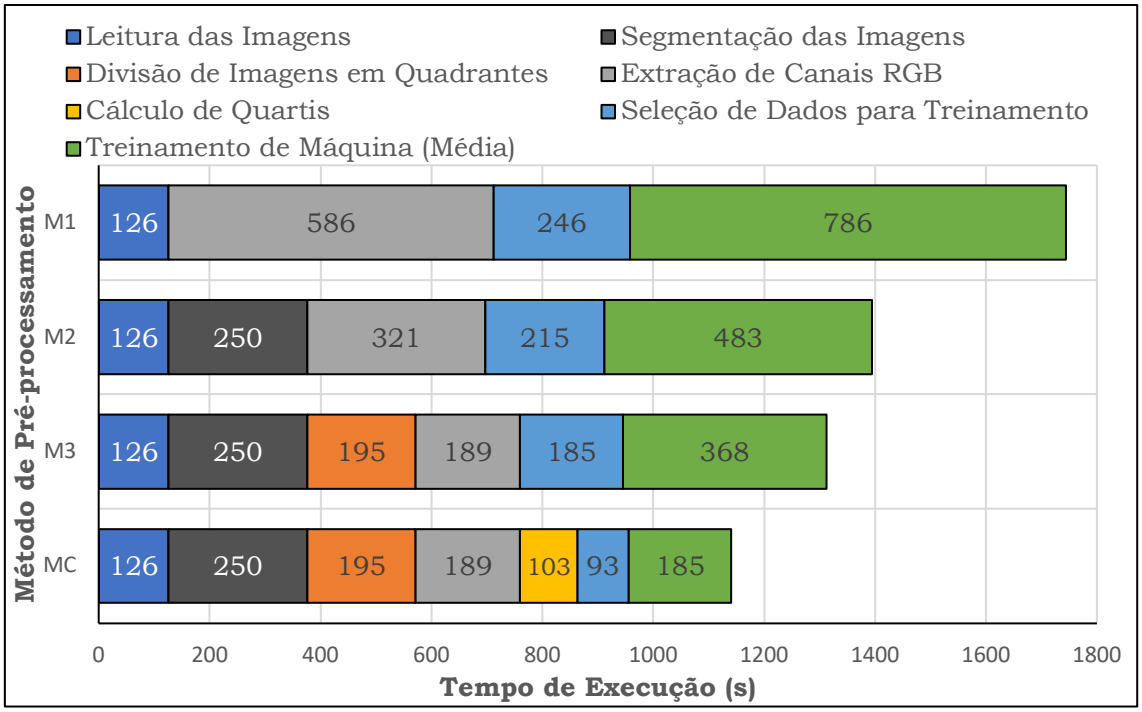

Figura 17 - Tempos gastos nas sub-etapas da fase de pré-processamento e fase de treinamento da máquina de aprendizado computacional para 3500 imagens da base de dados de treinamento. Fonte: Própria.

No decorrer da implementação do método proposto, a etapa de segmentação se destacou como a mais demorada no processo global do algoritmo. Isso se deve ao fato de que a segmentação representa a fase inicial do pré-processamento de dados, na qual as imagens são tratadas sem nenhum tipo de processamento prévio, e o algoritmo é aplicado à imagem como um todo.

No gráfico também é possível observar que o tempo de execução do algoritmo está diretamente associado às técnicas de pré-processamento aplicadas, o método M1 foi executado em 1744 segundos, ao utilizar uma técnica de pré-processamento, enquanto os métodos M2 e M3 foram executados em 1395 e 1313 segundos, utilizando 2 e 3 técnicas de pré-processamento, respectivamente. O método proposto foi executado em 1141 segundos, utilizando 4 técnicas de pré-processamento, o que justifica seu uso para diminuição do tempo de execução do algoritmo.

A Figura 18 demonstra o gráfico de tempo gasto, em milissegundos, para a realização da classificação para uma única imagem de 500x500p a partir dos métodos descritos na Tabela 3 da seção 4.3.

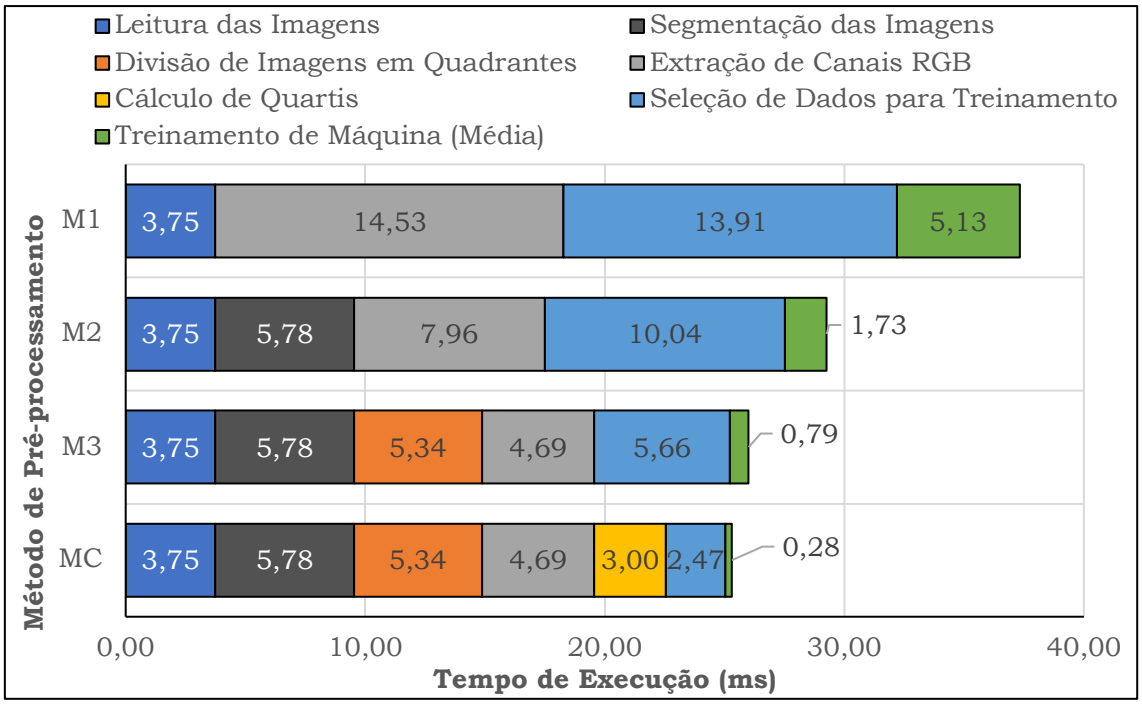

Figura 18 - Tempo gasto nas sub-etapas da fase de pré-processamento e classificação para 1 imagem de teste. Fonte: Própria.

Novamente, no processo de classificação utilizando método proposto, o processo de segmentação das imagens foi o mais longo, seguido da divisão de imagens e extração dos canais RGB. A classificação foi o processo mais curto, com apenas 0,28 milissegundos. O tempo total de execução foi de 25,31 milissegundos. Os métodos M2 e M3 obtiveram resultados similares, tendo utilizado 26,01 e 29,26 milissegundos para execução completa, respectivamente. O método M1 foi executado em 37,32 ms.

5.2 Resultados de Acurácia:

Os valores de acurácia são fundamentais para avaliar a qualidade de um algoritmo de aprendizado de máquina, pois indicam porcentagem de previsões corretas realizadas pelo algoritmo e, portanto, mostram como ele se comporta em relação aos dados de teste. Algoritmos como

Naive Bayes (NV), *Random Forest* (RF) e *KNN* são utilizados com frequência em diferentes aplicações e, portanto, é importante comparar seus desempenhos para escolher o que melhor se adequa a cada caso específico. Considerar valores de acurácia também permite comparar diferentes configurações de parâmetros e métodos, para selecionar aquele que oferece os melhores resultados.

Os valores descritos nessa seção foram obtidos a partir do uso de 2000 imagens para testes. Os testes foram realizados em etapas incrementais, conforme descrito na Tabela 2 da Seção 4.3, e as máquinas de treinamento foram configuradas de acordo com as especificações descritas na Seção 4.4.

A Figura 19 apresenta os resultados de acurácia (eixo y) para os métodos M1, M2, M3 e MC (eixo x) e aplicando os algoritmos de classificação Naive Bayes (NB), KNN e Random Forest (RF) e Perceptron Multicamadas (MLPC), curvas verde, azul, vermelha e amarela, respectivamente.

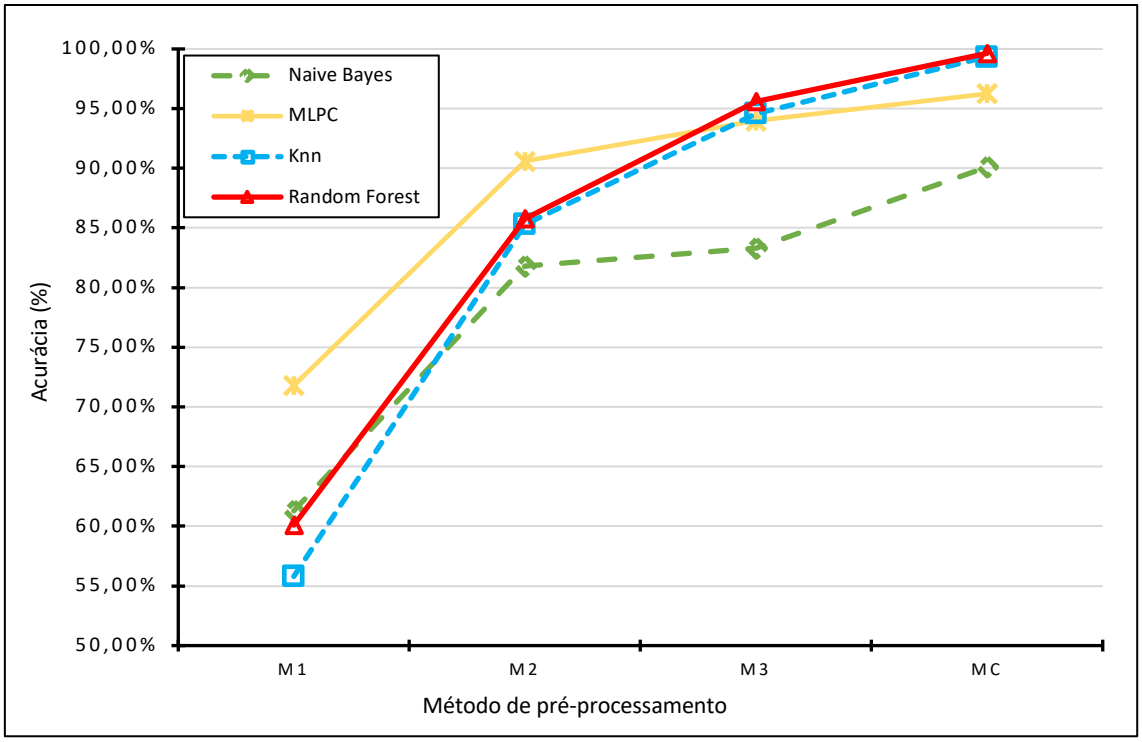

Figura 19 - Resultados de acurácia para diferentes métodos (M1, M2, M3 e MC) e algoritmos de classificação NV, MLPC, KNN e RF. Para os algoritmos Naive Bayes, Perceptron Multicamadas, KNN e Random Forest, os resultados foram, respectivamente: M1 = 61.3%, 71,8%, 55.8%, 60%; M2 = 81.8%, 90,6%, 85.3%, 85.8%; M3 = 83.3%, 93.96%, 94.6%, 95.6%; MC = 90.1%, 96.25%, 99.35%, 99.63%.

Na Figura 19, observa-se que o método M1 (Extração de valores RGB e filtragem de pixels com valores iguais) utilizado em conjunto com os algoritmos de classificação NB, MLPC, KNN e RF gerou valores de acurácia que variam entre 55,8% e 71,3%.

O método M2, onde há extração do primeiro plano das imagens e extração dos valores das componentes RGB gerou resultados melhores que o testes realizados com o método M1, com o qual a acurácia foi entre 81,8% e 90,6%. valores ainda abaixo dos artigos utilizados como referência, como Wu e Zhang (2018) e Almeida e Zhang (2022).

O método M3, onde foi realizado processo de extração do primeiro plano das imagens e divisão em quadrantes, gerou resultados favoráveis, sobretudo ao utilizar o algoritmo Random Forest para classificação, onde a acurácia foi superior a 95%.

No método MC, ao incluir o cálculo de quartis e seleção de dados, o algoritmo gerou resultados superiores a 90% de acurácia para todos os

algoritmos de treinamento, tendo 99,63% como melhor valor de acurácia, para o algoritmo *Random Forest*, com 150 árvores (*n_trees*) e 4 amostras mínimas para divisão do nó interno (*min_samples_split).*

A tabela 4 ilustra os principais resultados, as máquinas utilizadas e as configurações aplicadas.

Tabela 4 - Principais resultados obtidos ao utilizar o método proposto (MC), ordenado pelo valor de acurácia. Fonte: Própria.

Classificador	Configurações da Máquina	Acurácia (%)
Random Forest	N. árvores = 150 Min. Div. Nó = 4 Prof. Máx. = ilimitada	99.63%
Random Forest	N. árvores = (50, 200, 300) Min. Div. Nó = 4 Prof. Máx. = ilimitada	99.5%
Knn	Vizinhos = 1 Dist. Euclidiana = 2	99.34%
Knn	Vizinhos = 3 Dist. Euclidiana = 2	99.16%
Random Forest	N. árvores = 900 Min. Div. Nó = 4 Prof. Máx. = ilimitada	99.0%
MLPC	Tax. de Apr. = Constante Tax. de Regul. = 0.0001 Otimizador = *ADAM* Estado Aleat. = (42, 100, 200) Max. Inter. = 200	96.25%

O melhor resultado obtido, de 99,63% com utilização do método MC, indica que dentre as 2000 imagens testadas, o classificador RF cometeu 10 erros. Dentre estes erros, 3 se referem a imagens de florestas que o algoritmo classificou como incêndios e 7 de incêndios que o algoritmo classificou apenas como florestas.

A Figura 20 exemplifica as imagens onde o algoritmo cometeu erros.

Figura 20 - Exemplo das imagens classificadas erroneamente, contendo incêndio (acima) e contendo apenas florestas (abaixo). Fonte: Adaptada de Desfere (2022).

Nas imagens da parte superior da Figura 20, é possível perceber que os falsos negativos ocorreram devido a extração do primeiro plano da imagem original onde gerou-se como resultado imagens predominantemente verdes e com pequenos tons de amarelo ou cinza, sem a existência de fogo ou fumaça, o que ocasionou erro de classificação. O mesmo ocorre para os falsos positivos, onde o primeiro plano extraído das imagens originais possuía tons predominantemente cinzas.

A divisão por quadrantes auxilia para que esses erros ocorram com menos frequência, uma vez que é possível analisar de maneira mais precisa e isolada cada quadrante da imagem. No entanto, ainda é necessário um maior tratamento para imagens com pouca definição ou com tons que fogem do padrão da base de dados, como imagens contendo flores ou árvores com colorações diferenciadas.

A nível de comparação, é possível mencionar os trabalhos de Wu e Zhang (2018), Jadon e Varshney (2020) e Jiao e Zhang (2019) que, ao serem reproduzidos utilizando a mesma base de dados, *CorsicanFire Dataset (2017),* obtiveram resultados similares ao relatado originalmente, mas com tempo de processamento maior que o proposto, conforme mencionado na Tabela 5.

Tabela 5 - Comparação dos resultados obtidos ao submeter trabalhos disponíveis na literatura a mesma base de dados. Fonte: Própria.

Método	**Tempo de Processamento (s)**	**Tempo de Classificação por imagem (s)**	**Acurácia (%)**
Wu e Zhang (2018)	188,3	0,085	93,33%
Jadon e Varshney (2020)	211,0	0,113	96,33%
Jiao e Zhang (2019)	232,82	0,145	88,4%
Método Proposto	185,1	0,025	99,5%

5.3 Resultados de sensibilidade

A Figura 21, no eixo y, demonstra a evolução dos resultados de sensibilidade para os métodos M1, M2, M3 e MC apresentados no eixo x, ao longo da realização dos testes de aplicação dos algoritmos de classificação *Naive Bayes* (NB), *KNN* e *Random Forest* (RF), curvas verde, azul e vermelha, respectivamente.

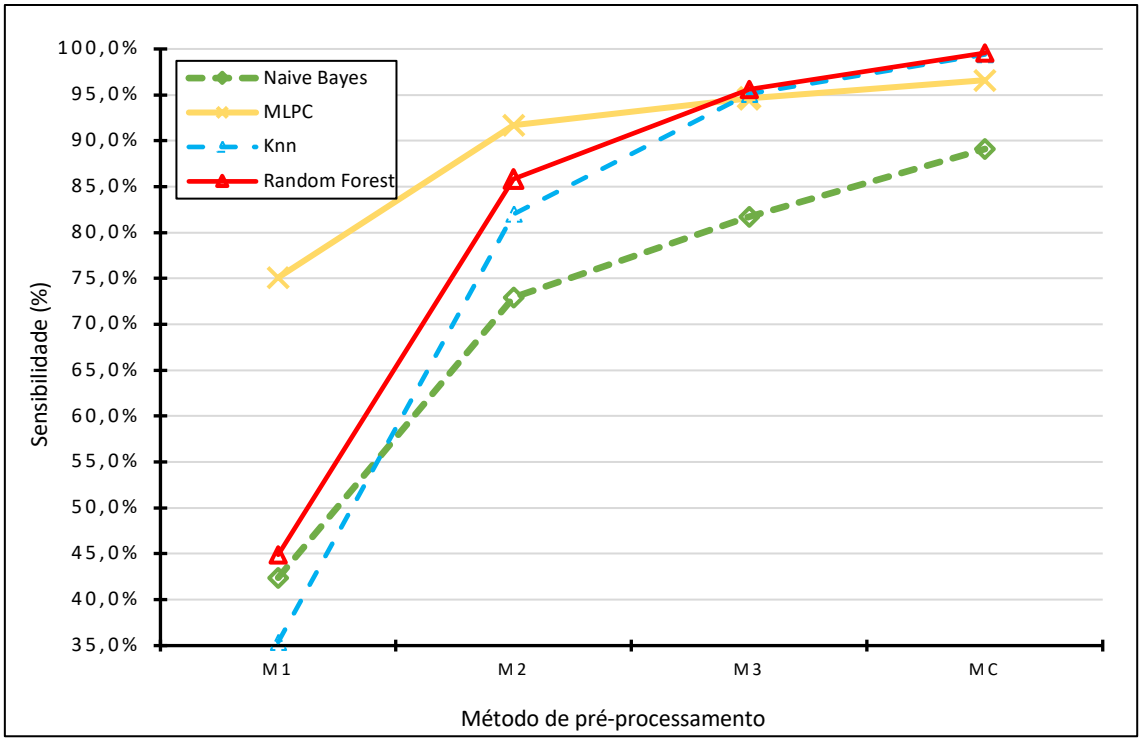

Figura 21 - Resultados de sensibilidade para os métodos propostos M1, M2, M3 e MC (Método Completo). Para os algoritmos Naive Bayes (NB), MLPC, KNN e Random Forest (RF), os resultados foram, respectivamente: M1 = 42.4%, 75.1%, 35.4%, 44.9%; M2 = 72.9%, 91.7%, 82.0%, 85.2; M3 = 81.7%, 94.6%, 95.1%, 95.6%; MC = 89.1%, 96.6%, 99.4%, 99.6%. Fonte: Própria.

Os piores resultados foram obtidos com o uso do método M1, com valores variando entre 35% e 75%. Ao considerar que a taxa de acurácia para esse método é de 55% a 71.8%, é possível afirmar que o método não possui eficácia suficiente para reconhecer padrões de incêndios florestais, resultando em uma alta taxa de falsos-negativos.

O método M2 gerou valores de sensibilidade superiores ao método M1 e compatível com os resultados de acurácia do método M2. Os resultados de sensibilidade variaram entre 72% e 91.7%, demonstrando que todos os modelos de classificação foram beneficiados ao utilizar o método M2, o que resultou em menor quantidade de falsos-negativos.

Com resultados ainda melhores que o método M2, o método M3 gerou resultados de sensibilidade que variaram entre 81% e 95.6%. A divisão da imagem em quadrantes auxiliou na classificação ao excluir partes desnecessárias para o treinamento, o que tornou a identificação de padrões muito mais precisa e simples.

Ao utilizar o método completo (MC), os resultados de sensibilidade foram superiores a 95,9%, tendo como melhor resultado 99.6%, para o classificador *Random Forest*, com 150 árvores. A inclusão do cálculo de quartis dos valores das componentes RGB auxiliou ainda mais na precisão da máquina de aprendizagem, sobretudo na exclusão de dados considerados desnecessário e na formação de padrões de cores para relacionar as fases de teste e treinamento.

É importante considerar os valores de sensibilidade nos testes realizados, pois ele fornece informações valiosas sobre a capacidade do modelo de prever corretamente as classes verdadeiras.

5.4 Resultados de especificidade:

A Figura 22 apresenta os resultados de especificidade, no eixo y, ao longo dos testes realizados utilizando os métodos M1, M2, M3 e MC, no eixo X, com a aplicação dos algoritmos de classificação *Naive Bayes* (NB), MLPC, KNN e *Random Forest* (RF), curvas verde, azul e vermelha, respectivamente.

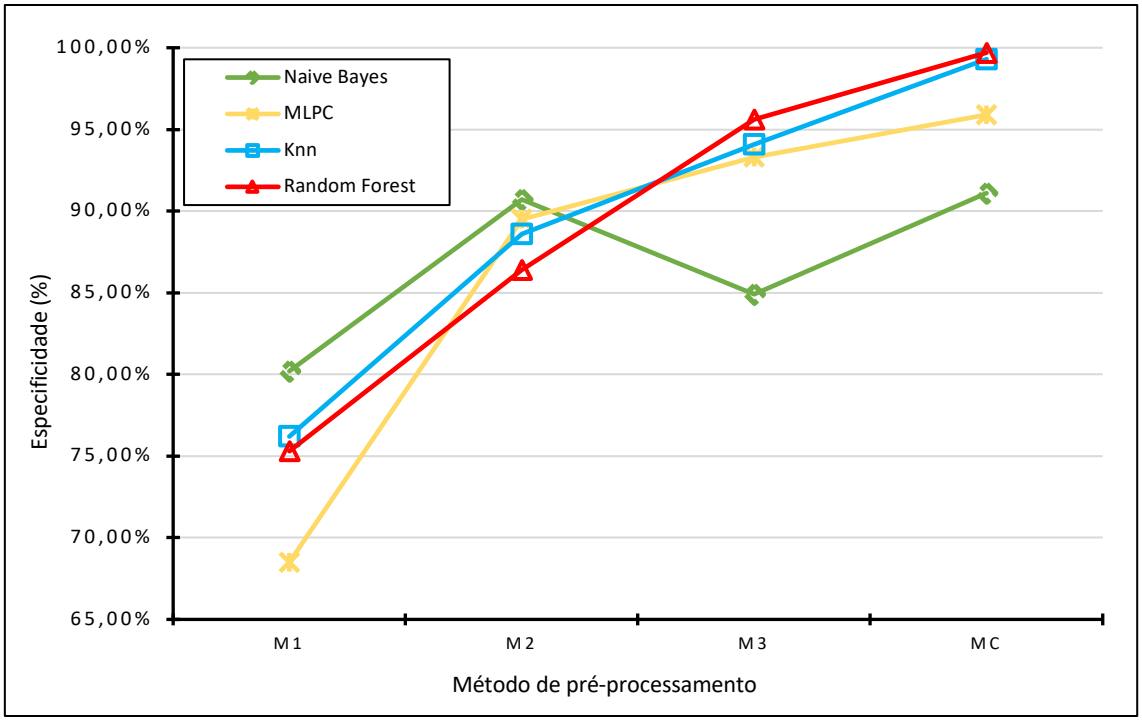

Figura 22 - Resultados de especificidade para os métodos propostos (M1, M2, M3 e MC), utilizando os algoritmos Nave Bayesa (NB), MLPC, KNN e Randon Forest (RF), os resultados foram, respectivamente: M1 = 80.20%, 68.5%, 76.20%, 75.30%; M2 = 90.70%, 89.5%, 88.60%, 86.40%; M3 = 84.9%, 93.3%, 94.1%, 95.6% e MC = 91.1%, 95.9% 99.3%, 99.7%. Fonte: Própria.

Na figura é possível notar que o método M1 gerou valores de especificidade de 80,20%, 68,5%, 76,20%, e 75,30% utilizando os algoritmos NV, MLPC, KNN e RF, respectivamente.

Para o método M2, gerou resultados de especificidade com variação entre 89,5% e 90,70%, mostrando valores superiores quando comparado ao método M1 e, com evolução dos resultados de especificidade similares aos resultados de acurácia. O mesmo ocorre para o método M3, que gerou resultados superiores aos do método M2 com valores entre 84,9% e 95,6%.

Os resultados de especificidade gerados pelo método completo (MC) também foram proporcionais aos resultados de acurácia e sensibilidade, resultando em valores no intervalo entre 91,1% e 99,7%.

É notável que, à medida que incrementamos as etapas, nos métodos M1, M2, M3 e MC, os resultados de especificidade aumentam,

mostrando que os algoritmos tornaram-se mais precisos na identificação de imagens sem incêndio.

É importante considerar os resultados de especificidade em nossa análise, pois eles nos permitem avaliar a capacidade do algoritmo de diferenciar corretamente entre casos negativos e positivos, o que é crucial para aplicações em que é importante evitar falsos positivos.

Dessa forma, é possível observar que o classificador *Random Forest* gerou os melhores resultados na identificação de todas as classes testadas. A combinação da alta sensibilidade e especificidade do algoritmo *Random Forest* o torna mais adequado para a tarefa em questão, mesmo que os algoritmos *Knn* e *MLPC* também tenham tido resultados positivos.

6 Conclusão

Neste trabalho foi proposto método de identificação de incêndios florestais com base em segmentação de imagens e aprendizado de máquina, utilizando dados das componentes RGB para identificação de padrões de intensidade de cores. O método projetado utiliza a análise de quartis na representação dos dados para melhorar a precisão de identificação, auxiliando na eliminação de dados desnecessários para o treinamento.

A segmentação foi um processo importante no método proposto utilizado para extrair o primeiro plano da imagem, potencializando os dados necessários para identificação de padrões. A divisão em quadrantes auxiliou no tratamento de dados, tornando a imagem menor e com identificação mais simples e precisa, bem como a extração das componentes de cor RGB e o cálculo de quartis, a fim de que a imagem pudesse ser traduzida como informações numéricas resumidas, simplificando os dados de entrada para os algoritmos de ML.

Foram realizados experimentos para avaliar o desempenho da proposta e o tempo de processamento para classificação, utilizando diferentes algoritmos de treinamento e métodos incrementais de pré-processamento. Para essa pesquisa, são utilizados três métricas de avaliação: acurácia, sensibilidade e especificidade.

A acurácia média da proposta MC desse trabalho é comparável aos disponíveis na literatura, que utilizam arquiteturas mais complexas, como para Zhang e Qu (2022) e bases de dados maiores, como em Chen e Hopkins (2022), obtendo como melhor resultado o a acurácia de 99,6% com a utilização do algoritmo *Random Forest*. Ao obter taxa de sensibilidade de 98,6%, é possível afirmar que o método MC proposto foi capaz de identificar incêndios florestais com alta precisão e baixo índice de falsos negativos.

Ao comparar os resultados entre imagens sem pré-processamento

(métodos M1, M2 e M3) e com a proposta do método (MC) é possível afirmar que a combinação entre a segmentação de imagens, divisão por quadrantes e extração de valores RGB foi bem-sucedida, já que a inserção incremental desses métodos mostrou alta evolução nos resultados obtidos. Além disso, por se tratarem de processos simples e rápidos, tais técnicas podem ser aplicadas a diferentes bases de dados com variações de luminosidade, locais de coleta e posicionamento de câmera.

Para trabalhos futuros, propõem-se a adaptação do método para o tratamento de dados extraídos de vídeos, respectiva análise do tempo de processamento, comparação com os resultados de imagens mais nítidas e frames de vídeos. Também se propõe o uso de algoritmos mais complexos de treinamento e classificação, advindos de técnicas de *deep learning,* a fim de aumentar a precisão e diminuir o tempo de classificação.

7 Referências

01 ZASLAVSKY, A.; PERERA, C.; GEORGAKOPOULOS, D. Sensing as a service and big data. arXiv 2013, arXiv:1301.0159.

02 BRANDÃO, A.; PEREIRA, E.; ESTEVES, M.; PORTELA, F.; SANTOS, M. F.; ABELHA, A.; MACHADO, J. A benchmarking analysis of open-source business intelligence tools in healthcare environments. Information, 2016, 7, 57.

03 REUTERS. "Desastres causam prejuízo de US$ 210 bi em 2020, dizem seguradoras". Exame, 7 de janeiro de 2021. Disponível em: https://exame.com/esg/desastres-causam-prejuizo-de-us-210-bi-em-2020-dizem-seguradoras/. Acesso em: 30 de jan. de 2023.

04 GIGLIO, L.; BOSCHETTI, L.; ROY, D. P.; HUMBER, M. L.; JUSTICE, C. O. The Collection 6 MODIS burned area mapping algorithm and product. Remote Sensing of Environment, 2018, 217, 72-85.

05 CHEN, X. et al. Wildland Fire Detection and Monitoring Using a Drone-Collected RGB/IR Image Dataset. IEEE Access, 2022, 10, 121301-121317. doi: 10.1109/ACCESS.2022.3222805.

06 ZHANG, S.; QU, N.; ZHENG, T.; HU, C. Series Arc Fault Detection Based on Wavelet Compression Reconstruction Data Enhancement and Deep Residual Network. IEEE Transactions on Instrumentation and Measurement, 2022, 71, 1-9. doi: 10.1109/TIM.2022.3158990.

07 KHRASHCHEV, V.; LARIONOV, R. Wildfire Segmentation on Satellite Images using Deep Learning. 2020 Moscow Workshop on Electronic and Networking Technologies (MWENT), Moscow, Russia, 2020, pp. 1-5. doi: 10.1109/MWENT47943.2020.9067475.

08 JADON, A.; VARSHNEY, A.; ANSARI, M. Low-Complexity High-Performance Deep Learning Model for Real-Time Low-Cost Embedded Fire Detection Systems. Procedia Computer Science, 2020, 171, 418-426. doi: 10.1016/j.procs.2020.04.044.

09 PREMAL, C. E.; VINSLEY, S. S. Image processing based forest fire detection using YCbCr colour model. 2014 International Conference on

Circuits, Power and Computing Technologies [ICCPCT-2014], Nagercoil, India, 2014, pp. 1229-1237. doi: 10.1109/ICCPCT.2014.7054883.

10 CHAKI, J.; DEY, N. A Beginner's Guide to Image Preprocessing Techniques. CRC Press, 2018. 114 páginas. ISBN: 9780429805110.

11 BOYKOV, Y. Y.; JOLLY, M. P. Interactive graph cuts for optimal boundary region segmentation of objects in n-d images. In: IEEE International Conference on Computer Vision, 2001, Vancouver, BC: Proceedings... IEEE, 2001, v. 1, p. 105-112.

12 ROTHER, C.; KOLMOGOROV, V.; BLAKE, A. "GrabCut": Interactive Foreground Extraction Using Iterated Graph Cuts. ACM Transactions on Graphics (TOG), 2004, v. 23, n. 3, p. 309-314.

13 GASPARI, Tiago De. Desenvolvimento de um m√©todo semiautomático para geração de ground truths de vídeos. São José do Rio Preto, 2015. 78 f. il., tabs.

14 KAKUMANU, P.; MAKROGIANNIS, S.; BOURBAKIS, N. A survey of skin-color modeling and detection methods. Pattern recognition, 2007, 40(3), 1106-1122.

15 CHAVES-GONZÁLEZ, J. M.; VEGA-RODRIGUEZ, M. A.; GÍMEZ-PULIDO, J. A.; SÃNCHEZ-PÉREZ, J. M. Detecting skin in face recognition systems: A colour spaces study. Digital Signal Processing, 2010, 20(3), 806-823.

16 FEITOSA, R. D. F. Modelos matemáticos para redução do espectro provável e detecção de tons de pele humana em imagens coloridas representadas nos espaços de cores RGB e HSV. Universidade Federal de Goiânia, Goiânia, 2015. 110 f. Dissertação (Mestrado em Ciência da Computação

17 JOARDER A. H. E FIROZZAMAN M. Quartiles for Discrete Data. Teaching Statistics, 2001, 23(3).

18 ZHANG, H. "The Optimality of Naive Bayes". FLAIRS 2004 conference. Disponível online em <http://www.cs.unb.ca/profs/hzhang/publications/FLAIRS04ZhangH.pdf>. Acesso em: 20 abr. 2020.

19 MARQUES, R. L.; DUTRA, I. Redes Bayesianas: o que s√£o, para que servem, algoritmos e exemplos de aplicações. Maio de 1999.

20 BISHOP, C. M. Pattern Recognition And Machine Learning. Springer Science+Business Media, LLC, 2006.

21 COVER, T. M. e HART, P. E. Nearest Neighbor Pattern Classification. Ieee Transactions On Information Theory, 1967, 13(1).

22 LANTZ, B. Machine Learning with R. Packt Publishing Ltd., 2¬™ edi√ß√£o, 2015.

23 BREIMAN, L. Random Forests. Machine Learning, 2001, 45, 5-32. doi: 10.1023/A:1010933404324.

24 AMIT Y, GEMAN D. Shape quantization and recognition with randomized trees. Neural Comput, 1997, 9, 1545-1588.

25 SILVA, I. N.; SPATTI, D. H.; FLAUZINO, R. A. Redes neurais artificiais para engenharia e ciências aplicadas: curso prático. São Paulo, SP: Artliber Editora Ltda., 2010.

26 HAYKIN, Simon S. Neural Networks and Learning Machines, Volume 10. Prentice Hall, 2009. 906 páginas.

27 CHEN, Y.; MA, Y.; KIM, D. H.; PARK, S. -K. Region-Based Object Recognition by Color Segmentation Using a Simplified PCNN. IEEE Transactions on Neural Networks and Learning Systems, vol. 26, no. 8, pp. 1682-1697, Aug. 2015. doi: 10.1109/TNNLS.2014.2351418.

28 JIAO, Z.; ZHANG, Y.; XIN, J.; YI, Y.; LIU, D.; LIU, H. Forest Fire Detection with Color Features and Wavelet Analysis Based on Aerial Imagery. 2018 Chinese Automation Congress (CAC), Xi'an, China, 2018, pp. 2206-2211. doi: 10.1109/CAC.2018.8623473.

29 WU, H.; LI, H.; SHAMSHOSHOARA, A.; RAZI, A.; AFHGAH, F. Transfer Learning for Wildfire Identification in UAV Imagery. 2020 54th Annual Conference on Information Sciences and Systems (CISS), Princeton, NJ, USA, 2020, pp. 1-6. doi: 10.1109/CISS48834.2020.1570617429.

30 WU, S.; ZHANG, L. Using Popular Object Detection Methods for Real Time Forest Fire Detection. 2018 11th International Symposium on Computational Intelligence and Design (ISCID), Hangzhou, China,

2018, pp. 280-284. doi: 10.1109/ISCID.2018.00070.

31 ALMEIDA, J. S.; HUANG, C.; NOGUEIRA, F. G.; BHATIA, S.; DE ALBUQUERQUE, V. H. C. EdgeFireSmoke: A Novel Lightweight CNN Model for Real-Time Video Fire–Smoke Detection. IEEE Transactions on Industrial Informatics, vol. 18, no. 11, pp. 7889-7898, Nov. 2022. doi: 10.1109/TII.2021.3138752.

32 BAKRI, N. S.; ADNAN, R.; SAMAD, A. M.; RUSLAN, F. A methodology for fire detection using colour pixel classification. 2018 IEEE 14th International Colloquium on Signal Processing & Its Applications (CSPA), Penang, Malaysia, 2018, pp. 94-98. doi: 10.1109/CSPA.2018.8368692.

33 TOULOUSE, T. et al. Computer vision for wildfire research: An evolving image dataset for processing and analysis. Fire Safety Journal, 2017, 92, pp.188-194. doi: 10.1016/j.firesaf.2017.06.012.

34 KHAN, A. et al. DeepFire: A Novel Dataset and Deep Transfer Learning Benchmark for Forest Fire Detection. Mobile Information Systems, vol. 2022, Article ID 5358359, 14 pages, 2022. doi: 10.1155/2022/5358359.

35 SHAMSOSHOARA, A. et al. The FLAME dataset: Aerial Imagery Pile burn detection using drones (UAVs). IEEE Dataport, November 19, 2020. doi: https://dx.doi.org/10.21227/qad6-r683.

36 KAGGLE. Forest Fire Images. Disponível em: https://www.kaggle.com/datasets/mohnishsaiprasad/forest-fire-images. Acesso em: 9 de outubro de 2023 às 19:26 horas.

37 PARK, Minsoo et al. Advanced wildfire detection using generative adversarial network-based augmented datasets and weakly supervised object localization. International Journal of Applied Earth Observation and Geoinformation, v. 114, p. 103052, 2022. ISSN 1569-8432. Disponível em: https://doi.org/10.1016/j.jag.2022.103052.

8 ANEXO

```
import os
import numpy as np
import cv2
from skimage.io import imread_collection, imsave
from tabulate import tabulate
import csv
import matplotlib.pyplot as plt
import itertools

def load_images(directory, file_spec='*.jpg'):
    load_pattern = os.path.join(directory, file_spec)
    return imread_collection(load_pattern)

def  grabcut_and_save_images(image_collection,  output_directory,
class_label):
    for x, image in enumerate(image_collection):
        mask = np.zeros(image.shape[:2], np.uint8)
        backgroundModel = np.zeros((1, 65), np.float64)
        foregroundModel = np.zeros((1, 65), np.float64)
        rectangle = (20, 100, 150, 150)

        cv2.grabCut(image, mask, rectangle,
                    backgroundModel, foregroundModel,
                    3, cv2.GC_INIT_WITH_RECT)

        mask2  =  np.where((mask  ==  2)  |  (mask  ==  0),  0,
1).astype('uint8')
        image = image * mask2[:, :, np.newaxis]

        filename = f"{class_label}_{x}.png"
        save_path = os.path.join(output_directory, filename)
        imsave(save_path, image)
        print(f"Saved: {save_path}")

def                load_and_process_images(input_directory_fire,
input_directory_nofire):
    ic_fire = load_images(input_directory_fire)
    ic_nofire = load_images(input_directory_nofire)
    return ic_fire, ic_nofire

def quantile_features(image_collection, class_label):
    quartil_data = []

    for x, image in enumerate(image_collection):
        for i in range(4):
            for j in range(4):
                recorte = image[i * 62:(i + 1) * 62, j * 62:(j +
1) * 62]
                canais = cv2.split(recorte)

                b = np.quantile(canais[0], [0.25, 0.5, 0.75, 1])
                g = np.quantile(canais[1], [0.25, 0.5, 0.75, 1])
                r = np.quantile(canais[2], [0.25, 0.5, 0.75, 1])

                quartil = [
                    [x, f'{i + 1}-{j + 1}', b[0], b[1], b[2],
b[3], g[0], g[1], g[2], g[3], r[0], r[1], r[2], r[3], class_label]
```

```
                ]
                quartil_data.extend(quartil)

    return quartil_data

def save_quantile_data_to_csv(data, output_filename):
    fieldnames = ['Imagem', 'Chanel', 'Blue', 'Blue', 'Blue',
'Blue', 'Green', 'Green', 'Green', 'Green', 'Red', 'Red', 'Red',
'Red', 'Fire']

    with open(output_filename, 'w', encoding='UTF8', newline='')
as f:
        writer = csv.writer(f)
        writer.writerow(fieldnames)
        writer.writerows(data)

def load_quantile_data_from_csv(filename):
    return pd.read_csv(filename)

def preprocess_quantile_data(data):
    data2 = []
    for row in data:
        if (row[2] == row[6] == row[10]) and (row[3] == row[7] ==
row[11]) and (row[4] == row[8] == row[12]) and (row[5] == row[9]
== row[13]):
            print("Skipping", row[0], "Quadrant", row[1])
        else:
            data2.append(row)
    return data2

def       train_knn_classifier(data,       target_column='Fire',
n_neighbors=3):
    from sklearn.model_selection import train_test_split
    from sklearn.neighbors import KNeighborsClassifier

    X_train,       X_test,       y_train,       y_test       =
train_test_split(data.iloc[:,    2:-1],    data[target_column],
test_size=0.20, random_state=1)
    knn = KNeighborsClassifier(n_neighbors=n_neighbors)
    knn.fit(X_train, y_train)

    return knn, X_test, y_test

def evaluate_classifier(classifier, X_test, y_test):
    from sklearn.metrics import accuracy_score, confusion_matrix,
plot_confusion_matrix

    y_pred = classifier.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    confusion = confusion_matrix(y_test, y_pred)

    print("Accuracy:", accuracy)
    print("Confusion Matrix:")
    print(confusion)

    plt
```