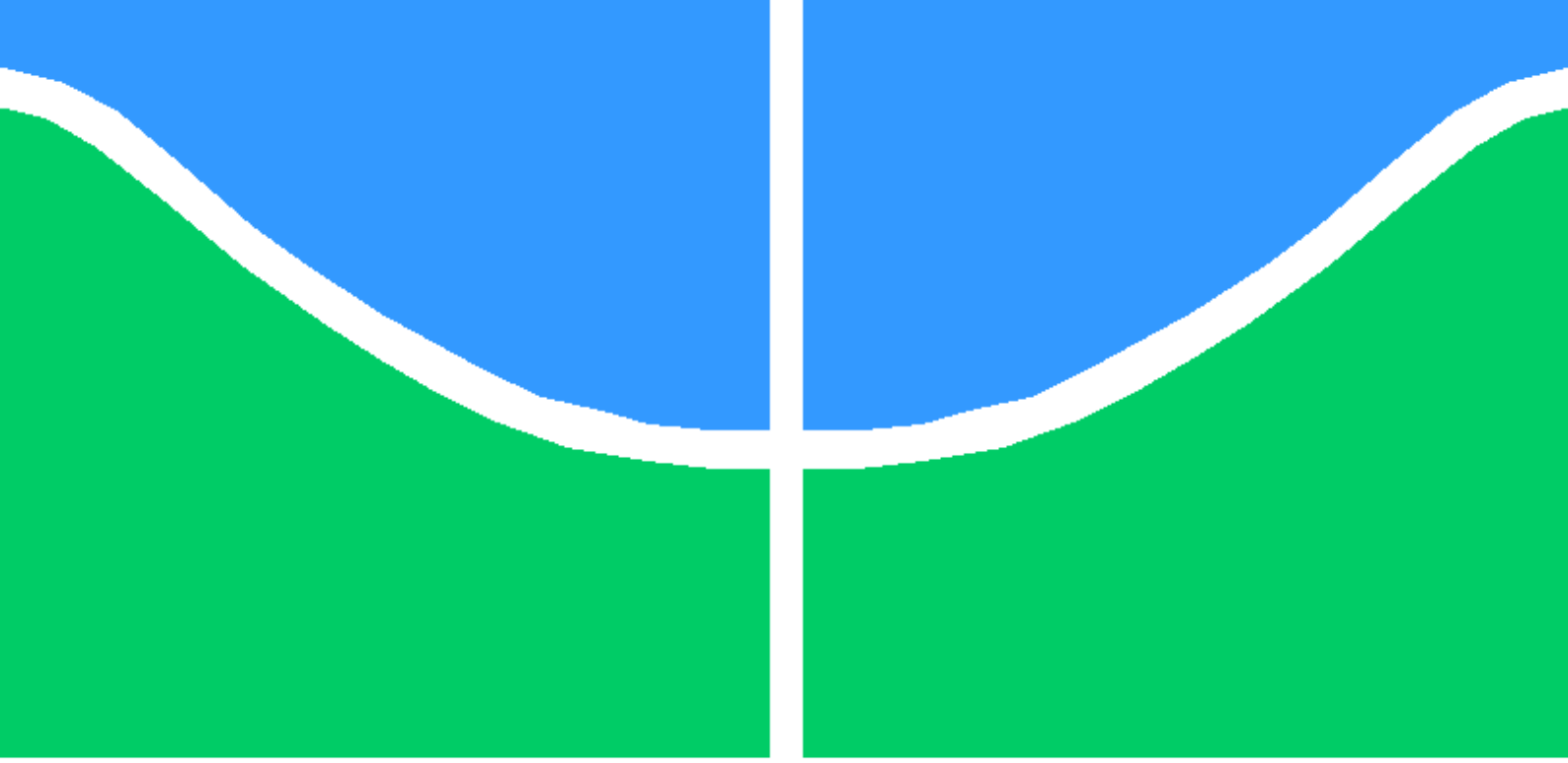

Trabalho Final de Graduação

Tipificação de Incêndios Florestais na Amazônia Legal através de Aprendizado de Máquina

Bruno Scholles Soares Dias

Brasília, Novembro de 2023

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASILIA
Faculdade de Tecnologia

Trabalho Final de Graduação

Tipificação de Incêndios Florestais na Amazônia Legal através de Aprendizado de Máquina

Bruno Scholles Soares Dias

Relatório submetido ao Departamento de Engenharia
Elétrica como requisito parcial para obtenção
do grau de Engenheiro de Redes de Comunicação

Banca Examinadora

Prof. D.Sc. Edson Mintsu Hung, ENE/UnB _____
Orientador

Profª. D.Sc. Mylene C. Q. de Farias, ENE/UnB _____
Examinadora Interna

D.Sc. Henrique Bernini, CENSIPAM _____
Examinador Externo

Agradecimentos

Com o término deste ciclo repleto de aprendizados, experiências e sentimentos intensos, expresso minha sincera gratidão à minha família, especialmente aos meus amados pais, Divina Soares e Marcelo Dias, minha avó Neri Dias e minha tia Márcia Dias, pelo apoio incondicional ao longo dessa jornada de crescimento pessoal e acadêmico. A eles, juntam-se meus calorosos agradecimentos à minha amada companheira, Beatriz Versiani, cuja presença constante e encorajadora foi um verdadeiro pilar durante minha trajetória universitária, enfrentando junto comigo os desafios, celebrando as conquistas e sempre me respaldando em cada decisão que tomei. Também estendo meus votos sinceros de que, em alguns anos, meu irmão, Caio Dias, alcance um brilhante sucesso em sua própria jornada acadêmica, seja pela conceituada Universidade de Brasília ou por qualquer outra renomada instituição de ensino.

Com imensa gratidão, agradeço a todos os amigos que fizeram parte desta jornada, compartilhando momentos de fraternidade e auxílio que jamais esquecerei. Especialmente, quero agradecer de coração a Julia Jamile, João Queiroz, Daniel Prado e Geovana de Melo pela sólida amizade construída durante esse período acadêmico! Meu reconhecimento também se estende aos amigos que trabalharam ao meu lado no desenvolvimento deste trabalho, com uma menção especial ao Ian Porto, Matheus Virgílio e Caio Saigg. Não posso deixar de agradecer aos meus amigos de longa data, cujo apoio e companheirismo foram inestimáveis em momentos importantes da minha vida; a Thaís Cardoso, Carolina Maia, Luís Felipe, Débora Sobreira e Gianvitor Magalhães minha gratidão eterna por sua presença constante ao meu lado.

Gostaria de expressar minha sincera gratidão a todos que compartilharam seus ensinamentos comigo. Agradeço especialmente ao meu orientador, Edson Mintsu, pela orientação fundamental neste trabalho. A contribuição inestimável do meu colega Nilson Donizete também foi essencial, proporcionando orientações valiosas ao longo do processo. Além disso, reconheço a minha admirável professora, Mylene Farias, cujos ensinamentos e sabedoria têm sido uma fonte contínua de inspiração em diversos projetos acadêmicos. Sou eternamente grato por todo o suporte e conhecimento oferecidos, pois sei que essas experiências moldarão significativamente meu caminho acadêmico e profissional.

Por fim, agradeço o apoio técnico e computacional do LISA - Laboratório de Imagens, Sinais e Acústica, da Universidade de Brasília, que conta com apoio do CNPq - Conselho Nacional de Pesquisa, da CAPES - Coordenação de Aperfeiçoamento do Pessoal de Nível Superior, da FAP-DF - Fundação de Amparo à Pesquisa do Distrito Federal e da FINATEC - Fundação de Empreendimentos Científicos e Tecnológicos. Além disso, desejo estender meus agradecimentos especiais aos laboratórios GPDS - Grupo de Processamento Digital de Sinais e ao UIoT - UnB Internet of Things, os quais também contam com os apoios mencionados acima e que desempenharam um papel fundamental na moldagem do meu conhecimento científico ao longo desta jornada.

Bruno Scholles Soares Dias

Resumo

Este estudo realiza uma abordagem abrangente da aplicação de técnicas de aprendizado de máquina para detectar eventos de incêndio na região da Amazônia Legal, em colaboração direta com o Centro Gestor e Operacional do Sistema de Proteção da Amazônia (CENSIPAM), visando integrar o trabalho realizado à plataforma de monitoramento de fogo, Painel do Fogo. Utilizando um conjunto inicial de dados rotulados pelo CENSIPAM, revelou-se a insuficiência de características para uma classificação precisa via algoritmos de aprendizado de máquina. A expansão do conjunto de dados foi alcançada por meio de revisão metodológica, destacando o GFED Amazon Dashboard, e consultas colaborativas com o CENSIPAM para identificar recursos essenciais à categorização precisa. Avaliando dois algoritmos, Random Forest e Multi-Layer Perceptron, os resultados demonstram acurácia geral em torno de 77%, apesar das classes "Sub-bosque" e "Desflorestamento" desafiarem a eficácia dos modelos, sendo a classe "Desflorestamento" de interesse primordial para o CENSIPAM. O estudo enfatiza a qualidade dos dados como fundamental, recomendando abordagem metodológica uniforme e correção manual para aprimorar a precisão dos modelos, representando um avanço para o monitoramento de incêndios na Amazônia Legal e destacando a constante necessidade de refinamento dos modelos e dados de entrada.

Palavras chave:

Sensoriamento remoto, classificação de queimadas, floresta amazônica, aprendizado de máquina, aprendizado profundo.

Abstract

This study presents a comprehensive approach to applying machine learning techniques for detecting fire events in the Legal Amazon region, in direct collaboration with the Amazon Protection System Management and Operations Center (CENSIPAM), aiming to integrate the work with the fire monitoring platform, Fire Panel. Utilizing an initial dataset labeled by CENSIPAM, the insufficiency of features for accurate classification through machine learning algorithms was revealed. Dataset expansion was achieved through methodological review, highlighting the GFED Amazon Dashboard, and collaborative consultations with CENSIPAM to identify essential resources for precise categorization. Evaluating two algorithms, Random Forest and Multi-Layer Perceptron, the results demonstrate an overall accuracy of around 77%, despite the challenges posed by the "Sub-bosque" and "Deforestation" classes to the model effectiveness, with the "Deforestation" class being of primary interest to CENSIPAM. The study emphasizes data quality as fundamental, recommending a uniform methodological approach and manual correction to enhance model accuracy, representing progress for fire monitoring in the Legal Amazon and underscoring the constant need for model and input data refinement.

Keywords:

Remote sensing, fire classification, Amazon forest, machine learning, deep learning.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE ABREVIATURAS

Acrônimos

CENSIPAM	Centro Gestor e Operacional do Sistema de Proteção da Amazônia
INPE	Instituto Nacional de Pesquisas Espaciais
PRODES	Programa de Desmatamento
DETER	Detecção de Áreas Degradadas em Tempo Real
GEE	Google Earth Engine
IA	Inteligência artificial
MLP	Multi-Layer Perceptron
GFED	Global Fire Emissions Database
FIRMS	Fire Information for Resource Management System
CAR	Cadastro Ambiental Rural

1 Introdução

O território abrangido pela Amazônia Legal, que totaliza 5.217.423 km^2, engloba oito estados brasileiros e representa significativos 58.9% do território nacional, estendendo-se por Acre, Amapá, Amazonas, Mato Grosso, Pará, Rondônia, Roraima, Tocantins e parte do Maranhão, além de ostentar 67% das florestas tropicais do planeta, conforme dados do IBGE [20].

A densa cobertura de floresta tropical na Amazônia Legal, historicamente, teve sua dinâmica influenciada por incêndios, cuja relevância remonta à era do Pleistoceno, quando as florestas eram relegadas a pequenas áreas em meio a pastagens [39]. A chegada dos primeiros habitantes e os incêndios resultantes retardaram o processo de recolonização das pastagens pelas florestas [5].

Anualmente, incêndios florestais consomem milhares de quilômetros de florestas amazônicas, gerando impactos socioeconômicos e ecológicos abrangentes. Esses impactos incluem a redução da biomassa florestal, mudanças na composição das espécies arbóreas, esgotamento do solo, perda de biodiversidade e prejuízos econômicos, além de efeitos indiretos associados [35].

A prática do uso do fogo varia em função de fatores biofísicos e econômicos. Em áreas de agricultura intensiva, a limpeza de terrenos é predominantemente conduzida por maquinaria pesada, enquanto em regiões de vegetação densa, a utilização indiscriminada do fogo se torna uma alternativa mais econômica para a limpeza de terrenos [31].

A classificação de incêndios florestais revela-se estrategicamente crucial para otimizar o desempenho das brigadas de incêndio na região amazônica. A prática cultural de queimadas, amplamente difundida no Brasil, está associada a métodos tradicionais de desmatamento para estabelecimento e manutenção de pastagens e terras agrícolas [28]. A diferenciação entre áreas de incêndios e aquelas recentemente exploradas (desmatadas) e áreas de manutenção de pastagens é essencial para orientar decisões durante o acionamento de equipes, considerando a disponibilidade de biomassa para queima, impactando diretamente na intensidade do fogo. Dada a possibilidade de o desmatamento estar vinculado a atividades ilegais, essa distinção é crucial para garantir a segurança das equipes de combate.

Plataformas como o Global Fire Data [2] e o "Painel do Fogo"desenvolvido pelo Centro Gestor e Operacional do Sistema de Proteção da Amazônia (CENSIPAM) em colaboração com corpos de bombeiros [7] têm contribuído para o monitoramento e acionamento de equipes em tempo quase real. Enquanto o primeiro define tipos de incêndios com base em taxas de desflorestamento, o segundo foi validado pelo Corpo de Bombeiros de Rondônia [12].

Adicionalmente, o Instituto Nacional de Pesquisas Espaciais (INPE) oferece plataformas como o Programa de Desmatamento (PRODES) para análise temporal mais extensa e o Detecção de Áreas Degradadas em Tempo Real (DETER) que emite alertas diários de desmatamento [21]. Embora essas ferramentas se concentrem em desflorestamento geral, não têm uma abordagem específica para queimadas.

Diante desse cenário, o presente trabalho, em colaboração com o CENSIPAM, busca desenvolver um sistema de previsão para classificação automatizada de incêndios na floresta amazônica. Esta iniciativa nasce da necessidade urgente do CENSIPAM em automatizar a classificação de queimadas e aplicar na plataforma Painel do Fogo, uma vez que, até o momento, essa tarefa é predominantemente realizada manualmente. Esse avanço representará uma contribuição significativa para o monitoramento eficaz e a resposta rápida diante dos desafios relacionados aos incêndios na região.

O presente estudo apresenta uma proposta de sistema automatizado de tipificação, fundamentado em aprendizado de máquina, utilizando um *dataset* fornecido pelo CENSIPAM. Para enriquecer o conjunto de dados, foram incorporadas informações relevantes provenientes de revisões em outros trabalhos, contribuindo para a caracterização mais abrangente dos diversos tipos de incêndios na região da Amazônia Legal. A construção desse banco de dados robusto impulsiona a aplicação de dois algoritmos de aprendizado de máquina, Random Forest e MLP. Dessa forma, busca-se não apenas demonstrar o desempenho e a viabilidade dessas técnicas, mas também destacar os desafios associados à implementação, utilizando estatísticas amplamente reconhecidas na comunidade científica como parâmetros de avaliação.

2 Revisão Literária

O monitoramento da floresta amazônica tem sido um tema importante de pesquisa nos últimos anos, devido a sua importância ecológica e econômica. O uso de tecnologias de monitoramento, como imagens de satélite, é uma ferramenta valiosa para entender a dinâmica da floresta e para tomar medidas para protegê-la. Assim, muitas obras abordam a detecção de incêndios florestais, além dos citados no capítulo da Introdução, especialmente utilizando a aprendizagem de máquina, todavia, a literatura carece de estudos para o bioma amazônico.

A principal referência para este trabalho é a do *Global Fire Emissions Database* (GFED) [2], que consiste em uma base de dados internacional fornecendo estimativas de emissões de gases de efeito estufa e partículas atmosféricas relacionadas a incêndios florestais. O GFED é amplamente utilizado como uma fonte autoritativa para avaliar a contribuição dos incêndios florestais para a mudança global do clima e a qualidade do ar. A classificação dos tipos de fogo é realizada em quatro categorias principais, incluindo savana/pastagem, pequenas clareiras, sub-bosque, e desmatamento. A classificação se baseia nas características da vegetação e condições climáticas que afetam o comportamento do fogo.

Seguindo, temos [26], no qual realizou um trabalho centrado no mapeamento de cicatrizes de queimadas na Amazônia a partir do Modelo Linear de Mistura Espectral de imagens de sensores MODIS (MOD09). Dessa forma, estudo foi baseado no DETER [21] utilizando um algoritmo de classificação não supervisionado com base na região. Os resultados mostraram que cerca de 50.000 km^2 da superfície amazônica foram queimados e que os dados diários do sensor MODIS são uma importante fonte de informação para o mapeamento de áreas queimadas.

Seguindo, temos um trabalho feito no Cerrado [36], no qual visava desenvolver um algoritmo automático para mapear áreas queimadas no Cerrado utilizando sensores orbitais. Foi realizada uma análise de índices espectrais e depois desenvolvido um algoritmo automático para mapear áreas queimadas no Landsat 8. Foram avaliados três núcleos e diferentes combinações dos parâmetros SVM-OC para verificar quais são os mais apropriados para mapear áreas queimadas. O núcleo radial mostrou a maior precisão, com um índice *kappa* (conhecido como *Kappa* de *Cohen*) de 0,98. Os resultados mostraram que 13% da área queimada cartografada eram cicatrizes sem foco ativo.

Temos também [12], que desenvolveu um indicador quase em tempo real da gravidade do fogo. Utiliza a informação dos satélites NOAA-20 e Suomi NPP para monitorar eventos de incêndio com base no nível de atenção que requerem e em quatro critérios: dimensão global, duração, extensão e intensidade. A extensão espacial deste indicador inclui os biomas da Amazônia brasileira e do Pantanal, e foi demonstrada num estudo de caso real voando com bombeiros a sudeste de Porto Velho, Brasil. Embora semelhante ao atual trabalho, o autor classificou os incêndios associados a áreas recém-colhidas, que são susceptíveis de ocorrer em áreas classificadas como vegetação nativa, bem como em áreas utilizadas por seres humanos, dependendo da estação em que os dados de cobertura e utilização são classificados e da época dos incêndios, enquanto o nosso trabalho foi

desenvolvido em cima de características similares, mas para tipificar os eventos de fogo ocorridos e não seus graus de severidade.

Também podemos contar com pesquisas feitas para o bioma do Cerrado. Os trabalhos de hoje em tal bioma são altamente precisos no mapeamento de cicatrizes de queimadas, conforme a pesquisa da [45] desenvolveu um método semiautomático para mapear áreas de queimadas no Cerrado, usando imagens *Landsat* e algoritmos de *Deep Learning* nas plataformas do *Google Earth Engine* (GEE) e *Google Cloud Storage*. Para a validação de 2.000 pontos, obteve-se uma acurácia de 97% e ao mapear as queimadas na área do Cerrado, acredita-se que 41% da área tenha sido queimada em algum momento. Este estudo mostra o potencial do uso de dados de sensoriamento remoto de séries temporais Landsat.

Por fim, temos o trabalho de Faria et al. [11], que propuseram uma abordagem de classificação de tipos de incêndios com propriedades extrínsecas como *Global Fire Emissions Database* (GFED), refletindo o foco de nossa pesquisa em andamento. Eles classificaram ocorrências de incêndio em categorias como desmatamento recente, utilizando o sistema DETER para mesclar incêndios ativos com áreas desmatadas. Suas descobertas revelaram que aproximadamente 21% dos eventos anuais de incêndio no território legal da Amazônia brasileira estavam relacionados ao desmatamento. Além disso, observaram que a estação seca influenciava predominantemente o período de ignição, ocorrendo tipicamente dois meses após o desmatamento, com o pico de ocorrências em agosto e setembro. No entanto, a metodologia deles tinha limitações, principalmente na diferenciação entre pequenas clareiras e grandes áreas queimadas, destacando a necessidade de uma classificação mais refinada dos eventos de incêndio.

Portanto, através da revisão literária, tais estudos sugerem que o uso de tecnologias de monitoramento, incluindo imagens de satélite e modelos de aprendizado de máquina, podem ser ferramentas valiosas para entender a dinâmica da floresta amazônica e para tomar medidas para protegê-la.

3 Referencial Teórico

Neste estudo, a análise de dados satelitais é empregada para monitorar a ocorrência de eventos de fogo florestais. A abordagem adotada se concentra na análise estatística dos dados de ocorrência de incêndios florestais e na avaliação de sua influência na classificação dos eventos de fogo. Portanto, nesta seção, os conceitos-chave utilizados no trabalho são apresentados para fornecer uma compreensão mais profunda dos temas abordados.

3.1 Sensoriamento Remoto

3.1.1 Conceito e Origem

O sensoriamento remoto é uma técnica que permite coletar informações sobre o meio ambiente usando sensores instalados em plataformas aéreas, terrestres ou orbitais. O objetivo principal do sensoriamento remoto é oferecer uma visão mais ampla e detalhada do ambiente, ajudando na tomada de decisões para questões relacionadas à gestão de recursos naturais, monitoramento climático e muitas outras aplicações.

A história do sensoriamento remoto começou durante a Primeira Guerra Mundial, quando as fotografias aéreas foram utilizadas para fins militares. Mais tarde, na década de 1930, essas fotografias foram utilizadas para fins de mapeamento e monitoramento ambiental [8].

A década de 1960 foi marcada pelo surgimento do sensoriamento remoto por satélite, permitindo uma cobertura mais ampla e frequente da Terra. Com o lançamento do satélite Landsat 1 em 1972, o sensoriamento remoto se tornou uma ferramenta indispensável para o monitoramento da Terra e a gestão de recursos naturais.

Desde então, o sensoriamento remoto tem evoluído continuamente, incluindo a introdução de novos sensores e tecnologias de processamento de dados. Além disso, a capacidade de capturar imagens de alta resolução e de diferentes bandas espectrais, como infravermelho e radar, ampliou ainda mais as possibilidades de uso do sensoriamento remoto [25].

Atualmente, o sensoriamento remoto é amplamente utilizado em uma variedade de setores, incluindo agricultura, meio ambiente, recursos hídricos, gestão de desastres e muitos outros. Além disso, o acesso a imagens de satélite de alta resolução e de baixo custo tornou o sensoriamento remoto acessível a uma ampla gama de usuários, incluindo governos, universidades, organizações não governamentais e empresas.

Assim, o sensoriamento remoto é uma ferramenta vital para a gestão do meio ambiente e para a tomada de decisões informadas. A história do sensoriamento remoto é rica e tem visto uma evolução contínua, tornando-se uma ferramenta acessível e eficaz para uma ampla gama de aplicações.

3.1.2 Sensoriamento Remoto na Amazônia

O Sensoriamento Remoto é uma ferramenta importante para a monitoração e compreensão do meio ambiente, especialmente na Amazônia. A Amazônia abriga uma grande diversidade de espécies, recursos naturais e populações humanas, e sua preservação é vital para a estabilidade do clima global e a manutenção da biodiversidade.

A utilização de técnicas de Sensoriamento Remoto permite a obtenção de informações precisas e frequentes sobre a floresta amazônica, como a cobertura florestal, a distribuição de espécies, o ciclo de vida da floresta, entre outros aspectos. Isso é possível graças às imagens de satélite, que oferecem uma visão geral da região, permitindo a identificação de tendências e padrões de mudança ao longo do tempo.

O Sensoriamento Remoto também pode ser usado para monitorar atividades humanas na Amazônia, como desmatamento, queimadas, mineração ilegal e exploração de recursos naturais, conforme visto na Figura 3.6. Isso é importante porque essas atividades podem ter impactos negativos irreversíveis na floresta e nas comunidades que dependem dela.

Figura 3.1 – Exemplo de sensoriamento remoto utilizado para mapear desmatamento na Amazônia. Fonte: INPE

No entanto, é importante destacar que o Sensoriamento Remoto não é uma ferramenta perfeita e tem limitações, como a resolução espacial limitada das imagens de satélite e a dificuldade de identificar certos tipos de atividades humanas na floresta, como pequenos desmatamentos ou exploração de recursos. Além disso, as imagens de satélite precisam ser interpretadas com cuidado, uma vez que a floresta amazônica é altamente variável e pode ser afetada por mudanças sazonais e outros fatores naturais.

Assim, o Sensoriamento Remoto é uma ferramenta valiosa para a compreensão e monitoramento da Amazônia, mas deve ser usado com cuidado e em conjunto com outras fontes de informação para garantir a precisão das informações obtidas.

3.1.3 QGIS

O Quantum Geographic Information System (QGIS) é um sistema de informações geográficas de código aberto desenvolvido com o objetivo de oferecer recursos avançados de SIG (Sistemas de Informações Geográficas) a usuários, desenvolvedores e empresas [41]. Desde sua primeira versão, lançada em 2002, o QGIS tem evoluído significativamente, tornando-se uma das opções mais populares e confiáveis para SIG, com uma ampla comunidade global de usuários e desenvolvedores.

Entre as funcionalidades oferecidas pelo QGIS, destacam-se: visualização de dados geográficos, edição de camadas, análise espacial, suporte a diversos formatos de arquivo, integração com outras ferramentas SIG, personalização por meio de plugins e acesso a serviços web, entre outros. Além disso, o QGIS é compatível com as principais plataformas operacionais, incluindo Windows, macOS e Linux, e oferece suporte a vários tipos de projeções, permitindo a visualização e análise precisa de dados geográficos em todo o mundo. Um exemplo de fisualização desses dados pode ser vista na Figura 3.2.

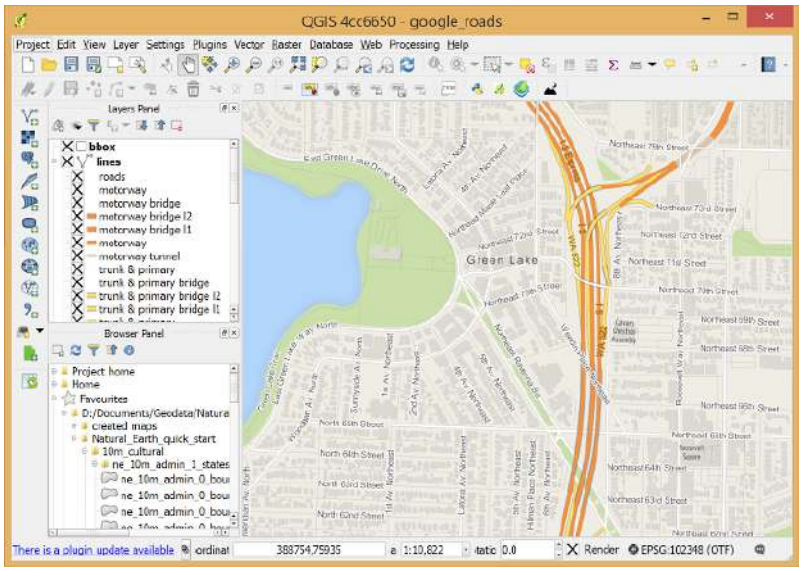

Figura 3.2 – Interface gráfica do QGIS.

O sucesso do QGIS pode ser atribuído a sua capacidade de atender às necessidades dos usuários, bem como à sua filosofia de colaboração e compartilhamento de conhecimento. A comunidade de usuários do QGIS é extremamente ativa, fornecendo suporte, documentação e desenvolvimento de novos recursos, e a equipe de desenvolvimento do projeto mantém uma abordagem aberta e transparente em relação ao desenvolvimento e evolução do software.

Portanto, o QGIS se mostrou uma excelente opção para aqueles que procuram uma solução SIG completa e confiável, com uma ampla gama de recursos e uma comunidade ativa e colaborativa e foi uma das ferramentas utilizadas para se analisar os dados do presente trabalho.

3.1.3.1 GeoTIFF

O GeoTIFF é um formato de arquivo de imagem raster que adiciona informações geográficas aos dados de imagem. O formato GeoTIFF permite que os dados de imagem sejam armazenados com informações sobre sua localização geográfica, permitindo que a imagem seja exibida e analisada dentro de um SIG (Sistema de Informações Geográficas).

A principal vantagem do formato GeoTIFF é sua capacidade de combinar dados de imagem com informações geográficas precisas, tornando-o uma escolha popular para aplicações GIS que requerem a visualização de dados geográficos. Além disso, o GeoTIFF é um formato aberto e amplamente aceito, o que significa que é compatível com uma ampla gama de softwares SIG [16].

Os arquivos GeoTIFF contêm informações como o sistema de coordenadas, a projeção e o modelo digital de elevação. Estas informações permitem que o SIG identifique a localização geográfica correta da imagem, permitindo a visualização e análise precisas dos dados.

Desta forma, o GeoTIFF é um formato de arquivo de imagem raster que permite combinar dados de imagem com informações geográficas precisas, tornando-o uma escolha popular para aplicações SIG que requerem a visualização de dados geográficos.

3.1.3.2 Shapefile

O Shapefile é um formato de arquivo vectorial amplamente utilizado em SIGs (Sistemas de Informações Geográficas) para armazenar dados geográficos, como camadas de linhas, pontos e polígonos [47]. O formato Shapefile foi desenvolvido pela ESRI (Environmental Systems Research Institute) e é suportado por uma ampla gama de softwares SIG.

A principal vantagem do formato Shapefile é sua capacidade de armazenar informações geográficas precisas e de alta qualidade, tornando-o uma escolha popular para aplicações SIG que requerem a visualização e análise de dados geográficos. Além disso, o formato Shapefile é fácil de usar e permite a transferência de dados geográficos entre diferentes softwares SIG sem perda de qualidade ou precisão.

Os arquivos Shapefile consistem em três ou mais arquivos, incluindo o arquivo .shp, que armazena a geometria dos objetos geográficos, o arquivo .dbf, que armazena as informações tabulares dos objetos geográficos, e o arquivo .shx, que armazena a indexação dos objetos geográficos.

O Shapefile é um formato de arquivo vectorial amplamente utilizado em SIGs para armazenar dados geográficos precisos e de alta qualidade, tornando-o uma escolha popular para aplicações SIG que requerem a visualização e análise de dados geográficos e foi extremamente utilizado ao longo deste trabalho.

3.2 Eventos de Fogo

O mapeamento dos incêndios florestais é importante por vários motivos. Em primeiro lugar, permite acompanhar a extensão e a intensidade dos fogos, avaliando seus efeitos na biodiversidade

e no meio ambiente. Além disso, pode ser usado para prevenir incêndios futuros, identificando áreas críticas de risco elevado, o que ajuda a implementar medidas preventivas.

Também é possível planejar os recursos de forma eficiente, graças ao mapeamento dos incêndios florestais. Isso permite a alocação adequada de equipes de combate a incêndios e equipamentos para responder a futuros incêndios.

Assim, ele também é útil no monitoramento da implementação de políticas públicas, permitindo avaliar a eficiência das políticas destinadas a prevenir e controlar incêndios florestais.

3.2.1 Focos de Calor

Os focos de calor são disponibilizados através da plataforma do *Fire Information for Resource Management System* (FIRMS) e *Visible Infrared Imaging Radiometer Suite* (VIIRS) [13], que fornece informações através de imagens de satélite e outras fontes de dados. Estes focos de calor são indicativos de incêndios florestais em andamento ou recentes.

As informações de focos de calor são atualizadas diariamente e incluem a data e hora do foco de calor, sua localização geográfica precisa, a intensidade da fonte de calor e, em alguns casos, a classificação do tipo de terreno queimado.

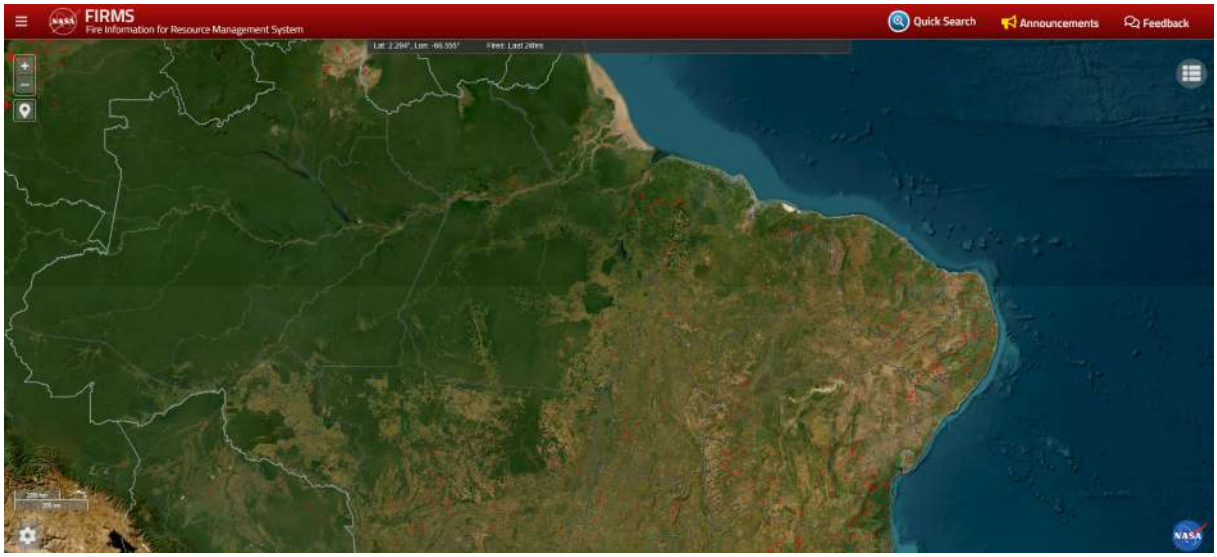

Figura 3.3 – Exemplo focos de calor apresentados pela plataforma do FIRMS, representados pelos pontos vermelhos.

O FIRMS, visto na Figura 3.3, é uma ferramenta valiosa para gerenciamento de recursos florestais, pois fornece informações atualizadas e precisas sobre incêndios florestais em andamento, permitindo que os responsáveis possam monitorar e responder a esses eventos de maneira mais eficaz. Além disso, o FIRMS também pode ser usado para planejar ações de prevenção e mitigação de incêndios florestais, bem como para avaliar o impacto desses eventos no meio ambiente e na economia local.

3.2.2 Tipificação do Fogo

A tipificação de eventos de fogo florestais é fundamental para entender a dinâmica dos eventos de fogo e desenvolver estratégias eficazes para prevenir e combater esses eventos de fogo.

De acordo com a Organização das Nações Unidas para a Alimentação e Agricultura (FAO), existem três tipos principais de fogos florestais: eventos de fogo de superfície, eventos de fogo de copas e eventos de fogo subterrâneos [14]. Cada tipo é caracterizado por diferentes dinâmicas de queima e requer abordagens distintas para o controle e extinção do fogo.

Além disso, a tipificação de fogos florestais é importante para avaliar o impacto dos eventos de fogo na biodiversidade e no meio ambiente. A tipificação também é importante para a investigação de eventos de fogo florestais, pois pode fornecer informações valiosas sobre as causas e circunstâncias do incêndio. Por exemplo, se for determinado que o incêndio foi causado por atividades humanas, isso pode indicar a necessidade de medidas de prevenção e gestão de fogos florestais mais rigorosas.

Portanto, a tipificação de fogos florestais é fundamental para entender a dinâmica dos eventos de fogo e desenvolver estratégias eficazes para prevenir e combater esses eventos de fogo. É importante para avaliar o impacto dos eventos de fogo na biodiversidade e no meio ambiente, bem como para a investigação de eventos de fogo florestais e a melhoria da gestão e prevenção de fogos florestais.

Como foi mencionado antes, este projeto foi realizado em colaboração com o CENSIPAM. Desta forma, o CENSIPAM trata de 4 tipos distintos na classificação de eventos de fogo, visto na Figura 3.4, na qual essas categorizações ajudam a determinar o tipo de terreno atingido pelo fogo e, assim, entender melhor suas consequências e as razões subjacentes. As quatro classificações utilizadas são:

- Savana/pastagem
- Pequenas clareiras
- Sub-bosque
- Desmatamento

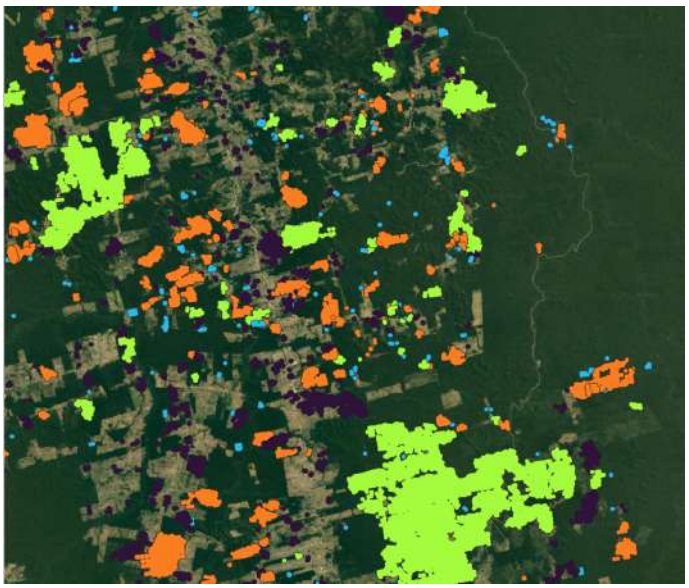

Figura 3.4 – Exemplo eventos de calor classificados pelo CENSIPAM, com diferentes tipagens de fogo, cada uma representada por uma cor.

Essa classificação é importante para entender a dinâmica dos eventos de fogo florestais na Amazônia e desenvolver estratégias eficazes para prevenir e combater esses eventos de fogo.

- **Fogo de Savanas/Pastagens**: Esses eventos de fogo ocorrem em áreas de savana ou pastagem, principalmente causados por ignições humanas ou condições climáticas secas. Eles afetam vegetação de baixa altura com biomassa escassa e seca. O CENSIPAM utiliza informações sobre a porcentagem de cobertura florestal e taxas históricas de desmatamento para distinguir esses eventos de fogo.

- **Fogo de Pequenas Clareiras (Roçado)**: Esses eventos de fogo são comuns em áreas com pequenas clareiras criadas para agricultura de subsistência ou pecuária. Podem se espalhar para florestas adjacentes e causar danos significativos. O CENSIPAM associa esses eventos de fogo com a cobertura do solo e as regras de área de tamanho pequeno relacionadas à degradação florestal. São classificados como pequenos eventos de fogo ou eventos de fogo agrícolas.

- **Fogo de Sub-bosque**: Esses eventos de fogo ocorrem frequentemente em florestas tropicais, originando-se abaixo do dossel arbóreo. São causados por negligência, como fogueiras, ou queimadas agrícolas. No entanto, na floresta amazônica, estão fortemente relacionados à dinâmica de desmatamento. Tendem a se desenvolver em áreas de degradação do solo com altas taxas de biomassa, bem como nas bordas de áreas desmatadas para queimar.

- **Fogo de Desmatamento**: eventos de fogo resultantes de práticas de desmatamento, realizadas por empresas ou agricultores para expandir suas terras. O uso do fogo para limpar áreas após o desmatamento aumenta o risco de eventos de fogo. Esses eventos de fogo frequentemente exibem uma alta emissão inicial de radiação devido ao acúmulo de detritos lenhosos, resultando em uma maior liberação de energia e persistência prolongada do fogo.

3.3 Aprendizado de Máquina

3.3.1 Origem

Atualmente, temas envolvendo Aprendizado de Máquina estão amplamente difundidos na comunidade científica e popular [44]. Tudo isso começou na Segunda Guerra Mundial, quando um importante criptanalista, chamado Turing, não pôde recorrer ao projeto de construir uma máquina de computação eletrônica de programa armazenado até à cessação das hostilidades na Europa em 1945. No entanto, durante a guerra, ele pensou consideravelmente na questão da inteligência das máquinas. Um dos colegas de Turing, Donald Michie (que mais tarde fundou o Departamento de Inteligência e Percepção de Máquinas na Universidade de Edimburgo), recordou mais tarde que Turing discutia frequentemente como os computadores poderiam aprender com a experiência, bem como resolver novos problemas através da utilização de princípios orientadores - um processo agora conhecido como resolução de problemas heurísticos. Assim o fenômeno da inteligência das máquinas começou a ser discutido no ano de 1950 em um artigo publicado por Alan Turing [9].

De acordo com Turing, uma inteligência artificial (IA) seria a capacidade de uma máquina imitar o comportamento humano. Quanto mais semelhantes suas ações fossem às de um humano, mais inteligente a máquina seria considerada. Para determinar a inteligência de uma IA, as pessoas criam um teste que não beneficia nem a pessoa nem a máquina que está sendo testada. Durante esse processo, três pessoas estão envolvidas: duas que fazem perguntas e uma que as responde. O teste de Turing requer dois respondentes para responder à pergunta de um questionador. Três participantes são selados em espaços separados e devem realizar vários projetos com temas diferentes. Esses projetos podem variar de fórmulas matemáticas a questões relacionadas à experiência de vida. O objetivo deste teste é separar a inteligência artificial da inteligência natural.

Desta forma, na mesma década, a Universidade Carnegie Mellon se tornou a primeira instituição acadêmica a estudar, de fato, a inteligência artificial. Cientistas como Allen Newell, Hebert Simon e outros pioneiros lideraram a pesquisa, buscando atender o anseio da humanidade por criarem máquinas que pudessem duplicar a inteligência e a ação humanas. Tais pesquisas levaram ambos os cientistas a ganharem o Prêmio A.M. Turing de 1975, sendo o maior prêmio acadêmico na área de computação, justamente pelas suas contribuições "básicas"para a inteligência artificial, psicologia da cognição humana e o processamento de listas.

Embora a inteligência artificial tenha se fortalecido graças ao progresso em seu desenvolvimento, ela também se beneficiou de um impulso na análise computacional desde sua criação. Auxiliados por isso, as máquinas, hoje, podem até analisar e criar novas vozes a partir da fala humana. Cientistas e pesquisadores perceberam que esse ramo da ciência tinha muito mais a se desenvolver e, como resultado, muitos decidiram buscar a criação de uma máquina que pudesse criar autoaperfeiçoamento, sentimentos e habilidades de linguagem, além da capacidade de pensar, nos levando à alta difusão de tal tecnologia nos dias atuais

3.3.2 Conceito

Conforme detalha [6], atualmente, o campo da aprendizagem de máquinas está organizado em torno de três focos primários:

- Estudos orientados para tarefas - desenvolvimento e análise de sistemas de aprendizagem para melhorar o desempenho num conjunto pré-determinado de tarefas (também conhecido como a "abordagem de engenharia").
- Simulação Cognitiva - a investigação e simulação por computador dos processos de aprendizagem humana.
- Análise Teórica - a exploração teórica do espaço de possíveis métodos e algoritmos de aprendizagem independentes do domínio de aplicação.

Dessa forma, instruir um computador para executar uma tarefa requer a definição de um algoritmo completo e correto para essa finalidade, seguido pela programação precisa desse algoritmo. Essas atividades geralmente demandam um esforço tedioso e demorado por parte do profissional responsável pela execução da tarefa. Nesse contexto, o setor de aprendizagem de máquinas busca abrir novas formas de instruir computadores, aliviando o ônus da programação manual em face do aumento constante de informações, que se tornam cada vez mais complexas a cada dia. Assim, a rápida expansão das aplicações e a ampla disponibilidade de computadores nos dias de hoje tornam essa perspectiva ainda mais atrativa e desejável.

3.3.3 Aprendizado Supervisionado

Conforme descrito por [1], ao criarmos um conjunto de treinamento com saídas e entradas corretas, os algoritmos de aprendizado supervisionado permitem que os modelos aprendam ao longo do tempo. Esses algoritmos usam uma medida de perda, como erro ou distância da saída desejada, para ajustar iterativamente seus resultados até que produzam a saída correta. Tal tipo de resolução foca em dois tipos de problemas, sendo elas a classificação e a regressão. O termo vem da IBM e é descrito na citação acima.

Os métodos de classificação contam com algoritmos para dividir conjuntos de dados em categorias usando dados de teste. Esses métodos procuram características específicas em um conjunto de dados e determinam como categorizá-las com base em suas observações. Os métodos de classificação comuns incluem Random Forest, classificadores lineares, SVMs, árvores de decisão, etc.

Já a regressão é usada para entender a relação entre variáveis dependentes e independentes. É comumente usado para fazer projeções, como por exemplo na receita de vendas que uma empresa espera obter. As pessoas costumam usar regressão linear, regressão logística, regressão polinomial, etc.

3.3.3.1 Random Forest

Random Forest é um algoritmo de aprendizado de máquina supervisionado, amplamente utilizado em diversas áreas, incluindo a análise de dados, classificação de imagens e previsão de séries temporais. Ele foi desenvolvido por Leo Breiman e Adele Cutler e apresentado pela primeira vez em 2001 [4].

O *Random Forest* é baseado em um conjunto de árvores de decisão, em que cada árvore é treinada com uma amostra aleatória dos dados de treinamento, e as previsões finais são feitas por votação dos resultados dessas árvores. Isso torna o algoritmo mais robusto e menos suscetível a overfitting em comparação a uma única árvore de decisão [10].

Cada árvore de decisão é treinada com uma amostra aleatória dos dados de treinamento, e as árvores são construídas com características aleatórias, o que resulta em uma combinação diferente de árvores [24]. Isso significa que o erro de uma árvore pode ser compensado pelo erro de outra árvore, levando a uma previsão mais precisa.

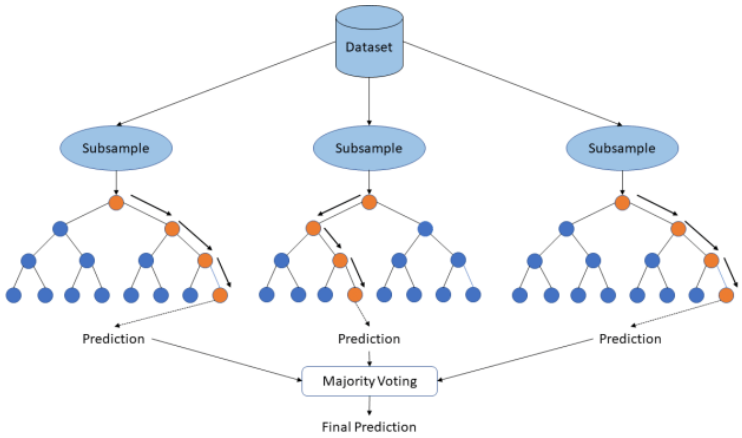

Figura 3.5 – Esquemático genérico do *Random Forest*. Fonte: ResearchGate

Acima, temos a Figura 3.5, que mostra a construção de cada árvore em um *Random Forest*, no qual envolve a seleção aleatória de uma amostra D' a partir do conjunto de dados de treinamento D. Cada árvore é treinada para otimizar uma medida de impureza, como a impureza de Gini ou a entropia. A impureza de Gini para um nó na árvore é dada por:

$$\text{Impureza}(D') = 1 - \sum_{i=1}^{K} p_i^2 \tag{3.1}$$

onde K é o número de classes e p_i é a proporção da classe i na amostra D'.

Uma vez que as árvores são construídas, as previsões são combinadas para formar a saída final do modelo. Para problemas de classificação, a combinação é feita por voto majoritário, enquanto para regressão é realizada por média. A previsão Y é definida como:

$$Y = \frac{1}{|F|} \sum_{i=1}^{|F|} f_i(X) \tag{3.2}$$

onde F é o conjunto de todas as árvores no *Random Forest* e $f_i(X)$ é a previsão da i-ésima árvore para a entrada X.

Os *Random Forests* oferecem vantagens significativas, incluindo a capacidade de lidar com dados não lineares, a robustez contra overfitting e a avaliação da importância das características. No entanto, é importante mencionar que a interpretabilidade do modelo pode ser comprometida devido à complexidade das florestas.

Elas também têm sido amplamente utilizados em uma variedade de domínios, incluindo bioinformática, finanças e reconhecimento de padrões. Sua capacidade de lidar com conjuntos de dados complexos faz deles uma escolha popular para problemas do mundo real.

Assim, elas representam uma abordagem eficaz para o aprendizado de máquina, equilibrando precisão e generalização. Sua aplicação abrange uma ampla gama de problemas, tornando-os uma ferramenta valiosa na caixa de ferramentas do cientista de dados.

3.3.4 Aprendizado não-supervisionado

Novamente, conforme detalha [1], o aprendizado de máquina não supervisionado permite que os algoritmos analisem e agrupem conjuntos de dados sem receber um rótulo. Isto é possível sem qualquer intervenção humana; os algoritmos descobrem padrões ou *clusters* ocultos nos dados. Ele pode identificar semelhanças e diferenças entre os dados, tornando-se a ferramenta perfeita para análise exploratória de dados, criação de estratégias de cross-marketing, segmentação de clientes e reconhecimento de imagens. Ele pode ser usado para três propósitos principais por meio de modelos de aprendizado não supervisionado: agrupamento, associação e redução da dimensionalidade da informação.

3.4 Aprendizado Profundo

3.4.1 Origem

Derivado do aprendizado de máquina, as redes de aprendizagem profunda, ou *Deep Learning*, tiveram origem na década de 1960, quando Ivakhnenko e Lapa publicaram o primeiro algoritmo de aprendizagem geral, utilizando *Multi-Layer Perceptrons* (MLP) supervisionados. As suas unidades tinham funções de ativação polinomial combinando adições e multiplicações em polinômios de Kolmogorov-Gabor.

Em 1971, Ivakhnenko já descrevia uma rede profunda com 8 camadas treinadas pelo "Método de Tratamento de Dados em Grupo", ainda popular na época. Dado um conjunto de formação de vetores de entrada com os correspondentes vetores de saída, as camadas eram incrementalmente cultivadas e treinadas por análise de regressão, depois podadas com a ajuda de um conjunto de validação separado, onde a regularização é utilizada para eliminar as unidades supérfluas.

Assim, com o passar do tempo, a aprendizagem profunda melhorou drasticamente o estado da arte em diversas áreas [46]. A sua natureza de arquitetura profunda conferiu à aprendizagem profunda a possibilidade de resolver muitas tarefas mais complicadas de IA [3]. Como resultado, os pesquisadores estão expandindo a aprendizagem profunda a uma variedade de diferentes domínios e tarefas modernas em adição às tarefas tradicionais como a detecção de objetos, reconhecimento facial, modelos linguísticos, etc.

3.4.2 Conceito

O aprendizado profundo, ou popularmente chamado de *Deep Learning* é um ramo do aprendizado de máquina que se concentra em modelos de redes neurais profundas e complexas para realizar tarefas como classificação, reconhecimento de imagens e linguística natural [23]. Ele se baseia em camadas consecutivas de processamento de dados, cada uma aprendendo a extrair características mais complexas a partir dos dados de entrada.

Ele é capaz de aprender representações hierárquicas de dados, o que o torna particularmente útil para problemas de processamento de imagem e de linguística natural, em que os dados são compostos por uma grande quantidade de características inter-relacionadas.

Assim, redes neurais profundas são compostas por muitas camadas, cada uma composta por uma grande quantidade de neurônios interconectados. Os neurônios nas camadas mais profundas aprendem a identificar padrões complexos a partir dos dados de entrada, enquanto as camadas mais externas realizam a classificação ou a previsão final.

Dessa forma, o *Deep Learning* tem sido aplicado com sucesso em uma ampla variedade de tarefas, incluindo classificação de imagens, reconhecimento de voz, processamento de linguística natural e controle de robôs. Além disso, o uso de técnicas avançadas de otimização, como o gradiente descendente estocástico, permitem aos modelos de Deep Learning aprender continuamente de novos dados, tornando-os mais precisos com o tempo [17].

Portanto, tal ramo é uma área em rápido crescimento do aprendizado de máquina que se concentra em modelos de redes neurais profundas para realizar tarefas complexas, como classificação de imagens e processamento de linguística natural. Ele se baseia em camadas consecutivas de processamento de dados e tem sido aplicado com sucesso em uma ampla variedade de tarefas, tornando-se uma ferramenta valiosa para a análise de dados.

3.4.3 Multi-Layer Perceptron (MLP)

O Multi-layer Perceptron (MLP), também conhecido como rede neural feedforward, é um tipo de modelo de aprendizado de máquina baseado em redes neurais que consiste em várias camadas de neurônios interconectados [42]. O MLP foi desenvolvido na década de 1950 e é uma das primeiras redes neurais a serem estudadas e aplicadas [17].

Em uma rede neural feedforward, as informações de entrada são passadas através de várias camadas de neurônios, cada uma realizando uma transformação matemática simples dos dados de entrada. A última camada da rede produz a saída desejada, como a classificação ou a previsão de

um determinado valor.

Ele é diferenciado de outros tipos de redes neurais, como as redes neurais recurrentes, pelo fato de que as informações de entrada só fluem na direção da saída, sem retorno a camadas anteriores. Isso permite que o MLP seja facilmente treinado usando técnicas de otimização, como o gradiente descendente, para ajustar os pesos dos neurônios de acordo com o erro de previsão.

A estrutura do Multi Layer Perceptron (MLP) consiste em três camadas de neurônios interligados. A camada de entrada é onde são recebidos os dados de entrada e processados para a próxima camada. A camada oculta é composta por vários neurônios que executam as operações matemáticas necessárias para processar os dados. O número de neurônios e subcamadas nesta camada pode ser ajustado dependendo da complexidade do problema. Por fim, a camada de saída produz a previsão ou a classificação do problema, que é a saída final da rede.

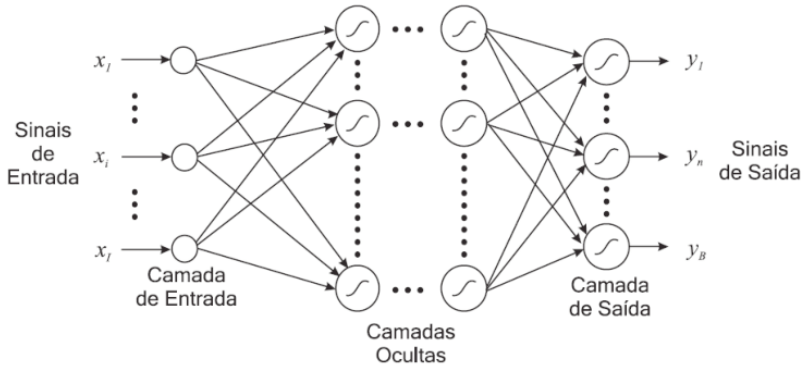

Figura 3.6 – Esquemático genérico de uma MLP.

Os neurônios em uma camada oculta, ilustrados na Figura 3.6, são conectados a todos os neurônios na camada anterior e na camada seguinte por meio de pesos. Cada peso representa a força da conexão entre dois neurônios. O objetivo do treinamento da rede neural é ajustar esses pesos de tal forma que a saída da rede seja a mais precisa possível.

A matemática por trás de uma camada oculta de MLP envolve a aplicação de uma função de ativação em cada neurônio. Esta função determina a resposta do neurônio aos sinais de entrada. Uma função comum é a função logística (também conhecida como função sigmóide), que produz uma saída compreendida entre 0 e 1. A saída de cada neurônio na camada oculta é dada por:

$$y = \sigma(w_1 x_1 + w_2 x_2 + ... + w_n x_n + b) \tag{3.3}$$

Assim, na equação 3.3, y é a saída do neurônio, σ é a função de ativação, $w_1, w_2, ..., w_n$ são os pesos da conexão entre os neurônios, $x_1, x_2, ..., x_n$ são os dados de entrada e b é o viés.

A saída da camada oculta é, então, usada como entrada para a próxima camada ou para a camada de saída, dependendo da arquitetura da rede. Esse processo é repetido até que a saída final da rede seja produzida.

O MLP têm sido amplamente utilizado em muitas áreas, incluindo a classificação de imagens,

o processamento de linguística natural e a previsão financeira. Além disso, o MLP foi uma das principais inspirações para o desenvolvimento de outros modelos de aprendizado de máquina, como as redes neurais convolutionais [23].

Assim, o MLP é um tipo de modelo de rede neural feedforward que consiste em várias camadas de neurônios interconectados. Ele foi desenvolvido na década de 1950 e é uma das primeiras redes neurais a serem estudadas e aplicadas. O MLP é fácil de treinar e tem sido amplamente utilizado em muitas áreas, incluindo a classificação de imagens e o processamento de linguística natural.

3.5 Python

O Python é uma linguagem de programação de alto nível que foi criada no final da década de 80 e se tornou uma das linguagens mais populares da atualidade [27]. Ela é conhecida por sua sintaxe clara, facilidade de aprendizado e capacidade de resolver problemas complexos de maneira eficiente. Além disso, o Python possui uma vasta comunidade de desenvolvedores e uma ampla gama de bibliotecas e ferramentas que permitem que ela seja utilizada em muitas áreas, incluindo ciência de dados, inteligência artificial, computação científica, desenvolvimento web e muito mais.

A seção em questão irá explorar os principais conceitos, ferramentas e bibliotecas que foram utilizados durante o projeto. Dessa forma, elas permitiram a realização de tarefas como manipulação de dados, visualização, modelagem e outras tarefas relacionadas ao processo de análise de dados.

3.5.1 Dataframe e GeoDataframe

Um DataFrame é uma estrutura de dados bidimensional utilizada na análise de dados. Ele é composto por linhas e colunas e pode ser pensado como uma tabela onde cada linha representa uma observação e cada coluna representa uma variável. Os dados em um dataframe podem ser do tipo numérico, categórico ou texto, e as operações que podem ser realizadas incluem a manipulação, seleção e agregação de dados.

DataFrames são amplamente utilizados em muitas áreas, incluindo finanças, saúde, marketing e ciência de dados. A utilização de DataFrames permite a manipulação e análise de grandes quantidades de dados de forma eficiente e intuitiva, tornando-os uma ferramenta valiosa para a exploração e compreensão de dados.

Já um GeoDataFrame é uma estrutura de dados baseada em dataframes do pandas que permite armazenar e manipular dados geográficos. Ele é uma extensão dos dataframes comuns e permite armazenar informações geográficas, como coordenadas e geometrias, além de dados tabulares como nomes, idades e endereços.

O uso combinado do GeoDataFrame e do Shapefile permite a manipulação eficiente de dados geográficos em SIGs, tornando possível realizar operações como a adição e remoção de camadas, a análise de dados geográficos e a geração de mapas. Além disso, o uso do GeoDataFrame permite a integração dos dados geográficos com os dados tabulares, tornando possível realizar análises mais complexas.

Assim, o uso combinado do GeoDataFrame e do Shapefile permite a manipulação eficiente de dados geográficos em SIGs, tornando possível realizar operações como a adição e remoção de camadas, a análise de dados geográficos e a geração de mapas.

3.5.2 Pandas e GeoPandas

Pandas é uma biblioteca de código aberto em Python que fornece estruturas de dados e ferramentas para análise de dados [29]. A principal estrutura de dados do Pandas é o DataFrame. O Pandas permite a realização de operações como filtragem, agrupamento, junção e muito mais de maneira rápida e eficiente. Além disso, ele é capaz de lidar com dados ausentes, duplicados e de vários tipos de formato.

Já o GeoPandas é uma biblioteca de código aberto que fornece ferramentas para manipulação e análise de dados geográficos [22]. Ele é construído sobre o Pandas e, portanto, herda as mesmas funcionalidades e facilidades para trabalhar com dados. Além disso, o GeoPandas permite a manipulação de informações geográficas, como shapefiles e informações sobre projeções de coordenadas, de maneira simples e intuitiva.

Ambas as bibliotecas, Pandas e GeoPandas, foram extremamente importantes para o trabalho, pois permitiram uma manipulação eficiente e rápida de dados geográficos. Além disso, as facilidades de análise de dados fornecidas pelo Pandas, combinadas com as funcionalidades geográficas do GeoPandas, permitiram a realização de análises complexas de maneira simples e eficiente.

3.5.3 Rasterio

Rasterio é uma biblioteca de software livre escrita em Python que fornece uma forma eficiente e fácil de trabalhar com dados de imagem raster, como arquivos GeoTIFF. Essa biblioteca permite que os dados raster sejam lidos, escritos, manipulados e visualizados com facilidade [**rasterio**].

Rasterio é baseado em Numpy [32], o que permite a integração eficiente de dados numéricos em seus processos de manipulação de dados. Além disso, Rasterio oferece uma interface para lidar com projeções de coordenadas e para trabalhar com dados georreferenciados, como imagens de satélite, mapas topográficos e outros tipos de dados geográficos.

Uma das grandes vantagens do Rasterio é que ele suporta uma ampla variedade de formatos de arquivo raster, incluindo GeoTIFF, JP2, PNG e muitos outros. Isso significa que é possível trabalhar com dados de diferentes fontes e formatos sem precisar convertê-los antes de usar.

Além disso, o Rasterio é compatível com outras bibliotecas Python populares, como NumPy, Matplotlib [19] e Geopandas, o que permite combinar facilmente dados raster com outros tipos of dados geográficos e analisá-los de forma integrada.

Em resumo, o Rasterio é uma ferramenta valiosa para quem precisa lidar com dados de imagem raster e integrá-los com outros dados geográficos para realizar análises avançadas. O uso desse pacote é fundamental para projetos que envolvem a manipulação de imagens de satélite, mapas topográficos e outros tipos de dados geográficos.

3.5.4 Scikit-learn

O scikit-learn [34] é uma biblioteca em Python de código aberto que oferece uma abordagem simples e eficaz para a construção de modelos de aprendizado de máquina. Essa biblioteca proporciona ferramentas robustas para pré-processamento de dados, seleção de recursos, construção de modelos e avaliação de desempenho, tornando o desenvolvimento de soluções de aprendizado de máquina mais acessível.

O scikit-learn também se baseia no Numpy e no SciPy, permitindo a manipulação eficiente de dados numéricos e científicos em suas operações de aprendizado de máquina. A interface coesa do scikit-learn facilita o fluxo de trabalho, desde a preparação dos dados até a avaliação dos modelos.

Uma das principais vantagens do scikit-learn é sua ênfase em modelos simples e bem documentados. A biblioteca oferece uma variedade de algoritmos de aprendizado de máquina supervisionados e não supervisionados, como regressão linear, classificação, agrupamento e redução de dimensionalidade. Cada algoritmo é cuidadosamente implementado e documentado, tornando-o uma escolha ideal para aqueles que estão começando no campo do aprendizado de máquina.

Além disso, o scikit-learn oferece uma abordagem consistente para ajuste de hiperparâmetros e validação cruzada. Isso significa que é possível otimizar os parâmetros dos modelos de maneira sistemática, melhorando seu desempenho e evitando problemas de overfitting ou underfitting.

Uma característica notável do scikit-learn é sua facilidade de uso e sua integração com outras bibliotecas populares, como o Matplotlib e o Pandas. Isso permite a visualização de resultados e a análise de dados de maneira eficiente, possibilitando insights valiosos durante o desenvolvimento e experimentação.

Portanto, o scikit-learn é uma ferramenta essencial para quem deseja entrar no mundo do aprendizado de máquina de forma prática e eficaz. Sua ampla gama de algoritmos, documentação detalhada, ênfase na simplicidade e integração com outras bibliotecas tornam-no uma escolha sólida tanto para iniciantes quanto para profissionais que buscam desenvolver soluções de aprendizado de máquina consistentes e bem fundamentadas.

3.5.5 PyTorch

O PyTorch [33] é uma biblioteca de código aberto desenvolvida em Python que oferece uma abordagem eficaz e flexível para a criação e treinamento de redes neurais e modelos de aprendizado profundo. Essa biblioteca possibilita a manipulação de tensores, o desenvolvimento de algoritmos de aprendizado de máquina e a construção de arquiteturas complexas com relativa facilidade.

Assim como o Rasterio e scikit-learn se baseiam no Numpy, o PyTorch também se apoia nessa biblioteca, o que facilita a integração de operações numéricas em suas tarefas de processamento de dados. No entanto, o PyTorch se destaca ao fornecer uma abstração dinâmica conhecida como "grafo computacional", que é uma estrutura que permite definir e executar operações em tempo real, facilitando a depuração e experimentação durante o desenvolvimento de modelos.

Uma das principais vantagens do PyTorch é sua flexibilidade na criação de modelos. Diferentemente de algumas bibliotecas que adotam uma abordagem estática, o PyTorch permite a

construção de arquiteturas de rede de maneira mais intuitiva e "Pythonic", o que é especialmente útil para pesquisadores e desenvolvedores que precisam iterar rapidamente em seus projetos.

Outro ponto forte do PyTorch é sua ênfase na computação na GPU. A biblioteca permite facilmente a transferência de dados e operações para placas gráficas, o que acelera significativamente o treinamento de modelos complexos, como redes neurais profundas. Isso torna o PyTorch uma escolha popular entre a comunidade de aprendizado profundo.

Além disso, o PyTorch também oferece ferramentas para trabalhar com autograd, uma técnica que calcula automaticamente gradientes para backpropagation, simplificando o processo de treinamento de modelos. Esse recurso é especialmente útil para ajustar parâmetros de modelos complexos e aprimorar sua precisão.

Uma característica notável do PyTorch é sua comunidade ativa e sua adoção pela academia e pela indústria. Muitas pesquisas e implementações de ponta em aprendizado profundo são desenvolvidas usando o PyTorch, o que contribui para uma rica base de conhecimento e recursos disponíveis.

Assim, sua abordagem dinâmica, integração com GPUs, suporte a autograd e comunidade ativa fazem dele uma escolha poderosa tanto para projetos de pesquisa quanto para aplicações de larga escala que demandam a utilização de técnicas de aprendizado profundo.

4 Metodologia e Arquitetura

A seguinte seção tem como objetivo descrever a metodologia utilizada para classificar eventos de fogo florestais na Amazônia Legal. Um relato detalhado da abordagem adotada na análise de dados, no desenvolvimento de modelos de classificação e na arquitetura computacional será fornecido. A lógica por trás dessas escolhas e as etapas tomadas para garantir a precisão e eficácia do sistema de classificação também serão explicadas minuciosamente. É essencial ter uma compreensão abrangente da metodologia e arquitetura para uma avaliação adequada e comparação com outras abordagens, bem como para replicar os resultados obtidos neste projeto. Em um segundo ponto desta pesquisa, oferecemos um método operacional disponível para ser integrado às ferramentas de painel de eventos de fogo.

Com base no pipeline apresentado na Figura 4.1, fornecemos uma visão do processo passo a passo empregado no trabalho atual, começando pela leitura de dados e incluindo a incorporação de novas características, as quais, como será detalhado, são cruciais para a classificação de eventos de fogo. Isso inclui pré-processamento de dados e o treinamento e teste de dados gerados usando os dois algoritmos de aprendizado de máquina, Random Forest e MLP.

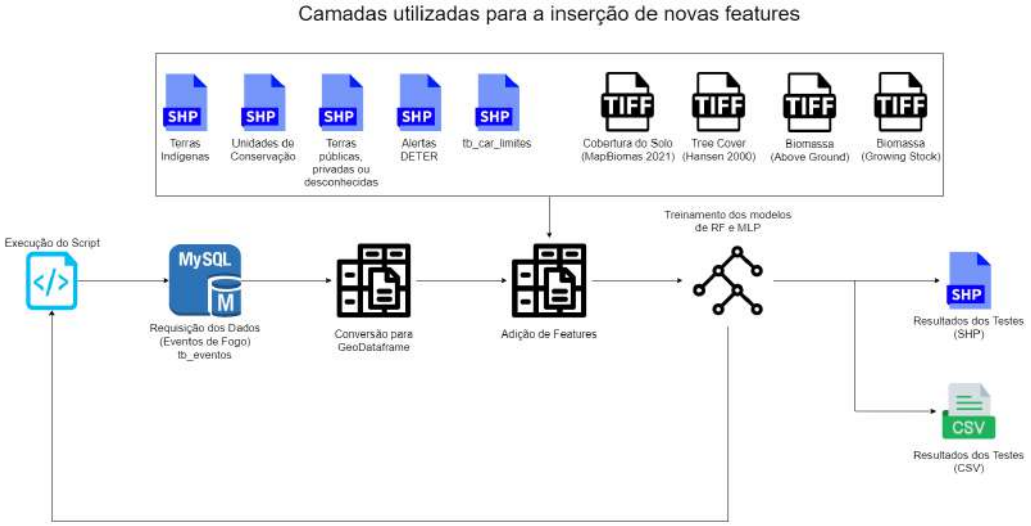

Figura 4.1 – Esquema detalhando o pipeline passo a passo. Fonte: Autor.

4.1 Dataset

O projeto realizado no CENSIPAM tinha como objetivo automatizar a classificação de eventos de incêndio na região da Amazônia Legal. Isso foi alcançado aproveitando um conjunto de dados que compreende vários eventos de incêndio meticulosamente mapeados pelo CENSIPAM no ano de 2020. O conjunto de dados é cuidadosamente elaborado, utilizando dados dos sensores orbitais VIIRS (satélites NOAA-20 e Suomi NPP) e MODIS (Aqua e Terra), que são instrumentais na detecção de eventos de fogo ativos.

Esses dados são então transformados em uma camada vetorial de "eventos de fogo ativos"chamada 'tb_eventos', indicando as localizações exatas desses eventos de incêndio ativos. Essa camada é atualizada diariamente. A resolução espacial das imagens usadas para gerar essa camada de pontos é de aproximadamente 375 metros para VIIRS e 1 quilômetro para MODIS. Para facilitar o processo de agrupamento, é utilizada uma abordagem de geometria artificial, e uma ferramenta de geoprocessamento de buffer é empregada para estabelecer conexões entre os pontos de incêndio ativos.

Tabela 4.1 – Atributos e rótulo do conjunto de dados disponibilizado pelo CENSIPAM.

Classe de Atributo	**Atributo**	**Descrição**
Características do Evento	ID	Número de identificação único do evento de incêndio
	Status	Fração (0-1) de células de grade de 550m desmatadas
	Data Mínima	Data de início das primeiras detecções
	Data Mais Recente	Última data de detecções
	Persistência	Duração (em dias)
	Número de Detecções	Número total de detecções focos de calor dentro do evento
	Área	Tamanho do incêndio em quilômetros quadrados
Rótulo	Tipo de Incêndio	Classificação mista do tipo de evento de incêndio, combinando rótulos manuais e do GFED

Para este estudo, o CENSIPAM forneceu um conjunto de dados exclusivo com mais de 150.000 eventos de incêndio rotulados de acordo com o Global Fire Emission Database, conforme detalhado na Tabela 4.1. Esses eventos de incêndio se enquadram em quatro categorias distintas identificadas pelo GFED: eventos de fogo em savanas/pastagens, pequenos eventos de fogo de desmatamento (corte e queima), eventos de fogo em sub-bosque e eventos de fogo de desmatamento. Cada categoria apresenta características únicas, especialmente ao considerar os atributos delineados na Tabela 4.1.

Embora muitos outros projetos se baseiem em metodologias que envolvem o monitoramento de eventos de fogo florestais por meio de imagens de satélite, nossa abordagem neste trabalho é a utilização de dataframes. Optamos por essa escolha devido à facilidade de processamento e armazenamento em grande escala desses conjuntos de dados, que são coletados com maior frequência pelos satélites em comparação com as imagens. Além disso, as imagens de satélite sofrem considerável impacto em sua qualidade devido a fatores climáticos, incidência de luz e posicionamento do satélite. Essas variáveis podem comprometer a eficácia da análise, especialmente em situações urgentes.

4.1.1 GFED *Amazon Dashboard*

Antes de avançar na investigação de dados na área de eventos de fogo florestais, é fundamental desenvolver um conjunto de dados adequado. O GFED Amazon Dashboard, assim como o CENSIPAM, classifica eventos de eventos de fogo florestais a partir da combinação de diversas fontes de dados, tais como os focos de calor fornecidos pelo FIRMS, modelos climáticos, medições terrestres e outras fontes de informações.

Essas fontes são combinadas para produzir estimativas mensais e anuais para cada evento de incêndio florestal identificado, além de classificar os tipos de queimadas presentes. Em primeiro lugar, foi considerado o conjunto de fatores utilizados pelo GFED na classificação dos tipos de eventos de eventos de fogo florestais, todos disponíveis no dataset fornecido por eles. Assim, temos:

- **Confiança**: (1) baixo, (2) moderado, (3) alto.
- **Status de Desflorestamento Local**: Fração, entre 0 e 1, de 550m de "grid cells" com desflorestação histórica.
- **Índice de Cobertura de Árvores**: Fração média de cobertura de árvores.
- **Biomassa**: Média de biomassa dentro do perímetro (polígono) do fogo.
- **FRP**: Média do Poder Radiativo de Fogo.
- **Contagem de focos de calor**: Número total de detecções de incêndio dentro do perímetro de incêndio.
- **Tamanho**: Tamanho do fogo em quilômetros quadrados.
- **Persistência**: Persistência média do fogo em células de 550m dentro do perímetro de fogo em número de dias.
- **De dia?**: Falso (0) se não ocorreu durante o dia e Verdadeiro (1) caso contrário.
- **Progressão**: Fração média da progressão do fogo em células de 550m dentro do perímetro, variando de 0 a 1.
- **Protegido**: Fração, entre 0 e 1, do fogo que ocorre dentro de uma área protegida ou terra indígena.
- **Dia de início**: Dia do novo incêndio começa a partir do dia do ano, variando de 1 a 366 dias.
- **Último dia**: Detecção ativa do incêndio mais recente dentro do perímetro do polígono a partir do dia do ano.
- **É novo?**: Analisa se o evento é um novo incêndio que começou dentro das últimas 24 horas, sendo classificado com verdadeiro (1) ou falso (0).

- **Está ativo?**: O fogo esteve ativo nos últimos 10 dias (1) ou não (0).
- **Bioma**: O fogo está dentro (1) ou fora (0) do bioma da Amazônia.
- **Tipo do Fogo**: Tipificação atribuída àquele evento de fogo.

Com base nas características examinadas, foi gerado um gráfico para determinar quais atributos devem ser destacados ao incorporar ao conjunto de dados do CENSIPAM, conforme a figura abaixo detalha:

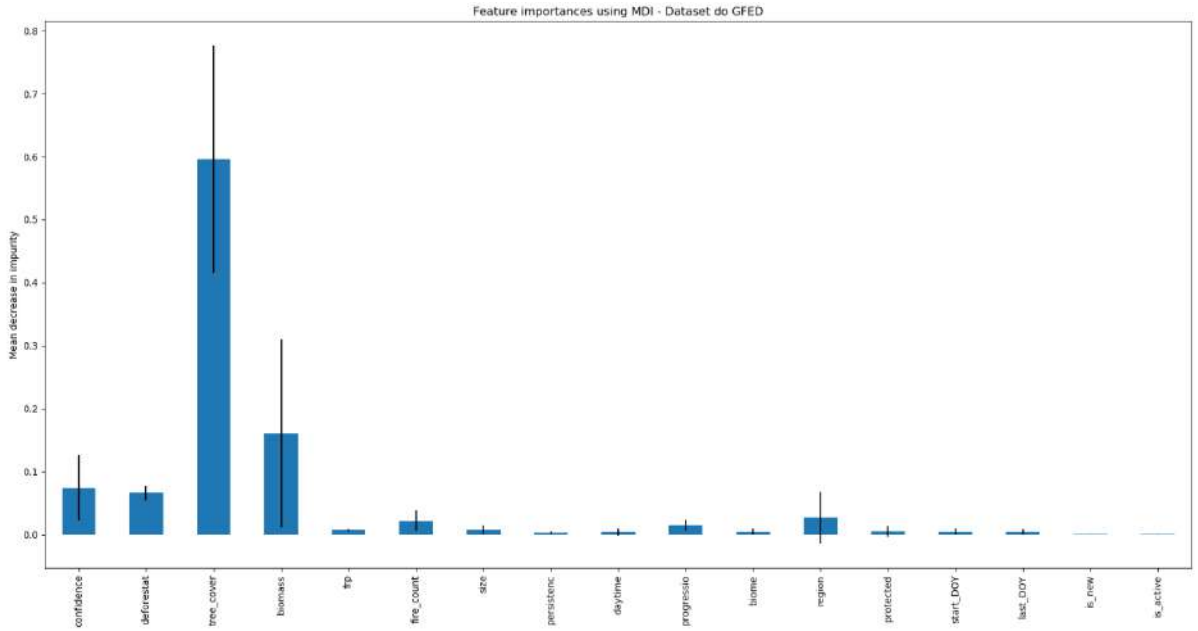

Figura 4.2 – Importância dos Atributos na classificação da tipagem do fogo para o dataset do GFED. Fonte: Autor.

Para determinar a importância de cada característica (*feature*), foi aplicada a técnica de *Feature Importance* utilizando o método de *Maximum Dependence Injection* (MDI), visto na Figura 4.2. Este método é utilizado para avaliar a importância de uma característica na previsão de uma variável dependente. A técnica consiste em inserir perturbações aleatórias em cada feature e verificar a variação na performance do modelo. A característica com a maior variação na performance é considerada a mais importante.

Assim, observamos que as características "Índice de Cobertura de Árvores", "Status de Desflorestamento Local", "Confiança" e "Biomassa" apresentaram-se como as mais relevantes para a classificação em questão e, consequentemente, foram as primeiras a serem incluídas no dataset do CENSIPAM. Contudo, é importante destacar que outras características também foram incluídas no processo posteriormente.

4.1.2 Features provindas de GeoTIFFs

Nesta subseção, o objetivo é apresentar as features adicionadas obtidas a partir de diversos GeoTIFFs. Para isso, foi desenvolvido um script geral que foi aplicado em todas as características

que serão descritas, exceto para a característica "cobertura do solo", para a qual foi elaborado um script específico que será detalhado posteriormente.

O script foi desenvolvido utilizando a linguagem de programação Python [15], com o auxílio das bibliotecas GeoPandas [22] e RasterIO [**rasterio**]. Essa função tem como objetivo calcular a média do valor de uma determinada feature (representada por um arquivo TIFF) dentro de polígonos em um shapefile de eventos. A ideia é que, para cada polígono no shapefile, seja recortado o valor da feature dentro do polígono e, em seguida, seja calculada a média desses valores.

A função aceita três parâmetros:

- **event_gdf**: um GeoDataFrame com os polígonos dos eventos
- **tiff_path**: caminho para o arquivo TIFF com a feature
- **feature_name**: nome da coluna a ser adicionada ao GeoDataFrame de saída com a média calculada.

Na Figura 4.3, temos a visualização do esquemático da função que abre o arquivo TIFF de entrada e, para cada polígono no GeoDataFrame de entrada, é feito o recorte da imagem para apenas conter a informação dentro do polígono. Em seguida, a média dos valores da feature dentro do polígono é calculada. Caso o polígono esteja em uma área sem valores (rios ou mares, por exemplo), o valor da média será 0.

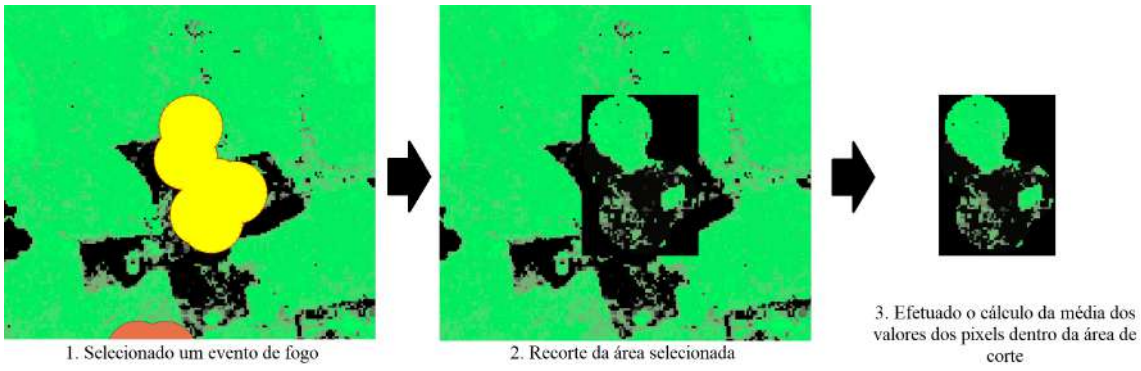

Figura 4.3 – Esquemático detalhando como é feito o cálculo do índice médio de cobertura de árvores em cada polígono. Fonte: Autor.

O resultado final é um GeoDataFrame com a coluna adicionada com a média da feature dentro de cada polígono. A função também imprime a porcentagem de polígonos já processados, para facilitar o acompanhamento do processo. A função implementada para realizar esta operação também pode ser visualizada na Figura 4.4.

```python
def mean_overlay_tiff_shp(event_gdf, tiff_path, feature_name):

    feature_array = []

    # Abre o raster de entrada e recorta o raster de saída
    with rasterio.open(tiff_path) as src:
        for index in range(len(event_gdf)):
            polygon = event_gdf.loc[index]  #Índice do polígono que está sendo analisado

            out_image, out_transform = rasterio.mask.mask(src, [polygon["geometry"]], crop=True, nodata= -1)
            out_meta = src.meta
            out_meta.update({
                    "height": out_image.shape[1],
                    "width": out_image.shape[2],
                    "transform": out_transform,}
            )

            # Transforma a matriz da imagem num array de uma dimensão
            pixels_values = out_image.ravel()

            # Pra casos em que o polígono está no rio ou mar, ou seja, 0
            if len(np.unique(pixels_values)) == 1 and np.unique(pixels_values)[0] == -1:
                feature_mean = 0
                feature_array.append(feature_mean)
            else:
                pixels_values = pixels_values[pixels_values != -1] # Desconsidera pixels fora da borda do polígono
                feature_mean = pixels_values.mean()
                feature_array.append(feature_mean)

            if index % 25000 == 0:
                print('Porcentagem da interseção (Cobertura de Árvores):', (round(((index/len(event_gdf))*100),2)), '%.')
                clear_output()
                print('Porcentagem da interseção (Cobertura de Árvores):', (round(((index/len(event_gdf))*100),2)), '%.')

    event_gdf[feature_name] = feature_array

    return event_gdf
```

Figura 4.4 – Script para realizar o overlay e cálculo descrito acima. Fonte: Autor.

4.1.2.1 Índice de Cobertura de Árvores

Conforme referenciado pelo próprio GFED, a base de dados que eles utilizam é a do *Hansen Global Forest Change* [18], visto na Figura 4.5. Ela consiste em uma base de dados global, em formato de GeoTiff, que fornece informações sobre mudanças na cobertura florestal, incluindo desmatamento e regeneração, usando imagens de satélite. É desenvolvido pelo Laboratório de Observação da Terra da Universidade de *Maryland* e é amplamente utilizado por pesquisadores, governos e organizações de conservação para avaliar a situação da floresta em todo o mundo. A base de dados inclui informações sobre mudanças na cobertura florestal desde 2000 e atualiza-se regularmente com informações atualizadas.

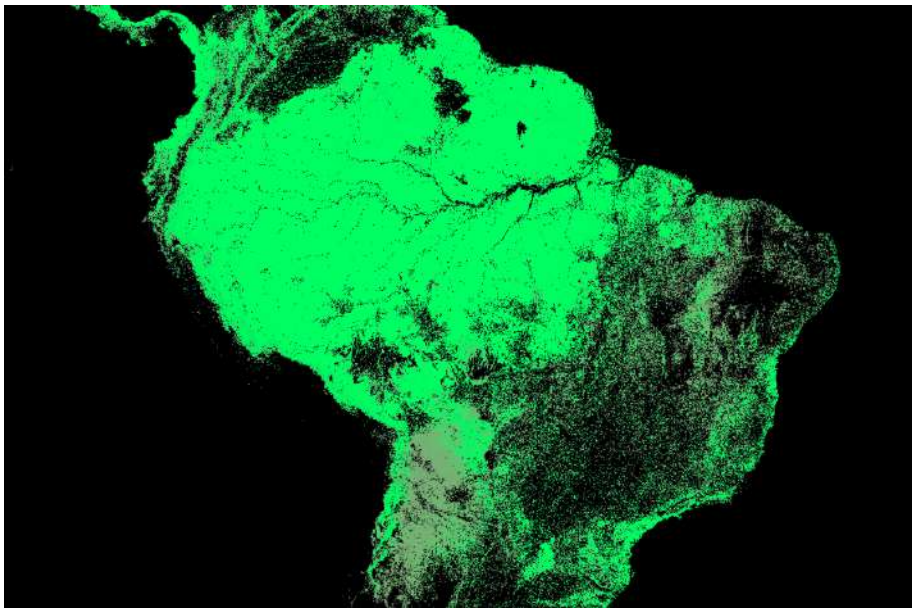

Figura 4.5 – Índice de cobertura de árvores. Fonte: *Hansen Global Forest Change*.

Cada pixel na imagem resultante da amostragem representa a porcentagem de cobertura de árvores naquele local, determinada pela fechamento da copa para toda a vegetação com mais de 5 metros de altura. Desse modo é feito o cálculo da média desse índice de cobertura de árvores dentro de cada área do evento de fogo e, assim, tais dados são adicionados ao GeoDataframe.

4.1.2.2 Biomassa

De acordo com a própria pesquisa do GFED, a biomassa na região de eventos de fogo florestais é relevante para determinar o tipo de fogo presente. No entanto, não está claro qual tipo de biomassa é usado nessa pesquisa. Por esse motivo, para esse estudo, utilizou-se a biomassa fornecida pelo GlobBiomass [43].

Os dados, representados pela Figura 4.6, incluem conjuntos globais de dados de estimativas de volume de estoque crescente (unidade: m3/ha) e biomassa acima do solo (unidade: toneladas/ha ou Mg/ha), ambos para o ano de 2010.

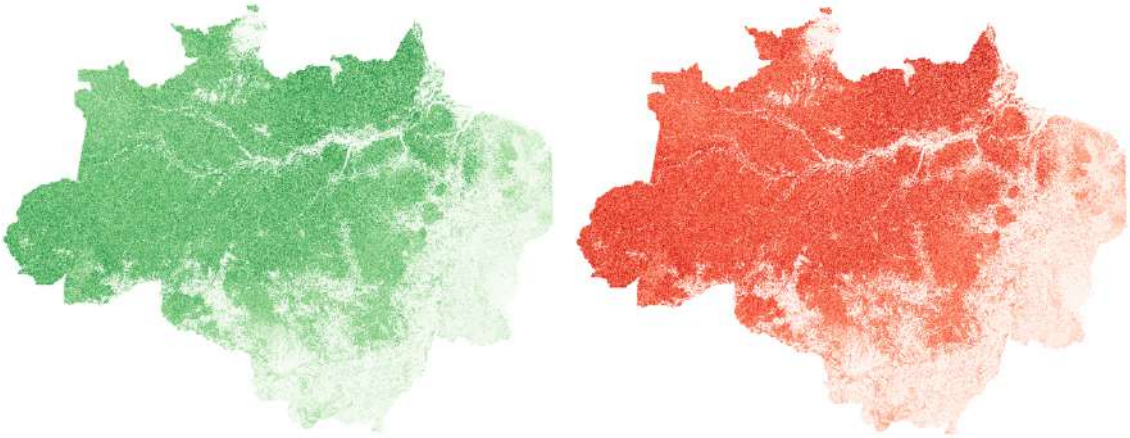

Figura 4.6 – Mapas de biomassa na respectiva ordem apresentada acima, para a Região da Amazônia Legal. Fonte: GlobBiomass

O volume de estoque crescente inclui o volume de todas as árvores vivas com mais de 10 cm de diâmetro a altura do peito, medido sobre a superfície do solo ou altura do tronco até um diâmetro de caule superior de 0 cm. Esse conjunto de dados exclui galhos, folhagem, flores, sementes, tronco e raízes. Já para a biomassa acima do solo, há a massa, expressa como peso seco ao forno, das partes lenhosas (caule, casca, galhos e galhos) de todas as árvores vivas, excluindo tronco e raízes.

A extração de features dos mapas é realizada de maneira uniforme para cada evento de incêndio, seguindo exatamente o mesmo processo utilizado para o índice médio da cobertura de árvores, conforme já demonstrado na figura 4.3. O algoritmo itera sobre cada polígono que representa um evento de fogo, e utiliza a geometria do polígono para selecionar a área correspondente na imagem GeoTiff. Em seguida, a média dos pixels na área é calculada e inserida no GeoDataframe.

4.1.2.3 Cobertura e uso do solo

Apesar de não ser utilizado pelo GFED, de acordo com realizados pelo Instituto Nacional de Pesquisas Espaciais (INPE) e pelo Instituto Brasileiro do Meio Ambiente e dos Recursos Naturais Renováveis (IBAMA), saber a cobertura do solo é essencial para se tipificar um incêndio florestal. Ela é dada pelo conjunto de vegetação e outros elementos presentes na superfície terrestre que cobrem e protegem o solo [30].

Tal informação utilizada para as equipes de combate ao incêndio, pois ajuda a determinar as melhores estratégias de combate e a prever o comportamento da queimada. Por exemplo, uma queimada em uma área com cobertura de floresta tropical seca será mais difícil de ser controlada do que uma queimada em uma área com cobertura de campo aberto, devido à densidade da vegetação e à presença de material inflamável na floresta. Além disso, a cobertura do solo também influencia na quantidade de água disponível para combater o incêndio, o que é importante para as equipes de combate ao incêndio.

A cobertura e o uso do solo no Brasil são monitorados pela plataforma colaborativa MapBiomas, visto na Figura 4.7. A plataforma foi criada com o objetivo de fornecer informações precisas e atualizadas sobre a evolução da cobertura florestal no país, bem como auxiliar na identificação de áreas de risco para eventos de fogo florestais e na tomada de decisões de conservação. Para produzir seus mapas, a plataforma utiliza dados de satélite combinados com tecnologias de inteligência artificial, e disponibiliza estes mapas gratuitamente na internet, atualizados anualmente [40]. Tal mapa conta com 63 tipos de cobertura do solo diferentes, listadas na documentação da plataforma.

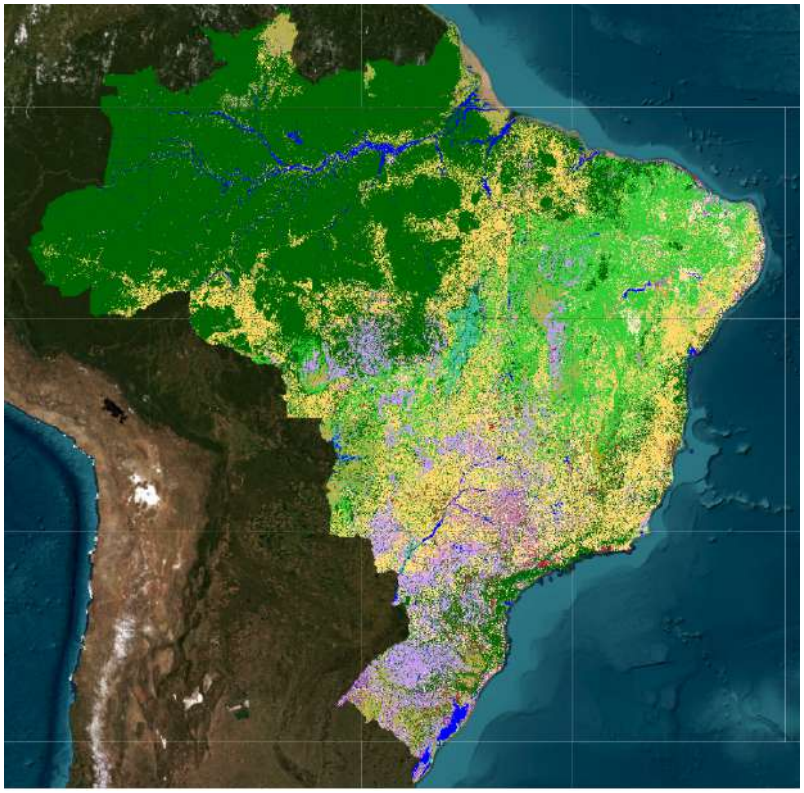

Figura 4.7 – Mapas de uso e cobertura do solo. Fonte: MapBiomas.

Assim, utilizamos o mapa de cobertura do solo de 2019, disponibilizado pela plataforma MapBiomas em formato GeoTIFF, para download. Durante esse processo, são extraídas características diferentes das mencionadas anteriormente a partir deste mapa.

O recorte da amostra, baseado na área do evento de incêndio florestal, é realizado da mesma forma anteriormente descrita, como visto na Figura 4.8. A única diferença está na estimativa da área ocupada por cada tipo de solo dentro do evento, a fim de determinar a porcentagem de cada tipo de cobertura de solo dentro daquele evento.

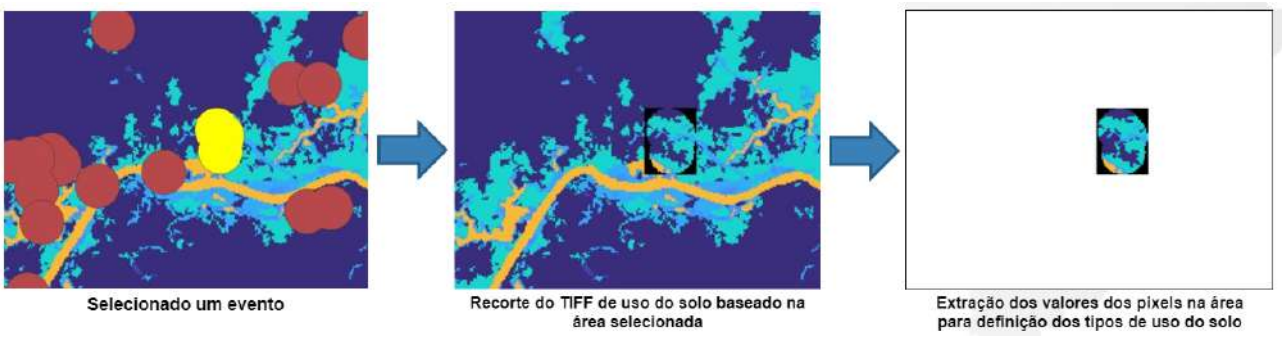

Figura 4.8 – Esquemático detalhando como é feito o recorte do tipo de cobertura do solo.

A resolução utilizada pelo MapBiomas para gerar este mapa é de 30 metros, com cada pixel representando uma área de aproximadamente 900 metros quadrados. A contagem dos pixels com determinado valor dentro da amostra é realizada, seguida da multiplicação deste valor por 900 para se obter a área estimada ocupada por cada tipo de solo na amostra, conforme visto na equação 4.1. Finalmente, a divisão da área de cada tipo de cobertura de solo pela área total do evento de incêndio florestal determina a porcentagem representada por cada tipo de cobertura dentro daquele evento.

$$A_{tipo_x} = n_{pixels} * 900 \tag{4.1}$$

A função em Python, para toda essa operação, é descrito no código disponibilizado abaixo:

```python
def coverage_features(event_gdf, tiff_path):
    # Criação de um array que abrigará todos os arrays referentes ao uso de solo e seu percentual
    area_root_array = []
    perc_root_array = []

    # Declaração dos 63 arrays que receberão append
    for i in range(63):
        area_root_array.append([])
        perc_root_array.append([])

    # Como desconsideramos os pixels de valor 0, já os setamos todos como 0
    area_root_array[0] = [0] * len(event_gdf)
    perc_root_array[0] = [0] * len(event_gdf)

    # Abre o raster de entrada e recorta o raster de saída
    with rasterio.open(tiff_path) as src:
        for index in range(len(event_gdf)):
            # Índice do polígono que está sendo analisado
            polygon = event_gdf.loc[index]

            out_image, out_transform = rasterio.mask.mask(src, [polygon["geometry"]], crop=True)
            out_meta = src.meta
            out_meta.update({
                    "height": out_image.shape[1],
                    "width": out_image.shape[2],
                    "transform": out_transform,}
            )

            for j in range(63):
                if j != 0:
                    contagem_pixels = (out_image == j).sum()
                    area_pixels = (contagem_pixels * 900)/1000000
                    area_root_array[j].append(round(area_pixels,5))
                    perc_root_array[j].append(round((area_pixels/polygon["area_km2"]),4))

            if index % 25000 == 0:
                print('Porcentagem da interseção (Uso do Solo):', (round(((index/len(event_gdf))*100),2)), '%.')
                clear_output()
                print('Porcentagem da interseção (Uso do Solo):', (round(((index/len(event_gdf))*100),2)), '%.')

    # Passagem das features extraídas para o GeoDataFrame
    for i in range(63):
        area_column_name = 'a_cover_' + str(i)
        perc_column_name = 'p_cover_' + str(i)
        event_gdf[area_column_name] = area_root_array[i]
        event_gdf[perc_column_name] = perc_root_array[i]

    clear_output()

    return event_gdf
```

Figura 4.9 – Função que realiza o overlay e extrai os dados de área e percentual ocupados por cada tipo de cobertura do solo presente nos evento de fogo.

4.1.3 Features provindas de Shapefiles

Nesta subseção, nosso foco é detalhar as features adicionadas por meio da interseção de dois conjuntos de polígonos em formato de shapefiles. Para isso, é essencial compreender o processo de interseção realizado entre os conjuntos de polígonos, utilizando a biblioteca GeoPandas, visando extrair as features a serem adicionadas ao dataset.

Primeiramente, foi então desenvolvido uma função para se realizar um overlay, conforme podemos ver no script disponibilizado abaixo.

```python
def overlay_two_shps(event_gdf, path_interest_shp):
    # Faz a leitura do shape que fará o overlay
    interest_gdf = gpd.GeoDataFrame.from_file(path_interest_shp, geometry='geometry')

    # Configura o CRS do evento para fazer a interseção entre ambos os SHPs
    try:
        event_gdf = event_gdf.to_crs(crs=4326)
        interest_gdf = interest_gdf.to_crs(crs=4326)
    except:
        try:
            event_gdf = event_gdf.to_crs(crs=4326)
            interest_gdf = interest_gdf.set_crs(crs=4326)
        except:
            try:
                event_gdf = event_gdf.set_crs(crs=4326)
                interest_gdf = interest_gdf.set_crs(crs=4326)
            except:
                event_gdf = event_gdf.set_crs(crs=4326)
                interest_gdf = interest_gdf.to_crs(crs=4326)

    # Realiza o overlay
    overlayed_shp = gpd.overlay(interest_gdf, event_gdf, how='intersection')

    # Converte as coordenadas para metros, para assim calcularmos a área dos polígonos
    aux_gdf = overlayed_shp.to_crs(crs=3857)
    inter_area = aux_gdf.area.to_list()
    inter_area = list(map(lambda x: x/1000000, inter_area))

    # Passa o valor da área em km2 para cada uma dos polígonos
    overlayed_shp['pol_area'] = inter_area

    # Deleta as variáveis que não possuem mais utilidade
    del interest_gdf, aux_gdf, inter_area

    return overlayed_shp
```

Figura 4.10 – Função que realiza o overlay entre dois shapefiles.

Ele realiza o overlay (ou interseção) entre dois shapefiles, "event_gdf"e "path_interest_shp", e retorna o shape resultante.

A leitura do shapefile "path_interest_shp"é feita utilizando a biblioteca GeoPandas, armazenando-o na variável "interest_gdf". Em seguida, o sistema de coordenadas (CRS) de ambos os shapefiles é ajustado para o padrão 4326. Essa etapa é importante para garantir a correta realização da interseção.

Após o ajuste do CRS, o overlay é realizado com a função "gpd.overlay"da biblioteca GeoPandas. A interseção dos polígonos é feita com o argumento "how=’intersection’".

As coordenadas dos polígonos resultantes são convertidas, então, para metros e a área desses polígonos é calculada, armazenando-se esse valor na coluna "pol_area". As variáveis que não são mais necessárias são deletadas antes de retornar o shapefile resultante do overlay.

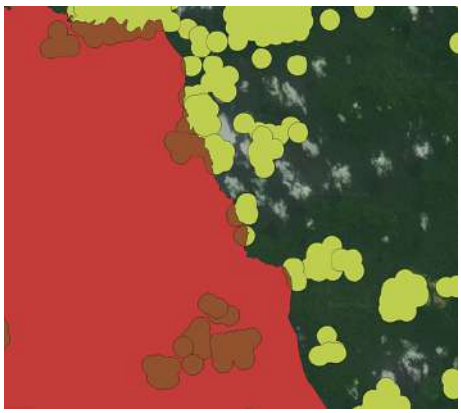

Figura 4.11 – Exemplo de overlay/interseção realizado pela função, representado pela cor marrom. Eventos representados pela cor amarela e feature a ser extraída de vermelho.

A seguir, temos uma função que tem como objetivo calcular a área resultante da interseção de dois shapefiles realizada previamente. Para isso, é necessário passar novamente como entrada o dataframe do evento "event_gdf"e o dataframe resultante da interseção "overlay_gdf", bem como o nome que se deseja dar a coluna que armazenará o valor da área.

```python
def calculate_area_from_overlay(event_gdf, overlay_gdf, area_name):
    # Nome da área de cada polígono formado na operação de overlay
    polygon_area = 'pol_area'

    new_areas = [0] * len(event_gdf)

    # Forma de busca e cruzamento de dados é feita pelo ID do evento
    intersections_ids = overlay_gdf['id_evento'].tolist()
    area_inside = overlay_gdf[polygon_area].tolist()

    for i in range(len(intersections_ids)):

        # Pega todo os polígonos formados no overlay e os soma, baseado no ID dos eventos
        index = event_gdf.loc[(event_gdf['id_evento'] == int(intersections_ids[i]))].index[0]
        total_Area_In = sum(overlay_gdf.loc[(overlay_gdf['id_evento'] == int(intersections_ids[i]))][polygon_area].to_list())
        new_areas[index] = total_Area_In

    event_gdf[area_name] =  new_areas

    del intersections_ids, area_inside, new_areas

    return event_gdf
```

Figura 4.12 – Função que realiza o cálculo da área resultante da interseção feita previamente.

A função começa por definir o nome da coluna que armazena a área dos polígonos resultantes da interseção "'pol_area'". Em seguida, são criados dois vetores, um para armazenar os IDs dos eventos que foram utilizados na interseção e outro para armazenar as áreas de cada polígono formado pela interseção.

Dessa forma, o código itera sobre cada um dos IDs dos eventos presentes na interseção, localiza o index no dataframe do evento correspondente ao evento em questão e soma todas as áreas dos polígonos resultantes da interseção para aquele evento. O resultado é então armazenado em uma lista de áreas, que é adicionada como uma nova coluna no dataframe do evento.

Após todas as iterações, a função retorna o dataframe do evento, agora com a coluna adicional que armazena a área resultante da interseção.

Por fim, temos mais uma função, que calcula a porcentagem da área adicionada. A função leva como entrada um GeoDataFrame "event_gdf"que representa o evento e um nome de coluna "area_name"que representa o nome da área da feature adicionada. Além disso, a função também precisa de um nome de coluna "percentage_area_name"para armazenar o resultado do cálculo.

```python
def percentage_area_in(event_gdf, area_name, percentage_area_name):

    total_area = event_gdf['area_km2'].tolist()
    target = event_gdf[area_name].tolist()

    # Divide os termos de X por Y (através do seu respectivo index)
    percentage_array = [x / y for x, y in zip(target, total_area)]

    # Tudo que for maior que 1, ele arredonda pra 1, já que os cálculo da área
    # das features não é 100% preciso
    percentage_array = [1 if i > 1 else i for i in percentage_array]

    event_gdf[percentage_area_name] = percentage_array

    # Deleta as variáveis que não possuem mais utilidade
    del percentage_array, total_area, target

    return event_gdf
```

Figura 4.13 – Função que realiza o cálculo da porcentagem ocupada pela área resultante da interseção feita previamente.

O cálculo é feito dividindo a área do da feature dentro da área do polígono total do evento de fogo, armazenando o resultado em uma lista. Em seguida, a função arredonda o resultado para 1 se o resultado for maior que 1, já que o cálculo da área das features pode não ser 100% preciso. Por fim, a função adiciona a lista como uma nova coluna ao GeoDataFrame "event_gdf"com o nome especificado em "percentage_area_name".

Finalmente, a função retorna o GeoDataFrame atualizado com as novas informações obtidas a partir do cálculo do overlay. Algumas funções adicionais foram necessárias para lidar com as informações específicas relacionadas ao DETER, mas em geral, as funções apresentadas foram suficientes para lidar com as outras features incluídas no dataset, que serão apresentadas a seguir.

4.1.3.1 Áreas Indígenas e Unidades de Conservação Ambiental

Como mostrado pelo GFED, é considerado o fato de se determinar se os eventos de incêndio ocorreram em territórios indígenas ou em unidades de conservação.

Os eventos de fogo florestais que ocorrem em reservas indígenas têm impactos significativos e duradouros na biodiversidade e nas comunidades locais. Esses eventos de fogo podem prejudicar a saúde dos ecossistemas e interferir nas formas de vida das comunidades indígenas. No entanto, algumas queimadas em zonas indígenas são utilizadas como uma técnica de manejo de terras para a agricultura, e são importantes para a fertilidade do solo e para a preservação de sistemas agrícolas tradicionais nas regiões amazônicas. Nesse sentido, é importante reconhecer que essas queimadas controladas não requerem a mesma atenção das forças de combate a incêndios, evitando ações desnecessárias e gasto de recursos públicos, e assim, as forças de combate a incêndios podem se

concentrar em eventos mais críticos.

Portanto, a fim de correlacionar os dados dos eventos de fogo fornecidos pelo CENSIPAM com as áreas indígenas, empregamos o shapefile intitulado "Áreas Indígenas na Amazônia Legal", disponibilizado pelo Terrabrasilis, conforme ilustrado na Figura 4.14 e disponível para download na aba correspondente [38].

Figura 4.14 – Áreas Indígenas na Amazônia Legal, visualizada no QGIS.

A identificação de incêndios florestais ocorrendo em unidades de conservação ambiental também é fundamental para a avaliação dos impactos desses eventos sobre o meio ambiente. Unidades de conservação ambiental são áreas designadas com o objetivo de preservar a biodiversidade, a paisagem e os recursos naturais, incluindo espécies ameaçadas de extinção. No entanto, essas áreas também estão sujeitas a incêndios florestais, que podem resultar na destruição de habitats e na perda de vida selvagem, além de serem frequentemente afetadas por queimadas relacionadas ao desflorestamento. Portanto, é importante conhecer a ocorrência de incêndios florestais em unidades de conservação ambiental para avaliar seus impactos e desenvolver estratégias adequadas de prevenção e resposta.

Dessa maneira, com o objetivo de estabelecer uma correlação entre os dados de eventos de fogo disponibilizados pelo CENSIPAM e as áreas de Unidade de Conservação Ambiental, conforme ilustrado na Figura 4.15, recorremos ao shapefile fornecido pelo Terrabrasilis, o qual está acessível na seção de downloads.

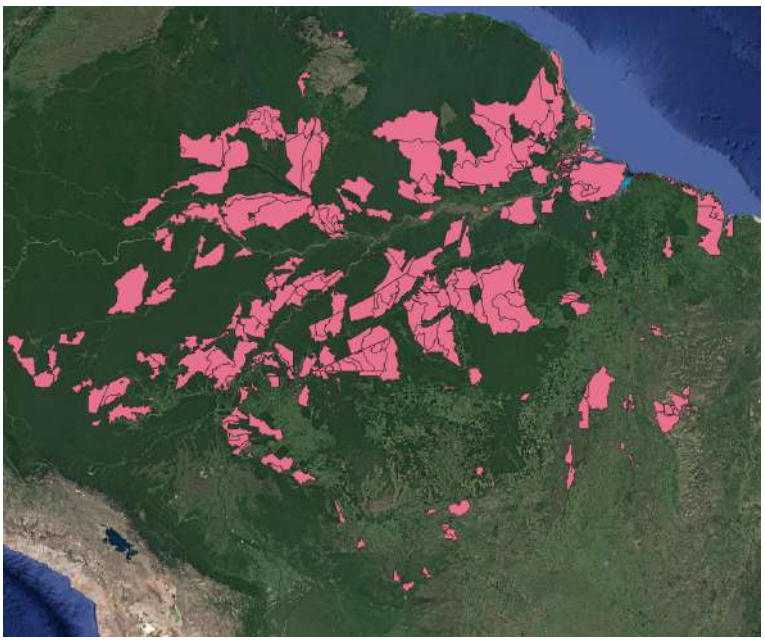

Figura 4.15 – Unidades de Conservação Ambiental na Amazônia Legal, visualizada no QGIS.

4.1.3.2 Status de Desmatamento

Com base nas informações fornecidas pelo GFED, a análise dos eventos de eventos de fogo florestais engloba a avaliação de sua ocorrência em regiões previamente sujeitas a desmatamento. Em colaboração com o CENSIPAM, foi acordado que, para a criação de atributos, se utilizaria como referência o mapa de alerta de desmatamento proveniente do Sistema de Detecção do Desmatamento em Tempo Real (DETER) [37].

A análise dos dados dos eventos de eventos de fogo florestais incluiu, então, a verificação da ocorrên- cia em áreas previamente desmatadas, com base no histórico de desmatamento monitorado pelo DETER. Para isso, é realizado um cruzamento de dados para identificar se houve algum histórico de desmatamento nas áreas afetadas pelos eventos de fogo. Caso positivo, as informações sobre a área desmatada e o percentual desta área dentro do evento de incêndio são incluídas como features na análise.

Além disso, empregou-se uma função adicional destinada à filtragem de um GeoDataFrame gerado pela sobreposição dos alertas de desmatamento do sistema DETER com os incidentes de incêndio. Essa filtragem opera com base nas datas registradas em ambas as bases de dados, com o propósito de eliminar alertas de desmatamento sem conexão com eventos de fogo ou que ocorram fora de uma janela temporal de 400 dias. A operação de filtragem compara as datas dos alertas de desmatamento com as datas dos incidentes de incêndio, eliminando os alertas que não atendem aos critérios predefinidos no GeoDataFrame resultante. A finalidade subjacente a essa filtragem é a seleção de alertas de desmatamento com maior probabilidade de estar diretamente ligados aos eventos de fogo, a fim de viabilizar uma análise mais precisa da relação entre esses dois fenômenos.

```python
def deter_filter(deter_overlay_gdf):

    # Cria uma variável auxiliar para manipular o GeoDataframe
    deter_overlay_gdf_aux = deter_overlay_gdf

    # Lista para receber a diferença de tempo no qual ocorreu entre o evento
    # de fogo e o alerta de desmatamento do DETER
    delta_time = []

    # Cria a lista que receberá os índices que deverão ser eliminados após
    # a aferição dos alertas que não se enquadram nos critérios
    indexes_out_400range = []
    indexes_alerts_after = []

    # Transforma as colunas das datas dos alertas de desmatamento e
    # eventos de fogo em listas, para uma melhor otimização
    desm_dates = deter_overlay_gdf_aux['VIEW_DATE'].to_list()
    event_dates = deter_overlay_gdf_aux['dt_minima'].to_list()

    for i in range(len(event_dates)):
        # Pega a data do evento e subtrai pela data do alerta de desmatamento
        try:
            event_date = datetime.strptime(event_dates[i][:10], '%Y-%m-%d')
        except:
            event_date = event_dates[i]

        desm_date = datetime.strptime(desm_dates[i], '%Y-%m-%d')
        delta = event_date - desm_date
        delta = delta.days
        delta_time.append(delta)

        # Para casos em que o alerta de desmtamento aconteceu depois do
        # evento de fogo, capturamos seus índices para depois os deletarmos
        if 0 > delta:
            indexes_alerts_after.append(i)

        # Para casos em que o alerta de desmtamento aconteceu depois do
        # evento de fogo ou antes de 400 dias, capturamos seus índices
        # para depois os deletarmos
        if 0 > delta or delta > 400:
            indexes_out_400range.append(i)

    # Aplica ambos os filtros, de acordo com as necessidades especificadas
    deter_overlay_gdf_aux['delta_date'] = delta_time
    deter_simple_filtered_gdf = deter_overlay_gdf_aux.drop(indexes_alerts_after)
    deter_filtered_400 = deter_overlay_gdf_aux.drop(indexes_out_400range)

    # Reseta o índice dos GeoDataframes
    deter_simple_filtered_gdf = deter_simple_filtered_gdf.reset_index()
    deter_filtered_400 = deter_filtered_400.reset_index()

    # Deleta as variáveis que não possuem mais utilidade
    del deter_overlay_gdf_aux, indexes_out_400range, indexes_alerts_after, delta_time

    return deter_simple_filtered_gdf, deter_filtered_400
```

Figura 4.16 – Código de filtragem dos overlays do DETER.

A primeira parte da função cria algumas listas vazias para armazenar resultados intermediários. Em seguida, a função percorre cada linha do GeoDataFrame "deter_overlay_gdf", extrai as datas dos eventos de fogo e dos alertas de desmatamento e calcula a diferença de tempo entre elas em dias. Essa diferença é armazenada na lista "delta_time".

Em seguida, a função adiciona o índice das linhas cujo alerta de desmatamento ocorreu depois do evento de fogo na lista "indexes_alerts_after"e o índice das linhas cujo alerta de desmatamento ocorreu antes de 400 dias após o evento de fogo na lista "indexes_out_400range".

Na próxima etapa, a função cria duas novas variáveis GeoDataFrame "deter_simple_filtered_gdf"e "deter_filtered_400", que correspondem ao GeoDataFrame original, mas sem as linhas cujos alertas de desmatamento ocorreram depois do evento de fogo e sem as linhas cujos alertas de desmatamento ocorreram antes de 400 dias após o evento de fogo, respectivamente.

Assim, além de calcular a área e percentual que esses eventos de fogo ocorreram dentro de um alerta de desmatamento, também temos uma função apelidada de "calculate_delta_time_from _deter". Ela tem como objetivo calcular a diferença em dias entre a data de cada evento de fogo

e a data mais recente de um alerta de desmatamento na mesma área.

```python
def calculate_delta_time_from_deter(event_gdf, overlay_gdf):

    # Forma de busca e cruzamento de dados é feita pelo ID do evento e geração da lista para saber o dia mais
    # recente que houve desmatamento na área
    min_delta_time = [9999] * len(event_gdf)
    intersections_ids = overlay_gdf['id_evento'].tolist()

    for i in range(len(intersections_ids)):

        #Observa, a partir do ID, qual foi a data mais recente que houve um alerta de desmatamento
        index = event_gdf.loc[(event_gdf['id_evento'] == int(intersections_ids[i]))].index[0]
        most_recent_date = min(overlay_gdf.loc[(overlay_gdf['id_evento'] == int(intersections_ids[i]))]['delta_date'].to_list())
        min_delta_time[index] = most_recent_date

    event_gdf['deter_days'] =  min_delta_time

    # Deleta as variáveis que não possuem mais utilidade
    del min_delta_time, intersections_ids

    return event_gdf
```

Figura 4.17 – Código cálculo de dia dos overlays do DETER.

A função utiliza o ID do evento de fogo para buscar na base de dados de alertas de desmatamento a data mais recente em que houve um alerta naquela mesma área. Para isso, são geradas duas listas de IDs (uma de eventos de fogo e outra de alertas de desmatamento) e é feita uma busca cruzada para obter a data mais recente de um alerta de desmatamento na área em que ocorreu o evento de fogo.

A diferença entre a data do evento e a data mais recente de um alerta de desmatamento é armazenada em uma nova coluna chamada "deter_days"no GeoDataFrame de eventos de fogo.

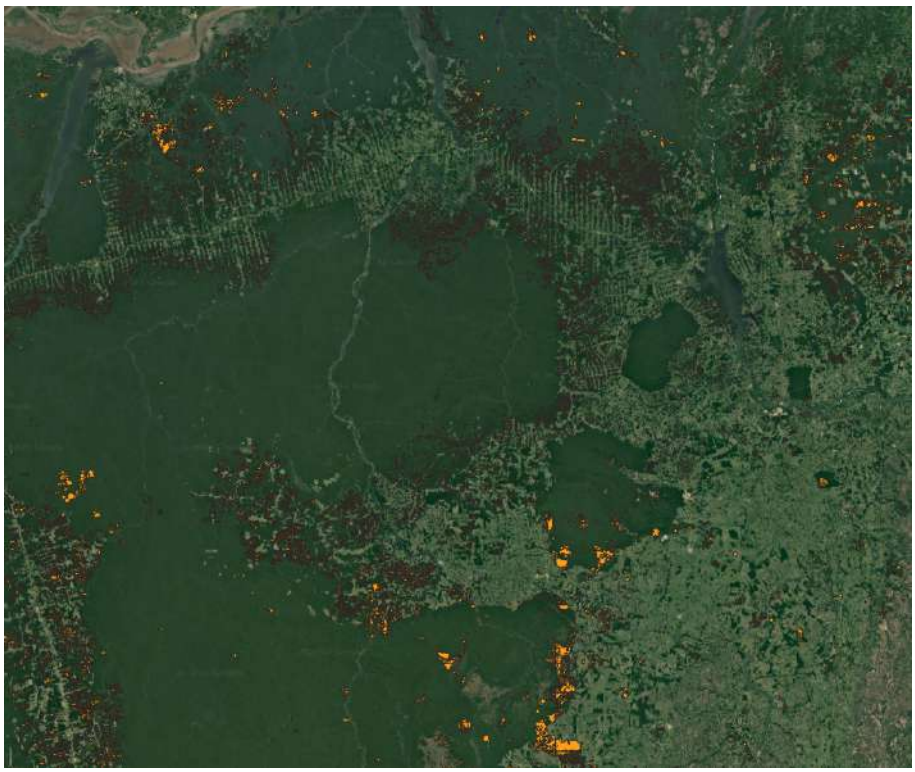

Figura 4.18 – Exemplo de alertas de desmatamento do DETER

4.1.3.3 Áreas Públicas, Privadas, Desconhecidas e Áreas de Cadastro Ambiental Rural

A classificação precisa de eventos de fogo com base na identificação das áreas afetadas pelo fogo e sua relação com diferentes tipos de propriedade (pública, privada ou desconhecida) desempenha um papel crucial na compreensão dos impactos ambientais e na efetividade das medidas de gestão e prevenção.

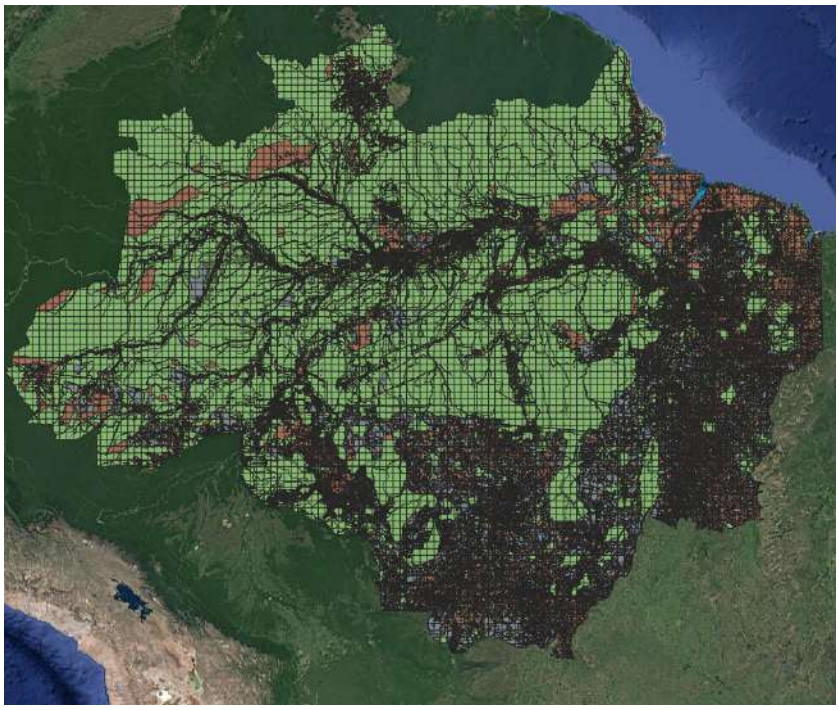

Figura 4.19 – Mapeamento das terras públicas, privadas e desconhecidas.

A determinação das áreas atingidas pelo fogo e sua vinculação com a propriedade da terra têm implicações profundas na responsabilização e tomada de medidas corretivas. Ao identificar se a área afetada pertence a terras públicas, privadas ou possui status desconhecido, as autoridades competentes podem adotar abordagens específicas para lidar com os impactos. Isso facilita a aplicação de políticas adequadas, como multas, direcionamento de recursos para recuperação e implementação de estratégias de prevenção de eventos de fogo.

A categorização dos eventos de fogo com base na propriedade da terra também é crucial para o combate ao fogo e a compreensão das dinâmicas socioeconômicas por trás dos eventos de fogo. Ao analisar como diferentes tipos de propriedade são afetados, é possível identificar padrões de uso da terra que podem estar relacionados à ocorrência de eventos de fogo. Isso ajuda a compreender as práticas de manejo, a gestão dos recursos naturais e a influência das atividades humanas nos eventos de fogo florestais.

Além disso, a relação entre a propriedade da terra e os eventos de fogo pode fornecer insights valiosos para a prevenção futura. Ao entender como fatores como posse da terra, gestão e uso afetam a incidência de eventos de fogo, as autoridades podem desenvolver estratégias direcionadas para minimizar os riscos. Isso pode incluir a implementação de regulamentações mais rigorosas em áreas com propriedade desconhecida ou a promoção de práticas de manejo sustentável em terras

privadas.

Seguindo a mesma lógica de utilização do registro de terras, a incorporação dos dados de Áreas de Cadastro Ambiental Rural (CAR) na classificação de eventos de fogo representa um avanço significativo na compreensão dos impactos ambientais e na eficácia das estratégias de categorização e gestão.

As informações contidas no CAR oferecem um panorama abrangente das características ambientais e do uso da terra em propriedades rurais. A inclusão desses dados na classificação de eventos de fogo permite uma análise mais detalhada das áreas afetadas, considerando não apenas a propriedade da terra, mas também aspectos como a presença de áreas de preservação permanente, reservas legais e outras delimitações previstas pela legislação ambiental.

Através da integração dos dados do CAR na categorização de eventos de fogo florestais, é possível identificar quais áreas possuem maiores restrições e obrigações legais de conservação. Isso contribui para classificar focos de incêndio, assim como direcionar os esforços de recuperação de acordo com os contextos específicos de cada propriedade, considerando as características naturais e as responsabilidades legais dos proprietários.

Além disso, a inclusão dos dados do CAR na análise de eventos de fogo proporciona uma visão mais aprofundada das práticas de manejo e do histórico de uso da terra. Isso pode revelar padrões de ocupação e exploração que influenciam a incidência de eventos de fogo. Ao entender como as atividades agrícolas, pecuárias e outras interações humanas afetam os eventos de fogo florestais, as estratégias de prevenção podem ser adaptadas para promover práticas sustentáveis.

4.1.4 Dataset Gerado

Através da metodologia proposta, aprimoramos significativamente o conjunto de dados original do CENSIPAM, enriquecendo-o com uma ampla variedade de recursos adicionais. Essa ampliação é fundamental para potencializar os algoritmos de aprendizado de máquina na detecção de padrões e informações relevantes relacionadas a eventos de incêndio. O resultado é um conjunto de dados substancialmente mais abrangente e robusto, capacitando os algoritmos a aprenderem a classificar eventos de incêndio em tempo quase real.

Ao integrar características identificadas como relevantes nos dados do GFED com aquelas disponibilizadas pelo CENSIPAM, construímos um conjunto de dados que incorpora uma diversidade significativa de recursos cruciais, todos detalhados na tabela 4.2. Esse conjunto de dados expandido e aprimorado desempenha um papel fundamental no aprimoramento das capacidades de classificação em tempo real de eventos de incêndio, proporcionando uma base sólida para análises subsequentes.

Portanto, foi formado um conjunto de dados composto por 220.739 eventos de incêndio. Dentre esse conjunto, 66% foi alocado para treinamento, enquanto 17% foi designado para validação e os 17% restantes para teste. É crucial observar que, como parte do pré-processamento para o MLP (Perceptron de Múltiplas Camadas), os dados passaram por normalização usando o Standard Scaler, que é uma técnica estatística que padroniza os dados ajustando-os para ter uma média de

Tabela 4.2 – Atributos e rótulo do conjunto de dados gerado.

Classe do Atributo	Atributo	Descrição
Atributos do Evento	ID	Número de identificação único do evento de incêndio
	Status	Fração (0-1) de células de grade de 550m desmatadas
	Data Mínima	Data de início das primeiras detecções
	Data Mais Recente	Última data de detecções de um evento
	Persistência	Duração (em dias)
	Número de detecções	Número total de focos de calor dentro da área do evento
	Área	Tamanho do incêndio em quilômetros quadrados
	Área de terra privada	Área dentro de terras privadas
	Percentual de terra privada	Percentual dentro de terras privadas
	Área de terra pública	Área dentro de terras públicas
	Percentual de terra pública	Percentual dentro de terras públicas
	Área de terras desconhecidas	Área dentro de terras desconhecidas
	Percentual de terras desconhecidas	Percentual dentro de terras desconhecidas
	Área do CAR	Área dentro de terras registradas no CAR
	Percentual do CAR	Percentual dentro de terras registradas no CAR
	Área desmatada (DETER)	Área previamente desmatada dentro da área do evento
	Percentual de desmatamento (DETER)	Percentual de desmatamento dentro da área do evento
	Biomassa Estocada no Solo	Índice de biomassa estocada no solo na área do evento
	Biomassa Acima do Solo	Índice de biomassa acima do solo na área do evento
	Cobertura de Árvores	Índice de cobertura arbórea na área do evento
	Área de Cobertura X (0 a 62)	Área dentro do evento para o tipo de uso do solo X
	Percentual de Cobertura X (0 a 62)	Percentual do tipo de uso do solo X, dentro do evento
Rótulo	Tipo de Incêndio	Classificação do tipo de evento de incêndio, combinando rótulos manuais e GFED

zero e um desvio padrão de um. Para o Random Forest, por outro lado, os dados brutos foram utilizados sem qualquer forma de transformação ou normalização.

4.2 Estrutura dos Algoritmos de Treino

Na presente seção, apresentamos uma análise meticulosa da estratégia utilizada na formulação dos códigos de treinamento direcionados aos algoritmos de Random Forest e Multi-Layer Perceptron (MLP). Ademais, detalharemos sobre os parâmetros empregados durante a etapa de treinamento dos modelos, cuja exposição minuciosa ocorrerá nas subseções subsequentes.

4.2.1 Random Forest

Este código foi criado com o propósito de realizar a classificação de eventos de fogo por meio da utilização do algoritmo Random Forest disponível na biblioteca scikit-learn. A abordagem adotada neste código abrange desde a configuração inicial até a análise estatística dos resultados, fornecendo uma compreensão completa do processo.

Inicialmente, são definidos parâmetros essenciais que guiarão a execução do código, conforme visto na Figura 4.20. Os caminhos para os conjuntos de dados de treinamento e predição são especificados, bem como as características e o alvo que serão empregados durante a classificação dos eventos.

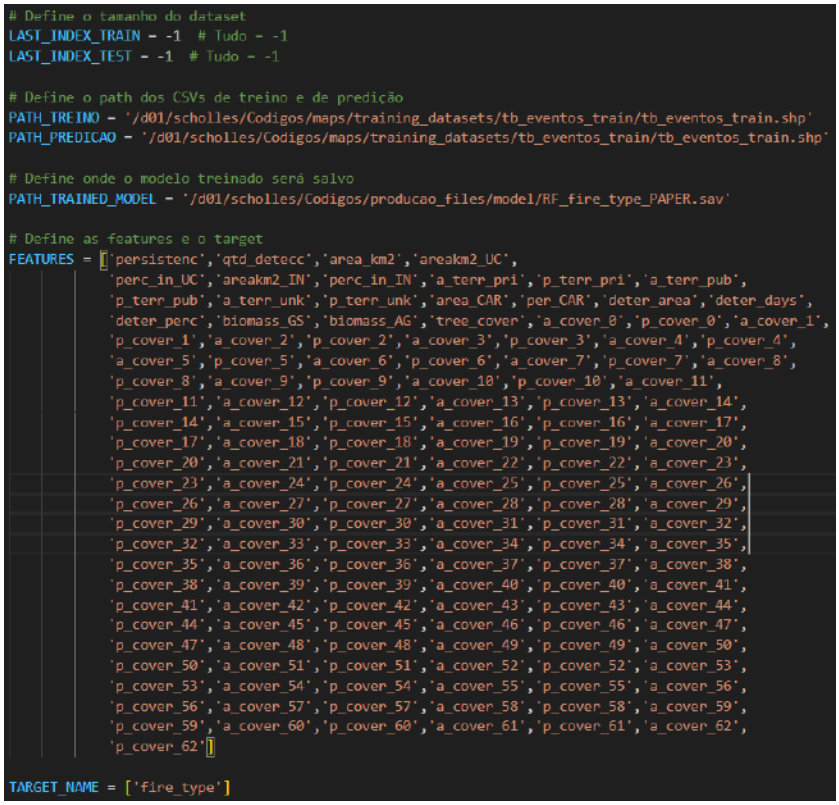

Figura 4.20 – Configuração inicial do código: especificação de conjuntos de dados, características e alvo para a classificação de eventos.

O próximo estágio envolve o pré-processamento dos dados e o treinamento do modelo, como visto na Figura 4.21. Os dados de treinamento são carregados, divididos em características (X) e rótulos (y) correspondentes aos eventos de fogo. O algoritmo Random Forest é escolhido como modelo de classificação, e o ajuste é realizado utilizando os dados de treinamento. Após o treinamento, o modelo resultante é armazenado em um arquivo no disco.

```
# Abre os CSVs de treino e de predição
csvTreino, csvPredicao = tools.open_data(PATH_TREINO, PATH_PREDICAO, LAST_INDEX_TRAIN, LAST_INDEX_TEST, FEATURES,
                                         TARGET_NAME)

X = csvTreino[FEATURES].to_numpy()
y = csvTreino[TARGET_NAME].to_numpy()

# Especifica o modelo de aprendizado de máquina e realiza o treinamento
learning_model = RandomForestClassifier(verbose=True)
print('Iniciado o treino!')
learning_model.fit(X, y.ravel())

# Salva o modelo treinado em disco
print('Iniciado o armazenamento do modelo treinado...')
pickle.dump(learning_model, open(PATH_TRAINED_MODEL, 'wb'))
```

Figura 4.21 – Treinamento do modelo Random Forest para classificação de eventos de fogo após pré-processamento dos dados.

Posteriormente, o código adentra a fase de avaliação estatística, como visto na Figura 4.22. Esta etapa compreende a previsão de rótulos nos dados de predição, utilizando o modelo previamente treinado, e a subsequente comparação dessas previsões com os rótulos reais dos eventos de fogo. O objetivo principal é avaliar a eficácia do modelo ao lidar com novos conjuntos de dados.

A análise das métricas de desempenho, tais como precisão, recall e f1-score, é fundamental para a compreensão do rendimento do modelo. O código faz uso da função classification_report para calcular e exibir essas métricas. Essa abordagem proporciona insights valiosos sobre o desempenho individual do modelo em relação a cada classe, permitindo identificar quais categorias estão sendo melhor ou pior classificadas. Por fim, uma matriz de confusão é gerada para visualizar, de forma mais tangível, os acertos e erros do modelo em cada classe.

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

print('Iniciada predição!')
x_pred = csvPredicao[FEATURES].to_numpy()
y_pred = learning_model.predict(x_pred)
y_true = csvPredicao[TARGET_NAME].to_numpy()

print(classification_report(y_true, y_pred, digits=4))

from sklearn.metrics import confusion_matrix

# Crie a matriz de confusão
cm = confusion_matrix(y_true, y_pred)
```

Figura 4.22 – Treinamento do modelo Random Forest para classificação de eventos de fogo após pré-processamento dos dados.

4.3 Multi-Layer Perceptron (MLP)

O processo de treinamento do modelo de aprendizado profundo Multi-Layer Perceptron (MLP) é executado por meio do código implementado utilizando a biblioteca PyTorch, utilizando a linguagem de programação Python. A escolha dessa biblioteca se deve à sua especialização no desenvolvimento de aplicações desse escopo. Ao longo do texto abaixo, procederemos a uma análise

da estrutura do código elaborado, explorando suas principais funções e classes que compõem o mesmo.

Começando pela classe do modelo MLP, representada de forma ilustrativa na Figura 4.23, desempenha um papel crucial na estruturação do nosso modelo de aprendizado de máquina. Sua concepção visa à compreensão de padrões complexos nos dados e à execução de tarefas de categorização. Em essência, a classe cria um "núcleo simulado"conhecido como Rede Neural Multilayer Perceptron (MLP). Essa abordagem é amplamente reconhecida e frequentemente aplicada em redes neurais. O papel fundamental desse núcleo é assimilar as relações entre os dados de entrada e suas respectivas saídas, assemelhando-se a uma relação de causa e efeito.

```python
class MLP(Module):
    def __init__(self, n_inputs):
        super(MLP, self).__init__()
        self.FL1 = Linear(n_inputs, LAYER1)
        self.act1 = ReLU()
        self.BN1 = BatchNorm1d(LAYER1)
        self.FL2 = Linear(LAYER1, LAYER2)
        self.act2 = ReLU()
        self.BN2 = BatchNorm1d(LAYER2)
        self.FL3 = Linear(LAYER2, LAYER3)
        self.act3 = ReLU()
        self.BN3 = BatchNorm1d(LAYER3)
        self.FL4 = Linear(LAYER3, LAYER4)
        self.act4 = ReLU()
        self.BN4 = BatchNorm1d(LAYER4)
        self.FL5 = Linear(LAYER4, NUM_CLASSES)
        self.BN5 = BatchNorm1d(NUM_CLASSES)
        self.softmax = Softmax(dim=-1)

    def forward(self, x):
        inter_values = []
        x = self.FL1(x)
        inter_values.append(x)
        x = self.act1(x)
        inter_values.append(x)
        x = self.BN1(x)
        inter_values.append(x)
        x = self.FL2(x)
        inter_values.append(x)
        x = self.act2(x)
        inter_values.append(x)
        x = self.BN2(x)
        inter_values.append(x)
        x = self.FL3(x)
        inter_values.append(x)
        x = self.act3(x)
        inter_values.append(x)
        x = self.BN3(x)
        inter_values.append(x)
        x = self.FL4(x)
        inter_values.append(x)
        x = self.act4(x)
        inter_values.append(x)
        x = self.BN4(x)
        inter_values.append(x)
        x = self.FL5(x)
        inter_values.append(x)
        x = self.BN5(x)
        inter_values.append(x)
        x = self.softmax(x)
        return x, inter_values
```

Figura 4.23 – Classe MLP utilizada no treinamento do modelo MLP.

Assim, construtor da classe `MLP`, é necessário fornecer o número de características de entrada

(`n_inputs`), que representa a dimensão das amostras de dados a serem processadas. A rede é composta por camadas totalmente conectadas (Fully Connected Layers), que são definidas sequencialmente. Cada camada é composta por uma transformação linear (`Linear`) seguida por uma função de ativação (`ReLU`) e uma camada de normalização em lote (`BatchNorm1d`).

O método `forward()` é onde ocorre o fluxo de informações durante a inferência. A entrada `x`, que representa os dados a serem processados, passa por cada camada definida no construtor. As ativações intermediárias em cada camada são armazenadas na lista `inter_values`, permitindo um acompanhamento detalhado do processo. As camadas de transformação linear (`Linear`), ativação (`ReLU`) e normalização em lote (`BatchNorm1d`) são aplicadas sequencialmente a `x`.

A última camada da rede é especializada para a tarefa de classificação e é definida por uma transformação linear final seguida por uma camada de normalização em lote. A função de ativação `Softmax` é aplicada para produzir uma distribuição de probabilidade sobre as classes de saída. O resultado final é a previsão da classe, juntamente com as ativações intermediárias armazenadas na lista `inter_values`.

Seguindo, temos a função `main()`, detalhada na Figura 4.27, ela desempenha um papel central no processo de treinamento do modelo. Seu objetivo é coordenar uma série de etapas cruciais para preparar, configurar e avaliar o desempenho do modelo de aprendizado de máquina. O código começa com uma mensagem de inicialização, marcando o início da normalização dos dados contidos nos arquivos SHP.

```python
def main():
    print('Inicializada a normalização geral dos CSVs...')
    csvTreino, csvPredicao = tools.normalizer_training(TARGET_NAME, 'standard', DF_TRAIN, DF_TEST)

    if  N_SAMPLES > 0:
        csvTreino = csvTreino.sample(n = N_SAMPLES)
        csvPredicao = csvPredicao.sample(n = N_SAMPLES)
    else:
        csvTreino = csvTreino.sample(frac = FRAC_SAMPLES)
        csvPredicao = csvPredicao.sample(frac = FRAC_SAMPLES)

    global WEIGHTS
    if not isinstance(WEIGHTS, type(None)):
        if isinstance(WEIGHTS, int):
            count_classes = csvTreino[TARGET_NAME].value_counts().sort_index()
            classes_freq = count_classes / count_classes.sum()
            inv_freq = 1/classes_freq
            WEIGHTS = (inv_freq/inv_freq.sum()).to_numpy()
        WEIGHTS = torch.Tensor(WEIGHTS).to(DEVICE)
    focal_loss_func = FocalLoss(weight=WEIGHTS, gamma=GAMMA)

    # TODO: implementar ideia de treinamento, validação e teste com subsets diferentes
    train_dl = DataLoader(CSVDataset(csvTreino), batch_size=BATCH_SIZE, shuffle=True)
    val_dl = DataLoader(CSVDataset(csvPredicao), batch_size=BATCH_SIZE, shuffle=False)
    model, optimizer, initial_epoch, initial_train_iter, initial_val_iter = init_or_load_checkpoint()

    print('**************************************Iniciado o treinamento!**************************************')
    run_train_on_all_epochs(model, optimizer, initial_epoch, initial_train_iter, initial_val_iter, train_dl, val_dl, foca
    print('**************************************Treinamento Encerrado!**************************************')

    print('**************************************Iniciado o teste!**************************************')
    test_and_log_metrics(model, val_dl)
    print('**************************************Encerrado o teste!**************************************')
```

Figura 4.24 – Função **main()** de execução do treinamento do modelo MLP.

A etapa de normalização é essencial para garantir que as características dos dados estejam em uma escala comum. Isso é alcançado por meio da função `tools.normalizer_training()`, que

aplica o método de escalonamento padrão aos conjuntos de treinamento e predição. Esse processo é fundamental para que o modelo possa extrair padrões e relacionamentos significativos dos dados.

Dependendo das configurações definidas, a amostragem dos dados é realizada para controlar o tamanho do conjunto de treinamento e predição. Se o número máximo de amostras (`N_SAMPLES`) for especificado, uma amostragem aleatória é executada. Caso contrário, uma fração dos dados (`FRAC_SAMPLES`) é amostrada. Essa etapa pode acelerar o treinamento e teste do modelo, especialmente quando lidando com grandes conjuntos de dados.

A definição de pesos para a função de perda é outra consideração importante. A variável `WEIGHTS` é usada para configurar os pesos da função de perda, que desempenha um papel crucial no processo de treinamento. Se `WEIGHTS` for um valor inteiro, os pesos são calculados de forma inversamente proporcional à frequência de cada classe, permitindo ao modelo dar mais atenção às classes menos frequentes.

Com as etapas de preparação concluídas, a função `main()` passa para a configuração do treinamento do modelo. Os dados de treinamento e validação são carregados em lotes usando a classe `DataLoader`, que oferece uma iteração eficiente sobre os dados durante o treinamento. O modelo é inicializado ou carregado a partir de um ponto de verificação anterior, e o processo de treinamento é iniciado.

O treinamento é executado por várias épocas, durante as quais o modelo é ajustado aos dados de treinamento para aprender padrões e representações relevantes. Uma vez que o treinamento é concluído, a função `test_and_log_metrics()` é chamada para avaliar o desempenho do modelo nos dados de validação. Isso envolve o cálculo de métricas relevantes que fornecem insights quantitativos sobre a capacidade do modelo de generalizar o aprendizado para dados não vistos.

Partindo para a função `run_train_on_all_epochs()`, temos um componente crucial na condução do processo de treinamento do modelo de machine learning. Ela é responsável por iterar sobre múltiplas épocas de treinamento e validação, monitorando o progresso e registrando métricas relevantes para análise posterior. Esta função encapsula uma série de etapas essenciais para melhorar o desempenho do modelo ao longo do tempo.

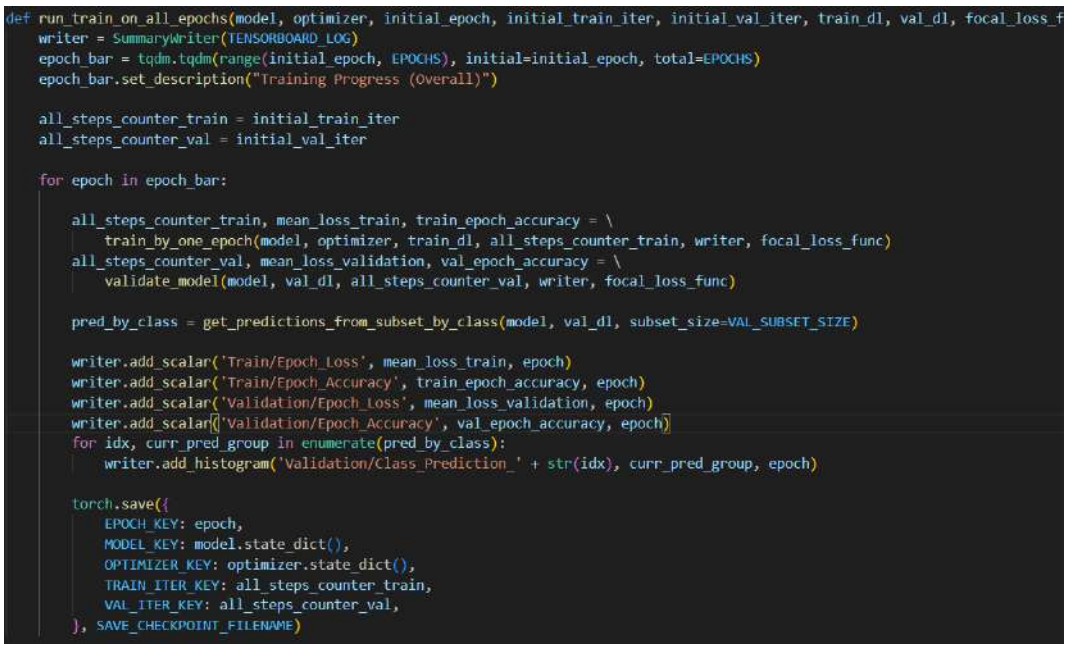

Figura 4.25 – Função **run_train_on_all_epochs()** para execução das épocas de treinamento e validação.

No início da função, é configurado um escritor (`SummaryWriter`) do TensorBoard, que permite a visualização interativa do treinamento e validação do modelo. Um barra de progresso (`tqdm.tqdm`) é inicializada para acompanhar as épocas em execução, fornecendo uma representação visual do progresso do treinamento.

Dentro do loop de épocas, a função itera sobre o conjunto de dados de treinamento e validação, executando o treinamento e a validação do modelo para cada época. A função `train_by_one_epoch()` é chamada para realizar o treinamento do modelo utilizando os dados de treinamento, otimizador e função de perda focal. Isso resulta em uma atualização dos pesos do modelo com o objetivo de aprender padrões relevantes nos dados de treinamento.

Em seguida, a função `validate_model()` é chamada para avaliar o desempenho do modelo nos dados de validação. Isso envolve a utilização dos dados de validação para calcular a perda média e a precisão da época. As previsões por classe também são obtidas usando a função `get_predictions_from_subset_by_class()`, permitindo uma análise mais detalhada do desempenho por classe.

As métricas de treinamento e validação, como a perda média e a precisão, são registradas no TensorBoard para acompanhamento e análise. Além disso, histogramas das previsões por classe são registrados, permitindo uma visualização das distribuições de previsões para cada classe ao longo das épocas.

Após o término de cada época, um ponto de verificação é salvo para permitir a retomada do treinamento a partir do ponto atual. Isso é realizado por meio da função `torch.save()`, que armazena informações como a época atual, os parâmetros do modelo e o estado do otimizador. Isso é fundamental para garantir a capacidade de retomar o treinamento e realizar análises posteriores

sem perder informações importantes.

A função `train_by_one_epoch()` desempenha um papel central na condução do processo de treinamento do modelo durante uma única época. Ela é responsável por iterar sobre os dados de treinamento em lotes, calcular a perda e atualizar os parâmetros do modelo para otimização. Essa função é fundamental para aprimorar os pesos do modelo, permitindo-lhe aprender representações significativas nos dados de treinamento.

```python
def train_by_one_epoch(model, optimizer, dataloader, all_steps_counter_train, writer, focal_loss_func):
    accuracy_fnc = Accuracy().to(DEVICE)
    training_bar = tqdm.tqdm(enumerate(dataloader), total=len(dataloader))
    training_bar.set_description("Training Progress (Epoch)")
    mean_loss_train = 0
    train_epoch_accuracy = 0

    for step_train,(x,y) in training_bar:
        x = x.cuda(DEVICE)
        y = y.cuda(DEVICE)
        y_hat, loss, inter_values = train_one_step(model, optimizer, focal_loss_func, x, y)
        mean_loss_train += loss

        y_hat_bin = y_hat.argmax(dim=1)[..., None]
        training_iteration_accuracy = accuracy_fnc(y_hat_bin.long(), y.long())
        train_epoch_accuracy += training_iteration_accuracy

        if step_train % LOG_INTERVAL == 0:
            writer.add_histogram('Train/Class_Prediction', y_hat_bin, all_steps_counter_train)
            writer.add_histogram('Train/GroundTruth', y, all_steps_counter_train)
            writer.add_scalar('Train/Iteration_Loss', loss, all_steps_counter_train)
            writer.add_scalar('Train/Iteration_Accuracy', training_iteration_accuracy, all_steps_counter_train)
            for name, param in model.named_parameters():
                if param.requires_grad:
                    writer.add_histogram('Gradients/' + name, param.grad, all_steps_counter_train)
                writer.add_histogram('Weights/' + name, param.data, all_steps_counter_train)
            for i, out in enumerate(inter_values):
                writer.add_histogram('Inter_Outputs/' + str(i), out[i], all_steps_counter_train)

        all_steps_counter_train += 1

    mean_loss_train /= len(dataloader)
    train_epoch_accuracy /= len(dataloader)

    return all_steps_counter_train, mean_loss_train, train_epoch_accuracy
```

Figura 4.26 – Função **train_by_one_epoch()** para execução de uma época de treinamento.

Na função da Figura 4.26, no início da função é inicializada uma instância da métrica de precisão (`Accuracy`) para avaliar o desempenho do modelo durante o treinamento. Um barra de progresso (`tqdm.tqdm`) é configurada para acompanhar o progresso do treinamento em lotes, fornecendo uma representação visual do progresso da época.Dentro do loop de treinamento, os dados de entrada (`x`) e os rótulos de saída (`y`) são carregados de cada lote. Esses dados são transferidos para a GPU para acelerar o processo de cálculo usando `cuda(DEVICE)`.

A função `train_one_step()` é chamada para executar uma única etapa de treinamento. Ela calcula as previsões do modelo (`y_hat`), a perda associada e as ativações intermediárias (`inter_values`). A perda é somada ao longo de todas as etapas de treinamento para calcular a perda média ao final da época. A métrica de precisão é calculada a partir das previsões binarizadas (`y_hat_bin`) e os rótulos reais (`y`). Isso fornece uma estimativa da precisão do modelo durante a iteração atual.

Em intervalos regulares definidos por `LOG_INTERVAL`, várias métricas e informações são registra-

das no TensorBoard por meio do escritor (`SummaryWriter`). Isso inclui histogramas das previsões de classe, rótulos reais, perda de iteração e precisão de iteração. Além disso, histogramas dos gradientes e pesos das camadas do modelo são registrados para acompanhamento e análise.

Após a conclusão do loop de treinamento, a perda média e a precisão média da época são calculadas dividindo os totais acumulados pelo número de iterações no dataloader. Essas métricas são retornadas pela função para uso posterior.

Seguindo, a função `train_one_step()` desempenha um papel responsável por calcular a perda e otimizar os parâmetros do modelo com base nos dados de entrada e rótulos de saída a cada iteração dentro de uma época. Essa função realiza uma única etapa de treinamento, refinando os pesos do modelo para melhor se ajustar aos dados de treinamento.

```python
def train_one_step(model, optimizer, loss_fnc, x, y):
    optimizer.zero_grad()
    y_hat, inter_values = model(x)
    one_hot_y = one_hot(y.long(), num_classes=NUM_CLASSES)
    one_hot_y = torch.squeeze(one_hot_y, dim=1)
    loss = loss_fnc(y_hat, one_hot_y.float())
    loss.backward()
    optimizer.step()

    return y_hat, loss, inter_values
```

Figura 4.27 – Função **train_one_step** para execução de uma iteração de treinamento.

No início da função, uma instância da métrica de precisão (`Accuracy`) é inicializada para avaliar o desempenho do modelo durante a validação. Um barra de progresso (`tqdm.tqdm`) é configurada para acompanhar o progresso da validação em lotes, proporcionando uma representação visual do progresso da época.

Dentro do loop de validação, os dados de entrada (`x`) e os rótulos de saída (`y`) são carregados de cada lote. Assim como no treinamento, esses dados são transferidos para a GPU para acelerar o cálculo usando `cuda(DEVICE)`. A função `model(x)` é chamada para fazer uma previsão (`y_hat`) com base nos dados de entrada (`x`). A perda é calculada usando a função de perda focal (`focal_loss_func`) e comparando as previsões do modelo com os rótulos *one-hot* dos dados de validação.

A métrica de precisão é calculada da mesma maneira que no treinamento, usando previsões binarizadas (`y_hat_bin`) e rótulos reais (`y`). Isso fornece uma estimativa da precisão do modelo em relação aos dados de validação na iteração atual.

Em intervalos regulares definidos por `LOG_INTERVAL`, várias métricas e informações são registradas no TensorBoard usando o escritor (`SummaryWriter`). Isso inclui histogramas das previsões de classe, rótulos reais, perda de iteração e precisão de iteração durante a validação.

Após a conclusão do loop de validação, a perda média e a precisão média da época são calculadas, assim como foi feito durante o treinamento. Essas métricas são retornadas pela função para uso posterior.

Por fim, temos a função `test_and_log_metrics()`, que é capaz de fornecer uma análise das

métricas de desempenho, precisão e recall. Essa função é responsável por avaliar o modelo em um conjunto de dados de teste independente, calcular métricas-chave e gerar um relatório detalhado das métricas de classificação.

```python
def test_and_log_metrics(model, dataloader):
    # Função customizada para gerar resultados finais e métricas sobre os resultados
    x, y = dataloader.dataset[:]
    x = torch.Tensor(x).to(DEVICE)
    y = torch.Tensor(y).to(DEVICE).long()

    y_pred, _ = model(x)
    y_pred = y_pred.argmax(axis=1)[..., None]

    accuracy = Accuracy(num_classes=NUM_CLASSES, average=None).to(DEVICE)(y_pred, y)
    precision = Precision(num_classes=NUM_CLASSES, average=None).to(DEVICE)(y_pred, y)
    recall = Recall(num_classes=NUM_CLASSES, average=None).to(DEVICE)(y_pred, y)
    f1 = F1Score(num_classes=NUM_CLASSES, average=None).to(DEVICE)(y_pred, y)
    support = torch.unique(y, return_counts=True)[1]

    weighted_accuracy, weighted_precision, weighted_recall, weighted_f1 = 0, 0, 0, 0
    report = []
    for i in range(NUM_CLASSES):
        weight = support[i] / len(y)
        weighted_accuracy += weight * accuracy[i]
        weighted_precision += weight * precision[i]
        weighted_recall += weight * recall[i]
        weighted_f1 += weight * f1[i]
        report.append(['class_' + str(i), accuracy[i].item(), precision[i].item(), recall[i].item(), f1[i].item(), support[i].ite
    report.append(['weighted', weighted_accuracy.item(), weighted_precision.item(), weighted_recall.item(), weighted_f1.item(), 1

    report_df = pd.DataFrame(report, columns=['-', 'accuracy', 'precision', 'recall', 'f1', 'support'])
    report_df.set_index(report_df.columns[0])
    print('************************************ REPORT ************************************')
    print(report_df.to_markdown(index=False))
    report_df.to_csv(STATS_FILENAME, sep = ';', index=False)

    pred_df = pd.DataFrame(torch.concat([y, y_pred], axis=1).cpu(), columns=['GT', 'predictions'])
    pred_df.to_csv(RESULTS_FILENAME, sep = ';', index = False)
```

Figura 4.28 – Função **train_one_step** para execução da validação durante o treinamento.

No início da função, os dados de teste (`x` e `y`) são carregados a partir do dataloader do conjunto de teste. Os dados são transferidos para a GPU usando `torch.Tensor` para acelerar o cálculo. A função `model(x)` é chamada para fazer previsões (`y_pred`) com base nos dados de entrada (`x`). As previsões são convertidas em classes previstas usando o índice do valor máximo ao longo do eixo das classes.

Métricas essenciais de classificação, como precisão, recall e F1-score, são calculadas usando métricas do PyTorch, como `Accuracy`, `Precision`, `Recall` e `F1Score`. As métricas são calculadas para cada classe individualmente e para uma média ponderada de todas as classes, considerando o tamanho de suporte (número de amostras) de cada classe. O desempenho ponderado das métricas é calculado, levando em consideração o tamanho de suporte de cada classe, o que fornece uma avaliação mais realista do desempenho geral do modelo em relação a diferentes classes.

Um relatório detalhado é gerado e exibido, bem como salvo em um arquivo CSV para análises posteriores. O relatório inclui métricas como precisão, recall, F1-score e suporte para cada classe individualmente, bem como métricas ponderadas para avaliar o desempenho geral do modelo. Além disso, as previsões do modelo e os rótulos reais são salvos em um arquivo CSV para permitir uma análise mais detalhada das previsões e resultados.

5 Análise dos Resultados

A seção atual se concentrará nos resultados derivados da implementação e comparação dos modelos de aprendizagem de máquina, Random Forest e MLP, cujas aplicações foram detalhadas anteriormente. A análise visa identificar o modelo com o melhor desempenho na classificação de eventos de fogo, proporcionando uma apresentação clara e resumida dos resultados, incorporando métricas de desempenho, tabelas e gráficos para visualizar as discrepâncias entre os modelos.

5.1 Random Forest

No primeiro experimento, empregou-se o algoritmo Random Forest para avaliar seu desempenho em um conjunto de dados previamente não utilizado para o treinamento. Essa amostra foi selecionada de maneira aleatória e abrangeu métricas estatísticas como precisão, recall e f1-score para cada classe, assim como uma medida de acurácia global para o modelo, conforme apresentado na tabela abaixo. Este procedimento permitiu a obtenção de uma visão abrangente do desempenho do modelo e sua capacidade de generalização.

Tabela 5.1 – Métricas do resultado do teste realizado com o Random Forest.

Classe	**Precisão**	**Recall**	**F1-Score**	**Acurácia**	**Número de Dados**
1 - Savana e Pastagem	84.59%	87.20%	85.88%	87.20%	16758
2 - Pequenas Clareiras	74.98%	81.79%	78.24%	81.79%	13831
3 - Sub-bosque	55.78%	7.98%	13.96%	7.94%	1028
4 - Desflorestamento	60.60%	50.16%	54.89%	50.15%	5173
Acurácia Geral				77.74%	36790

Ao analisarmos os resultados apresentados na Tabela 5.1, podemos extrair insights valiosos sobre o desempenho do modelo de classificação em relação a cada classe e à sua acurácia global.

Ao examinarmos a classe "Savana e Pastagem", identificamos uma taxa de Precisão de 84.59%, indicando que a maior parte das previsões positivas atribuídas pelo modelo estava correta. O Recall, alcançando 87.20%, sinaliza que o modelo conseguiu identificar a maioria das instâncias reais pertencentes a essa categoria. O F1-Score, com uma pontuação de 85.88%, reforça o equilíbrio apropriado entre a precisão e a capacidade de recuperar as instâncias corretas, apontando para um desempenho sólido na detecção de "Savana e Pastagem".

Quanto à classe "Pequenas Clareiras", a análise revela uma Precisão de 74.98%, o que implica que a maioria das previsões positivas foi feita de maneira precisa. O Recall, atingindo 81.79%, sugere que o modelo foi capaz de identificar a maior parte das instâncias reais dessa classe, apesar de algumas previsões equivocadas. O F1-Score, calculado em 78.24%, reforça o compromisso entre a precisão e a capacidade de lembrar das instâncias, indicando um desempenho satisfatório.

Contudo, ao avaliarmos a classe "Sub-bosque", encontramos desafios notáveis. A Precisão

de 55.78% reflete previsões substancialmente corretas; no entanto, um Recall de apenas 7.98% aponta para dificuldades na identificação da maioria das instâncias reais dessa categoria. O F1-Score, marcando 13.96%, reflete as complicações em alcançar um equilíbrio eficaz entre a precisão e a capacidade de recuperar as instâncias, destacando a necessidade de ajustes para melhorar o desempenho nessa classe específica.

Analisando a classe "Desflorestamento", encontramos uma Precisão de 60.60%, indicando um número considerável de previsões corretas. O Recall de 50.16% sugere que o modelo conseguiu identificar aproximadamente metade das instâncias reais dessa categoria. O F1-Score, situado em 54.89%, sinaliza um desempenho intermediário, sugerindo oportunidades para otimizações futuras.

Em relação à Acurácia Geral do modelo, que atinge 77.74%, devemos salientar que essa métrica representa a média das classificações corretas para todas as classes. Contudo, é imperativo considerar as implicações práticas das previsões incorretas em cada categoria específica. Essa análise global oferece uma visão abrangente do desempenho do modelo, destacando áreas que podem ser alvo de ajustes para aprimoramentos futuros.

Em síntese, a análise das métricas da tabela **??** proporciona uma compreensão detalhada do desempenho do modelo de classificação. A visualização dos valores de Precisão, Recall e F1-Score para cada classe destaca tanto as conquistas notáveis, como na classe "Savana e Pastagem"e "Pequenas Clareiras", quanto as áreas que requerem atenção, como a classe "Sub-bosque"e "Desflorestamento". Esses insights críticos orientarão futuros esforços para otimizar o modelo no futuro, a fim de aprimorar sua capacidade de generalização e precisão em cenários do mundo real.

Dessa forma, a partir da execução do teste, procedemos à construção de uma matriz de confusão com o propósito de proporcionar uma visualização mais clara e reveladora do processo de classificação dos eventos de fogo pelo modelo. A matriz de confusão é uma ferramenta essencial que nos permite não apenas compreender as previsões corretas e incorretas feitas pelo algoritmo, mas também identificar de maneira detalhada os tipos de erros cometidos.

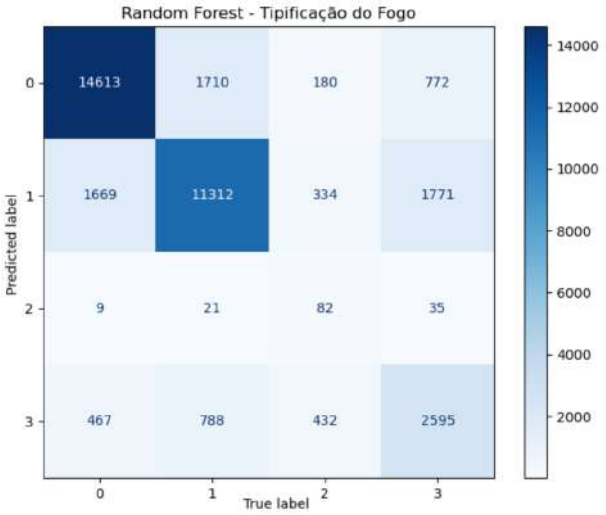

Figura 5.1 – Matriz de confusão do resultado do teste realizado com o Random Forest.

Assim, a partir da figura 5.1, temos que as classes são dadas por:

0. Fogo de Savana
1. Fogo de Pequenas Clareiras
2. Fogo de Sub-bosque
3. Fogo de Desflorestamento

Ao analisar a matriz de confusão, é possível distinguir quatro categorias distintas: verdadeiros positivos, verdadeiros negativos, falsos positivos e falsos negativos, que desempenham papéis fundamentais na avaliação do desempenho do modelo. Com base nessa estrutura, podemos realizar uma análise mais detalhada do modelo de classificação em relação a cada classe e suas interações. A matriz de confusão, ao comparar as previsões do modelo com os valores reais das classes, oferece uma visão completa do desempenho, destacando tanto os acertos quanto as áreas que requerem melhorias.

Assim, na classe de número 0 na matriz, que representa a classe "Savana e Pastagem", observamos que o modelo teve um número significativo de previsões corretas (14613) e um número moderado de falsos negativos (2662). Seguindo, a presença de falsos positivos (2145) indica que, embora o modelo tenha tido um desempenho relativamente sólido, existem casos em que houve dificuldade em distinguir corretamente entre essa classe e outras.

Na classe de número 1 na matriz, que representa a classe "Pequenas Clareiras", o modelo alcançou um bom número de previsões corretas (11312) e um número menor de falsos positivos (1919). No entanto, os falsos negativos (3774) sugere que ainda há espaço para melhorias na identificação precisa dessa classe.

Na classe de número 2 na matriz, que representa a classe "Sub-bosque", temos desafios notáveis, com um número muito baixo de previsões corretas (82) em relação aos falsos negativos (65) e falsos positivos (946). Isso indica que o modelo teve dificuldades em identificar e classificar adequadamente as instâncias reais dessa classe, resultando em uma representação subestimada.

Ao explorarmos a classe "Desflorestamento", de número 3 na matriz, observamos um número moderado de previsões corretas (2595), mas também um número considerável de falsos positivos (2578) e falsos negativos (1687). Isso sugere que o modelo enfrentou desafios em identificar com precisão essa classe e houve casos em que ocorreram confusões com outras classes.

Em síntese, classe "Savana e Pastagem", observamos um equilíbrio notável entre previsões corretas e aquelas incorretas, indicando a necessidade de aprimorar a diferenciação dessa classe das demais. Na classe "Pequenas Clareiras", embora tenhamos alcançado uma taxa satisfatória de previsões corretas, a presença de falsos positivos e falsos negativos aponta para áreas em que a precisão pode ser refinada. A classe "Sub-bosque"emerge como um desafio distinto, com um número limitado de previsões corretas e uma proporção desproporcional de falsos positivos e falsos negativos, sugerindo a necessidade de aperfeiçoar a identificação dessa classe. Por fim, ao examinarmos a classe "Desflorestamento", destacamos a importância de ajustes para equilibrar a precisão e minimizar as previsões incorretas. Essa avaliação abrangente serve como guia para

futuras melhorias e refinamentos do modelo de classificação, visando um desempenho mais sólido e confiável.

Ademais, no intuito de aprofundar nossa análise e compreensão, geramos um gráfico de importância de características (*Feature Importance*), visto na Figura 5.2. Este gráfico nos permite visualizar de maneira clara e elucidativa quais características exercem um papel mais proeminente na tarefa de classificação. Através dessa abordagem, podemos identificar as principais contribuições das diferentes características para o desempenho do modelo.

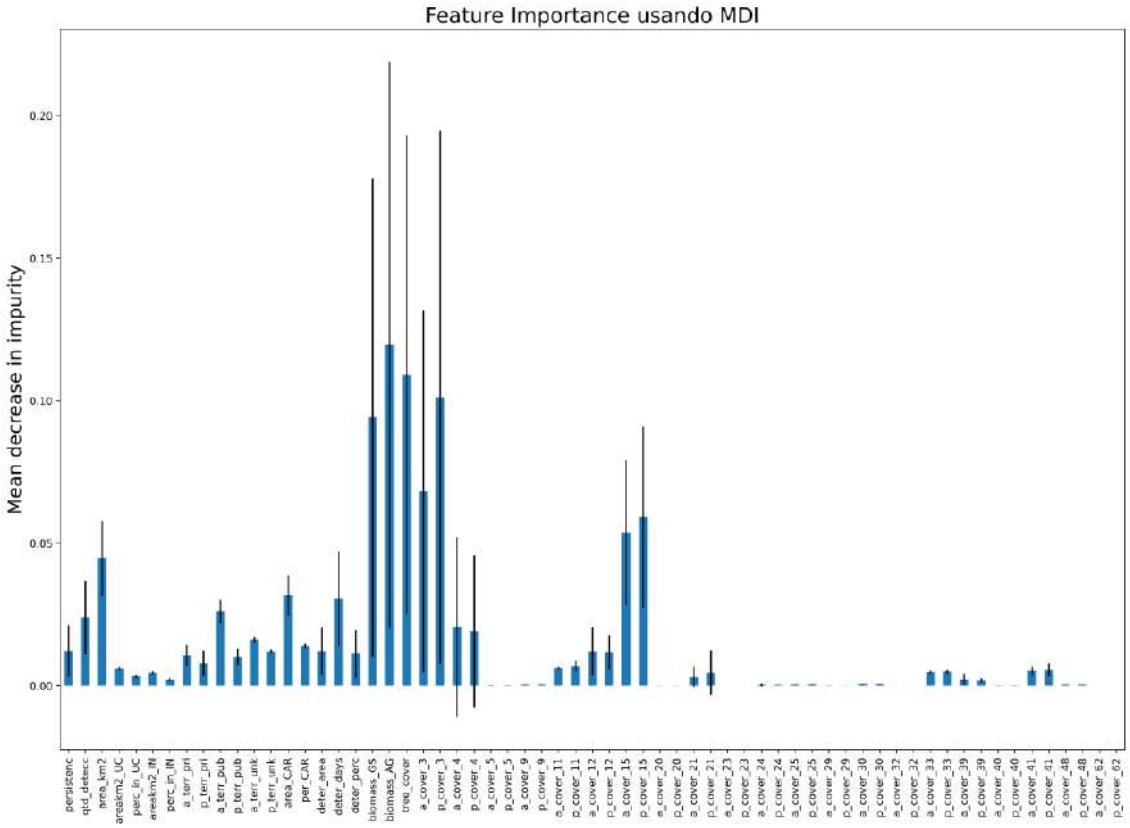

Figura 5.2 – Atributos mais importantes para a tomada de decisão na hora de classificação dos eventos de fogo.

A concepção do Gráfico de Importância de Características envolveu um processo de filtragem, no qual foram identificados e excluídos os atributos que se revelaram irrelevantes durante a fase de treinamento, expressando-se por valores zerados. A visualização resultante deste gráfico proporciona uma abordagem esclarecedora, permitindo a extração de *insights* significativos.

Nesse contexto, é possível identificar características que exercem um papel significativo na classificação dos eventos. Em destaque, a área em quilômetros quadrados (km^2), representada pela coluna "area_km2", emerge como um fator substancialmente influente na tomada de decisões do modelo Random Forest. Além disso, dois outros elementos também demonstraram relevância no processo de classificação. A análise da localização dos eventos se revelou pertinentemente valiosa, alcançada por meio da interseção de dados relacionados à localização geográfica, notavelmente retratada pela extensão de terras públicas (a_terr_pub). Adicionalmente, a distinção entre terras

privadas ou não registradas e a presença de um Cadastro Ambiental Rural (CAR) também desempenharam um papel discernível na classificação, especialmente quando se considera a proporção que essas áreas representam dentro dos eventos de fogo (area_CAR).

Também podemos observar que algumas características têm uma forte correlação com as conclusões da análise de importância das características no conjunto de dados GFED. Vale destacar especialmente o papel desempenhado pela biomassa e pela cobertura de árvores. Esses aspectos, representados pelas variáveis Estoque de Biomassa no Solo (biomass_GS), Biomassa Acima do Solo (biomass_AG) e cobertura de árvores (tree_cover), contribuem de maneira significativa. É relevante mencionar o destaque dado à cobertura de árvores, uma vez que essa mesma variável é usada no GFED. Esses resultados reforçam as evidências do GFED, enfatizando a importância desses fatores na avaliação de eventos de fogo e indicando a concordância das conclusões entre diferentes abordagens.

Por fim, merece destaque a notável influência de certos tipos de uso do solo, notadamente o uso do solo de tipo 3, que caracteriza formações florestais, o tipo 4, que engloba áreas de formação savânica, como o cerrado, e o tipo 15, associado ao uso do solo para pastagens. É interessante observar que tais conclusões estão em consonância com dados estatísticos mais abrangentes e alinham-se com as regiões de foco do CENSIPAM em suas iniciativas de monitoramento de eventos de fogo. Esses resultados reforçam a relevância das características locais na identificação e previsão de ocorrências de incêndios florestais, ampliando nossa compreensão sobre as nuances que moldam tais fenômenos e direcionando futuras estratégias de análise e intervenção.

5.2 Multi-Layer Perceptron (MLP)

Neste experimento, utilizou-se o algoritmo Multi-Layer Perceptron com o propósito de examinar sua eficácia em um conjunto de dados previamente não empregado no processo de treinamento, similarmente à abordagem anterior, visto na tabela 5.2. Mais uma vez, essa seleção de amostra foi realizada de modo aleatório e englobou métricas estatísticas como precisão, recall e f1-score para cada categoria, juntamente com uma métrica de acurácia global para a avaliação do modelo, conforme evidenciado na tabela a seguir.

Tabela 5.2 – Métricas do resultado do teste realizado no MLP.

Classe	Precisão	Recall	F1-Score	Acurácia	Número de Dados
1 - Savana e Pastagem	85.14%	86.15%	85.64%	86.15%	16984
2 - Pequenas Clareiras	73.96%	81.47%	77.53%	81.47%	13684
3 - Sub-bosque	34.92%	10.17%	15.75%	10.17%	1013
4 - Desflorestamento	59.48%	49.32%	53.93%	49.30%	5109
Acurácia Geral				77.20%	36790

Assim, repetindo a análise feita para o modelo anterior, examinamos a classe "Savana e Pastagem"e identificamos uma taxa de Precisão de 85.14%, indicando que a maior parte das previsões positivas atribuídas pelo modelo estava correta. O Recall, alcançando 86.15%, sinaliza que o modelo conseguiu identificar a maioria das instâncias reais pertencentes a essa categoria. O F1-Score,

com uma pontuação de 85.64%, reforça o equilíbrio apropriado entre a precisão e a capacidade de recuperar as instâncias corretas, apontando para um desempenho sólido na detecção de "Savana e Pastagem".

Quanto à classe "Pequenas Clareiras", a análise revela uma Precisão de 73.96%, o que implica que a maioria das previsões positivas foi feita de maneira precisa. O Recall, atingindo 81.47%, sugere que o modelo foi capaz de identificar a maior parte das instâncias reais dessa classe, apesar de algumas previsões equivocadas. O F1-Score, calculado em 77.53%, reforça o compromisso entre a precisão e a capacidade de lembrar das instâncias, indicando um desempenho satisfatório.

Contudo, ao avaliarmos a classe "Sub-bosque", encontramos desafios notáveis. A Precisão de 34.92% reflete previsões substancialmente corretas; no entanto, um Recall de apenas 10.17% aponta para dificuldades na identificação da maioria das instâncias reais dessa categoria. O F1-Score, marcando 15.75%, reflete as complicações em alcançar um equilíbrio eficaz entre a precisão e a capacidade de recuperar as instâncias, destacando a necessidade de ajustes para melhorar o desempenho nessa classe específica.

Analisando a classe "Desflorestamento", encontramos uma Precisão de 59.48%, indicando um número considerável de previsões corretas. O Recall de 49.32% sugere que o modelo conseguiu identificar aproximadamente metade das instâncias reais dessa categoria. O F1-Score, situado em 53.93%, sinaliza um desempenho intermediário, sugerindo oportunidades para otimizações futuras.

Em relação à Acurácia Geral do modelo, que atinge 77.20%, devemos salientar que essa métrica representa a média das classificações corretas para todas as classes. Contudo, novamente, é imperativo considerar as implicações práticas das previsões incorretas em cada categoria específica.

A seguir, mais uma vez, ao executarmos o teste, seguimos o processo detalhado de criação de uma matriz de confusão.

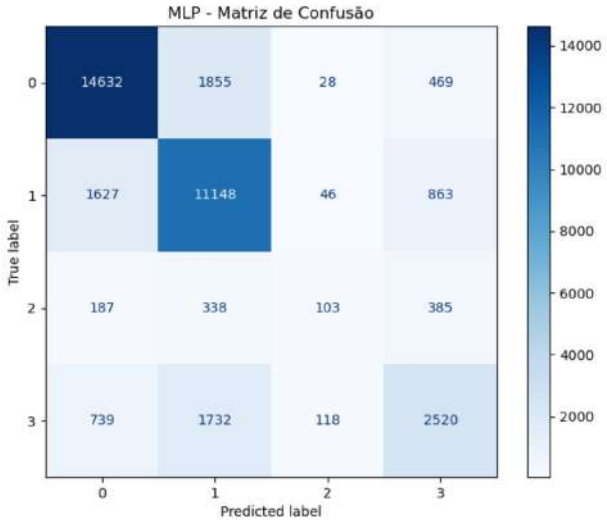

Figura 5.3 – Matriz de confusão do resultado do teste realizado com o MLP.

Assim, a partir da figura 5.3, temos que as classes são dadas por:

0. Fogo de Savana
1. Fogo de Pequenas Clareiras
2. Fogo de Sub-bosque
3. Fogo de Desflorestamento

Vale lembrar que a matriz presente mantém a mesma estrutura da matriz anterior, ilustrada na Figura 5.1. Assim, referente à categoria "Savana e Pastagem", notamos que o modelo obteve um volume significativo de previsões exatas (14632) nessa classe e também um número moderado de resultados falsos negativos (2352). Além disso, a presença de resultados falsos positivos (2553) indica que, embora o desempenho do modelo tenha sido relativamente sólido, ele teve dificuldades em discernir com precisão entre essa classe e outras categorias.

No que tange à classe "Pequenas Clareiras", o modelo atingiu um número satisfatório de previsões corretas (11148) e uma quantidade menor de falsos positivos (2536). No entanto, a ocorrência de falsos negativos (3925) sugere que ainda é possível aprimorar a identificação precisa dessa classe.

No caso da classe "Sub-bosque", encontramos desafios notáveis, com uma contagem muito baixa de previsões corretas (103) em comparação com os falsos negativos (192) e os falsos positivos (675). Isso evidencia que o modelo enfrentou dificuldades ao identificar e classificar adequadamente as instâncias reais dessa classe, resultando em uma representação subestimada.

Por último, ao investigar a classe "Desflorestamento", notamos uma quantidade moderada de previsões corretas (2520), porém também um número considerável de falsos positivos (1717) e falsos negativos (2589). Isso indica que o modelo teve desafios em identificar essa classe com precisão, e ocorreram situações em que houve confusões com outras classes.

Em suma, a análise das diferentes classes revela nuances significativas no desempenho do modelo de classificação. Enquanto algumas categorias demonstraram resultados promissores, como as categorias "Savana e Pastagem"e "Pequenas Clareiras", com suas previsões corretas substanciais, outras, como a classe "Sub-bosque", expuseram desafios distintos, revelando a necessidade de ajustes para uma representação mais precisa. A classe "Desflorestamento", por sua vez, apresentou um quadro misto de previsões corretas e falsos positivos, apontando para áreas nas quais aprimoramentos são possíveis.

Finalmente, examinamos os resultados obtidos através do *TensorBoard* ao longo do processo de treinamento do modelo em consideração. Compreender o comportamento da rede durante o treinamento assume um papel fundamental, permitindo-nos determinar se houve uma convergência satisfatória, ou seja, se a rede alcançou um nível apropriado de aprendizado sem ceder a tendências enviesadas, o que é conhecido como overfitting.

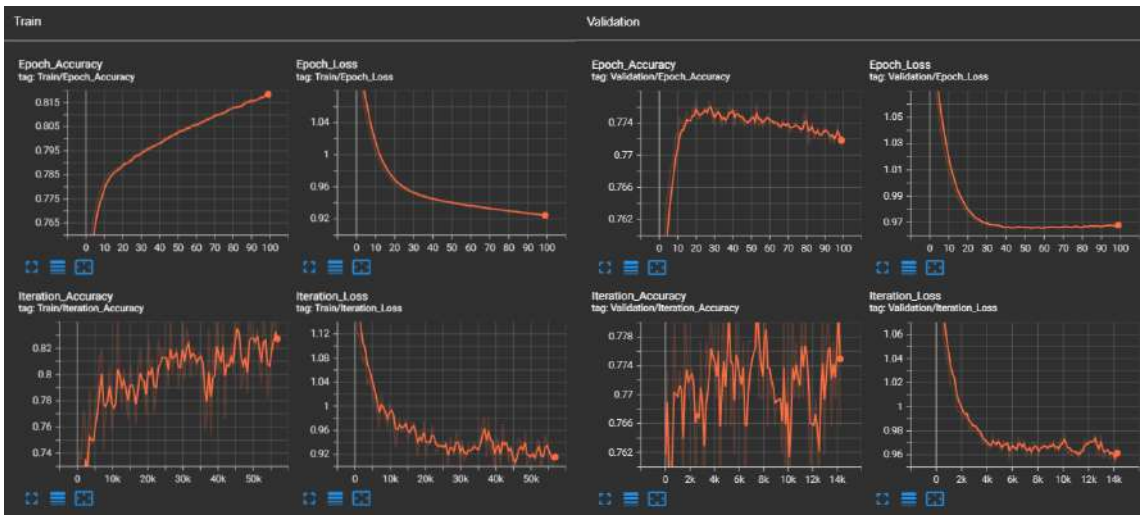

Figura 5.4 – Gráficos de perda e acurácia durante o treinamento, para os conjuntos de treinamento e validação, visualizados através do Tensorboard.

Além disso, é de grande interesse analisar o histograma gerado durante o treinamento, a fim de visualizar a distribuição dos dados previstos em comparação com os valores reais, avaliando se ambas as distribuições são semelhantes.

Ao observarmos os gráficos que ilustram as épocas ao longo do processo de treinamento, na Figura 5.4, notamos que emergem padrões significativos no conjunto de treinamento e na fase de validação. O gráfico da métrica de perda (*loss*) mostra uma tendência inicial de decréscimo seguida de estabilidade, sobressaindo durante a validação. Similarmente, ao examinarmos a métrica de acurácia, observamos um comportamento satisfatório com uma tendência ascendente seguida de estabilidade, especialmente na validação. Essa análise evidencia um comportamento esperado de uma rede neural que convergiu de forma apropriada, demonstrando sua capacidade de generalizar o aprendizado até um ponto específico, evitando viéses tendenciosos (*overfitting*). Tal análise do comportamento da rede neural realça sua capacidade de convergência adequada, indicando a aquisição da habilidade de generalização até um ponto ótimo, prevenindo possíveis problemas de ajuste excessivo.

Ao analisarmos a Figura 5.5 logo abaixo, é possível perceber que os histogramas que retratam a distribuição dos dados previstos em comparação com os valores reais exibem semelhanças notáveis, especialmente nas duas primeiras categorias. Contudo, é importante destacar que há uma discrepância mais notável na categoria de "sub-bosque", o que contrasta com a matriz de confusão evidenciada na Figura 5.3.

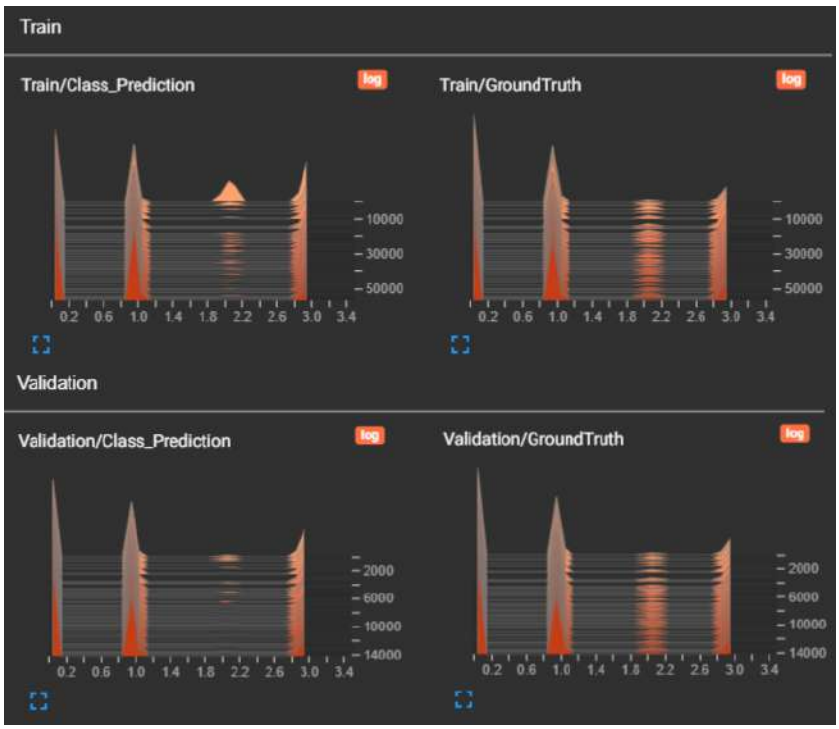

Figura 5.5 – Histogramas comparando dados reais com suas respectivas predições para os conjuntos de treinamento e validação, visualizados através do Tensorboard.

6 Conclusão

Os modelos gerados evidenciaram um desempenho satisfatório ao categorizar as classes de ecossistemas de savana/pastagem e pequenas clareiras. Todavia, é crucial destacar suas limitações na classificação da classe de desflorestamento, a qual é de interesse fundamental para o CENSIPAM, e, de forma ainda mais acentuada, na identificação da classe de sub-bosque, na qual a performance foi substancialmente deficiente. Essa dificuldade pode ser possivelmente atribuída à heterogeneidade presente na rotulagem dos dados empregados no treinamento dos modelos.

Presume-se que a inclusão de rótulos provenientes de múltiplas fontes possa estar exercendo um efeito adverso sobre a capacidade de aprendizado dos modelos. Para atenuar esse cenário, o atual trabalho está sendo incorporado ao sistema de vigilância de incêndios do CENSIPAM, denominado Painel do Fogo. Essa implementação envolve a adoção de uma abordagem metodológica mais homogênea, concomitantemente com a introdução de um procedimento de correção manual dos dados. A expectativa é que essa estratégia conjunta possua o potencial de substancialmente elevar a qualidade do conjunto de dados de entrada, resultando, por conseguinte, em aprimoramentos consideráveis na precisão dos modelos gerados.

Simultaneamente, está em andamento a implementação de um projeto adicional com o propósito de se integrar ao contexto do Painel do Fogo. Este projeto adota uma abordagem metodológica análoga, aproveitando integralmente a metodologia empregada neste estudo para a obtenção das características utilizadas, porém se distingue ao examinar a evolução temporal dos eventos desde sua detecção inicial. Esse procedimento é conduzido através da aplicação de Redes Neurais Recorrentes com Memória de Longo Prazo (LSTM), uma técnica especializada no tratamento de sequências temporais.

Por fim, como perspectiva para pesquisas futuras, o aprofundamento das investigações em direção a uma modalidade de modelagem que englobe o comportamento de propagação do fogo por meio de Inteligência Artificial demonstra um alto potencial de utilidade. Esse avanço teria a capacidade de significativamente ampliar as habilidades de combate a incêndios na região amazônica.

Bibliografia

[1] Mohamed Alloghani et al. “A Systematic Review on Supervised and Unsupervised Machine Learning Algorithms for Data Science”. Em: jan. de 2020, pp. 3–21. ISBN: 978-3-030-22474-5. DOI: <10.1007/978-3-030-22475-2_1>.

[2] Niels Andela et al. “The Global Fire Atlas of individual fire size, duration, speed and direction”. Em: *Earth System Science Data* 11 (abr. de 2019), pp. 529–552. DOI: <10.5194/essd-11-529-2019>.

[3] Yoshua Bengio. “Learning Deep Architectures for AI”. Em: *Foundations and Trends® in Machine Learning* 2.1 (2009), pp. 1–127. ISSN: 1935-8237. DOI: <10.1561/2200000006>. URL: <http://dx.doi.org/10.1561/2200000006>.

[4] Leo Breiman. “Random forests”. Em: *Machine learning* 45.1 (2001), pp. 5–32.

[5] Gerardo Budowski. *Tropical savannas, a sequence of forest felling and repeated burning*. Turrialba, Costa Rica, 1956.

[6] Jaime G. Carbonell, Ryszard S. Michalski e Tom M. Mitchell. “An Overview of Machine Learning”. Em: *Machine Learning: An Artificial Intelligence Approach*. Ed. por Ryszard S. Michalski, Jaime G. Carbonell e Tom M. Mitchell. Berlin, Heidelberg: Springer Berlin Heidelberg, 1983, pp. 3–23. ISBN: 978-3-662-12405-5. DOI: <10.1007/978-3-662-12405-5_1>. URL: <https://doi.org/10.1007/978-3-662-12405-5_1>.

[7] CENSIPAM. *Painel do Fogo*. Acessado em: 26/10/2022. URL: <https://paineldofogo.sipam.gov.br>.

[8] R. N. Colwell. *Remote Sensing of Environment: An Earth Resource Perspective*. Prentice Hall, 1997.

[9] “COMPUTING MACHINERY AND INTELLIGENCE”. Em: *Mind, Volume LIX, Issue 236* 2.2 (1950), pp. 433–460. DOI: <10.1093/mind/LIX.236.433>.

[10] Adele Cutler e Leo Breiman. “Random Forests”. Em: *Encyclopedic Dictionary of ICT and Computing*. IGI Global, 2007.

[11] Daniela R. G. de Faria et al. “Análise da dinâmica do intervalo entre desmatamento e fogo: bases para tipificação de fogo de desmatamento”. Em: *SBSR 2023 - XX Simpósio Brasileiro de Sensoriamento Remoto*. 2023, pp. 1484–1487.

[12] Daniela R. G. de Faria et al. “Multicriteria Severity Indicator Using Remote Sensing for Forest Firefighting Dispatch in the Brazilian Amazon”. Em: *IGARSS 2022 - 2022 IEEE International Geoscience and Remote Sensing Symposium*. 2022, pp. 5740–5743. DOI: <10.1109/IGARSS46834.2022.9884033>.

[13] *Fire Information for Resource Management System (FIRMS)*. <https://firms.modaps.eosdis.nasa.gov/>. Accessed: [Insert date here].

[14] *Forest fires*. Food e Agriculture Organization of the United Nations, 2021. URL: <http://www.fao.org/forestry/16084-0c68f127467a9e074d4ed3e2cd63f3d1f.pdf>.

[15] Python Software Foundation. *Python Programming Language*. <https://www.python.org/>. 1991.

[16] GeoTIFF. *What is a GeoTIFF?* 2021. URL: <https://www.geotiff.org/what_is_a_geotiff>.

[17] Ian Goodfellow, Yoshua Bengio e Aaron Courville. *Deep learning*. MIT Press, 2016.

[18] M. C. Hansen et al. *High-Resolution Global Maps of 21st-Century Forest Cover Change*. 2013. DOI: <10.1038/nclimate1944>.

[19] John D. Hunter. "Matplotlib: A 2D Graphics Environment". Em: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95.

[20] IBGE. *Amazônia Legal*. URL: <https://www.ibge.gov.br/geociencias/cartas-e-mapas/mapas-regionais/15819-amazonia-legal.html?=&t=o-que-e>.

[21] INPE. *DETER*. Acessado em: 27/10/2022. URL: <http://www.obt.inpe.br/OBT/assuntos/programas/amazonia/deter/deter>.

[22] James Kelley e Other Contributors. *Geopandas*. <https://geopandas.org/>. 2021.

[23] Yann LeCun, Yoshua Bengio e Geoffrey Hinton. "Deep learning". Em: *Nature* 521.7553 (2015), pp. 436–444.

[24] Andy Liaw e Matthew Wiener. "Classification and regression by randomForest". Em: *R news* 2.3 (2002), pp. 18–22.

[25] L. S. Lillesand e R. Kiefer. *Principles of Remote Sensing*. Wiley, 2000.

[26] Andre Lima et al. "Mapeamento de Cicatrizes de Queimadas na Amazônia Brasileira a partir da aplicação do Modelo Linear de Mistura Espectral em imagens do sensor MODIS". Em: (jan. de 2009).

[27] M. Lutz. *Learning Python, 5th Edition*. O'Reilly Media, Inc., 2013.

[28] A. B. Martins, A. C. Filho e J. L. da Costa. "Dinâmica de desmatamento e queimadas na Amazônia Legal". Em: *XV jornada de iniciação científica do PIBIC/CNPq/FAPEAM/INPA* 103 (2006), pp. 73–74.

[29] Wes McKinney e Other Contributors. *Pandas*. <https://pandas.pydata.org/>. 2021.

[30] Instituto Brasileiro do Meio Ambiente e dos Recursos Naturais Renováveis (IBAMA). *Combate a incêndios florestais*. 2021. URL: <https://www.ibama.gov.br/queimadas/combate-incendios-florestais>.

[31] Douglas Morton et al. "Cropland Expansion Changes Deforestation Dynamics in the Southern Brazilian Amazon". Em: *Proceedings of the National Academy of Sciences of the United States of America* 103 (out. de 2006), pp. 14637–41. DOI: <10.1073/pnas.0606377103>.

[32] Travis Oliphant e et al. *Numpy*. <https://numpy.org/>. 2021.

[33] Adam Paszke et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. Em: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. URL: <http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf>.

[34] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. Em: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[35] Thales Vaz Penha. “Detecção de áreas queimadas na Amazônia utilizando imagens de média resolução espacial, técnicas de GEOBIA e mineração de dados”. Programa de Pós-Graduação do INPE em Sensoriamento Remoto. Instituto Nacional de Pesquisas Espaciais (INPE), 2018. URL: <http://urlib.net/sid.inpe.br/mtc-m21c/2018/04.29.22.45>.

[36] Allan Arantes Pereira. “Automatic mapping of wildfires in the Cerrado biome using orbital sensors”. Em: (ago. de 2017).

[37] Instituto Nacional de Pesquisas Espaciais (INPE). *DETER: Early Detection of Deforestation in the Amazon*. Rel. técn. Instituto Nacional de Pesquisas Espaciais, 2004.

[38] Instituto Nacional de Pesquisas Espaciais (INPE). *TerraBrasilis: A Platform for Monitoring and Analysis of Remote Sensing Data*. Accessed on: 2023-02-10. 2021. URL: <https://terrabrasilis.dpi.inpe.br/>.

[39] Ghillean T. Prance. *Biological Diversification In The Tropics*. Columbia University Press, 1982.

[40] *Projeto MapBiomas - Série Anual de Mapas da Cobertura e Uso do Solo do Brasil*. <https://mapbiomas.org/>. Acessado em 11/11/2022.

[41] QGIS Development Team. *QGIS 3.16 LTR*. 2022. URL: <https://qgis.org/en/site/>.

[42] David E Rumelhart, Geoffrey E Hinton e Ronald J Williams. “Learning representations by back-propagating errors”. Em: *Nature* 323.6088 (1986), pp. 533–536.

[43] M. Santoro et al. *GlobBiomass global above-ground biomass and growing stock volume datasets*. 2018. URL: <http://globbiomass.org/products/global-mapping>.

[44] Jaime Simão Sichman. “Inteligência Artificial e sociedade: avanços e riscos”. Em: *Estudos Avançados* 35.101 (abr. de 2021), pp. 37–50. DOI: <10.1590/s0103-4014.2021.35101.004>. URL: <https://doi.org/10.1590/s0103-4014.2021.35101.004>.

[45] Vera Laísa da Silva Arruda. “Mapeamento de cicatrizes de áreas queimadas no Cerrado usando imagens Landsat, Google Earth Engine e Deep Learning”. Em: (abr. de 2021).

[46] Haohan Wang e Bhiksha Raj. *On the Origin of Deep Learning*. 2017. DOI: <10.48550/ARXIV.1702.07800>. URL: <https://arxiv.org/abs/1702.07800>.

[47] *What is a Shapefile?* 2021. URL: <https://desktop.arcgis.com/en/arcmap/10.3/manage-data/shapefiles/what-is-a-shapefile.htm>.