Universidade Federal de Santa Catarina
Centro Tecnológico, de Ciências Exatas e Educação de Blumenau
Departamento de Engenharia de Controle e Automação e Computação

Franciele Kuwahara de Matos Silva

Aprendizado de Máquina aplicado à Predição de incêndio em Canavial

Blumenau
2021

Franciele Kuwahara de Matos Silva

Aprendizado de Máquina aplicado à Predição de incêndio em Canavial

Trabalho de Conclusão de Curso apresentado à Universidade Federal de Santa Catarina como parte dos requisitos necessários para a obtenção do Título de Engenheiro de Controle e Automação.
Orientador: Prof. Dr. Maiquel de Brito

Universidade Federal de Santa Catarina
Centro Tecnológico, de Ciências Exatas e Educação de Blumenau
Departamento de Engenharia de
Controle e Automação e Computação

Blumenau
2021

Franciele Kuwahara de Matos Silva

Aprendizado de Máquina aplicado à Predição de incêndio em Canavial

Trabalho de Conclusão de Curso apresentado à Universidade Federal de Santa Catarina como requisito parcial para a obtenção do título de Engenheiro de Controle e Automação.

Comissão Examinadora

Documento assinado digitalmente
Maiquel de Brito
Data: 30/09/2021 09:05:49-0300
CPF: 979.657.150-15
Verifique as assinaturas em https://v.ufsc.br

Prof. Dr. Maiquel de Brito
Universidade Federal de Santa Catarina
Orientador

Documento assinado digitalmente
Fabio Rafael Segundo
Data: 01/10/2021 16:14:09-0300
CPF: 901.593.909-82
Verifique as assinaturas em https://v.ufsc.br

Prof. Dr. Fábio Rafael Segundo
Universidade Federal de Santa Catarina

Documento assinado digitalmente
Mauri Ferrandin
Data: 30/09/2021 13:10:42-0300
CPF: 018.580.869-73
Verifique as assinaturas em https://v.ufsc.br

Prof. Dr. Mauri Ferrandin
Universidade Federal de Santa Catarina

Blumenau, 30 de setembro de 2021

Dedico este trabalho aos meus pais e meu irmão, que sempre foram meu alicerce em todos os momentos da minha vida.

Agradecimentos

Gostaria de agradecer, primeiramente, aos meus pais, Cecília e Carlos, por todo o incentivo e por toda a educação que me proporcionaram em minha vida. Serei eternamente grata a vocês. Eu também gostaria de agradecer ao meu irmão, Vinícius, que sempre me apoiou e me ajudou em minhas decisões. Obrigada por ser meu exemplo de vida e sempre me mostrar como ser uma pessoa melhor.

Aos meus amigos de longa data e aos que conheci durante o percurso da graduação e que de alguma forma colaboraram para o meu crescimento pessoal e intelectual. Agradeço à todos!

Agradeço imensamente aos meus professores, em especial, ao meu orientador, Maicon de Brito, a quem dirijo respeito como profissional e ser humano. Agradeço por sempre estar a disposição para tirar dúvidas, conversar a respeito do desenvolvimento deste trabalho e ser compreensivo em todos os momentos. Além disso, obrigada pelas aulas tão bem ministradas durante o curso que foram fundamentais para a construção da minha base de aprendizado para o desenvolvimento deste trabalho de conclusão de curso.

Por fim, agradeço às pessoas que se dispuseram a me orientar sobre este trabalho na empresa em que ele foi desenvolvido. A todas as pessoas que de alguma forma contribuíram para a realização deste trabalho, meu muito obrigada!

"A person who never made a mistake, never tried anything new"
(Albert Einstein)

Resumo

Este trabalho descreve o desenvolvimento de um modelo de aprendizado de máquina que prediz incêndio em um canavial de uma empresa da região noroeste do estado de São Paulo. Inicialmente é apresentada a fundamentação teórica sobre a matéria-prima cana-de-açúcar, assim como também são retratados conceitos relacionados a modelos de aprendizado de máquina. São apresentadas as etapas desenvolvidas até se alcançar um modelo com um resultado satisfatório. Foi mostrada a metodologia aplicada durante o progresso do algoritmo. Assim, o projeto foi caracterizado a partir da seleção de variáveis, tratamento dos dados, criação de novos atributos, transformações de dados, técnicas de pré-processamentos, além da validação do modelo produzido. Por fim, os resultados da implementação do projeto são discutidos, os quais mostram que o sistema proposto é aceitável, servindo de apoio para áreas de gerenciamento de risco de focos de incêndio da empresa.

Palavras-Chave: 1. Incêndios. 2. Canavial. 3. Inteligência Artificial. 4. Modelos Preditivos. 5. XGBoost.

Abstract

This work describes the development of a machine learning model that predicts the fire in a sugarcane plantation of a company in the northwest region of the state of São Paulo. Initially, the theoretical foundation on the raw material of sugarcane is presented, as well as concepts related to machine learning models are also portrayed. The steps developed until reaching a model with a satisfactory result are displayed. The methodology applied during the progress of the algorithm was shown. Thus, the project was characterized from the selection of variables, data treatment, creation of new attributes, data transformations, pre-processing techniques, in addition to the validation of the model produced. Finally, the results of the project's implementation are discussed, which show that the proposed system is acceptable, serving as a support for the company's fire areas risk management.

Keywords: 1. Fires. 2. Cane Field. 3. Artificial Intelligence. 4. Predictive Models. 5. XGBoost.

Lista de figuras

Lista de tabelas

Lista de Siglas e Abreviaturas

AM	*Aprendizado de Máquina*
XGBoost	*Extreme Gradient Boosting*
Inpe	*Instituto Nacional de Pesquisas Espaciais*
k-NN	*k-Nearest Neighbors*
ML	*Machine Learning*
MAE	*Mean Absolute Error*
MAPE	*Mean Absolute Percentage Error*
MSE	*Mean Squared Error*
RF	*Random Forest*
OMS	*Organização Mundial da Saúde*
RMSE	*Root Mean Squared Error*
SMOTE	*Synthetic Minority Oversampling TEchnique*
SMOTE-NC	*Synthetic Minority Oversampling TEchnique - Nominal Continuous*
SVM	*Support Vector Machine*
NDVI	*Normalized Difference Vegetation Index*

Sumário

1 Introdução

1.1 Contexto

Tempo seco e altas temperaturas são fatores que favorecem o aumento de incêndios ambientais. Na região Centro-Sul do Brasil, as queimadas ocorrem principalmente entre abril e novembro, em que é justamente um período de pouca chuva. No ano de 2020, as condições ambientais não foram favoráveis para as áreas de cultivo de cana-de-açúcar. Assim, o prejuízo com queimadas continua ameaçando as usinas da safra do ano de 2021. Devido a isso, assuntos relacionados à prevenção de incêndios estão em foco nas unidades industriais.

Além de causarem perdas da matéria-prima, degradação do solo, poluição do ar entre outros fatores negativos, focos de incêndio também levam a gastos referentes ao deslocamento de carros e caminhões de bombeiros até o local do incêndio. De acordo com o Jornal Cana [1], os incêndios em canaviais têm causado perda de aproximadamente 30% da matéria-prima em 2020 e áreas afetadas podem demorar dois anos ou mais para se recuperarem. Além disso, estimativas revelaram que, em 2020, as áreas atingidas por incêndios tiveram uma perda da ordem de 15% de massa de cana (toneladas) e que o prejuízo de um grupo do setor sucroenergético representou uma perda de mais de 40 milhões de reais [2].

De acordo com a União da Indústria de Cana-de-açúcar (Unica), este cenário de baixa precipitação e altas temperaturas vai se manter no ano de 2021 durante a estação mais seca. Com isso, aumenta-se a preocupação de focos de incêndios iniciados de maneira criminosa ou acidental nos canaviais. Geralmente, a origem dos incêndios ocorre nas áreas próximas a rodovias ou onde a população possui fácil acesso. Alguns agentes desses incidentes são bitucas de cigarro, limpeza de terrenos, fogueiras entre outros.

Atualmente, aproximadamente 99,6% de toda cana-de-açúcar cultivada em São Paulo é colhida sem queima [3], ou seja, a cana é colhida de forma mecanizada. Ações de comunicação, tais como placas informativas e aplicativos de celular, são realizadas junto à população para ajudar a evitar atitudes que gerem risco de incêndio.

Neste cenário de tempo seco e o aumento da preocupação com riscos de incêndios, é interessante que se possa monitorar esses riscos com antecedência. No entanto, a associação de condições climáticas com fatores de localização geográfica para o acompanhamento prévio de focos de incêndio não é algo simples. Com isso, essa questão que é mais complexa pode ser abordada com o uso de *Machine Learning (ML)*, também conhecido por aprendizado de máquina. A aplicação de aprendizado de máquina, que no caso deste trabalho é um sistema que serve para reconhecer padrões nos dados analisados, pode ajudar

a evitar que incêndios em áreas canavieiras aconteçam ou se espalhem. Isso pode ser feito a partir da determinação da possibilidade de uma área pegar fogo, contribuindo nas tomadas de decisões e melhorando o desempenho de tarefas que auxiliam no combate a incêndios.

1.2 Problemática da Pesquisa

Devido ao clima desfavorável do ano 2020 para o cultivo da cana de açúcar [1] e a preocupação com o risco de queimadas nos canaviais, optou-se em desenvolver um modelo de aprendizado de máquina. O objetivo deste modelo é indicar se determinada região tem chances ou não de pegar fogo. E como não se tem conhecimento de um sistema que indique previamente a probabilidade de ocorrência de incêndio em certas áreas do canavial da empresa sucroenergética da região noroeste do estado de São Paulo, este trabalho de conclusão de curso propõe o desenvolvimento desse modelo preditivo.

A elaboração deste modelo, no contexto em que será desenvolvido e aplicado, apresenta alguns desafios. Um problema que pode interferir na implementação do modelo preditivo é o fato da área a ser analisada não ser tão extensa. Isso pode fazer com que valores das variáveis referentes às condições climáticas de uma região para outra sejam próximos e com baixa diferença entre si, dificultando o aprendizado de padrões para o modelo. E outro problema que também podem afetar a previsibilidade do modelo está relacionado à confiabilidade dos dados medidos, podendo muitas vezes acontecer erros de medição e até mesmo a falta de valores para realizar as análises.

1.3 Objetivos

O presente trabalho tem como objetivo desenvolver um modelo de aprendizado de máquina para predizer a probabilidade de incêndio em canavial. Para isso, foram utilizados dados dos anos 2017 até 2020 para treinar o modelo e fazer com que ele aprenda a identificar padrões nos dados utilizados como entrada.

Como objetivos específicos para o desenvolvimento do projeto, tem-se:

- Selecionar variáveis que possuem maior importância para a análise;
- Desenvolver métodos para pré-processamento, limpeza e tratamentos necessários nos dados;
- Analisar a melhor métrica de validação de modelo;
- Refinar algoritmo com a otimização de hiperparâmetros;
- Utilizar as predições como auxílio nas tomadas de decisões.

1.4 Estrutura do documento

Este trabalho está dividido em cinco capítulos. Este primeiro capítulo presente é a seção introdutória. O Capítulo 2 traz a fundamentação teórica, abordando conceitos bastante utilizados entre assuntos de aprendizado de máquina. No Capítulo 3, é abordada a metodologia aplicada no desenvolvimento do sistema de predição. Os resultados obtidos são apresentados no Capítulo 4. E, no final deste documento, o Capítulo 5 se refere à conclusão do trabalho e pontuações de possíveis melhorias a serem aplicadas.

2 Fundamentação Teórica

Este capítulo visa explicar de maneira sucinta conceitos que são relevantes para o entendimento deste trabalho. Faz-se uma breve explicação sobre canaviais e, também, como as queimas nessas plantações afetam o meio ambiente e a população ao redor. Além disso, apresenta-se, de forma resumida, a teoria do aprendizado de máquina, alguns de seus algoritmos e métodos que são utilizados hoje em dia.

2.1 Incêndios em canaviais

A cana-de-açúcar, planta originária da Nova Guiné, representou uma importante época da história do Brasil com o Ciclo da Cana-de-Açúcar entre os séculos XVI e XVIII. No final do século XVI, o Brasil já era considerado o maior produtor e fornecedor mundial de açúcar, produto que representou a primeira grande riqueza agrícola e industrial do país [4].

O Brasil continua sendo o país que mais produz cana-de-açúcar do mundo e o cultivo da cana ocupa mais de oito milhões de hectares tendo o estado de São Paulo como o maior produtor [5]. O etanol e o bagaço, que são materiais obtidos a partir da cana, mantém no Brasil o maior sistema de produção de energia comercial de biomassa do mundo [6].

Sendo o açúcar um dos produtos essencias no dia a dia das pessoas, é muito importante que o canavial seja bem cuidado e preservado para que a produção saia como o esperado e não tenha prejuízos na economia. No entanto, um fator que ameaça as plantações são as queimadas, que ocorrem princialmente em tempos de pouca chuva, tanto de forma acidental ou criminosa. De forma a evitar isso, as usinas investem em tecnologia e no uso de satélites para o monitoramento dos canaviais.

Gases como o monóxido de carbono (CO), o metano (CH_4) e o óxido nitroso (NO_2) são enviados para a atmosfera em grande quantidade em decorrência da queima da cana-de-açúcar e isto contribui com o aumento do aquecimento global como também com a formação de ozônio na baixa atmosfera [7]. Inflamações, infeccções crônicas e até mesmo um câncer podem surgir em quadros de problemas respiratórios devido às substâncias encontradas na fumaça da queima. Segundo a OMS (Organização Mundial da Saúde), índices de umidade relativa do ar inferiores à 30% são considerados críticos e esses níveis levam as pessoas a sentirem desconfortos físicos. Essa baixa umidade relativa do ar favorece a ignição de queimadas, podendo piorar ainda mais a qualidade do ar.

As plantações de canaviais geralmente não possuem grandes distâncias de outros tipos de vegetações. Dessa forma, áreas florestais próximas ao canavial se tornam suscetíveis à queima acidental além de ameaçar a biodiversidade animal pela perda de seu habitat e,

em muitos casos, por atropelamentos em rodovias devido a suas fugas dos incêndios.

Até 2014, a queima da palha da cana-de-açúcar facilitava a colheita em razão da maior facilidade para os trabalhadores realizarem o corte da planta. No entanto, com o avanço da tecnologia das colheitadeiras, a colheita mecanizada se tornou mais vantajosa. Em agosto de 2007 foi firmado um protocolo de intenções entre a Secretaria de Meio Ambiente (SMA) do estado de São Paulo. Neste documento, a prática da queima da palha da cana devia ser gradativamene eliminada até 2014 em áreas que são mecanizáveis e até 2017 para as áreas restantes (não mecanizáveis). No caso de fornecedores de cana-de-açúcar para as usinas, estes possuem até o ano de 2021 para pararem com a queima [5]. Essa proibição leva a uma diminuição na emissão de poluentes, melhora na qualidade do solo, biodiversidade e qualidade de vida da população ao redor. Apesar da criação deste protocolo, a transição da colheita feita a partir da queima da palha da cana para a mecanizada seria inevitável com o avanço da tecnologia que oferece uma maior produtividade para as empresas. No entanto, essa mudança da colheita manual para a colheita mecanizada de cana crua (não queimada) implicou em algumas complicações sociais já que uma colheitadeira pode substituir em torno de 90 cortadores de cana. Sendo assim, o perfil necessário do trabalhador agrícola está mudando e, com isso, é preciso capacitar essas pessoas para que haja sua inserção neste ramo de trabalho.

2.2 Aprendizado de Máquina

O aprendizado de máquina, do inglês *Machine Learning* (ML), ramo da inteligência artificial, procura resolver problemas do mundo real através de programas de computador. Com a capacidade de aprender a partir de dados que ofereçam informações suficientes, esses sistemas conseguem alcançar uma solução de forma automática. A partir da evolução de algoritmos computacionais projetados para emular a inteligência do homem aprendendo com o ambiente circundante, o aprendizado de máquina é um ramo em evolução [8]. Sendo assim, o objetivo do aprendizado de máquina é encontrar padrões e tomar decisões com o mínimo possível de intervenção humana. Trata-se de uma ferramenta poderosa que possui diversos algoritmos que podem apresentar diferentes performances de acordo com cada caso. Assim, pode-se definir como aprendizado algo que melhorou o desempenho a partir de uma mudança de comportamento [9].

Um programa de computador aprende a partir da experiência E com relação a alguma classe de tarefas T e mede o desempenho P se seu desempenho nas tarefas em T, conforme medido por P, melhora com a experiência E [10]. Sendo assim, o aprendizado de máquina permite que um programa de computador identifique padrões e encontre relações entre os dados de forma automatizada, a partir do aprendizado e experiência com os dados.

Pode-se dividir o aprendizado de máquina em dois sub-grupos: não-supervisionado e supervisionado. O primeiro se refere a um aprendizado em que o conjunto de dados não

é rotulado. Já o segundo grupo é definido pelo uso de dados rotulados para o treino do algoritmo, ou seja, tem-se uma base de dados especialmente para treinar o modelo para que ele dê como saída o resultado desejado.

2.2.1 Aprendizado de Máquina Supervisionado

Algoritmos com aprendizado supervisionado são aqueles que utilizam um conjunto de dados para treinarem e aprenderem padrões nas relações entre entradas e saídas [11]. Após aprenderem as relações entre entradas e saídas, retornam as saídas apropriadas para novas entradas. Ou seja, os dados utilizados nestas técnicas de aprendizado supervisionado são acompanhados com a resposta esperada (valor da variável desejada). O sistema então se molda fazendo com que dado uma entrada, seja emitida como saída a resposta esperada para aquelas características. Estes modelos geralmente são usados em dois tipos de problemas: Classificação (Seção 2.3.1) e Regressão (Seção 2.3.2).

2.2.2 Aprendizado de Máquina Não Supervisionado

Para os modelos não supervisionados, são passados somente dados de entrada para que o algoritmo encontre padrões nesse conjunto de informações. A partir disso, o algoritmo tenta identificar possíveis tendências e situações que possam ocorrer novamente para obter as saídas esperadas. Este tipo de aprendizado é útil quando se deseja descobrir relações implícitas em um determinado conjunto de dados não rotulados. Um caso comum de uso desse tipo de algoritmo é a identificação de imagens similares [12].

2.3 Tipos de problemas

São várias as funções que sistemas de *Machine Learning* conseguem desempenhar em uma empresa. Pode-se citar como exemplo a identificação de falhas ou futuras falhas em produtos, classificação de materiais de acordo com suas especificações, entre outros. Dois problemas de modelagem preditiva que diferem entre si são explicados nos tópicos a seguir. Basicamente suas diferenças são relacionadas ao fato de um dos problemas preverem um rótulo (uma classe) e o outro retornar um valor quantitativo.

2.3.1 Classificação

Um modelo de classificação tira conclusões de valores observados para identificar uma determinada classe entre as possibilidades existentes. Este método se ajusta de acordo com os exemplos contidos no conjunto de dados em que o algoritmo identifica padrões para fazer futuras classificações, ou seja, o modelo é treinado [13]. Sendo assim, dadas

uma ou mais entradas, o modelo tentará prever a qual classe cada uma das entradas pertence.

2.3.2 Regressão

Os algoritmos de regressão preveem valores numéricos de saída com base nos dados que alimentam a entrada do sistema. O algoritmo constrói um modelo com base no seu treinamento sobre os recursos dos dados de entrada e utiliza este modelo para prever valores quantitativos dos novos dados. Da mesma forma que modelos de classificação, problemas de regressão são resolvidos a partir do treinamento de modelos para a identificação de padrões com dados rotulados. Sendo assim, esta técnica tem como objetivo gerar uma equação que relacione a variável resposta e uma ou mais variáveis explicativas (também chamadas de variáveis regressoras ou independentes), permitindo-se fazer a predição de valores da variável de interessse.

2.4 Métodos de Modelos Preditivos

Modelos preditivos são aqueles que são capazes de identificar padrões em uma massa de dados e a partir disso oferecer a probabilidade de resultados futuros com base em dados históricos.

Para cada problema de aprendizado de máquina, existem vários algoritmos que podem ser aplicados. Esta seção descreve alguns destes algoritmos. Neste trabalho será utilizada a biblioteca *XGBoost* que é altamente eficiente, flexível e portátil que trabalha com árvore de decisão resolvendo problemas de ciência de dados com maior velocidade que outros modelos preditivos [14]. Esta biblioteca será detalhada mais adiante na Seção 2.4.8.

2.4.1 Regressão Linear

Método que permite estimar um valor da variável resposta em função das variáveis preditoras, a Regressão Linear busca relações em variáveis que possuem linearidades entre si. Seu objetivo é encontrar os melhores valores para os coeficientes α e β de forma que os erros na predição da variável dependente a partir da(s) variável(s) preditora(s) sejam os menores possíveis. Modelos de regressão que não podem ser traduzidos a partir de uma função linear são denominados como não-lineares.

A Regressão Linear Simples (regressão com apenas uma variável preditora) pode ser definida pela expressão 2.1:

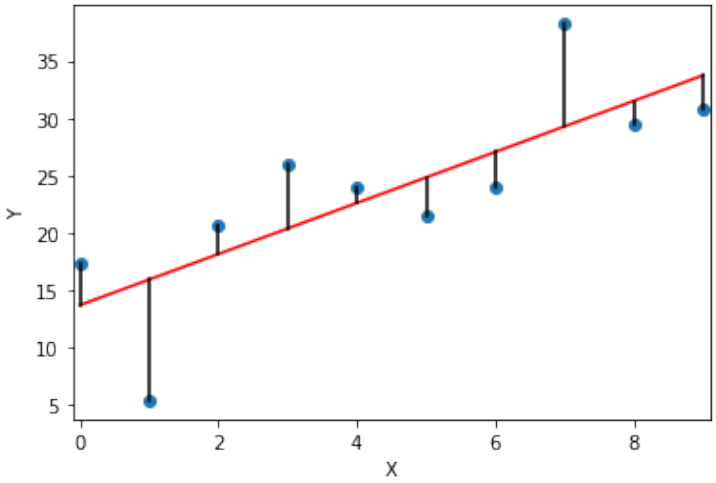
$$y = \alpha + \beta x + \varepsilon \tag{2.1}$$

Onde:

- y: variável dependente (valor a ser previsto)
- x: variável independente (variável preditora)
- α : constante
- β : coeficiente angular
- ε : variação de y que não é explicada pelo modelo (erro)

Existem muitas técnicas para ajustes da reta em meio ao conjunto de dados. Entre elas, há a técnica de Mínimos Quadrados que é uma das mais amplamente utilizadas [15] e devido a isso será brevemente descrita neste trabalho. Método de otimização que busca fazer com que a distância vertical entre os pontos de dados e a reta seja a menor possível (Figura 1), esta técnica procura minimizar a soma dos quadrados das diferenças entre o valor previsto e o valor real. Assim, a técnica dos Mínimos Quadrados busca aumentar o grau de ajuste do modelo aos dados observados.

Figura 1 – Reta Preditora



Fonte: A autora.

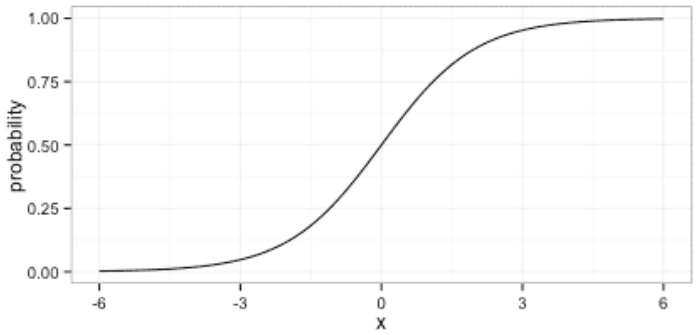
2.4.2 Regressão Logística

Outra alternativa de regressão, dessa vez não-linear, a regressão logística é um modelo que é capaz de indicar a possibilidade de um evento acontecer. Diferente da regressão linear que resulta em uma estimativa numérica, a regressão logística dá como resultado um valor binário (zero ou um) ou também pode-se obter as probabilidades dos eventos acontecerem. É uma técnica que é recomendada para situações em que a variável a

ser prevista é de natureza binária enquanto que as variáveis preditoras podem ser tanto categóricas quanto contínuas. Tem-se abaixo a equação da Regressão Logística Simples.

$$g(x_i) = \beta_0 + \beta_1 x \tag{2.2}$$

Figura 2 – Curva da Regressão Logística



Fonte: Tobias Madsen [16].

Na Equação 2.2, os coeficientes β_0 e β_1 são calculados a partir do conjunto dados utilizando-se o método da máxima verossimilhança. Este método encontra uma combinação de coeficientes que maximiza a probabilidade da amostra ter sido observada [17]. A variável x é a variável independente. Observa-se na Figura 2 a curva da regressão logística. Ela possui um comportamento probabilístico no formato da letra S, sendo uma característica deste tipo de regressão.

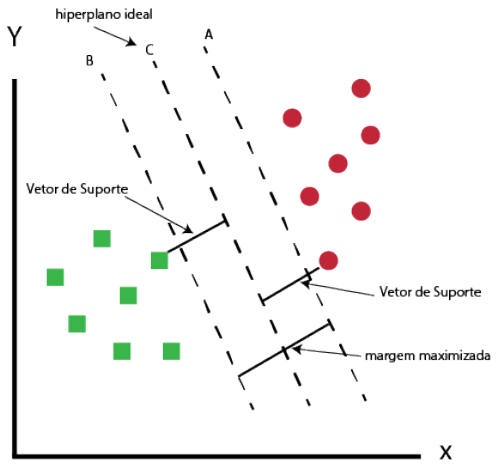
2.4.3 Máquina de Vetor de Suporte

SVM (*Support Vector Machine*, em português, Máquina de Vetor de Suporte) é um algoritmo que pode ser usado tanto para casos de classificação quanto regressão. O algoritmo separa as classes criando-se uma linha ou hiperplano entre os dados. Dessa forma, o *SVM* encontra pontos que estão dividindo as classes presentes no conjunto de dados e calcula a distância desses pontos até as linhas (Figura 3). Assim, com este algoritmo, busca-se encontrar a reta que maximize a distância entre todas as classes generalizando-se a divisão entre os dados.

2.4.4 Árvores de decisão

Árvores de decisão são um método de aprendizado supervisionado que pode ser usado tanto para classificação quanto regressão. Seu objetivo é prever valores de saídas, dadas pelas folhas das árvores, a partir de decisões sobre um atributo que ocorrem nos nós.

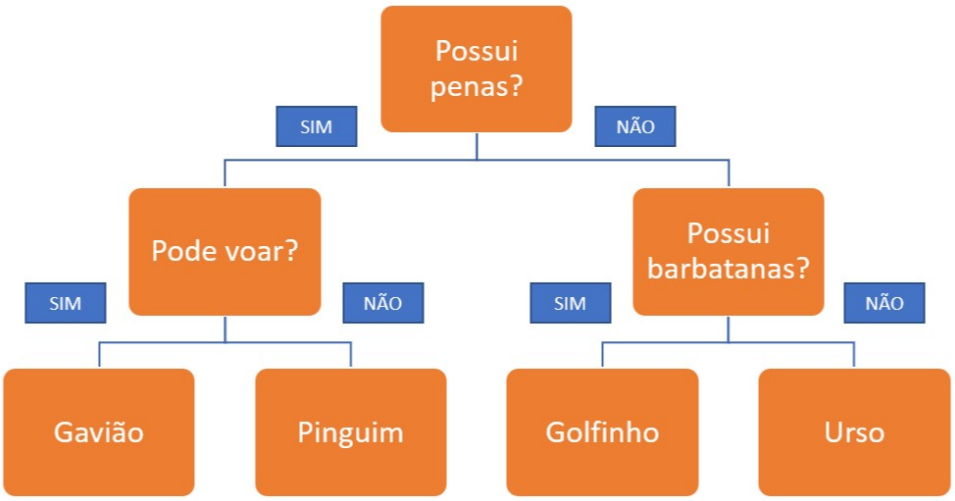
Figura 3 – *SVM - Support Vector Machine*



Fonte: Código Fluente [18].

Assim, cada filho de um nó representa os possíveis valores do atributo. Utilizam-se, na prática, conjuntos de árvores de decisão para obter melhores resultados porque, muitas vezes, somente uma árvore de decisão normalmente não é forte o suficiente para ser utilizada. Dessa forma, é comum usar várias árvores para se chegar a um único resultado [19]. Ferramenta de suporte à decisão, este método iterativo permite abordar o problema de forma estruturada e sistemática para se chegar a uma conclusão lógica. Observa-se na Figura 4 um exemplo de árvore de decisão. Nesta árvore, questionamentos sobre os dados são feitos e, de acordo com as respostas, dados são classificados.

Figura 4 – Árvore de Decisão

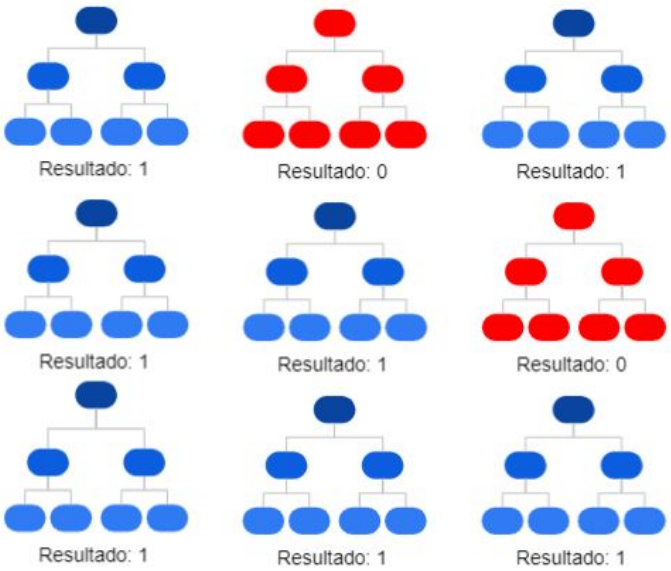


Fonte: A Autora.

2.4.5 *Random Forest*

Random Forest (RF), em português, Floresta Aleatória, é um método de regressão, e também de classificação, que consiste em um grande número de árvores individuais de decisão que operam como um conjunto. Cada árvore individual neste conjunto de árvores resulta em uma predição de resultado e, no fim, a classe com a maior quantidade de votos (para classificações) ou a média dos resultados das árvores (para regressão) se torna o resultado final de predição do modelo [20]. Como mostra a Figura 5, tem-se um exemplo de classificação para o *Random Forest* em que sete árvores individuais deram como resultado uma predição para o valor 1 e duas árvores resultaram no valor zero. Sendo assim, o resultado final é o valor resultante de maior frequência, nesse caso, um. O efeito disso é que enquanto algumas árvores podem estar erradas, muitas outras podem estar certas, ou seja, as árvores protegem umas às outras por seus erros individuais.

Figura 5 – Floresta Aleatória



Fonte: A Autora.

2.4.6 *k-Nearest Neighbors*

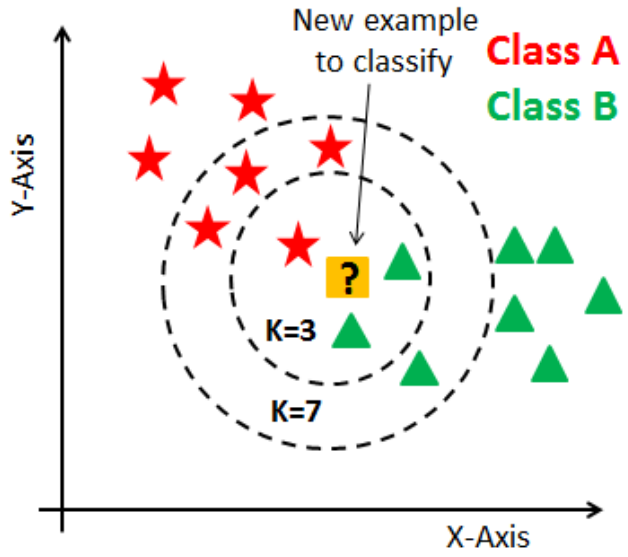
O algoritmo *k-NN*, de *k-Nearest Neighbors* (k-vizinhos mais próximos, em português) é um algoritmos de aprendizagem supervisionada em que o aprendizado é baseado na similaridade (também chamado por distância ou proximidade) dos dados.

Por exemplo, para uma amostra ainda não classificada, indicada por um ponto de interrogação na Figura 6, este algoritmo realiza o cálculo da distância desse ponto até as amostras já classificadas em suas classes mais próximas a ele. Com isso, após definida

a quantidade (k-vizinhos) de amostras mais próximas a serem analisadas, o algoritmo classifica o novo dado de acordo com as características dos exemplos mais próximos. Observa-se na Figura 6 duas opções da quantidade k de vizinhos mais próximos, sendo uma opção 3 e a outra 7. Se o valor de k é igual a 3, então o novo valor será classificado como sendo da classe B. Entretanto, se o valor de k é igual a 7, o novo exemplo seria classificado como pertencente a classe A.

Sendo assim, para este algoritmo, informações similares estão próximas entre elas e ele pode ser utilizado tanto para regressão e classificação [21]. No entanto, pelo fato de realizar cálculos a cada ciclo para determinar os vizinhos mais próximos, este algoritmo demanda de alta capacidade de processamento computacional.

Figura 6 – *k-NN* para k=3 e k=7



Fonte: kdnuggets [22].

2.4.7 *Naive Bayes*

Naive Bayes é um classificador que se baseia na probabilidade de cada instância pertencer a uma determinada classe. Este algoritmo não precisa de muitos dados para teste para se ter um resultado de boa precisão [23], sendo um algoritmo simples e ágil. O termo *"Naive"* (ingênuo, em português) é dado porque o algoritmo em questão não leva em conta as correlações entre as variáveis, ou seja, trata cada uma delas de forma independente [24].

2.4.8 *Extreme Gradient Boosting*

XGBoost, que vem de *Extreme Gradient Boosting* (Aumento de Gradiente Extremo, em português), é uma biblioteca de código aberto em C++, Java, Python, R e Julia, Perl, e Scala. Baseada em árvores de decisão, esta biblioteca fornece uma implementação eficaz e utiliza uma estrutura de *Gradient Boosting*, referindo-se a uma classe de algoritmos de aprendizado de máquina de conjunto usada para minimizar as perdas. O *XGBoost* foi amplamente reconhecido pelo seu bom desempenho nos aprendizados de máquina e, também, em desafios de mineração de dados. Isso pode ser observado pela quantidade de desafios vencidos utilizando-se esta biblioteca no site *kaggle*, próprio para competições de aprendizado de máquina. Entre 29 desafios que foram resolvidos e publicados no *blog Kaggle* em 2015, 17 soluções usaram o *XGBoost* [25].

A ideia de impulsionar (*boosting*) uma árvore de decisão surgiu do pensamento de que um modelo fraco pode ser alterado para virar um classificador melhor. E o primeiro *boosting* a ser aplicado com sucesso foi o *AdaBoost*, de *Adaptive Boosting* (Impulso Adaptativo). Neste algoritmo, os modelos fracos são as árvores que possuem somente uma única divisão. Seu funcionamento consiste em ponderar as observações colocando-se um maior peso nas instâncias difíceis de classificar. Dessa forma, modelos fracos são acrescentados sequencialmente a fim de que o treinamento se concentre nos padrões mais difíceis [26]. A prediçãofinal é realizada a partir da maior quantidade de resultados feitos pelos modelos fracos.

O *XGBoost* impulsiona a árvore de decisão fazendo com que a cada iteração realizada pelo algoritmo o modelo tente corrigir o erro cometido na iteração anterior. Foi projetado para possuir uma rápida execução e ser altamente eficaz [27]. A escalabilidade do *XGBoost* é o fator que levou ao seu sucesso. Seu sistema funciona além de 10 vezes mais rápido em comparação com as soluções populares [25]. E isso foi possível devido aos vários importantes sistemas e otimizações dos algoritmos. O princípio do *Gradient Boosting* é combinar saídas de classificadores menos eficientes (normalmente árvores de decisões) para resultar em um conjunto forte de decisão.

Utilizado para aprendizado de máquina supervisionado, o *XGBoost* buscou aperfeiçoar o desempenho do *Gradient Boosting* com a otimização tanto do *software*, com o conceito de regularização, quanto do *hardware*, a partir da paralelização e manejo de memória em cache. A regularização foi incorporada para evitar modelos que tendem a gerar sobreajuste (bastante conhecido pela expressão em inglês, *overfitting*) que é quando o modelo se ajusta bem ao conjunto de dados de treinamento mas não consegue prever de forma eficaz os dados do conjunto de teste. Ou seja, o modelo não consegue generalizar as respostas, apenas responder sobre exatamente aquilo que foi treinado. Já sobre a paralelização e o manejo de memória em cache, estes foram otimizados para diminuir o tempo computacional.

2.5 Métricas de avaliação de desempenho do modelo

Apesar de possuírem uma capacidade de processamento e armazenamento de informações muito maior do que um ser humano, algoritmos estão sujeitos a errarem suas previsões. Para isso, técnicas de avaliação de algoritmos são necessárias para entender o grau de confiança que se pode empregar sobre os resultados obtidos.

2.5.1 Métricas para classificação

Para problemas de classificação, normalmente são utilizadas as seguintes métricas: precisão, revocação ou especificidade, sensibilidade, acurácia, F1 e curva ROC_AUC. Para entender melhor essas métricas, a seguir tem-se uma explicação sobre conceitos de matriz de confusão em que auxiliará no entendimento dos conceitos dessas métricas como também nos posteriores resultados obtidos neste trabalho.

2.5.1.1 Matriz de Confusão

Em uma matriz de confusão, é possível observar os erros e acertos do modelo em relação aos valores reais. Para classificações com apenas duas classes, a matriz é do tipo 2x2. A diagonal principal dessa matriz é composta pelos dois tipos da classificação correta. Já a diagonal oposta representa os dois tipos de classificações incorretas. As classificações corretas são dadas pelos tipos: Verdadeiros Positivos (TP) e Verdadeiros Negativos (TN). E, portanto, as classificações com erro são as do tipo Falso Positivo (FP) e Falso Negativo (FN). Observa-se essas classificações distribuídas na Figura 7, em que:

- Verdadeiros Positivos (TP): correta classificação da classe Positivo;
- Falsos Negativos (FN): ocorre quando o resultado é previsto incorretamente como da classe Negativo quando é realmente da classe Positivo;
- Falsos Positivos (FP): classificação incorreta em que o modelo preveu como classe Positivo valores que deveriam ser da classe Negativo;
- Verdadeiros Negativos (TN): correta classificação da classe Negativo.

Para facilitar o entendimento, considere-se a matriz da Figura 8. Aproveitando o contexto desse trabalho de conclusão de curso, imagine que os valores ali expostos são resultantes de um modelo preditivo da possibilidade de haver queimada em uma área ambiental e que a classe Positiva é dada pela "Queima" e a classe Negativa é dada pela "Não Queima".

Com isso, observa-se que o modelo:

- Acertou sua predição 3 vezes para queima;

Figura 7 – Categorias da Matriz de Confusão

		Valores Preditos	
		Não Queima	Queima
Valores Reais	Não Queima	TN	FP
	Queima	FN	TP

Fonte: A Autora.

Figura 8 – Exemplo de Matriz de Confusão

		Valores Preditos	
		Não Queima	Queima
Valores Reais	Não Queima	4	1
	Queima	2	3

Fonte: A Autora.

- Acertou sua predição 4 vezes para não queima;
- Errou sua predição 2 vez para queima;
- Errou sua predição 1 vezes para não queima.

A métrica precisão retorna a probabilidade do modelo ter classificado corretamente as observações da classe Positiva. Ou seja, dentre as classificações que o modelo realizou da classe positiva, quantas delas estão corretas. É a proporção de instâncias que realmente têm classe "x", considerando todas que foram classificadas como "x".

$$Precisão = \frac{TP}{TP + FP} \tag{2.3}$$

Considerando a Figura 8, a precisão resulta em 0,75 ou 75%.

Outra métrica, Revocação ou Sensibilidade, calcula a taxa de acerto da classe positiva quando os valores esperados são dessa classe.

$$Sensibilidade = \frac{TP}{TP + FN} \tag{2.4}$$

De acordo com o exemplo da Figura 8, a sensibilidade é de 0,6 ou 60%.

Por outro lado, à especificidade diz respeito a capacidade do modelo de identificar resultados da classe Negativa quando este é a classe de interesse. Ou seja, esta métrica retorna a taxa de acerto da classe negativa.

$$Especificidade = \frac{TN}{FP + TN} \tag{2.5}$$

Com os valores da Figura 8, obtém-se uma especificidade de 0,8 ou 80%.

Tem-se, também, a acurácia. Ela basicamente indica a taxa de acerto de todas as classes presentes no modelo, ou seja, sua performance geral. Seu cálculo consiste no número de acertos divididos pelo número total de exemplos do conjunto de dados. Essa métrica é útil quando se tem a mesma quantidade de dados em cada classe. Casos de classes desbalanceadas (classes que apresentam diferenças nas quantidades de dados entre si) podem resultar em uma falsa impressão de que esta métrica está gerando um bom valor.

$$Acurácia = \frac{TP + TN}{TP + FP + TN + FN} \tag{2.6}$$

A acurácia para a matriz da Figura 8 é de 0,7 ou 70%.

Outra métrica bastante utilizada para casos de classificação é a F1. Ela consiste em uma média harmônica entre a precisão e a sensibilidade e é uma boa métrica em casos de classes desbalanceadas . Em geral, quanto maior o valor resultante desta métrica, melhor.

$$F1 = 2 * \frac{Precisão * Sensibilidade}{Precisão + Sensibilidade} \tag{2.7}$$

Por fim, o F1 Score resultante de acordo com a matriz de confusão da Figura 8 é de 0,66 ou 66%.

Além das métricas citadas anteriormente, outras duas métricas muito usadas para medir o desempenho de um modelo de aprendizado de máquina são: Características operacionais do receptor (em inglês, Receiver Operating Characteristics - ROC) e Área sob a curva (em inglês, Area Under the Curve - AUC). O cálculo dessas métricas pode ser feito ao plotar a sensibilidade e (1 - especificidade).

A curva ROC é uma curva de probabilidade que traça a taxa de Verdadeiros Positivos contra a taxa de Falsos Positivos de acordo com vários valores do parâmetro *Threshold*. Com este parâmetro, pode-se converter a probabilidade de predição em um rótulo de classe ao definir um valor limite de decisão. Por padrão, este valor é 0,5 para probabilidades previstas normalizadas ou probabilidades que ficam entre os valores 0 e 1 [28]. Neste trabalho, foi utilizado o valor padrão deste parâmetro. Dessa forma, para problemas de classificações binárias com rótulos de classe entre 0 e 1 e um *Threshold* de 0,5, tem-se:

- Predições $<$ 0,5 pertencem à classe 0;
- Predições $>=$ 0,5 pertencem à classe 1.

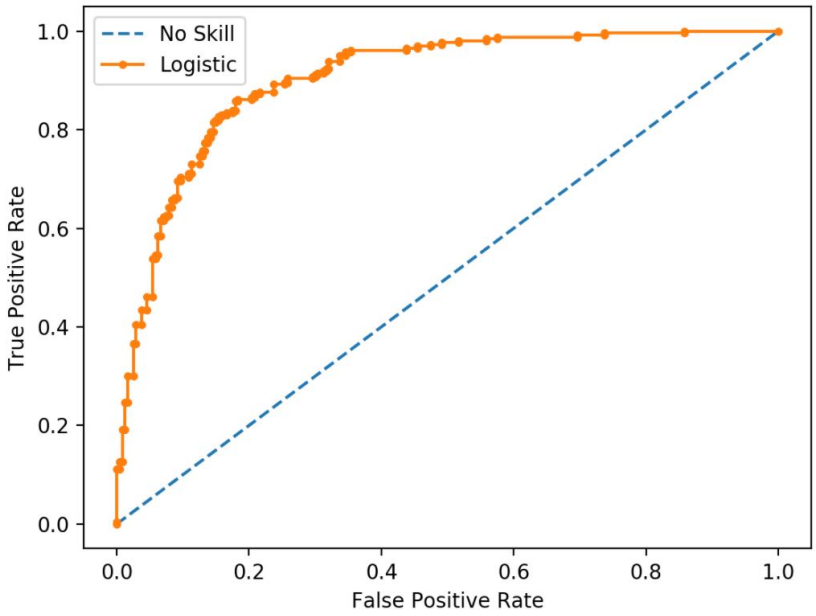
A área sob a curva (AUC) mede a capacidade de um classificador distinguir entre duas classes e é usada como um resumo da curva ROC. Quanto maior o valor da métrica AUC, maior é a capacidade do modelo de distinguir entre as classes positivas e negativas [29]. Seu valor varia entre 0 e 1. Assim, quanto mais próximo de 1, maior é o desempenho do modelo em generalizar suas predições para novos dados de entrada.

Na Figura 9, observa-se um exemplo de curva ROC-AUC. A linha pontilhada azul indica um valor AUC de 0,5 (modelo sem capacidade de distinguir classe positiva e classe negativa) e uma linha laranja, que é de um modelo de regressão logística. Neste gráfico, quanto maior o valor do eixo "x", maior é a taxa de Falsos Positivos do que Verdadeiros Negativos. E, quanto maior o valor do eixo "y", maior é a taxa de Verdadeiros Positivos do que Falsos Negativos. Dessa forma, o ponto no gráfico que define um modelo perfeito em termos de classificação está localizado no canto superior esquerdo, coordenada (0,1) no plano cartesiano. É neste local que se possui o maior valor para a especificidade e sensibilidade.

2.5.2 Métricas para regressão

Métricas mais comumente utilizadas para avaliar modelos de regressão são: Erro médio Absoluto (*Mean Absolute Error - MAE*), Erro Quadrático Médio (*Mean Squared Error - MSE*), Raiz do Erro Quadrático Médio (*Root Mean Squared Error - RMSE*) e o Erro Absoluto Médio Percentual (*Mean Absolute Percentage Error - MAPE*) e serão brevemente explicadas em seguida. Os cálculos dessas métricas são compostos pelo pelo valor real y_j da variável a ser prevista, o valor previsto $\hat{y}_j$ e pelo número "n" de observações selecionadas para teste.

Figura 9 – Exemplo de Curva ROC-AUC



Fonte: Machine Learning Mastery [30].

O Erro Médio Absoluto, é o cálculo da média da soma dos erros (diferença entre o valor real e o valor previsto) para um número "n" de observações de teste. Quanto menor o valor desta métrica, melhor.

$$MAE = \frac{1}{n}\sum_{j=1}^{n}[y_j - \hat{y}_j] \tag{2.8}$$

Outra métrica utilizada para problemas de regressão é o Erro Quadrático Médio, *MSE*. Esta métrica é muito utilizada para o cálculo de acurácias de modelos e é bastante sensível a grandes erros pelo fato da exponencial presente em sua operação, conforme a Equação (2.9).

$$MSE = \frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2 \tag{2.9}$$

A Raiz Quadrada do Erro Médio Absoluto, também conhecida por *RMSE*, é bastante utilizada para detectar efeitos de valores que estão fora do padrão entre os dados, uma vez que os erros são elevados ao quadrado antes da média ser calculada. Erros muito grandes

possuem um peso maior no cálculo do *RMSE* e devido a isso essa medida é muito útil em casos em que grandes erros não são aceitáveis. A equação do *RMSE*, Equação (2.10), é basicamente a raiz quadrada do resultado do *MSE*, Equação (2.9).

$$RMSE = \sqrt{\frac{\sum_j (y_j - \hat{y}_j)^2}{n}} \tag{2.10}$$

O Erro Percentual Absoluto Médio, *MAPE*, calcula o erro como uma porcentagem e, devido a isso, é possível fazer comparações de resultados entre modelos. Pelo fato dessas comparações ser de fácil visualização, esta métrica é muito utilizada em relatórios de gerenciamento.

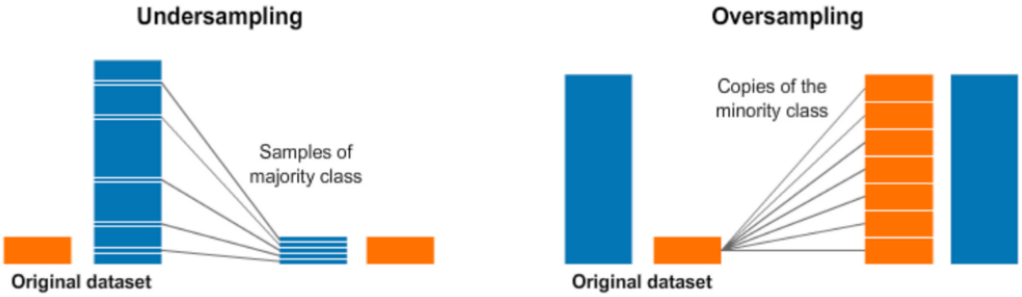
$$MAPE = \frac{1}{n} \sum_{j=1}^{n} \left| \frac{y_j - \hat{y}_j}{y_j} \right| \tag{2.11}$$

2.6 Base de dados desbalanceadas

Uma base de dados é considerada desbalanceada quando há muitos exemplos de uma classe em comparação com a outra classe (Figura 10). Quando se trata de um modelo de classificação, este ficará enviesado, aprendendo a classificar os novos dados de acordo com a classe que possui mais exemplos. Sendo assim, diz-se que as classes que possuem uma maior quantidade de dados são favorecidas enquanto as classes minoritárias possuem pequena taxa de reconhecimento [31]. Um caso bastante comum de desbalanceamento de dados é a detecção de fraudes [32] em que o número de fraudes é bem menor do que as operações legítimas. Em muitos casos, tem-se um interesse muito maior em classificar corretamente os exemplos das classes minoritárias, o que demanda um maior custo em casos de erros de classificação em relação aos elementos das classes majoritárias.

Existem métodos que consistem na reamostragem dos dados para lidar com bases de dados desbalanceadas e em muitos casos envolvem o aumento (sobreamostragem) do número de exemplos da classe minoritária, ou seja, a que possui menor número de dados. Também tem métodos que diminuem (subamostragem) o número de dados da classe que possui o maior número de exemplos. E, além desses métodos, há aqueles que combinam os dois tipos: sobreamostragem e subamostragem.

Figura 10 – Métodos de Reamostragens



Fonte: Semantix [33].

2.7 Otimização de hiperparâmetros

Antes de se treinar um modelo de aprendizado de máquina, é interessante encontrar um conjunto de valores de hiperparâmetros que oferença o melhor desempenho dos dados. Este processo é chamado de otimização ou ajuste de hiperparâmetros [34]. Hiperparâmetros são muito importantes para a performance de um modelo de aprendizado de máquina. Assim, devem ser escolhidos cuidadosamente. No entanto, na prática, é necessário treinar vários modelos com diferentes combinações de hiperparâmetros durante o processo de ajuste. Assim, a forma de se otimizar os hiperparâmetros tornou-se um ponto chave no desenvolvimento de modelos de aprendizado de máquina.

A busca por hiperparâmetros otimizados pode ser feita, principalmente, de duas formas: manual ou automática. A forma manual depende basicamente da intuição e experiência dos usuários que conseguem visualizar combinações de valores que afetem positivamente o modelo. No entanto, isso não é fácil de ser feito, visto que quanto maior a quantidade de combinações, mais difícil fica para um ser humano lidar com os números.

Para ajudar com a busca de hiperparâmetros otimizados, foram criados algoritmos que realizam buscas automáticas. Tem-se, por exemplo, a grade de pesquisa, também conhecida por *Grid Search*, em inglês. Este algoritmo tem como princípio a busca exaustiva [34]. Neste método, é criada uma matriz com possíveis valores de hiperparâmetros. Cada iteração realiza uma combinação com os hiperparâmetros e essa combinação é ajustada ao modelo. Depois disso, a performance do modelo é medida e a combinação de valores de hiperparâmetros que resultarem em um melhor desempenho do modelo é retornada.

3 Metodologia

Este capítulo está organizado em 3 seções e tem como objetivo retratar a metodologia aplicada para o desenvolvimento do modelo de predição de incêndio. A Seção 3.1 descreve brevemente sobre a empresa para a qual o modelo de predição de incêndio foi desenvolvido assim como conceitos relacionados ao canavial, o que deve facilitar o entendimento das próximas seções. A Seção 3.2 retrata as variáveis que foram usadas até se chegar ao modelo preditivo final. A Seção 3.3 descreve a metodologia para implementação do processo de *ML* para predição de incêndio.

Com o objetivo de promover a organização do capítulo, a Seção 3.3 é dividida em oito subseções. A primeira, Seção 3.3.1, explica a escolha das primeiras variáveis a serem dadas como entradas para o *XGBoost*. A Seção 3.3.2 contempla o tratamento de valores ausentes e a criação de novas variáveis. A próxima, Seção 3.3.3, comenta sobre o processo de categorização dos dados. Depois vem a explicação da divisão dos dados entre treino e teste, conforme Seção 3.3.4. Em seguinda, técnicas de reamostragem dos dados são expostas na Seção 3.3.5. Depois, comenta-se sobre o desempenho do modelo na Seção 3.3.6. E, por fim, ocorre a seleção das variáveis mais relevantes e os testes realizados cmo os pré-processadores, Seção 3.3.7 e Seção 3.3.8, respectivamente.

3.1 Caracterização da organização

A empresa para a qual o modelo de predição de incêndio descrito neste trabalho foi desenvolvido está localizada na região noroeste do estado de São Paulo e trabalha com a matéria-prima *cana-de-açúcar*. Os produtos finais do processamento da cana podem ser tanto o açúcar quanto o etanol, além de outros produtos relacionados. O bagaço da cana-de-açúcar também é utilizado para a geração de eletricidade a partir do vapor.

Com um canavial extenso, é fundamental a empresa possuir sistemas de acompanhamento das condições meteorológicas e também de focos de queima a fim de que a área como um todo seja monitorada e cuidada. Para isso, a companhia conta com estações meteorológicas espalhadas pelos canaviais onde há sistemas de sensoriamento remoto para a obtenção de dados como temperatura do ambiente, quantidade de chuva, umidade do solo, radiação solar, velocidade do vento entre outras. Além do sensoriamento remoto feito nessas estações localizadas no meio do canavial, a empresa também conta com satélites para o monitoramento de suas plantações, implantados principalmente para enfrentarem os problemas relacionados às queimas que atingem o setor sucroenergético. O acompanhamento é realizado a partir da utilização de 13 satélites e é baseado no georreferenciamento orbital. Dessa forma, é possível realizar um monitoramento remoto em que há a emissão

de alertas automaticamente para a Central de Controle quando focos de incêndios são detectados. Este sistema traz uma maior agilidade no combate a incêndios e serve de suporte para a alocação de recursos como a área de brigada de incêndio.

Além de agilizarem as tomadas de decisões quanto aos incêndios, os satélites também enviam informações da situação do canavial referentes ao Índice de Vegetação por Diferença Normalizada (*Normalized Difference Vegetation Index - NDVI*). Este índice faz a análise da condição da vegetação em que mede a intensidade da clorofila nas plantas, permitindo-se fazer várias aplicações que diminuem as perdas e aumentam a produtividade. É bastante utilizada para o monitoramento de plantações, detecção de secas e localização de pragas. Um índice de vegetação que é derivado do *NDVI* e também é obtido pelos mesmos satélites é o *HHI*, Índice de Homogeneidade. Este índice também é bastante usado para o monitoramento agrícola.

Para facilitar a compreensão dos próximos tópicos deste capítulo, a seguir tem-se uma breve explicação sobre alguns termos utilizados para definirem áreas do canavial e suas divisões, além de conceitos utilizados sobre a plantação.

- Unidade industrial: a empresa é composta por mais de uma unidade industrial. Todas as unidades processam a cana-de-açúcar, mas possuem diferenças particulares em processos e produtos finais;
- Canavial: extensa área com aglomerado de canas-de-açúcar;
- Bloco: área do canavial composta por vários talhões;
- Talhão: unidade mínima de cultivo em que seu comprimento é ideal para preencher a capacidade de um caminhão de transbordo (caminhão que transfere a cana no campo para o caminhão que vai até a usina);
- Maturação: processo natural da cana-de-açúcar em que há a redução da produção de hormônio para o crescimento e seu metabolismo é direcionado para o acúmulo de sacarose;
- Corte: a cana-de-açúcar pode ser cortada ou colhida em média 5 vezes sem precisar replantar. Sendo assim, cada vez em que a plantação é cortada, ela quantifica um número para o corte;
- Ambiente: determina a característica do solo onde a cana está implantada e é indicado pelas letras de A até a E; na medida em que os ambientes tramitam de A para E, a produtividade da cultura vai decrescendo.

3.2 Dados utilizados

Como citado na Seção 1.3, o modelo de predição visa prever probabilidades de ocorrência de incêndios. As variáveis de entrada para o modelo são compostas por dados independentes uns dos outros. Cada um desses dados é um atributo (ou *feature*). A variável de saída, que é a variável a ser predita pelo modelo, pode ser chamada de dependente pois seu valor depende dos valores das variáveis de entrada. Para o modelo, a variável de saída possui o nome de "queima" e ela pode assumir o valor 0, indicando que não há queima, e 1, quando há queima. Estas informações são armazenadas no banco de dados da empresa.

Como descrito no parágrafo anterior, o modelo de predição tem como objetivo prever a probabilidade de ter ou não queima em partes do canavial em função da combinação de variáveis que, inicialmente, eram: unidade industrial, blocos dos canaviais, ambiente, corte, maturação, *NDVI*, *HHI*, temperatura (mínima, média e máxima), chuva, umidade do solo (profundidade de 5cm e 25cm), radiação solar, centroide dos blocos, bases georreferenciadas tanto das rodovias do Brasil além das áreas urbanas da região noroeste do estado de São Paulo. Ainda assim, esse conjunto de atributos (características ou propriedades) inicial foi refinado ao longo do desenvolvimento do modelo por meio do uso de métodos de seleção de atributos, como discutido na Seção 3.3.7 e também houveram novas variáveis criadas a partir dos dados brutos. Pode-se citar como exemplo a média de temperatudas de dias anteriores à queima ou colheita do canavial. O conjunto de informações que foi utilizado para prever a queima foi usado para treinar o algoritmo e todas as variáveis foi obtido do banco de dados *PostgreSQL*. Nele estão contidos dados de diversos sistemas, tanto da área corporativa como industriais e agrícola. E foi utilizado o ambiente *Jupyter Notebook* para o desenvolvimento do algoritmo.

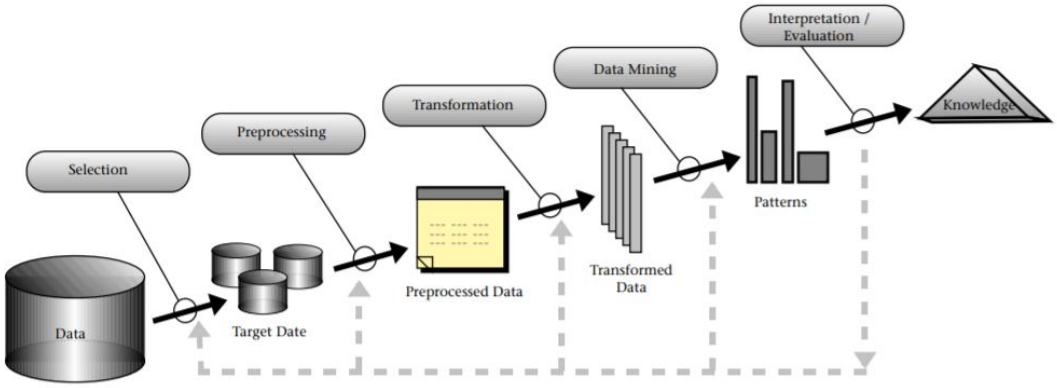
Nos experimentos para o desenvolvimento do algoritmo de predição foram utilizados dados de quatro anos, mais especificamente 2017, 2018, 2019 e 2020. Não foram utilizados dados anteriores a estes pelo fato de as informações sobre a entrada de cana de açúcar queimada nas usinas não ter diferenciação entre canas queimadas propositalmente, em que ainda eram permitidas por lei, e as queimadas acidentalmente por causas naturais ou criminosas.

3.3 Descrição da Metodologia Adotada

A Figura 11 mostra um diagrama do fluxo de processos para *Machine Learning*. Neste diagrama, mostra-se os processos pelos quais os dados brutos passam até serem "transformados" em conhecimento para um sistema. A metodologia aplicada a este trabalho pode ser explicada por este diagrama em que, a partir de dados, pode-se prever a probabilidade de ocorrência de incêndio em partes do canavial.

É importante que ocorra um processo de tratamento das variáveis a fim de se remover informações que não são relevantes ou que podem dificultar o aprendizado do modelo. Neste trabalho, para iniciar os experimentos, foi realizada uma seleção das possíveis variáveis que pudessem contribuir com o modelo, conforme a Seção 3.3.1. Com os atributos selecionados, foi feito um pré-processamento com valores ausentes, em que estes foram substituídos conforme explicado brevemente na Seção 3.3.2.1. Também foi realizada a transformação dos dados de acordo com o que a biblioteca *XGBoost* exige para ser aplicada. Alguns atributos que possuem suas instâncias no formato textual passaram por uma transformação para que suas instâncias fossem numéricas (Seção 3.3.3). Além disso, uma outra etapa do pré-processamento dos dados brutos é a seleção das instâncias que são importantes para o aprendizado do modelo de *ML*. Dados redundantes ou irrelevantes atrapalham o desempenho dos algoritmos e por isso é de suma importância que os dados passem por um pré-processamento. A seleção das variáveis mais relevantes pode ser observada na Seção 3.3.7.

Figura 11 – Passos para extração de conhecimento dos dados



Fonte: AI Magazine [35].

Sendo assim, o fluxo de processos da metodologia adotada neste trabalho pode ser seguido pela Figura 11 em que iniciou-se com a seleção das variáveis de interesse; após, ocorreu o pré-processamento dos dados trabalhando-se sobre os atributos ausentes. Em seguida, algumas variáveis foram transformadas de forma que a sintaxe do algoritmo *XGBoost* exige. Depois, o conjunto de dados foi submetido a um processo de seleção de variáveis onde destes considerou-se apenas as relevantes e são utilizados como entrada para o algoritmo *XGBoost*. Após o classificador identificar e aprender os padrões dos dados, seu desempenho foi avaliado para observar seu desempenho.

3.3.1 Seleção de Variáveis

Para o desenvolvimento do modelo de predição de incêndio, foram selecionadas, inicialmente, possíveis variáveis que pudessem impactar positivamente para a predição. De forma a identificar a probabilidade de ocorrência de queimadas, variáveis relacionadas a condições meteorológicas foram escolhidas, visto que impactam diretamente nas incidências de queimas. Da mesma forma, outras variáveis que caracterizam os cenários do solo e da cana-de-açúcar também foram selecionadas. As variáveis utilizadas nos experimentos durante o desenvolvimento do algoritmo preditivo foram: unidade industrial, blocos dos canaviais, ambiente, corte, maturação, *NDVI*, *HHI*, temperatura (mínima, média e máxima), chuva, umidade do solo (profundidade de 5cm e 25cm), radiação solar, centroide dos blocos, bases georreferenciadas tanto das rodovias do Brasil além das áreas urbanas da região noroeste do estado de São Paulo.

3.3.2 Pré-processamento

Nesta seção serão apresentadas algumas técnicas de pré-processamento dos dados que foram aplicadas. Essas técnicas foram fundamentais para preparar, organizar e estruturar os dados para que o algoritmo tenha um bom desempenho. Além disso, é necessário deixar os dados com o formato de acordo com o que a biblioteca *XGBoost* exige.

3.3.2.1 Tratamento dos Valores Ausentes

Uma etapa muito importante no processo de análise exploratória dos dados é o pré-processamento, em que nela ocorre a limpeza e correção dos dados.

Alguns problemas são comuns em diferentes tipos de conjunto de dados. Dessa forma, muitas técnicas de análise exploratória dos dados são aplicadas similarmente nos mais diversos casos de desenvolvimento de modelos de *Machine Learning*. Neste trabalho, valores ausentes das variáveis foram identificados entre os dados brutos e isto foi tratado a partir de diferentes técnicas.

Após selecionar as possíveis variáveis preditoras, foi realizado o tratamento dos valores ausentes apenas das variáveis que possuíam valores nulos. Assim, os primeiros valores nulos identificados foram das seguintes variáveis: temperatura e chuva. Valores nulos da variável "temperatura" foram tratados substituindo-se estes espaços pela média dos valores obtidos dessa variável. Assim, seja por um erro de medição ou falha no sensor, os valores nulos dessa variável foram substituídos por valores próximos aos já coletados. Para a variável "chuva", espaços nulos foram substituídos pelo valor *zero*. Com isso, pode-se dizer que estes valores nulos foram considerados como dias sem chuva. Essa substituição não afetou o restante dos dados devido à baixa quantidade de valores nulos em comparação com a quantidade total de dados coletados.

Outra variável que possuía valores nulos foi o "centroide" dos blocos. Este atributo define a localização de cada bloco no canavial através da longitude e latitude. Dessa forma, blocos que não possuíam o centroide definido no banco de dados foram excluídos da análise. Essa falta de informação pode ter origem em atualizações nas localizações de alguns blocos no sistema. No entanto, isto também não afetou o restante dos dados presentes.

Assim, somente estas três variáveis (temperatura, chuva e centroide) apresentaram valores nulos durante a escolha e seleção inicial das variáveis. Após a criação de novas variáveis a partir das já existentes (Seção 3.3.2.2), surgiram novos valores nulos devido às técnicas utilizadas e seus tratamentos serão citados brevemente na próxima seção.

3.3.2.2 Criação de Novas Variáveis

Um processo que faz o uso de técnicas para criar novas variáveis que podem ser relevantes para aumentar o poder preditivo do algoritmo de aprendizado de máquina é a Engenharia de Recursos, muito conhecida por *Feature Engineering*. Com ela, pode-se extrair mais informações das variáveis brutas ou que ainda não foram transformadas.

Inicialmente, a variável temperatura (mínima, média e máxima) e a chuva foram usadas para criar novas variáveis a partir delas. Novas colunas foram adicionadas em que possuem a média da temperatura e chuva de dias anteriores. Assim, utilizou-se a função *rolling()* da biblioteca *Pandas*. Para exemplificar, observa-se a Tabela 1. Nesta tabela há o dia (formato ano - mês - dia) em que a informação foi coletada, a temperatura máxima medida neste dia e a nova coluna criada ("temp_max4").

Tabela 1 – Exemplo da aplicação de Engenharia de Recursos

Criação da coluna "temp_max4"		
Dia	temp_max	temp_max4
2017-04-01	29.320000	NaN
2017-04-02	30.830000	NaN
2017-04-03	31.510000	NaN
2017-04-04	34.759998	31.605000
2017-04-05	31.250000	32.087500

Assim, pode-se observar na Tabela 1, que os primeiros valores da coluna "temp_max4" são nulos pois não havia dados anteriores para ele realizar o cálculo, visto que as informações coletadas no banco de dados são do período de abril de 2017 até dezembro de 2020. A média dos 4 primeiros valores da coluna "temp_max" resulta no primeiro valor não nulo da coluna "temp_max4". E foi dessa forma que todos os outros valores foram calculados. Com isso, utilizando-se a função *rolling()*, foram criadas novas colunas para a chuva e temperatura com a média de quatro, oito, quinze, vinte, trinta e quarenta dias anteriores. Nota-se que surgiram espaços nulos nestas novas colunas. Assim, eles foram

substituídos aplicando-se o mesmo método comentado na Seção 3.3.2.1: para a temperatura, substituiu-se os espaços nulos pelo valor da média da coluna e, para a chuva, atribuiu-se o valor zero. Outras variáveis que receberam a mesma transformação foram a radiação solar e a umidade no solo (profundidade 5cm e 25cm). Para elas, também foram criadas novas colunas fazendo-se a média dos valores. E, os espaços nulos que surgiram foram trocados pela média dos valores da própria coluna.

Outra *Feature Engineering* aplicada foi para a formação de uma variável que indica a distância dos blocos do canavial até as rodovias mais próximas ("dist_rodovia"). E outra variável que aponta a distância dos blocos até as cidades mais próximas ("dist_urbano"). As distâncias dos blocos do canavial até rodovias e áreas urbanas foram originadas a partir de bases de dados georreferenciados. Assim, com as coordenadas geográficas necessárias, fez-se o cálculo distância de cada bloco do canavial até as rodovias e cidades mais próximas a eles.

3.3.3 Transformação dos Dados

A transformação do conjunto de dados se refere ao processo de edição de texto de algumas instâncias de variáveis de maneira a deixar a sintaxe correta para que o *XGBoost* possa receber os dados. Entre as seis variáveis escolhidas finais para comporem as entradas do modelo, quatro delas passaram por uma transformação.

A primeira variável a sofrer alteração foi a "unidade_numerica", que define o nome das unidades industriais. Cada nome foi trocado por um número porque o *XGBoost* não processa dados em formato de texto. A fim de exemplificar a transformação realizada com os nomes das unidades industriais mantendo o sigilo da empresa, ilustra-se na Tabela 2 como foram atribuídos os valores de acordo com o nome das unidades, não sendo a quantidade de unidades, de fato, o real.

Tabela 2 – Transformação da variável "unidade_numerica"

Transformação da variável "unidade_numerica"	
Unidade	Valor atribuído
Unidade W	1
Unidade X	2
Unidade Y	3
Unidade Z	4

Depois, a segunda variável que passou pela transformação foi a que retorna a distância entre os blocos dos canaviais até as rodovias mais próximas. Ela foi dividida em intervalos e para cada intervalo um valor foi atribuído, categorizando-se a variável. Com isso, sete categorias para as distâncias foram criadas (Tabela 3) a fim de facilitar para o modelo a identificação de queima através da frequência de casos de acordo com cada intervalo.

Antes de realizar esta categorização, o modelo recebia milhares de valores distintos. Há infinitos números entre dois valores. Após a categorização dos dados, o modelo possui somente sete intervalos para reconhecer um padrão nos dados de acordo com os intervalos. Essa divisão dos dados em sete categorias foi feita da mesma forma em que ocorreu para a variável "temp_max40", como mostrado mais adiante. Assim, foi analisada a distribuição de frequência e, após, decidiu-se separar em 7 conjuntos.

Tabela 3 – Transformação da variável "dist_rodovia"

Transformação da variável "dist_rodovia"	
Intervalo	Valor atribuído
Até 5 km	1
De 5 km até 10 km	2
De 10 km até 15 km	3
De 15 km até 20 km	4
De 20 km até 25 km	5
De 25 km até 30 km	6
Maior que 30 km	7

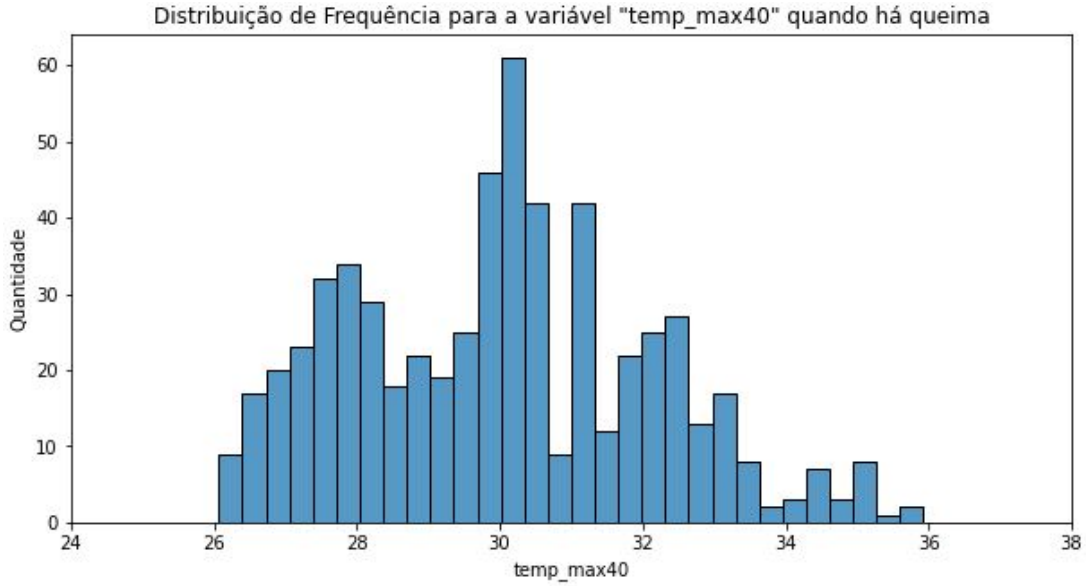
A variável "tipo_maturacao", que mostra o tipo de maturação da cana-de-açúcar, também passou por transformações. Como as informações que compõem o tipo de maturação da cana são em formato de texto, sendo os campos dados por "Precoce", "Média" ou "Tardia", atribuiram-se os números 1, 2 e 3 para elas, respectivamente (Tabela 4).

Tabela 4 – Transformação da variável "tipo_maturacao"

Transformação da variável "tipo_maturacao"	
Maturação	Valor atribuído
Precoce	1
Média	2
Tardia	3

Outra variável que passou pelo processo de transformação foi a temperatura. O melhor resultado das métricas de avaliação do desempenho do modelo entre as temperaturas brutas e criadas foi quando utilizou-se a média das temperaturas máximas dos últimos quarenta dias antes da queima ou da colheita do canavial. Sendo assim, foi escolhida a variável "temp_max40" como uma variável de entrada para o modelo. A Figura 12 mostra a distribuição de frequência desta variável para casos em que somente houve queima. Assim, para facilitar o aprendizado do modelo, seus valores foram separados em sete intervalos e foram enumerados de 1 até 7 (Tabela 5). Com isso, antes haviam infinitos valores para o modelo identificar padrões. Agora, há somente sete.

Figura 12 – Distribuição de Frequência da variável “temp_max40” para casos de queima



Fonte: A Autora.

Tabela 5 – Transformação da variável “temp_max40”

Transformação da variável “temp_max40”	
Intervalo	Valor atribuído
Até 26 ºC	1
De 26 ºC até 28 ºC	2
De 28 ºC até 30 ºC	3
De 30 ºC até 32 ºC	4
De 32 ºC até 34 ºC	5
De 34 ºC até 36 ºC	6
Maior que 36 ºC	7

3.3.4 Separação entre Treino e Teste

Em aplicações de *Machine Learning*, é comum dividir os dados em dois subconjuntos: treino e teste (Figura 13). Com estas divisões dos dados, pode-se acontecer um sobreajuste ou subajuste do modelo sobre os dados de treino. Dessa forma, muitas vezes ocorre o particionamento em três subconjuntos para se evitar estes problemas: treino, validação e teste. Inicialmente, o modelo foi avaliado somente com a divisão entre treino e teste. Após os primeiros resultados e depois de aplicar as técnicas de reamostragem, ele passou a ser avaliado com as três subdivisões descritas acima, mostrado na Seção 3.3.6.

Depois de aplicadas as etapas de pré-processamentos e transformações dos dados, foi realizada a divisão dos dados entre subconjuntos de treino e teste. O subconjunto de treino contém informações se houve queima ou não, de acordo com as variáveis preditoras. Assim,

Figura 13 – Divisão em Treino e Teste

Fonte: A Autora.

o modelo aprende a partir desses dados de forma a estar generalizado para receber novos dados posteriormente. E o subconjunto de teste é usado para verificar se as predições do modelo estão próximas com a realidade.

Para a divisão entre treino e teste, foi utilizada a biblioteca do *Scikit-Learn*, que é bastante usada especificamente para *Machine Learning* [36], e foi aplicado o método *train_test_split*. A porcentagem em relação à quantidade total dos dados para treinar o modelo foi de 70% e o restante, 30%, ficou para o testar o modelo. Essa separação de porcentagens 70/30 é bastante comum. Outras separações comumente utilizadas são: 67/33 e 50/50. A escolha dessas porcentagens dependem dos objetivos do projeto. O custo computacional sobre o treinamento do modelo, sobre a avaliação e a representatividade dos conjuntos de treino e teste podem ser levados em consideração [37].

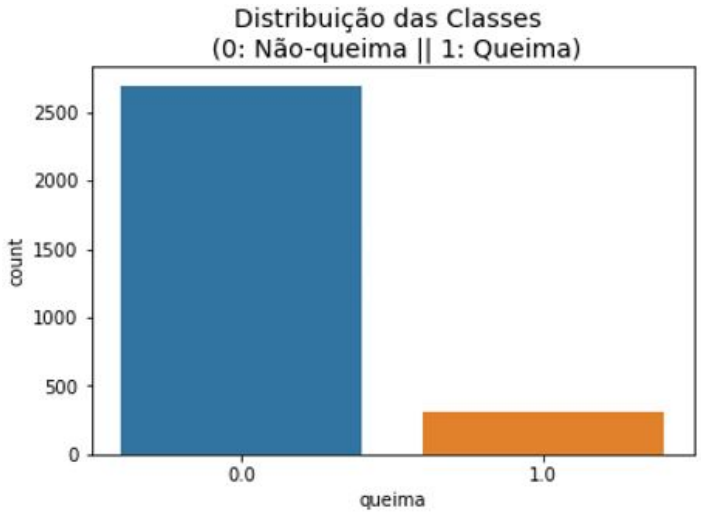
3.3.5 Técnicas de Reamostragem

Após a etapa de divisão dos dados entre treino e teste, na Seção 3.3.4, foram aplicadas técnicas de reamostragem no subconjunto de treinamento. Observa-se, na Figura 14, a distribuição dos dados de treinamento em relação à variável a ser prevista ("queima"). Nota-se, nesta figura, que a quantidade de amostras referentes à classe "não-queima" (2701 exemplos) é muito maior do que a quantidade de exemplos da classe "queima" (308 exemplos). Somente um pouco mais de 10% dos dados totais do subconjunto de treinamento pertencem à classe positiva. Isto é conhecido por *classes desbalanceadas*.

Classes desbalanceadas podem enviesar as decisões do modelo. Dessa forma, é importante que técnicas de reamostragem dos dados sejam aplicadas. Isto é feito para que o modelo não se ajuste demais à classe que possui a maior quantidade de amostras e, consequentemente, não faça boas predições da classe que possui a menor quantidade de amostras onde, muitas vezes, é o foco da atenção.

Para problemas de classes desbalanceadas, uma saída para lidar com isso é rebalancear as classes. Isto pode ser feito retirando-se amostras da classe que possui uma maior

Figura 14 – Distribuição dos dados da variável a ser prevista



Fonte: A Autora.

quantidade de dados (método de subamostragem) ou aumentando-se a quantidade de amostras da classe que possui a menor quantidade de dados (método de sobreamostragem) [38]. E, ainda, há técnicas que utilizam tanto a subamostragem quanto a sobreamostragem em conjunto.

Neste trabalho foram aplicados os seguintes métodos de reamostragem:

- *SMOTE()* ou *Synthetic Minority Oversampling TEchnique*;
- *RandomUnderSampler()*
- Combinação de *RandomOverSampler()* e *RandomUnderSampler()* ;
- Combinação de *SMOTE()* e *RandomUnderSampler()* ;
- *SMOTE-NC()* ou *Synthetic Minority Oversampling TEchnique - Nominal-Continuous.*

A classe *SMOTE()*, técnica de sobreamostragem minoritária sintética, é um dos métodos dominantes na literatura que atinge essa geração de amostra extra [38]. De maneira sucinta, este método gera pontos em um segmento de linha. Um ponto de dado selecionado aleatoriamente é conectado até uns de seus k-vizinhos mais próximos e, por meio de operações matemáticas, um ou mais novos pontos de dados são criados neste segmento de linha [39].

Uma variação da classe *SMOTE()*, é a classe e *SMOTENC()*. A diferença entre as duas consiste no fato de que a classe *SMOTENC()* é utilizada quando há dados categórios além de numéricos entre os dados.

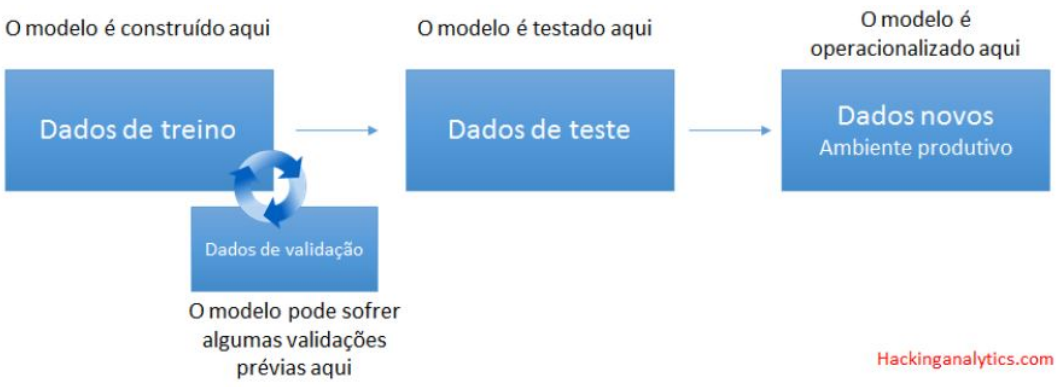
Para realizar a técnica de subamostragem, foi aplicada a classe *RandomUnderSampler()*. A forma mais simples desta técnica consiste em diminuir as amostras dos dados majoritários de forma aleatória a fim de balancear a distribuição das classes [40]. Esta técnica, apesar de simples e eficiente, pode deletar dados importantes, já que os dados a serem excluídos são escolhidos aleatoriamente. De forma análoga à classe *RandomUnderSampler()*, a classe *RandomOverSampler()* cria dados aleatórios para a classe que possui a menor quantidade de amostras nos dados de treinamento.

Neste trabalho, a biblioteca utilizada para criar dados extras de treinamento para todas as técnicas foi a *imbalanced-learn*, em que providencia ferramentas para lidar com classificações de classes desbalanceadas. Os experimentos com cada pré-processador foram descritos na Seção 3.3.8.

3.3.6 Desempenho do modelo

Para saber o desempenho de um modelo em relação ao seu aprendizado sobre os padrões dos dados, é realizada uma técnica de validação cruzada. Assim, é feito um particionamento do conjunto de dados em subconjuntos mutuamente exclusivos do mesmo tamanho. A partir desta divisão, primeiramente, alguns subconjuntos são usados para o treinamento do modelo. Após, o restante dos dados, que não foram usados no treino do modelo, são utilizados para a validação e teste (Figura 15). Este processo funciona como uma simulação de como o modelo se comportaria frente à entrada de novos dados, que antes nunca foram vistos, em ambiente produtivo.

Figura 15 – Tipos de conjuntos de dados no processo de modelagem

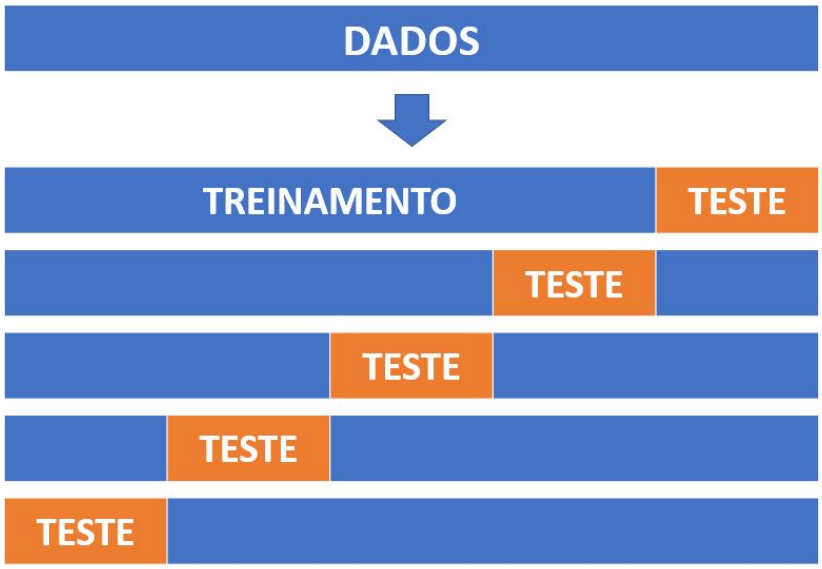


Fonte: Hacking analytics [15]

Durante o processo de treino do modelo, pode-se dividir os dados de treino em diferentes subconjuntos. Estes são chamados de dados de validação e são usados para validações iniciais durante a aprendizagem do modelo.

No algoritmo deste trabalho, foi utilizada a técnica de *cross validation*. O tipo aplicado foi o *repeated k-fold cross validation*, em que ele consiste em dividir os subconjuntos em k grupos. Um desses grupos é usado para teste e o restante, *k-1*, para a estimação do desempenho do modelo a partir das métricas com os dados de treinamento do modelo. Este processo é realizado k vezes alternando de forma circular o subconjunto de teste. Dessa forma, ao final das k iterações, a capacidade do modelo em generalizar suas decisões é calculada, obtendo-se, com isso, uma medida mais confiável. A vantagem de aplicar essa técnica consiste no fato de que diferentes porções dos dados são aplicados ao modelo. Pode-se treinar e testar o modelo com todos os dados disponíveis, evitando-se a variância. Isso faz com que o modelo conheça previamente diferentes combinações de conjuntos de dados, aumentando a sua chance de ter melhores resultados em dados nunca antes vistos.

Figura 16 – Reamostragem com Validação Cruzada

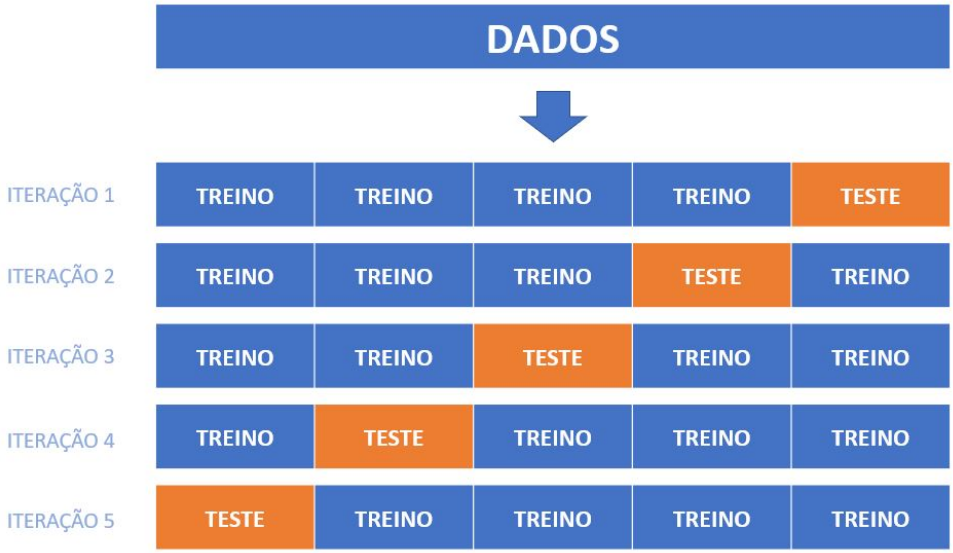


Fonte: A autora.

A Figura 16 mostra a separação dos dados em grupos de treino e teste. Os dados são misturados a cada iteração e são separados em k número de grupos. Assim, a cada repetição tem-se um conjunto distinto de dados para treino e teste, conforme mostra a Figura 17.

Neste trabalho foi utilizada a função *cross_val_score()* para aplicar a técnica de reamostragem. Foi usada uma quantidade $k = 10$, que é um valor padrão popular [41]. E, a métrica para medir o desempenho do modelo foi a "ROC_AUC". Para obter esse valor final da métrica escolhida, é realizada a média dos valores resultantes de todas as iterações. A métrica AUC foi obtida utilizando-se o método *roc_auc_score()* da biblioteca *sklearn*.

Figura 17 – Iterações na Validação Cruzada



Fonte: A autora.

3.3.7 Seleção das variáveis relevantes

A importância de uma variável de entrada para um modelo preditivo se refere ao quanto ela é útil para a previsão dos valores de saída. Isso acontece através de *pontuações de importância*, que indicam o quanto um recurso é importante para a predição oferencendo *insights* sobre os dados. Também, ajuda a selecionar apenas as variáveis que melhoram a eficiência do modelo preditivo. Em muitos casos, nem sempre todas as variáveis preditoras fornecem informações relevantes para o classificador. A etapa de seleção das variáveis relevantes, de acordo com a Figura 11, ocorre após a avaliação do modelo.

Após aplicadas as métricas para verificar o desempenho do modelo quanto às variáveis preditoras utilizadas, as variáveis que apresentam baixo impacto positivo são eliminadas e passam a não ser utilizadas voltando à etapa de seleção de variáveis. Essa redução de informações para o modelo pode diminuir o tempo de processamento e os requisitos de armazenamento, aumentando-se, assim, a eficiência de processamento [42].

As pontuações de importância podem ser usadas para problemas de regressão e classificação. Para obter essas pontuações, existem muitas maneiras e modelos que podem ser aplicados para se chegar ao ranqueamento. O cálculo da importância das variáveis neste trabalho foi baseado em modelos de árvores de decisão já que a biblioteca utilizada (*XGBoost*) tem sua estrutura baseada em árvore e esta é uma das maneiras mais rápidas de se obter a importância das variáveis [43]. A técnica foi aplicada sobre os dados de treinamento.

Para o ranqueamento baseado em árvores de decisão, cada nó da árvore é uma condição

de como se dividir os valores para um único resultado, de modo que valores semelhantes da variável dependente terminem no mesmo conjunto após a divisão. A condição para o caso específico deste trabalho, que se trata de classificação, é a impureza de Gini ou ganho de informação. Este índice de Gini mede o grau de heterogeneidade dos dados. Por isso pode ser usado para quantificar o grau de impureza de um nó [44]. Para casos de regressão, a impureza é a variância (medida que mostra o quão distante um valor está da média dos valores do conjunto de dados). Sendo assim, para ambos os casos, o cálculo é realizado de forma a se verificar o quanto cada variável impacta na diminuição desta impureza.

Dentre todas as variáveis preditoras selecionadas para compor o modelo de predição de incêndio, as que apresentaram um melhor resultado nas métricas de avaliação de desempenho do modelo, foram:

- "unidade_numerica"
- "categorias_distancias"
- "tipo_maturacao"
- "categorias_temperatura"
- "umidade_solo_5"
- "dist_urbano"

3.3.8 Experimentos

Nesta seção, serão abordados os experimentos realizados, em que em cada um deles, utilizou-se um pré-processador diferente. Além disso, métodos de transformação dos dados para se chegar no modelo de predição de incêndio final também serão descritos resumidamente. Ademais, serão apresentadas as variáveis selecionadas.

O desenvolvimento dos experimentos iniciou-se com a seleção das variáveis (Subseção 3.3.1). Após selecionadas as primeiras variáveis de interesse, foi realizado o tratamento dos valores ausentes, conforme citado na Subseção 3.3.2.1. Em seguida, a fim de se obter outras variáveis com provável relevância para as predições, foram criados novos atributos a partir dos já existentes (Subseção 3.3.2.2). Posteriormente, foram aplicadas transformações em variáveis que não estavam no formato exigido pelo algoritmo *XGBoost* (Subseção 3.3.3).

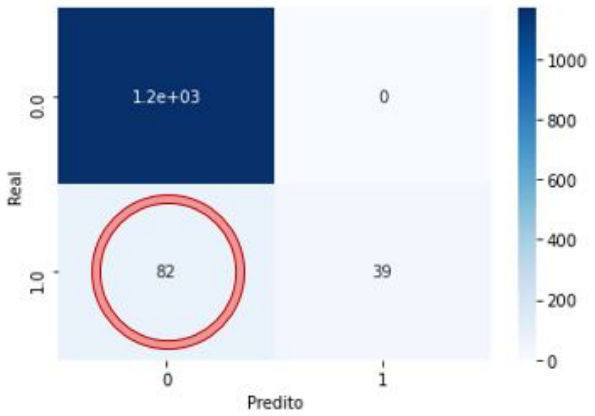
Depois das transformações iniciais, os dados foram divididos entre treino e teste, sendo 70% deles usados para treino e o restante para teste. Posteriormente, a biblioteca *XGBoost* foi utilizada para ser a classificadora.

Após observar o resultado do modelo com as variáveis selecionadas, foram efetuadas novas seleções de variáveis relevantes para o modelo, apresentada na Subseção 3.3.7. Além disso, depois de se ter as variáveis consideradas mais relevantes, etapas com técnicas de reamostragem dos dados foram realizadas a fim de se obter um equilíbrio entre as quantidades de dados de queima e não-queima para que as decisões do modelo não fossem enviesadas.

Seguindo o fluxo de processos da metodologia adotada (Figura 11), após o modelo reconhecer padrões nos dados analisados, ele é avaliado e um ciclo composto por avaliação e aplicação de novas técnicas é iniciado. Dessa forma, o modelo foi constantemente analisado e apenas as variáveis e técnicas que possuíram a maior relevancia para a predição final foram selecionadas.

Os experimentos iniciais foram com as seguintes variáveis: ambiente, corte, maturação, *NDVI*, *HHI*, temperatura média do dia, chuva do dia, além da variável *target* queima. Com essas variáveis de entrada aplicadas ao *XGBoost*, obteve-se a matriz de confusão da Figura 18. Observa-se que a matriz de confusão indica que o modelo de predição acertou todos os casos de não-queima (número 0 para os verdadeiros negativos) porém está com um valor alto para os falsos negativos, situação preocupante pois uma queima traz muito prejuízo para a empresa. Este comportamento de acertar todos todos os valores de apenas uma classe é comumente observado quando se tem quantidades muito diferentes de dados em cada classe. Situações de conjuntos de dados com um grande número de exemplos de uma classe e poucos exemplos da outra classe é conhecida por "classes desbalanceadas" e uma forma de se corrigir isto é através do balanceamento dos dados.

Figura 18 – Matriz de Confusão Inicial

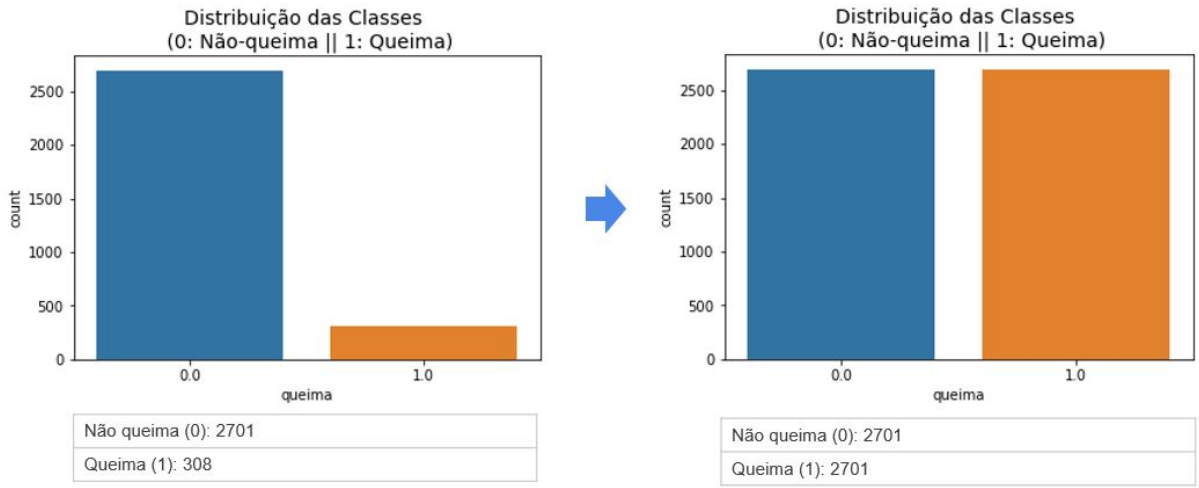


Fonte: A Autora.

Tem-se, na Figura 19, que o número de exemplos relativos à classe queima na variável a ser prevista era apenas de 308 em comparação com 2701 exemplos da classe não-queima.

Com isso, o algoritmo ficou mais especializado em reconhecer casos de não-queima porém não soube identificar adequadamente situações de queima. Após a aplicação da técnica de balanceamento SMOTE, a quantidade de exemplos das classes presentes na variável a ser predita ficaram equivalentes. Este método, SMOTE, é normalmente utilizado quando a classe desejada para se analisar está sub-representada [45].

Figura 19 – Balanceamento dos Dados com método SMOTE



Fonte: A Autora.

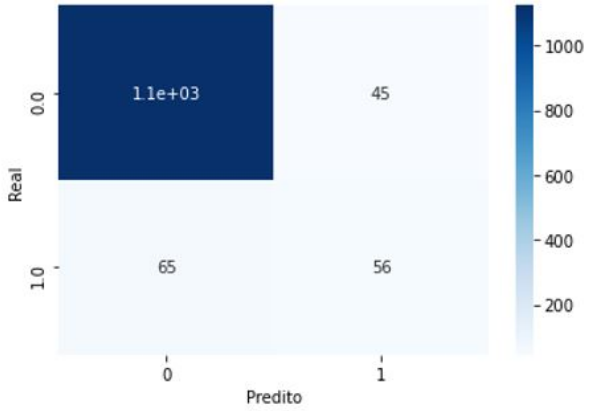
A técnica SMOTE aumenta as instâncias de casos minoritários dentro do conjunto de dados. Como se pode observar na Figura 19, há 2701 instâncias da classe *não-queima* e 308 da classe *queima*. Dessa forma, a classe minoritária é a classe *queima*, pois ela contém a menor quantidade de exemplos. Sendo assim, a técnica SMOTE gera novos exemplos dessa classe, igualando-a à quantidade de exemplos classe majoritária e não alterando a quantidade de exemplos desta classe.

Além do balanceamento dos dados, uma técnica de normalização também foi aplicada. Essa técnica foi aplicada a fim de que as variáveis tivessem seus valores em uma mesma escala, facilitando a comparação entre elas e melhorando a performance do modelo [46]. Este pré-processamento é útil quando se tem parâmetros com escalas muito diferentes pois alguns algoritmos levam em consideração o tamanho do valor, colocando-se um maior peso em uma característica que possui um maior valor. Feita a normalização dos atributos com o pré-processador *MinMaxScaler()*, todos passaram a ter seus valores no intervalo de zero à um, respeitando as diferenças nos intervalos de valores.

Após aplicadas as técnicas de normalização e balanceamento dos dados, houve uma diminuição dos falsos positivos na matriz de confusão, como mostra a Figura 20. Observa-se, também, que os falsos negativos aumentaram. No entanto, tratando-se do negócio da empresa, é muito mais vantajoso ter um modelo que alerte a possibilidade de que haja

queima em um local mesmo não existindo essa chance de queima ao invés de indicar a probabilidade de não-queima para uma região que está com os fatores propícios a iniciar uma queimada. O custo de deslocamento da brigada de incêndio até o local indicado a se ter uma alta probabilidade de queima é muito menor do que quando o canavial pega fogo.

Figura 20 – Matriz de Confusão após Normalização e Balanceamento dos dados



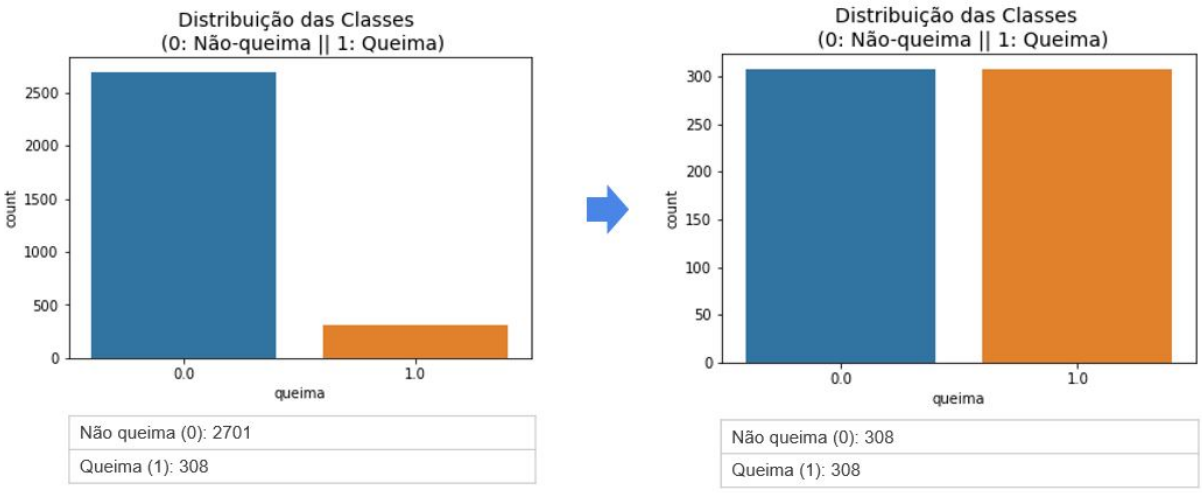
Fonte: A Autora.

Feito isso, outras técnicas de reamostragem foram aplicadas. Além do método de sobreamostragem SMOTE, foram aplicados métodos de subamostragem, como a Subamostragem Aleatória e métodos que mesclam tanto a subamostragem quanto a sobreamostragem.

O método de subamostragem aleatória consiste em diminuir a quantidade de exemplos da classe majoritária deixando-a com a mesma quantidade de exemplos da classe minoritária. Na Figura 21, observa-se que a quantidade de exemplos da classe *não-queima* era de 2701 antes do balanceamento dos dados e, após a aplicação da subamostragem com o pré-processador *RandomUnderSampler()*, este número diminuiu para 308 (quantidade de exemplos da classe minoritária). A matriz resultante desse método (Figura 22) mostra que o campo de falsos positivos teve um aumento na quantidade de valores, não sendo bom para o modelo desejado. Dessa forma, o método SMOTE se mantém como o melhor pré-processamento até o momento.

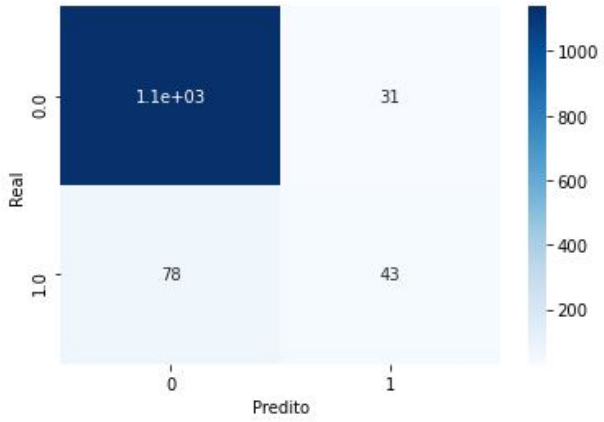
Além dos métodos descritos acima, também foram utilizados em pré-processamentos de dados combinações de métodos. Sendo assim, foi realizado um experimento aplicando-se o método de sobreamostragem aleatória em conjunto com subamostragem aleatória. Neste processo, aumenta-se o número dos casos minoritários até uma determinada quantidade (ainda menor que a quantidade de exemplos da classe majoritária) e, depois, diminui-se a quantidade de exemplos da classe majoriária para a mesma quantidade atual da classe minoritária (Figura 23). Após aplicado este pré-processamento, observa-se que o número

Figura 21 – Balanceamento dos Dados com Subamostragem Aleatória



Fonte: A Autora.

Figura 22 – Matriz de Confusão com método de Subamostragem Aleatória



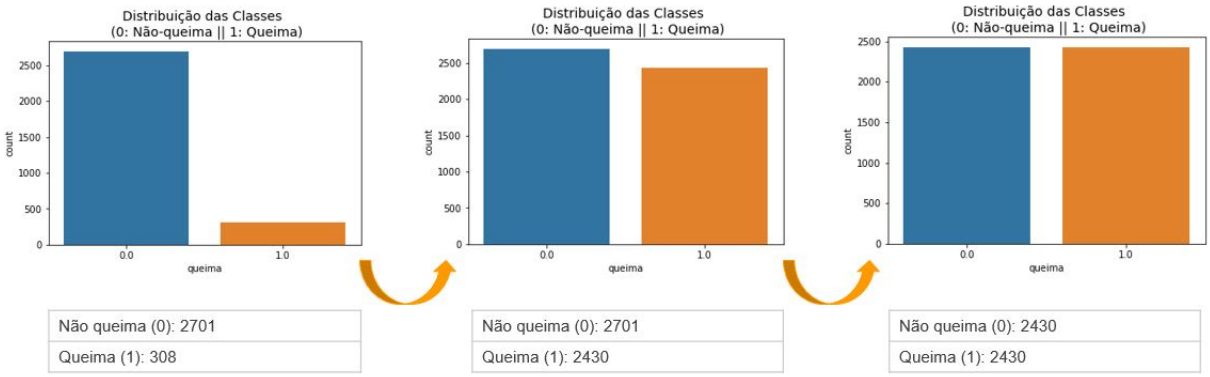
Fonte: A Autora.

de falsos positivos diminuiu em comparação com a matriz da Figura 22. No entanto, o método SMOTE continuou apresentando o melhor resultado.

Outro teste envolvendo combinações de métodos de pré-processamento foi o de sobreamostragem com o método SMOTE e subamostragrem aleatória (Figura 25). O processo é exatamente igual ao descrito anteriormente, com a mesma quantidade de exemplos para cada classe. O resultado na matriz de confusão (Figura 26) também foi bem próximo. Ainda assim, as predições do modelo utilizando-se o pré-processador SMOTE permaneceu sendo as melhores.

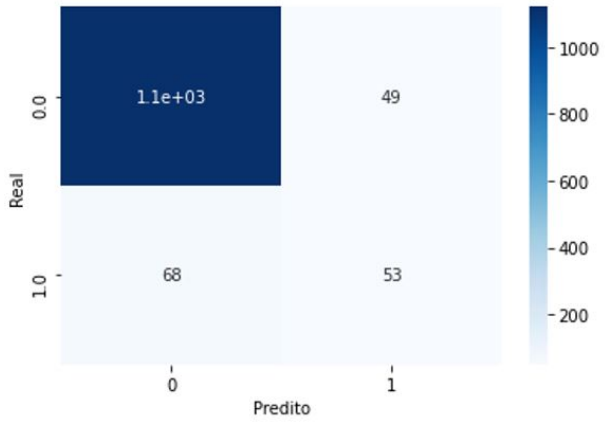
O método SMOTE possui uma variação, o Synthetic Minority Oversampling TEch-

Figura 23 – Balanceamento dos Dados com Sobreamostragem Aleatória e Subamostragem Aleatória



Fonte: A Autora.

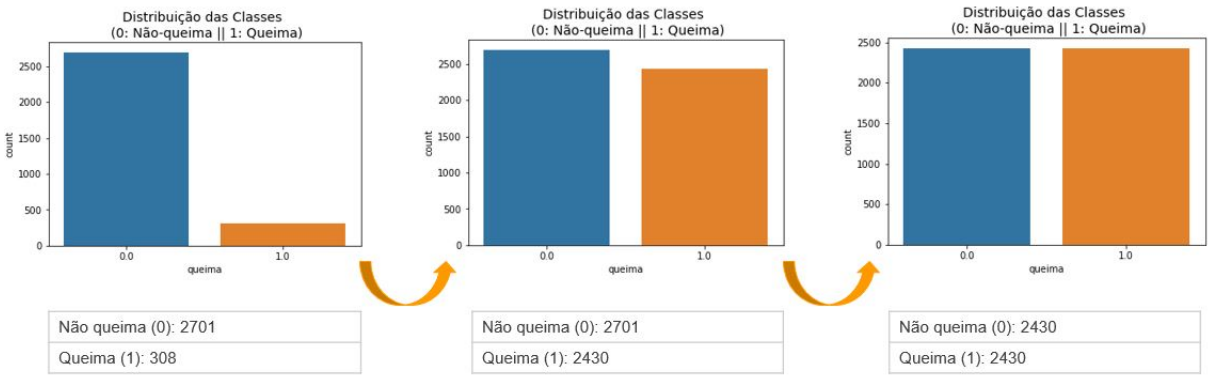
Figura 24 – Matriz de Confusão com Sobreamostragem Aleatória e Subamostragem Aleatória



Fonte: A Autora.

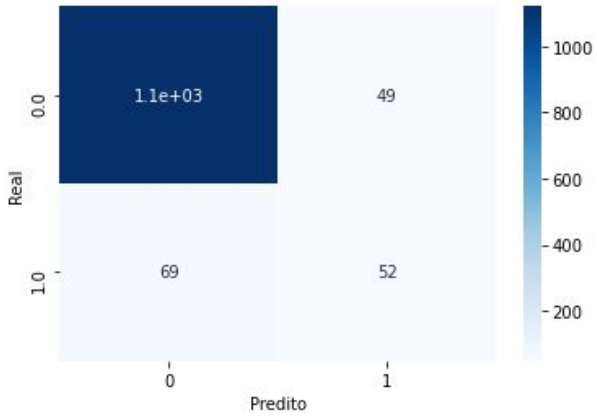
nique - Nominal Continuous (SMOTE-NC), que é próprio para conjunto de dados que possuem variáveis contínuas e categóricas, assim como o conjunto de dados utilizado para o desenvolvimento do modelo deste trabalho. Este método cria uma nova amostra dos dados categóricos ao invés de criar dados sintéticos, evitando-se criar valores que não fazem sentido para o algoritmo. O SMOTE-NC aumenta a quantidade de exemplos da classe minoritária para a mesma quantidade da classe majoritária (Figura 27), assim como o SMOTE. Aplicando-se o SMOTE-NC, o número de falsos positivos diminuiu porém os falsos negativos aumentaram. De qualquer forma, o resultado da Figura 30 ainda é o melhor pelo fato do modelo conseguir identificar corretamente mais casos de queima e isto ser de maior interesse para a empresa.

Figura 25 – Balanceamento dos Dados com método SMOTE e Subamostragem Aleatória



Fonte: A Autora.

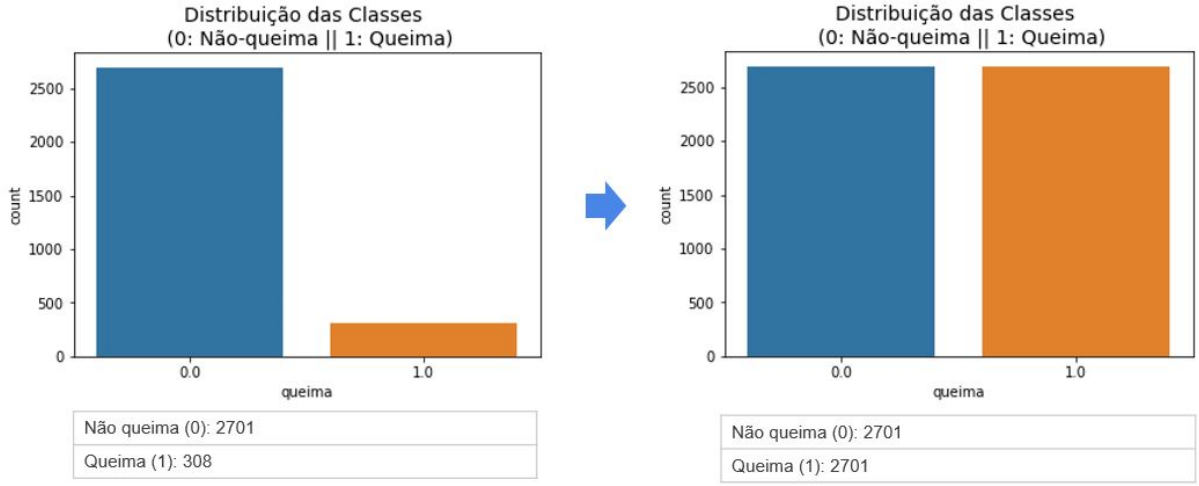
Figura 26 – Matriz de Confusão com método SMOTE e Subamostragem Aleatória



Fonte: A Autora.

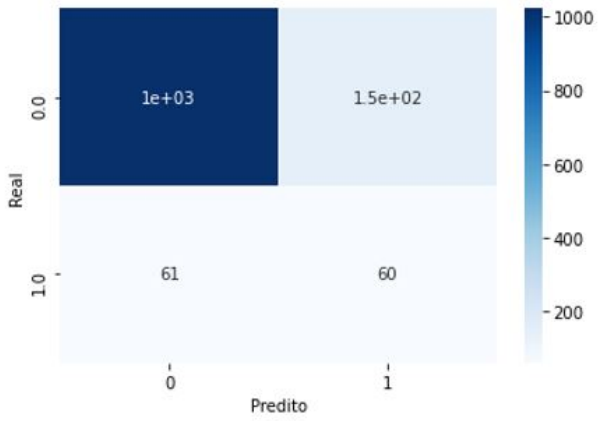
De acordo com os resultados obtidos nas matrizes de confusão, o método SMOTE-NC apresentou o melhor resultado dentre as preferências da empresa e foi escolhido para constituir o pré-processamento dos dados deste trabalho. Para facilitar a visualização e comparação das sensibilidades, observa-se na Tabela 6 os valores resultantes utilizando-se os dados de teste, em que há 121 exemplos da classe “queima” e 1169 da classe “não-queima”.

Figura 27 – Balanceamento dos Dados com método SMOTE-NC



Fonte: A Autora.

Figura 28 – Matriz de Confusão com SMOTE-NC



Fonte: A Autora.

Tabela 6 – Comparações das Sensibilidades

Sensibilidades	
Matriz	Sensibilidade (%)
Figura 18	32,2
Figura 20	46,3
Figura 22	35,5
Figura 24	43,8
Figura 26	42,9
Figura 30	49,6

4 Resultados

Este capítulo consiste em apresentar e explicar o resultado final dos testes realizados na Seção 3.3.8. O resultado final consiste na escolha das variáveis de entrada e no método de pré-processamento escolhido. Também, será exposto os valores finais dos hiperparâmetros otimizados da biblioteca *XGBoost*. A análise dos experimentos foi feita a partir da matriz de confusão e das seguintes métricas: sensibilidade, especificidade e f1-score. Os resultados serão apresentados, primeiramente, mostrando-se as variáveis selecionadas. Depois, a escolha do método de pré-processamento. Após, a seleção dos hiperparâmetros e resultados das métricas de avaliação.

4.1 Modelo de predição final

O modelo de predição final visa retornar a probabilidade de ocorrência de incêndio em áreas do canavial de uma empresa da região noroeste do estado de São Paulo. É um algoritmo desenvolvido utilizando-se a biblioteca *XGBoost* e possui como entradas variáveis relacionadas com as condições climáticas da região, estado ambiental do canavial e georreferenciamento.

As variáveis preditoras que sinalizaram uma maior importância para o resultado final do modelo foram: “unidade_numerica”, “categorias_distancias”, “tipo_maturacao”, “categorias_temperatura”, “umidade_solo_5” e “dist_urbano” (expostas na Tabela 7).

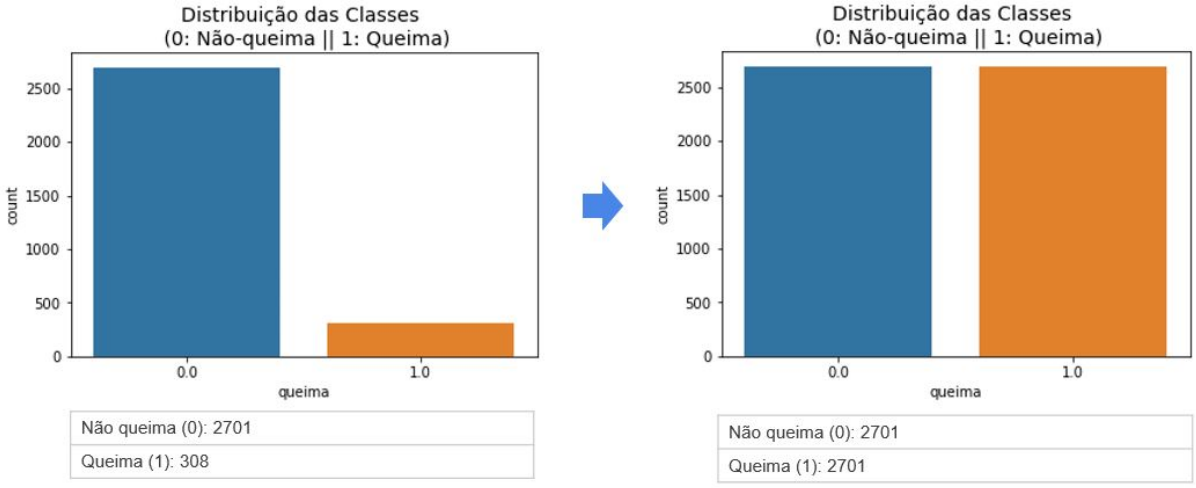
Tabela 7 – Variáveis de Entrada Finais

Tratamento dos Valores Ausentes
Nome da Variável
unidade_numerica
categorias_distancias
tipo_maturacao
categorias_temperatura
umidade_solo_5
dist_urbano
queima

A técnica de reamostragem que apresentou um melhor desempenho para o *XGBoost* foi com o *SMOTENC()*, conforme explicado na Seção 3.3.5. Assim, a quantidade de 308 dados de treinamento da classe “queima”, passou a ser 2701 exemplos (Figura 29).

A métrica que teve maior prioridade nas análises foi a sensibilidade, visto que ela é bastante relacionada com os verdadeiros positivos da matriz de confusão. A sensibilidade indica a taxa de acertos do modelo desenvolvido quanto à predição de ter queimadas.

Figura 29 – Balanceamento dos Dados com método SMOTE-NC



Fonte: A Autora.

A proporção dos dados entre treino e teste foi de 70% e 30%, respectivamente. E a biblioteca utilizada desde o início do desenvolvimento do algoritmo, foi o *XGBoost*.

Para refinar o modelo de aprendizado de máquina *XGBoost*, o módulo *GridSearchCV* da biblioteca *Scikit Learn* foi aplicado. Dessa forma, os valores dos hiper-parâmetros selecionados pelo *GridSearchCV* está na Tabela 8.

Tabela 8 – Otimização de hiperparâmetros com *GridSearchCV*

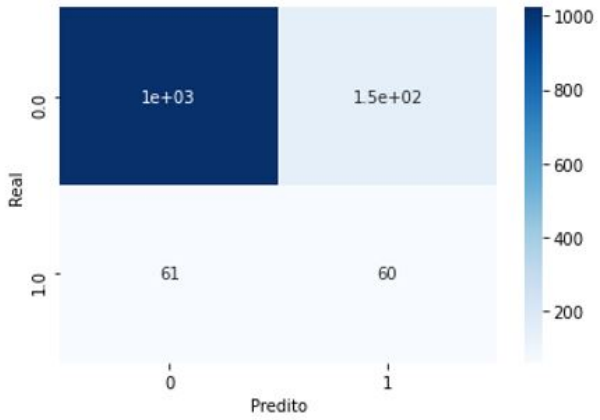
Hiperparâmetros - *XGBoost*	
Hiperparâmetro	Valor
objective	"binary : logistic"
n_estimators	500
learning_rate	0.3
max_depth	7
colsample_bytree	0.8
scale_pos_weight	1
reg_lambda	1
reg_alpha	0

Durante o desenvolvimento do algoritmo de predição, os experimentos foram analisados de acordo com as matrizes de confusão e métricas de avaliação de desempenho do modelo (Subseção 3.3.8). Na Figura 30 é apresentada a matriz de confusão final. Nesta imagem, pode-se observar que o modelo é capaz de acertar 50% dos casos referentes à classe "queima" e 87% em relação à classe "não-queima".

Ainda que o resultado obtido seja bom, levando-se em consideração a complexidade do problema e do algoritmo, o modelo de predição de incêndio pode ser melhorado. Como foi

comentado anteriormente, é de grande importância para as empresas do setor sucroenergético evitar a ocorrência de incêndios em seus canaviais. Dessa forma, identificando-se previamente áreas com grandes chances de pegar fogo, equipes de brigadas de incêndio podem se preparar e se alocar próximas dessas regiões, diminuindo-se o tempo de deslocamento caso houver focos de queimadas. Com isso, para que o modelo fique ainda melhor, é necessário que o valor a métrica sensibilidade possua uma porcentagem maior. No entanto, como um sistema de apoio para as áreas de gerenciamento de riscos de incêndio, o modelo desenvolvido já pode ser utilizado como suporte.

Figura 30 – Matriz de Confusão Final



Fonte: A Autora.

5 Conclusões

O presente trabalho teve como objetivo desenvolver um algoritmo de predição que indicasse a probabilidade da ocorrência de incêndio. Para isso, foi proposto pela equipe de análise de dados da empresa em questão, a utilização de aprendizado de máquina com a biblioteca *XGBoost*. O resultado esperado é retratar como é a criação de um algoritmo capaz de identificar se uma área do canavial possui altas ou baixas chances de pegar fogo de acordo com as condições meteorológicas, do solo, além da localização dessas áreas.

Este trabalho descreveu o projeto e a implementação de um algoritmo de predição de incêndio em um canavial da região noroeste do estado de São Paulo. Para isso, foi utilizado o ambiente *Jupyter Notebook* para o desenvolvimento do algoritmo. Em relação aos dados, todas as informações foram coletadas do gerenciador de banco de dados *PostgreSQL*. Com as pesquisas realizadas, foram estudadas técnicas de seleção de variáveis, pré-processamentos de dados e métricas de avaliação de desempenho de modelos de aprendizado de máquina. Além disso, também foi estudado sobre otimização de modelos através de hiperparâmetros.

Dos resultados obtidos e análises feitas, é possível observar que o modelo apresentou um bom resultado e satisfaz as expectativas dos interessados pelo projeto na empresa. Mesmo com uma área não tão extensa, o modelo consegue identificar padrões nos dados e retornar métricas que indicam um bom desempenho. No âmbito financeiro, o modelo de predição serve de suporte às equipes de gerenciamento de risco contra incêndios, indicando quais áreas possuem maiores probabilidades de iniciar um incêndio. Assim, é possível se preparar com antecedencia, evitando-se que focos de incêndio se iniciem ou se espalhem. Dessa forma, a empresa pode cuidar de seu canavial de forma preditiva, evitando-se prejuízos com queimadas.

Para aprimoramento dos resultados finais deste trabalho, propõe-se aplicar outras técnicas de pré-processamento dos dados assim como definir novos valores de *Threshold*, de forma a se ajustar aos interesses da empresa. Além disso, pode-se testar outros algoritmos de *Machine Learning* para comparar as métricas de desempenho.

Referências Bibliográficas

1 REIS, A. *Qual é o custo de incêndios no canavial e como prevenir e controlar*. <https://jornalcana.com.br/qual-e-o-custo-de-incendios-no-canavial-e-como-prevenir-e-controlar/>.

2 REIS, A. *Fogo no canavial: usinas se preparam para evitar prejuízo com incêndios*. <https://jornalcana.com.br/fogo-no-canavial-usinas-se-preparam-para-evitar-prejuizo-com-incendios/>.

3 UNICA. *Incêndios colocam canaviais em risco em São Paulo*. <https://unica.com.br/noticias/incendios-colocam-canaviais-em-risco-em-sao-paulo/#:~:text=O%20tempo%20seco%2C%20associado%20%C3%A0s,o%20risco%20de%20inc%C3%AAndios%20ambientais.&text=Atualmente%2C%20aproximadamente%2099%2C6%25,queima%20como%20m%C3%A9todo%20pr%C3%A9%2Dcolheita>.

4 MEDINA, J. *Cana-de-açúcar: a Cultura que Potencializou o Brasil!* <https://agropos.com.br/cana-de-acucar/>.

5 RONQUIM, C. C. Queimada na colheita da cana-de-açúcar: impactos ambientais, sociais e econômicos. 2010.

6 MACEDO, I. d. C. Sugar cane's energy: twelve studies on brazilian sugar cane agribusiness and its sustainability. São Paulo: União da Agroindústria Canavieira do Estado de São Paulo, 2007.

7 IPCC. Climate change 1994: radiative forcing of climate change and an evaluation of the IPCC IS92 emission scenarios. Intergovernmental Panel on Climate Change, p. 339, 1995.

8 NAQA RUIJIANG LI, M. J. M. I. E. Machine learning in radiation oncology. Springer, Cham, v. 1, 2015. ISSN 0020-0255.

9 WITTEN EIBE FRANK, M. a. H. I. H. *Data Mining: Practical Machine Learning Tools and Techniques*. [S.l.: s.n.], 2011. (Morgan Kaufmann Publishers). ISBN 0123748569.

10 MITCHELL, T. *Machine Learning*. [S.l.]: McGraw-Hill, 1997. (McGraw-Hill International Editions). ISBN 9780071154673.

11 SUPERVISED Learning. [S.l.]: IBM Cloud Education. <https://www.ibm.com/cloud/learn/supervised-learning>.

12 FUCHS, K. *Machine Learning: Classification Models*. <https://medium.com/fuzz/machine-learning-classification-models-3040f71e2529#:~:text=Classification%20models%20include%20logistic%20regression,level%20at%20some%20of%20these>.

13 EWERT, N. *Algoritmos de Classificação: uma introdução*. <https://awari.com.br/algoritmos-de-classificacao-uma-introducao/#:~:text=Muito%20conhecido%20pela%20sigla%20em,s%C3%A3o%20classificados%20de%20uma%20forma>.

14 DEVELOPERS xgboost. *XGBoost Documentation*. <https://xgboost.readthedocs.io/en/latest/>.

15 UEL. *ENSINO SUPERIOR :: Álgebra Linear: Metodo dos mínimos quadrados*. <http://www.uel.br/projetos/matessencial/superior/alinear/mmq.htm>.

16 MADSEN, T. *Locally Weighted Logistic Regression*. <http://tobiasmadsen.com/2015/03/26/locally-weighted-logistic-regression/>.

17 USP. *REGRESSÃO LOGÍSTICA*. <https://edisciplinas.usp.br/pluginfile.php/3769787/mod_resource/content/1/09_RegressaoLogistica.pdf>.

18 FLUENTE, C. *Scikit-Learn – Support Vector Machine ou máquina de vetores de suporte*. <https://www.codigofluente.com.br/aula-08-scikit-learn-maquina-de-vetores-de-suporte/>.

19 INTRODUCTION to Boosted Trees. <https://xgboost.readthedocs.io/en/latest/tutorials/model.html>.

20 YIU, T. *Understanding Random Forest*. <https://towardsdatascience.com/understanding-random-forest-58381e0602d2>.

21 HARRISON, O. *Machine Learning Basics with the K-Nearest Neighbors Algorithm*. <https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761>.

22 GOKTE, S. A. *Most Popular Distance Metrics Used in KNN and When to Use Them*. <https://www.kdnuggets.com/2020/11/most-popular-distance-metrics-knn.html>.

23 BECKER, L. *Algoritmo de Classificação Naive Bayes*. <https://www.organicadigital.com/blog/algoritmo-de-classificacao-naive-bayes/>.

24 SCIKIT-LEARN. *Classificação com Naive Bayes*. <https://www.datageeks.com.br/naive-bayes/>.

25 CHEN, C. G. T. XGBoost: A Scalable Tree Boosting System. SIGKDD.

26 BROWNLEE, J. *A Gentle Introduction to the Gradient Boosting Algorithm for Machine Learning*. <https://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-machine-learning/>.

27 BROWNLEE, J. *Extreme Gradient Boosting (XGBoost) Ensemble in Python*. <https://machinelearningmastery.com/extreme-gradient-boosting-ensemble-in-python>.

28 BROWNLEE, J. *A Gentle Introduction to Threshold-Moving for Imbalanced Classification*. <https://machinelearningmastery.com/threshold-moving-for-imbalanced-classification/>.

29 BHANDARI, A. *AUC-ROC Curve in Machine Learning Clearly Explained*. <https://www.analyticsvidhya.com/blog/2020/06/auc-roc-curve-machine-learning/>.

30 BROWNLEE, J. *How to Use ROC Curves and Precision-Recall Curves for Classification in Python*. <https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-classification-in-python/>.

31 CASTRO, A. P. B. Cristiano Leite de. Aprendizado supervisionado com conjuntos de dados desbalanceados. *Sba: Controle Automação Sociedade Brasileira de Automatica*, scielo, v. 22, p. 441 – 466, 10 2011. ISSN 0103-1759.

32 FAWCETT, F. P. T. Adaptive fraud detection. In: *Data Mining and Knowledge Discovery*. [S.l.: s.n.], 1997. p. 291–316.

33 OS 5 Algoritmos de Amostragem Que Todo Cientista de Dados Precisa Saber. <https://www.semantix.com.br/os-5-algoritmos-de-amostragem-que-todo-cientista-de-dados-precisa-saber/>.

34 WU, J. et al. Hyperparameter optimization for machine learning models based on bayesian optimizationb. *Journal of Electronic Science and Technology*, v. 17, n. 1, p. 26–40, 2019. ISSN 1674-862X.

35 FAYYAD GREGORY PIATETSKY-SHAPIRO, P. S. U. From data mining to knowledge discovery in databases. AI Magazine, v. 17, n. 3, p. 37–54, 1996.

36 TECH, D. *A biblioteca scikit-learn – Pyhton para machine learning*. <https://didatica.tech/a-biblioteca-scikit-learn-pyhton-para-machine-learning/>.

37 BROWNLEE, J. *Train-Test Split for Evaluating Machine Learning Algorithms*. <https://machinelearningmastery.com/train-test-split-for-evaluating-machine-learning-algorithms/>.

38 ELREEDY, D.; ATIYA, A. F. A comprehensive analysis of synthetic minority oversampling technique (smote) for handling class imbalance. *Information Sciences*, v. 505, p. 32–64, 2019. ISSN 0020-0255.

39 CHAWLA KEVIN W. BOWYER, L. O. H. W. P. K. N. V. SMOTE: Synthetic Minority Over-sampling Technique. *Artificial Intelligence Research*, arXiv.org, v. 16, p. 321–357, 2002.

40 BROWNLEE, J. *Undersampling Algorithms for Imbalanced Classification*. <https://machinelearningmastery.com/undersampling-algorithms-for-imbalanced-classification/>.

41 BROWNLEE, J. *How to Configure k-Fold Cross-Validation*. <https://machinelearningmastery.com/how-to-configure-k-fold-cross-validation/>.

42 BEHNAMIAN KOREEN MILLARD, S. N. B. L. W. M. R. J. P. A. A Systematic Approach for Variable Selection With Random Forests: Achieving Stable Variable Importance Values. *IEEE GEOSCIENCE AND REMOTE SENSING LETTERS*, IEEE, v. 14, n. 11, 2017.

43 RADECIC, D. *3 Essential Ways to Calculate Feature Importance in Python*. <https://towardsdatascience.com/3-essential-ways-to-calculate-feature-importance-in-python-2f9149592155>.

44 PUC-RIO. *Árvore de Decisão*. <https://www.maxwell.vrac.puc-rio.br/7587/7587_4.PDF>.

45 SMOTE. <https://docs.microsoft.com/pt-br/azure/machine-learning/algorithm-module-reference/smote>.

46 FEATURE Normalization. <https://python-data-science.readthedocs.io/en/latest/normalisation.html>.