Trabalho Prático - Machine Learning

Bruno Rodrigues Faria
Pontifícia Universidade Católica De Minas Gerais
Belo Horizonte, Minas Gerais, Brasil

Guilherme Dantas Caldeira Fagundes
Pontifícia Universidade Católica De Minas Gerais
Belo Horizonte, Minas Gerais, Brasil

Laura Iara Silva Santos Xavier
Pontifícia Universidade Católica De Minas Gerais
Belo Horizonte, Minas Gerais, Brasil

Maria Luisa Tomich Raso
Pontifícia Universidade Católica De Minas Gerais
Belo Horizonte, Minas Gerais, Brasil

Matheus Rangel de Figueiredo
Pontifícia Universidade Católica De Minas Gerais
Belo Horizonte, Minas Gerais, Brasil

ABSTRACT

Este artigo irá tratar sobre a escolha do Dataset e suas mais características, além de mostrar resultados de uma aplicação de um algoritmo de Machine Learning. Que foi aplicado sem as devidas etapas de pré-processamento antes. Trazendo assim os resultados a serem analisados pelos estudantes e tratados na próxima etapa de desenvolvimento do trabalho.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning approaches**.

KEYWORDS

dataset, atributos, supervisionado, machine learning, detecção de incêndio

ACM Reference Format:
Bruno Rodrigues Faria, Guilherme Dantas Caldeira Fagundes, Laura Iara Silva Santos Xavier, Maria Luisa Tomich Raso, and Matheus Rangel de Figueiredo. 2018. Trabalho Prático - Machine Learning. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUÇÃO

O objetivo deste trabalho é analisar o dataset [1] retirado do Kaggle, e mostrar suas características através da aplicação de um algoritmo de aprendizado de máquina.

PUC MINAS , September 17–10, 2022, Belo Horizonte, MG

ACM ISBN 978-1-4503-9999-9/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

O arquivo apresenta cerca de 60.000 instâncias com dados de diversos ambientes passíveis de incêndio, viabilizando um diagnóstico sobre a a existência ou não de fumaça no local. Os resultados obtidos, podem ser utilizados no funcionamento de detectores de fumaça que captam essas variáveis.

2 DESCRIÇÃO DA BASE

O dataset é composto por dados numéricos distribuídos em 13 colunas, dentre elas, o atributo binário de classificação "Fire Alarm" reperesentando a presença de fumaça no local, em função dos outros 12 valores.

Cada um desses atributos possui um limite máximo e mínimo, e uma descrição da sua representação dentro do contexto:

- Temperature (°C): representa a temperatura do ar, possui como mínimo -22,01°C e máximo como 59,93°C.
- Humidity (%): representa a porcentagem de humidade do ar, possui como mínimo 10,74% e 75,2% como máximo.
- TVOC (ppb): valor numérico em partes por bilhão que representa o total de compostos orgânicos voláteis. O valor mínimo é 0 e o valor máximo é 60.000.
- eCO2 (ppm): valor numérico em partes por milhão que representa a concentração total de dióxido de carbono no ambiente. O mínimo é 400 e o máximo é 60.000.
- Raw H2 (valor numérico bruto): valor que representa a quantidade de moléculas de H2 no ambiente. O mínimo é 10.668 e o máximo é 13.803.
- Raw Ethanol (valor numérico bruto): valor que representa a quantidade de moléculas de gás etanol no ambiente. O mínimo é de 15.317 e o máximo de 21.410
- Pressure (hPA): valor em hectopascal (100 x pascal) do ambiente. Menor valor é de 930,852 e o maior de 939,861
- PM1.0 (valor numérico bruto): valor que representa a quantidade de partículas com menos de 1µm. Menor valor é 0 e maior é 14.333,69

- PM2.5 (valor numérico bruto): valor que representa a quantidade de partículas com menos de 2.5µm. Menor valor é 0 e maior é 45.432,26
- NC0.5 (valor numérico): valor que representa a concentração de partículas com menos de 0.5µm. Menor valor é 0 e maior é 61.482,03
- NC1.0 (valor numérico): valor que representa a concentração de partículas com menos de 1µm. Menor valor é 0 e maior é 51.914,68
- NC2.5 (valor numérico): valor que representa a concentração de partículas com menos de 2.5µm. Menor valor é 0 e maior é 30.026,438

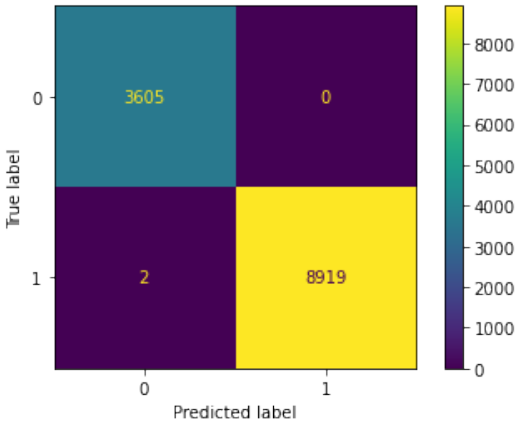
3 CÓDIGO FONTE

O código fonte pode ser encontrado no repositório do projeto no GitHub, onde está separado por etapas realizadas ou também em um JupyterNotebook completo com todas as etapas concatenadas.

Resultados e Métricas

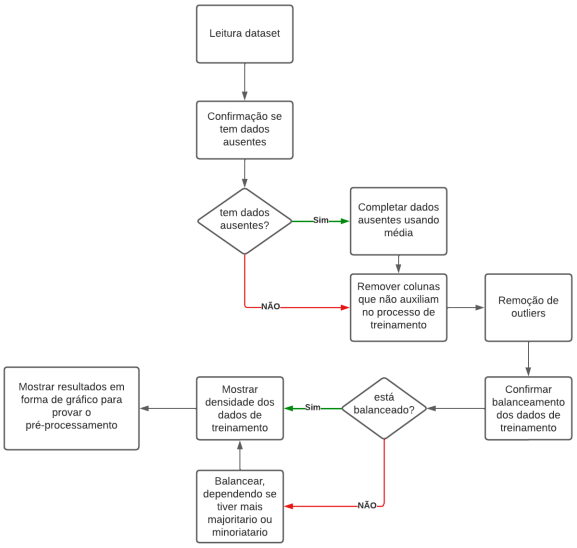
Após a finalização do algoritmo, estes foram os resultados encontrados. A matriz de confusão indicou um ótimo desempenho, ja que apenas 2 de 12.526 instâncias foram classificadas incorretamente. Portanto, as taxas VP e VN foram altíssimas.

As demais métricas também apresentaram resultados acima do esperado. A Precisão foi de 0,99 enquanto o recall, acurácia e F-measure foram 1.



Implementação do Primeiro Tratamento de Dados

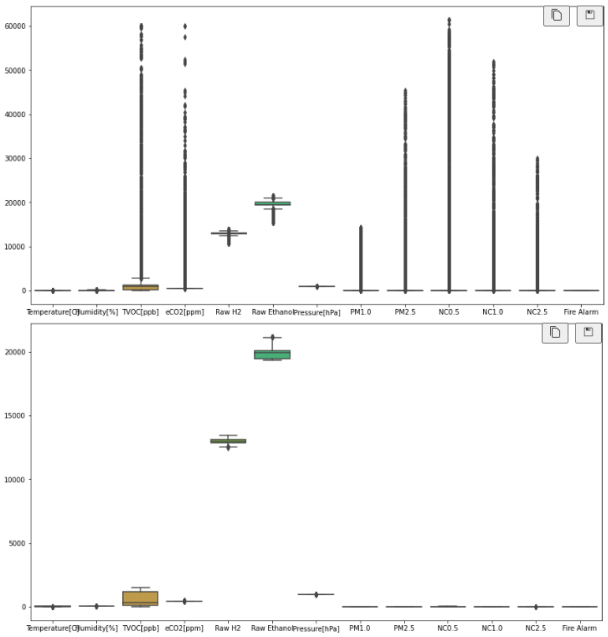
Nessa etapa, foram feitos diversos tipos de tratamento como verificaçào de dados nulos, remoção de outliers, balanceamento de dados e análise da estrutura interna e variação dos dados da base através do algoritmo PCA. O passo a passo realizado para o pré-processamento da base pode ser exemplificado pelo seguinte workflow:



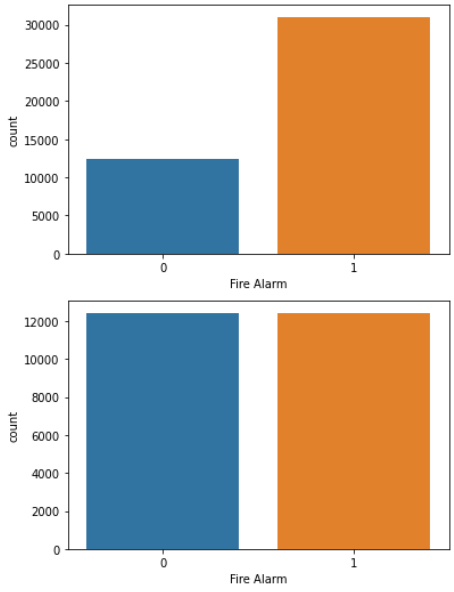
Após a análise da base durante esse primeiro tratamento, foi percebido que não há nenhum dado nulo e que os atributos presentes são todos númericos, não sendo necessária a aplicação de um algoritmo de transformação de dados. Assim, nenhum tratamento nesse sentido foi implementado.

```
#   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Unnamed: 0      62630 non-null  int64
 1   UTC             62630 non-null  int64
 2   Temperature[C]  62630 non-null  float64
 3   Humidity[%]     62630 non-null  float64
 4   TVOC[ppb]       62630 non-null  int64
 5   eCO2[ppm]       62630 non-null  int64
 6   Raw H2          62630 non-null  int64
 7   Raw Ethanol     62630 non-null  int64
 8   Pressure[hPa]   62630 non-null  float64
 9   PM1.0           62630 non-null  float64
 10  PM2.5           62630 non-null  float64
 11  NC0.5           62630 non-null  float64
 12  NC1.0           62630 non-null  float64
 13  NC2.5           62630 non-null  float64
 14  CNT             62630 non-null  int64
 15  Fire Alarm      62630 non-null  int64
dtypes: float64(8), int64(8)
```

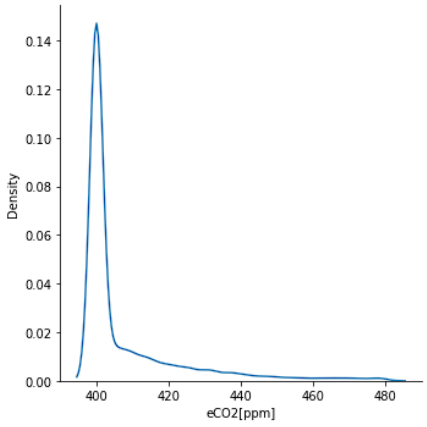
A partir disso, percebeu-se uma grande quantidade de outliers na base e então,foi feito o tratamento desses dados removendo-os, como é mostrado nox bloxplots a seguir.

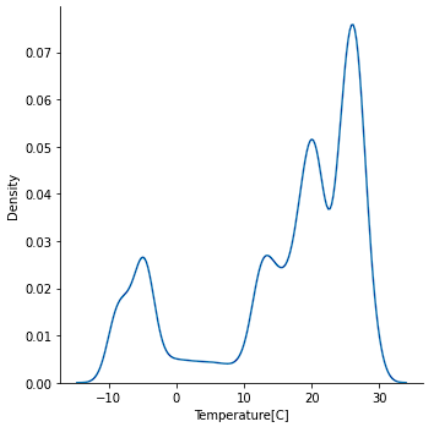


Em seguida, verificou-se que a base é desbalanceada, possuindo muito mais valores verdadeiros do que falsos no atributo de classificação. Portanto, para que um balanceamento adequado fosse alcançado, foi utilizada a técnica de undersampling.



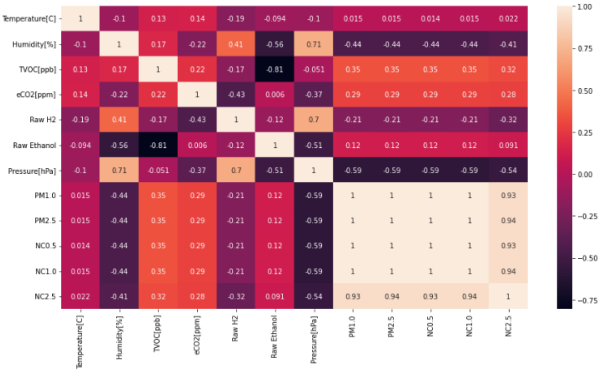
Além disso, uma breve visualização da distribuição dos dados foi feita para maior compreensão do estado do dataset. Pode-se observar que a distribuição varia dependendo da natureza do dado dos atributos de teste, seguem alguns exemplos (não foi possível adicionar todos os gráficos, já que a tabela possui muitas colunas, mas eles podem ser analisados no arquivo JupyterNotebook):





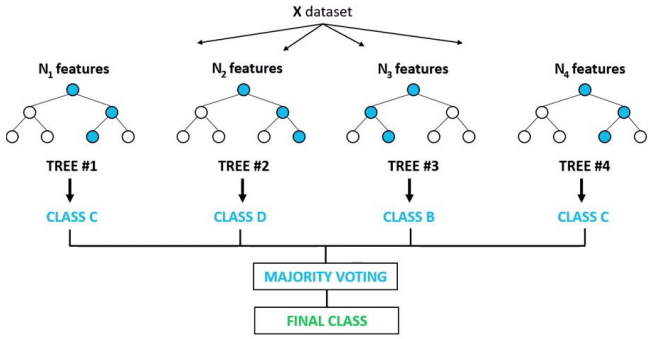
Por fim, com o intuito de analisar a estrutura interna e a variação dos dados, foi aplicado o algoritmo PCA (principal component analisys). Esse algoritmo tem o foco em encontrar os principais componentes da base de dados, traçando correlações entre os atributos. Ele é útil em casos onde a base possui diversas colunas, ignorando as de menor significância e, por consequência, reduzindo a dimensão do dataset.

Por se fundamentar em uma combinação linear de diversos atributos, pode ser difícil para humanos extrair valores de maneira simples do resultado gerado. Entretanto, a aplicação desse conceito matemático facilita o processamento de outros algoritmos de machine learning.



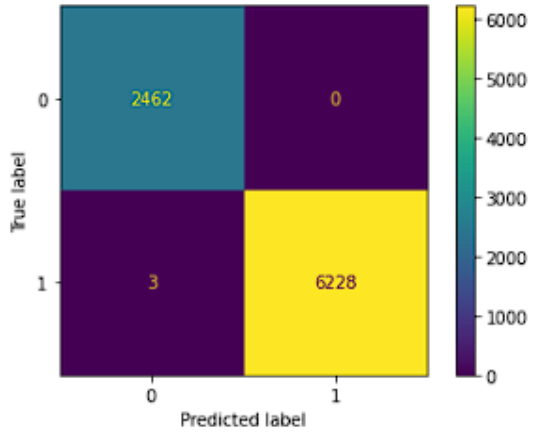
4 IMPLEMENTAÇÃO RANDOM FOREST

A aplicação do random forest tem como intuito, aumentar o número de árvores de decisão para ver se a precisão da Inteligência Artifical aumenta, pois segundo os estudos, quanto mais árvores de decisão tivermos, mais preciso fica o resultado. Então, Random forest é um algoritmo de aprendizagem de máquina que constrói um conjunto de classificadores de árvore de decisão e os combina de forma a melhorar a precisão geral do modelo. Ele funciona treinando vários classificadores de árvore em subconjuntos aleatórios do conjunto de dados e usando a média das previsões de cada árvore para fazer previsões finais, caso a previsão seja númerica.



Resultados e Métricas

Após a aplicação do algoritmo podemos perceber que a Precisão foi de 0.999, onde nosso algoritmo junto com as etapas de pré processamento está prevendo com muita qualidade os resultados, sendo possível melhorar ainda mais com outras etapas de pré-processamento e outros algoritmos mais eficientes de classificação. Para que assim, possa chegar o mais próximo de 1 a Precisão, pode-se observar a matriz de confusão dos testes realizados na imagem abaixo.



Foram efetuados testes com 100 a 500 árvores, porém o algoritmo manteve a mesma precisão, fazendo assim chegarmos a conclusão que mesmo com o número de árvores aumentando não iria modificar nada, porém ao modificar o parametro max features, o qual consiste em mudar a quantidade de features que o algoritmo irá considerar para o cálculo do índice de Gini, que pode ser configurado para alguns valores, sendo o valor padrão de None, ou seja, que o próprio algoritmo irá escolher o melhor valor. Com isso, após testes foi determinado o número 3, pois foi a quantidade de features que manteve a melhor Precisão para o algoritmo Random Forest.

REFERENCES

[1] Stefan Blattmann. 2019. Smoke Detection Dataset. (2019). https://www.kaggle.com/datasets/deepcontractor/smoke-detection-dataset