

**PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS**  
**NÚCLEO DE EDUCAÇÃO A DISTÂNCIA**  
**Pós-graduação *Lato Sensu* em Ciência de Dados e Big Data**

**Talita Santos Andrade**

**ANÁLISE E APLICAÇÃO DE MODELOS DE MACHINE LEARNING PARA A  
CLASSIFICAÇÃO DE OCORRÊNCIAS AERONÁUTICAS**

Belo Horizonte

2023

**Talita Santos Andrade**

**ANÁLISE E APLICAÇÃO DE MODELOS DE MACHINE LEARNING PARA A  
CLASSIFICAÇÃO DE OCORRÊNCIAS AERONÁUTICAS**

Trabalho de Conclusão de Curso apresentado  
ao Curso de Especialização em Ciência de  
Dados e Big Data como requisito parcial à  
obtenção do título de especialista.

Belo Horizonte

2023

## SUMÁRIO

<b>1. Introdução.....</b>	<b>4</b>
<b>1.1. Contextualização .....</b>	<b>4</b>
<b>1.2. O problema proposto .....</b>	<b>5</b>
<b>2. Coleta de Dados .....</b>	<b>7</b>
<b>3. Processamento/Tratamento de Dados .....</b>	<b>12</b>
<b>4. Análise e Exploração dos Dados .....</b>	<b>16</b>
<b>5. Criação de Modelos de Machine Learning .....</b>	<b>22</b>
<b>5.1. Métricas de Avaliação dos Modelos .....</b>	<b>24</b>
<b>5.2. Aplicação dos Modelos.....</b>	<b>25</b>
<b>5.2. Utilização de Hiperparâmetros para melhoria de performance dos Modelos .....</b>	<b>28</b>
<b>6. Interpretação dos Resultados .....</b>	<b>32</b>
<b>7. Apresentação dos Resultados .....</b>	<b>36</b>
<b>8. Links.....</b>	<b>37</b>
<b>REFERÊNCIAS.....</b>	<b>38</b>

## 1. Introdução

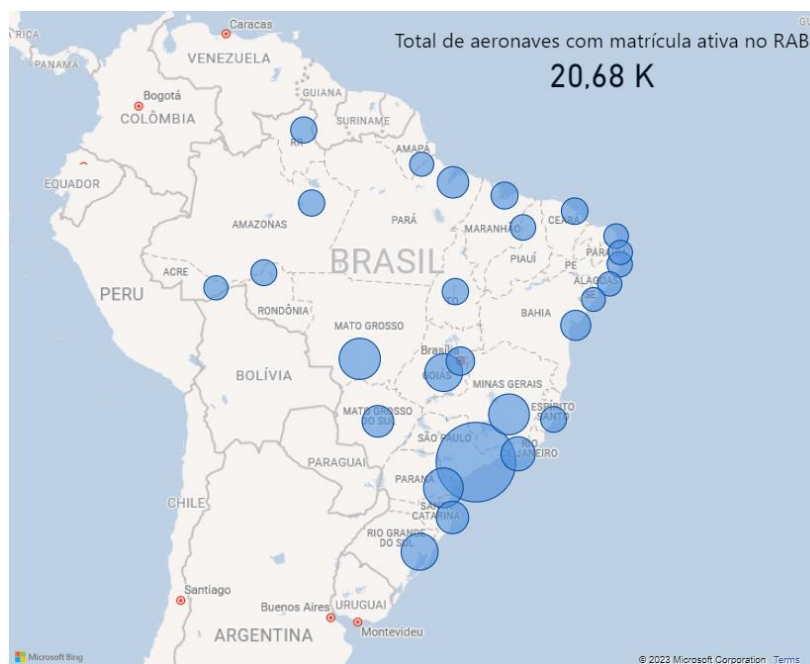
### 1.1. Contextualização

Por se tratar de um país com dimensões continentais, o Brasil possui um dos mais fortes sistemas de transporte aéreo do mundo, sendo responsável pelo transporte de milhões de passageiros e toneladas de carga anualmente, em 2019, o Brasil registrou 119,4 milhões de passageiros transportados por via aérea, segundo dados da ANAC.

O transporte aéreo é de suma importância para o desenvolvimento econômico de um país, permitindo o deslocamento de pessoas e bens, impulsionando atividades comerciais e turísticas. Devido à extensão do país e as dificuldades enfrentadas no transporte terrestre ou marítimo, o transporte aéreo por vezes é o único meio de acesso a determinadas localidades.

Segundo dados divulgados pela Agência Nacional de Aviação Civil (ANAC), até abril de 2023, existem cerca de 20.680 aeronaves registradas no Registro Aeronáutico Brasileiro (RAB), delas 15.267 são aeronaves em situação aeronavegável.

**Figura 1: Total de aeronaves com matrícula ativa no RAB**

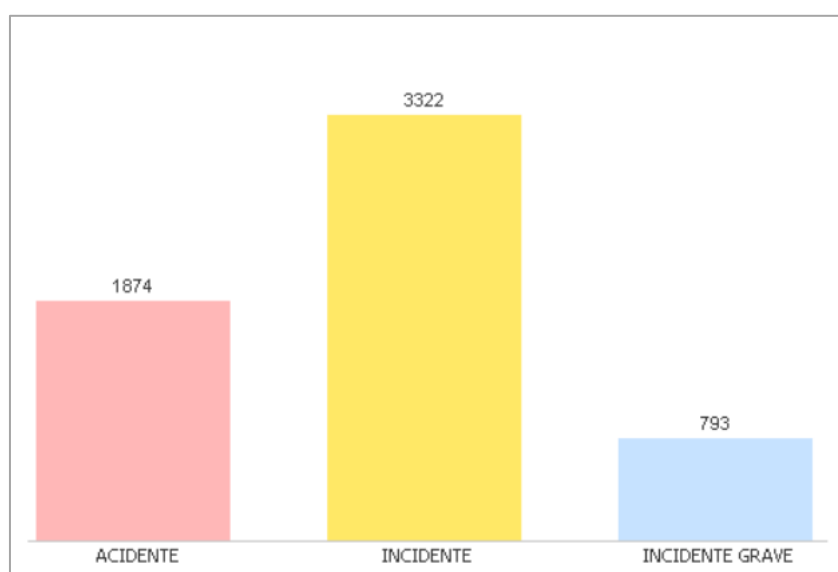


Por ser amplamente utilizado e pela importância do transporte aéreo no âmbito nacional, o Centro de Investigação e Prevenção de Acidentes Aeronáuticos (CENIPA)

foi criado em 1971 com o objetivo de investigar e promover a "prevenção de acidentes aeronáuticos", em concordância com normas internacionais.

De acordo com os dados com dados do Centro de Investigação e Prevenção de Acidentes Aeronáuticos, entre 2012 e abril de 2023, foram registradas 5.984 ocorrências aeronáuticas na aviação civil brasileira, incluindo acidentes e incidentes. Dessas ocorrências, 1.874 foram classificadas como "acidentes", 793 como "incidentes graves" e 3.322 como "incidentes".

**Figura 2: Ocorrências aeronáuticas por categoria**



Os dados do CENIPA são referentes a eventos ocorridos em território brasileiro envolvendo aeronaves civis registradas no Brasil ou no exterior, além de aeronaves militares em operação conjunta com a aviação civil.

Cabe ressaltar que o CENIPA define incidente como um evento não intencional que, embora não tenha resultado em danos às pessoas ou à aeronave, tem potencial para tal. Já o termo acidente é usado para eventos que resultam em morte, lesão grave ou destruição significativa da aeronave.

## **1.2. O problema proposto**

Levando em consideração toda a relevância social e econômica do sistema aeronáutico do Brasil, esse trabalho busca responder a seguinte questão:

*De acordo com as informações registradas pelo CENIPA na base de dados de ocorrências aeronáuticas brasileiras nos anos de 2010 a 2022, qual é a classificação de cada ocorrência registrada.*

Para direcionar o entendimento do problema e solução a ser proposta foi utilizada a técnica do 5Ws, que consiste em responder as seguintes perguntas:

**Why?** A quantidade de ocorrências registradas e a importância da aviação no cenário brasileiro faz com que a análise dos eventos ocorridos e melhor entendimento deles seja importante.

**Who?** Informações registradas pelo CENIPA em sua base de dados de acidentes aeronáuticos.

**What?** Gerar um algoritmo que seja capaz de classificar corretamente uma ocorrência como “indecente”, “incidente grave” e “acidente”.

**Where?** Todos os dados apresentados na pesquisa tratam de eventos ocorridos em território brasileiro envolvendo aeronaves civis registradas no Brasil ou no exterior.

**When?** O trabalho compreende a análise de dados de 2010 à 2022.

O CENIPA define Acidente Aeronáutico como:

(...) toda ocorrência relacionada com a operação de uma aeronave, havida entre o período em que uma pessoa nela embarca com a intenção de realizar um voo, até o momento em que todas as pessoas tenham dela desembarcado e, durante o qual, pelo menos uma das situações abaixo ocorra:

a) qualquer pessoa sofra lesão grave ou morra como resultado de estar na aeronave, em contato direto com qualquer uma de suas partes, incluindo aquelas que dela tenham se desprendido, ou submetida à exposição direta do sopro de hélice, rotor ou escapamento de jato, ou às suas consequências. Exceção é feita quando as lesões resultem de causas naturais, forem auto ou por terceiros infligidas, ou forem causadas a pessoas que embarcaram clandestinamente e se acomodaram em área que não as destinadas aos passageiros e tripulantes;

b) a aeronave sofra dano ou falha estrutural que afete adversamente a resistência estrutural, o seu desempenho ou as suas características de voo; exija a

substituição de grandes componentes ou a realização de grandes reparos no componente afetado. Exceção é feita para falha ou danos limitados ao motor, suas carenagens ou acessórios; ou para danos limitados a hélices, pontas de asa, antenas, pneus, freios, carenagens do trem, amassamentos leves e pequenas perfurações no revestimento da aeronave;

c) a aeronave seja considerada desaparecida ou o local onde se encontre seja absolutamente inacessível.

## 2. Coleta de Dados

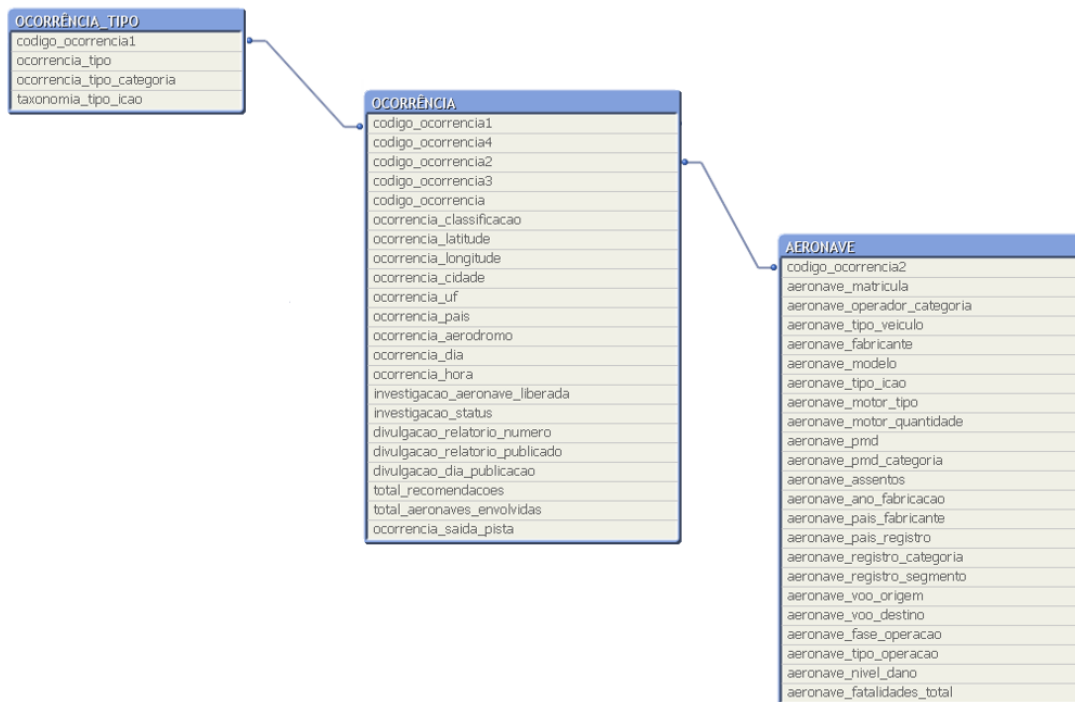
Os dados desse trabalho foram coletados através do Portal Brasileiro de Dados Abertos, responsável por manter mais de 12.000 conjuntos de dados referente a diversas organizações públicas ou vinculadas ao governo federal.

Para o trabalho, foi utilizada as bases de dados de ocorrências aeronáuticas, encontrada no endereço <https://dados.gov.br/dados/conjuntos-dados/ocorrencias-aeronauticas-da-aviacao-civil-brasileira>, que é gerenciada pelo Centro de Investigação e Prevenção de Acidentes Aeronáuticos e nela consta todas as ocorrências notificadas ao CENIPA, assim como informações das aeronaves envolvidas, local, data, número de fatalidades, dentre outros dados. Todas as informações divulgadas procuram resguardar a privacidade das pessoas físicas/jurídicas envolvidas, a fim de não ferir a Lei de Acesso à Informação (Redação dada pela Lei nº 13.853, de 2019).

Os dados contendo as informações de ocorrência de 2010 à 2022 foram disponibilizados em **OCORRÊNCIA.csv**, os dados referentes ao tipo de ocorrência em **OCORRÊNCIA\_TIPO.csv**. As informações sobre as aeronaves envolvidas nos acidentes, assim como o número de fatalidades pode ser visualizado na tabela **AERONAVE.csv**.

O relacionamento entre as tabelas utilizadas no projeto pode ser verificado na **Figura 3**.

**Figura 3: Relacionamento entre os datasets**



A coleta destes dados é feita utilizando a biblioteca Pandas no Python, cada comando `pd.read_csv` é responsável por coletar as informações de um arquivo csv (Comma-separated values) diferente, conforme mostrados nas **Figuras 4 e 5**.

**Figura 4: Importação de biblioteca Pandas**

#### ▼ Importação de Bibliotecas

✓ [43] # Importação das bibliotecas necessárias para execução do trabalho

✓ [122] #Importação de biblioteca Pandas  
import pandas as pd

**Figura 5: Carregamento de dados dos datasets dentro do DataFrame Pandas**

Coleta de dados

```

[45] # Carregamento dos dados dos datasets dentro do DataFrame
df_ocorrencia = pd.read_csv("/content/drive/MyDrive/TCC - Ciência de Dados e Big Data/06-Ocorrencias-Aeronauticas/ocorrencia.csv", sep=';')
df_aeronave = pd.read_csv("/content/drive/MyDrive/TCC - Ciência de Dados e Big Data/06-Ocorrencias-Aeronauticas/aeronave.csv", sep=';')
df_ocorrencia_tipo = pd.read_csv("/content/drive/MyDrive/TCC - Ciência de Dados e Big Data/06-Ocorrencias-Aeronauticas/ocorrencia_tipo.csv", sep=';')

df_fator_contribuinte = pd.read_csv("/content/drive/MyDrive/TCC - Ciência de Dados e Big Data/06-Ocorrencias-Aeronauticas/fator_contribuinte.csv", sep=';')
df_recomendacao = pd.read_csv("/content/drive/MyDrive/TCC - Ciência de Dados e Big Data/06-Ocorrencias-Aeronauticas/recomendacao.csv", sep=';')

[46] # Análise de informações sobre os datasets
df_ocorrencia.info()
df_aeronave.info()
df_ocorrencia_tipo.info()
  
```



Cada tabela de dados utilizada possui um conjunto de informação diferente e a integração das mesmas é essencial para gerar informações para o modelo de classificação, os dados contidos em cada um dos datasets são mostrados nas tabelas a seguir:

**Tabela 1: Descrição dos campos/colunas dos datasets**

<b>Dataset</b>	<b>Ocorrência</b>	<b>Arquivo "ocorren- cia.csv"</b>
Nome da coluna/campo	Descrição	Tipo
<i>codigo_ocorrencia</i>	ID da ocorrência	Numérico (inteiro)
<i>codigo_ocorrencia1</i>	ID da ocorrência	Numérico (inteiro)
<i>codigo_ocorrencia2</i>	ID da ocorrência	Numérico (inteiro)
<i>codigo_ocorrencia3</i>	ID da ocorrência	Numérico (inteiro)
<i>codigo_ocorrencia4</i>	ID da ocorrência	Numérico (inteiro)
<i>ocorrencia_classificacao</i>	Classificação da ocorrência	Texto
<i>ocorrencia_latitude</i>	Latitude em que houve a ocorrência	Texto
<i>ocorrencia_longitude</i>	Longitude em que houve a ocorrência	Texto
<i>ocorrencia_cidade</i>	Cidade da ocorrência	Texto
<i>ocorrencia_uf</i>	Estado da ocorrência	Texto
<i>ocorrencia_pais</i>	Pais da ocorrência	Texto
<i>ocorrencia_aerodromo</i>	Aeródromo em que houve a ocorrência	Texto
<i>ocorrencia_dia</i>	Dia da ocorrência	Texto
<i>ocorrencia_hora</i>	Hora da ocorrência	Texto
<i>investigacao_aeronave_liberada</i>	Liberação da investigação	Texto
<i>investigacao_status</i>	Status da investigação sobre a aeronave	Texto
<i>divulgacao_relatorio_numero</i>	Código do relatório divulgado	Texto
<i>divulgacao_relatorio_publicado</i>	Status da publicação do relatório	Texto
<i>divulgacao_dia_publicacao</i>	Dia da publicação do relatório	Texto
<i>total_recomendacoes</i>	Número total de recomendações	Numérico (inteiro)
<i>total_aeronaves_envolvidas</i>	Número de aeronaves envolvidas	Numérico (inteiro)
<i>ocorrencia_saida_pista</i>	Ocorrência durante a saída de pista (SIM/NÃO)	Texto

Dataset	Aeronave	Arquivo "aeronave.csv"
Nome da coluna/campo	Descrição	Tipo
codigo_ocorrencia2	ID da ocorrência	Numérico (inteiro)
aeronave_matricula	Matrícula da aeronave	Texto
aeronave_operador_categoria	Tipo de categoria do operador da aeronave	Texto
aeronave_tipo_veiculo	Tipo de aeronave	Texto
aeronave_fabricante	Nome da fabricante da aeronave	Texto
aeronave_modelo	Modelo da aeronave	Texto
aeronave_tipo_icao	Código gerado pela Organização da Aviação Civil Internacional	Texto
aeronave_motor_tipo	Tipo de motor da aeronave	Texto
aeronave_motor_quantidade	Quantidade de motores da aeronave	Texto
aeronave_pmd	Peso máximo de decolagem da aeronave	Numérico (inteiro)
aeronave_pmd_categoria	Categoria do peso máximo de decolagem	Numérico (inteiro)
aeronave_assentos	Quantidade de assentos	Numérico (inteiro)
aeronave_ano_fabricacao	Ano de fabricação da aeronave	Data
aeronave_pais_fabricante	País de fabricação	Texto
aeronave_pais_registro	País de registro	Texto
aeronave_registro_categoria	Categoria do registro da aeronave	Texto
aeronave_registro_segmento	Segmento do registro da aeronave	Texto
aeronave_voo_origem	Origem do voo	Texto
aeronave_voo_destino	Destino do voo	Texto
aeronave_fase_operacao	Fase de operação da aeronave no momento da ocorrência	Texto
aeronave_tipo_operacao	Tipo da operação da aeronave	Texto
aeronave_nivel_dano	Nível do dano da aeronave durante a ocorrência	Texto
aeronave_fatalidades_total	Quantidade de fatalidades	Numérico (inteiro)

Dataset	Tipo de Ocorrência	Arquivo "ocorren- cia_tipo.csv"
Nome da coluna/campo	Descrição	Tipo
codigo_ocorrendia1	ID da ocorrência	Númerico (inteiro)
ocorrendia_tipo	Tipo de ocorrência	Texto
ocorrendia_tipo_categoria	Categoria do tipo de ocorrência	Texto
taxonomia_tipo_icao	Classificação do tipo de ocorrência utilizando a taxonomia da Organização Internacional de Aviação Civil	Texto

Utilizando o método Pandas DataFrame info(), foi possível coletar informações como o número de colunas, rótulos de coluna, tipos de dados da coluna e o número de células em cada coluna (valores não nulos), conforme mostrado na figura a seguir:

**Figura 6: Informação dos datasets utilizando o método info()**

```
[46] # Análise de informações sobre os datasets
df_ocorrendia.info()
df_aeronave.info()
df_ocorrendia_tipo.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6769 entries, 0 to 6768
Data columns (total 22 columns):
#   Column                                     Non-Null Count  Dtype
---  ---
0   codigo_ocorrendia                         6769 non-null   int64
1   codigo_ocorrendia1                       6769 non-null   int64
2   codigo_ocorrendia2                       6769 non-null   int64
3   codigo_ocorrendia3                       6769 non-null   int64
4   codigo_ocorrendia4                       6769 non-null   int64
5   ocorrencia_classificacao                 6769 non-null   object
6   ocorrencia_latitude                      5135 non-null   object
7   ocorrencia_longitude                     5135 non-null   object
8   ocorrencia_cidade                        6769 non-null   object
9   ocorrencia_uf                            6769 non-null   object
10  ocorrencia_pais                          6769 non-null   object
11  ocorrencia_aerodromo                     6769 non-null   object
12  ocorrencia_dia                           6769 non-null   object
13  ocorrencia_hora                           6767 non-null   object
14  investigacao_aeronave_liberada            6531 non-null   object
15  investigacao_status                       6428 non-null   object
16  divulgacao_relatorio_numero              5987 non-null   object
17  divulgacao_relatorio_publicado            6769 non-null   object
18  divulgacao_dia_publicacao                 1781 non-null   int64
19  total_recomendacoes                      6769 non-null   int64
20  total_aeronaves_envolvidas                6769 non-null   int64
21  ocorrencia_saida_pista                    6769 non-null   object
dtypes: int64(7), object(15)
memory usage: 1.1+ MB
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6339 entries, 0 to 6338
Data columns (total 23 columns):
#   Column                                     Non-Null Count  Dtype
---  ---
0   codigo_ocorrendia2                       6339 non-null   int64
1   aeronave_matricula                       6306 non-null   object
2   aeronave_operador_categoria              6306 non-null   object
3   aeronave_tipo_veiculo                   6306 non-null   object
4   aeronave_fabricante                      6306 non-null   object
5   aeronave_modelo                          6306 non-null   object
6   aeronave_tipo_icao                        6306 non-null   object
7   aeronave_motor_tipo                     6289 non-null   object
8   aeronave_motor_quantidade                6306 non-null   object
9   aeronave_pmd                            6306 non-null   float64
10  aeronave_pmd_categoria                   6306 non-null   float64
11  aeronave_assentos                        6064 non-null   float64
12  aeronave_ano_fabricacao                  6078 non-null   float64
13  aeronave_pais_fabricante                 6306 non-null   object
14  aeronave_pais_registro                   6306 non-null   object
15  aeronave_registro_categoria              6306 non-null   object
16  aeronave_registro_segmento               6306 non-null   object
17  aeronave_voo_origem                      6305 non-null   object
18  aeronave_voo_destino                     6305 non-null   object
19  aeronave_fase_operacao                   6306 non-null   object
20  aeronave_tipo_operacao                   6306 non-null   object
21  aeronave_nivel_dano                      6306 non-null   object
22  aeronave_fatalidades_total               6306 non-null   float64
dtypes: float64(5), int64(1), object(17)
memory usage: 1.1+ MB
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7100 entries, 0 to 7099
Data columns (total 4 columns):
#   Column                                     Non-Null Count  Dtype
---  ---
0   codigo_ocorrendia1                       7100 non-null   int64
1   ocorrencia_tipo                           7099 non-null   object
2   ocorrencia_tipo_categoria                 7099 non-null   object
3   taxonomia_tipo_icao                       7099 non-null   object
dtypes: int64(1), object(3)
memory usage: 222.0+ KB
```

### 3. Processamento/Tratamento de Dados

O Processamento de dados é uma etapa fundamental para a identificação de padrões e tendências, nesta etapa, os dados são tratados e transformados em informações úteis para a análise e tomada de decisão.

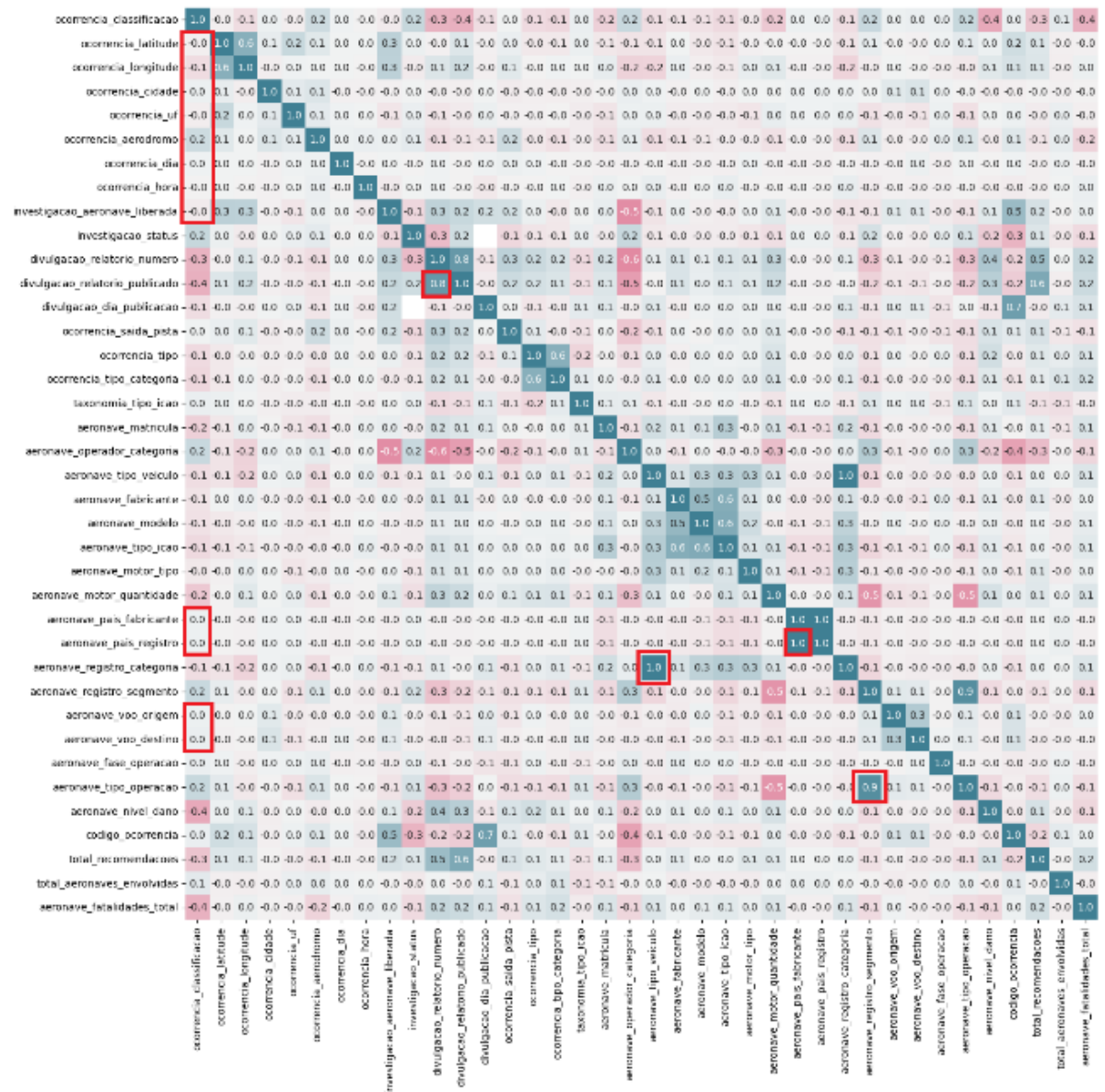
O primeiro processo efetuado para tratamento dos dados foi a filtragem de colunas da tabela que não ajudariam na execução do modelo, o processo de filtragem é uma técnica importante pois permite selecionar as variáveis mais relevantes e eliminar aquelas que possuem pouco ou nenhum impacto na análise. Essa técnica tem como objetivo reduzir a complexidade dos dados, tornando-os mais fáceis de serem analisados e processados.

Uma das principais vantagens da filtragem de colunas é a redução de ruído nos dados, ou seja, a eliminação de variáveis que não possuem relação significativa com o resultado final da análise. Isso melhora a qualidade dos resultados e torna as análises mais precisas e confiáveis.

Para execução do processo de filtragem, foi feita uma avaliação da relevância das colunas utilizando o método de correlação da biblioteca Pandas no Python, conforme mostrado na **Figura 7**.

Além da análise do resultado do “*Linear Correlation*”, também foi realizada uma exploração manual e verificação de colunas com grande número de valores nulos. Ao final da avaliação, foram excluídas as colunas: `codigo_ocorrendia1`, `codigo_ocorrendia2`, `codigo_ocorrendia3`, `codigo_ocorrendia4`, `ocorrendia_latitude`, `ocorrendia_longitude`, `ocorrendia_cidade`, `ocorrendia_uf`, `ocorrendia_pais`, `ocorrendia_aerodromo`, `investigacao_aeronave_liberada`, `investigacao_status`, `divulgacao_relatorio_numero`, `divulgacao_relatorio_publicado`, `divulgacao_dia_publicacao`, `total_recomendacoes`, `taxonomia_tipo_icao`, `aeronave_matricula`, `aeronave_operador_categoria`, `aeronave_tipo_icao`, `aeronave_pmd`, `aeronave_pmd_categoria`, `aeronave_assentos`, `aeronave_voo_origem`, `aeronave_voo_destino`, `aeronave_tipo_operacao`.

**Figura 7: Resultado da correlação linear com exemplos de colunas que foram suprimidas do *dataset* final após a análise de correlação**



**Tabela 2: Colunas do dataset final com valores nulos**

ocorrencia_latitude	1661
ocorrencia_longitude	1661
ocorrencia_hora	2
investigacao_aeronave_liberada	249
investigacao_status	348
divulgacao_relatorio_numero	800
divulgacao_dia_publicacao	5271
ocorrencia_tipo	1
ocorrencia_tipo_categoria	1
taxonomia_tipo_icao	1
aeronave_matricula	574
aeronave_operador_categoria	574

aeronave_tipo_veiculo	574
aeronave_fabricante	574
aeronave_modelo	574
aeronave_tipo_icao	574
aeronave_motor_tipo	592
aeronave_motor_quantidade	574
aeronave_pmd	574
aeronave_pmd_categoria	574
aeronave_assentos	824
aeronave_ano_fabricacao	810
aeronave_pais_fabricante	574
aeronave_pais_registro	574
aeronave_registro_categoria	574
aeronave_registro_segmento	574
aeronave_voo_origem	575
aeronave_voo_destino	575
aeronave_fase_operacao	574
aeronave_tipo_operacao	574
aeronave_nivel_dano	574
aeronave_fatalidades_total	574

**Figura 8: Código em Python com a seleção das colunas mais relevantes do dataset para o modelo de Machine Learning**

```
# Filtro de colunas que serão utilizadas para a aplicação dos modelos de ML, colunas não relevantes para a análise foram filtradas
# levando em consideração as informações obtidas com a análise de correlação linear e exploração dos dados
df_ocorr_aeronauticas = df_ocorr_aeronauticas_full[['codigo_ocorrencia',
'total_recomendacoes',
'total_aeronaves_envolvidas',
'ocorrencia_saida_pista',
'ocorrencia_tipo',
'ocorrencia_tipo_categoria',
'aeronave_tipo_veiculo',
'aeronave_motor_tipo',
'aeronave_motor_quantidade',
'aeronave_registro_categoria',
'aeronave_registro_segmento',
'aeronave_fase_operacao',
'aeronave_tipo_operacao',
'aeronave_nivel_dano',
'aeronave_fatalidades_total',
'ocorrencia_classificacao']]
df_ocorr_aeronauticas.info()
```

Por se tratar de um banco de dados com diversos campos com inconsistência, a próxima etapa para tratamento das informações foi efetuar uma substituição de textos como '\*\*\*' ou 'NULL' em valores mais coerentes para a análise, para fins de padronização, foi utilizado o texto 'NÃO INFORMADO' para substituir os valores '\*\*\*' e nulos das colunas ocorrencia\_tipo, ocorrencia\_tipo\_categoria, aeronave\_tipo\_veiculo, aeronave\_fabricante, aeronave\_motor\_tipo e aeronave\_motor\_quantidade. Para a coluna aeronave\_ano\_fabricação foi adotado a opção de substituir os registros com texto igual "NULL" para o valor 0. Todo o tratamento executado é mostrado na **Figura 9**. Foram utilizados os métodos *replace* e *fillna* para substituição dos dados.

**Figura 9: Código em Python para substituição de valores nulos ou '\*\*\*'**

```
# Substituição de valores '***' por texto 'NÃO INFORMADO' para uniformização da base de dados
df_ocorr_aeronauticas = df_ocorr_aeronauticas.replace('***', 'NÃO INFORMADO')

# Substituição de valores NULOS de colunas com informação do tipo texto por valor 'NÃO INFORMADO'
df_ocorr_aeronauticas[df_ocorr_aeronauticas.select_dtypes(
    include=['object']).columns] = df_ocorr_aeronauticas.select_dtypes(
    include=['object']).fillna('NÃO INFORMADO')

# Substituição de valores NULOS da coluna aeronave_fatalidades_total por zero (0)
df_ocorr_aeronauticas['aeronave_fatalidades_total'] = df_ocorr_aeronauticas['aeronave_fatalidades_total'].fillna(0)
df_ocorr_aeronauticas.info()
```

Com aplicação dos métodos acima, foram substituídos 574 valores nulos na coluna `aeronave_fatalidades_total`, para os valores '\*\*\*', foram alteradas as seguintes quantidades de registros para cada coluna:

**Figura 10: Código em Python para contagem de valores '\*\*\*'**

```
count = df_ocorr_aeronauticas.apply(lambda x: x[x=='***'].count())
print(count)
```

<code>codigo_ocorrencia</code>	0
<code>total_recomendacoes</code>	0
<code>total_aeronaves_envolvidas</code>	0
<code>ocorrencia_saida_pista</code>	0
<code>ocorrencia_tipo</code>	12
<code>ocorrencia_tipo_categoria</code>	12
<code>aeronave_tipo_veiculo</code>	173
<code>aeronave_motor_tipo</code>	280
<code>aeronave_motor_quantidade</code>	108
<code>aeronave_registro_categoria</code>	173
<code>aeronave_registro_segmento</code>	83
<code>aeronave_fase_operacao</code>	28
<code>aeronave_tipo_operacao</code>	130
<code>aeronave_nivel_dano</code>	57
<code>aeronave_fatalidades_total</code>	0
<code>ocorrencia_classificacao</code>	0

A escolha do zero para a coluna `aeronave_fatalidade_total` foi efetuada após identificação que muitos dos dados não informados estavam relacionados a ocorrências do tipo incidente, que tem como característica não ter fatalidades envolvidas.

A última etapa executada no processamento dos dados foi a normalização dos dados numéricos. A normalização de dados é uma técnica importante no pré-processamento de dados para o treinamento de modelos de aprendizado de máquina, pois ajuda a evitar problemas com diferentes escalas de unidades de medida nos

dados, além de tornar os dados mais consistentes, removendo os efeitos de valores extremos e outras anomalias nos dados. Isso pode ajudar a melhorar a precisão e a estabilidade dos modelos de aprendizado de máquina.

O processo de normalização foi efetuado utilizando o método `fit_transform` do pacote `MinMaxScaler` e foi aplicado para as colunas `total_recomendacoes`, `aeronave_fatalidades_total` e `total_aeronaves_envolvidas`, conforme mostrado na **Figura 11**.

**Figura 11: Código em Python para normalização de colunas numéricas**

```
# Normalização dos dados numéricos
# Selecionar apenas as colunas numéricas que deseja normalizar
cols_to_normalize = ['total_recomendacoes', 'total_aeronaves_envolvidas', 'aeronave_fatalidades_total']

# Cria uma instância do objeto MinMaxScaler
scaler = MinMaxScaler()

# Aplica a normalização min-max nas colunas selecionadas
df_ocorr_aeronauticas[cols_to_normalize] = scaler.fit_transform(df_ocorr_aeronauticas[cols_to_normalize])
```

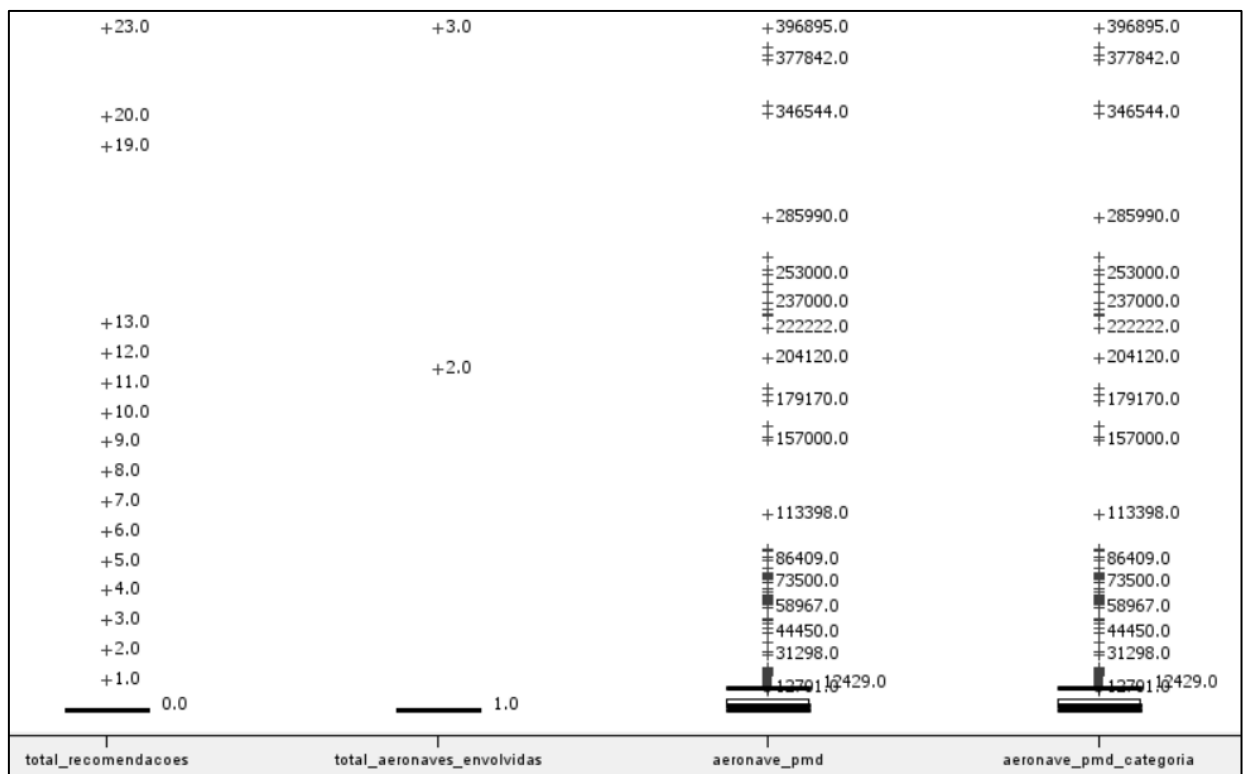
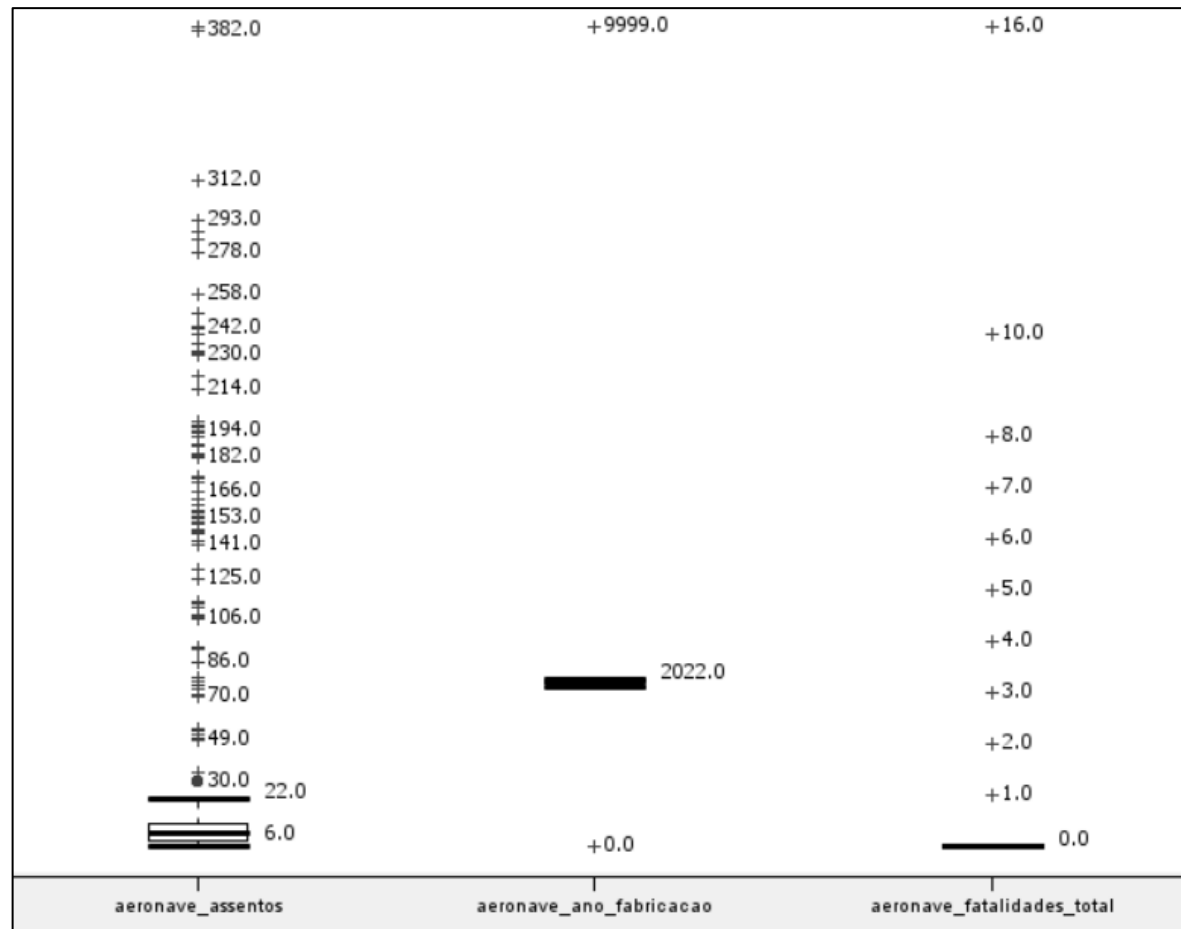
#### 4. Análise e Exploração dos Dados

Após a coleta de dados, a fase de exploração das informações obtidas da database foi iniciada com o objetivo de obter insights, identificar padrões e verificar quais dados são relevantes para a próxima etapa do projeto, a criação de Modelos de Machine Learning.

A primeira etapa de análise foi verificar o comportamento dos dados numéricos através de gráfico de boxplot e análise estatísticas das informações obtidas. Na **Figura 12** é mostrado os resultados do boxplot para as colunas `'total_recomendacoes'`, `'total_aeronaves_envolvidas'`, `'aeronave_fatalidades_total'`, `'aeronave_pmd'`, `'aeronave_pmd_categoria'`, `'aeronave_assentos'` e `'aeronave_ano_fabricacao'`.



**Figura 12: Gráfico de bloxplot de columnas numéricas**



O gráfico boxplot mostra a distribuição de um conjunto de dados numéricos, com base em cinco estatísticas resumo: valor mínimo, primeiro quartil (Q1), mediana (Q2), terceiro quartil (Q3) e valor máximo. Os valores que ultrapassam o limite inferior e superior mostram possíveis valores discrepantes ou atípicos. Nos gráficos anteriores é possível perceber que o campo 'aeronave\_ano\_fabricação' possui valores incorretos de inserção 0 ou 9999. Para a coluna 'aeronave\_assentos', apesar da quantidade de assentos variar de forma significativa, todos os valores estão em uma faixa considerada correta, uma vez que dependendo do porte da aeronave, a mesma pode possuir apenas alguns assentos, como jatos executivos, enquanto outras podem acomodar centenas de passageiros, como os aviões comerciais de grande porte.

Dando sequência à análise exploratória dos dados, foi realizada a consulta estatística da base, utilizando o método describe do Pandas e o método displot do Seaborn, com as informações obtidas foi possível saber quantos valores vazios existiam em cada coluna, valores de mínimo e máximo, desvio padrão e o histograma de cada campo.

**Figura 13: Código Python para análise de estatísticas do dataset**

```
# Obter informações estatísticas do dataset
stats = df_ocorr_aeronauticas.describe()

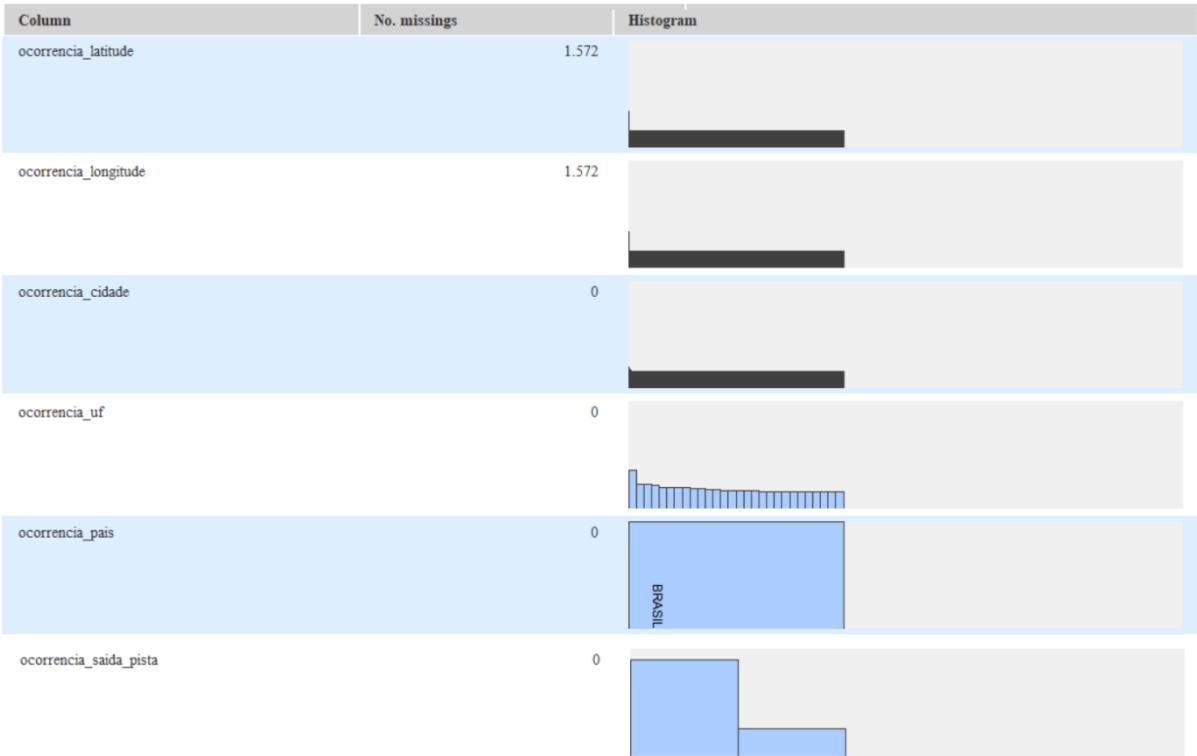
# Imprimir as informações estatísticas
print(stats)

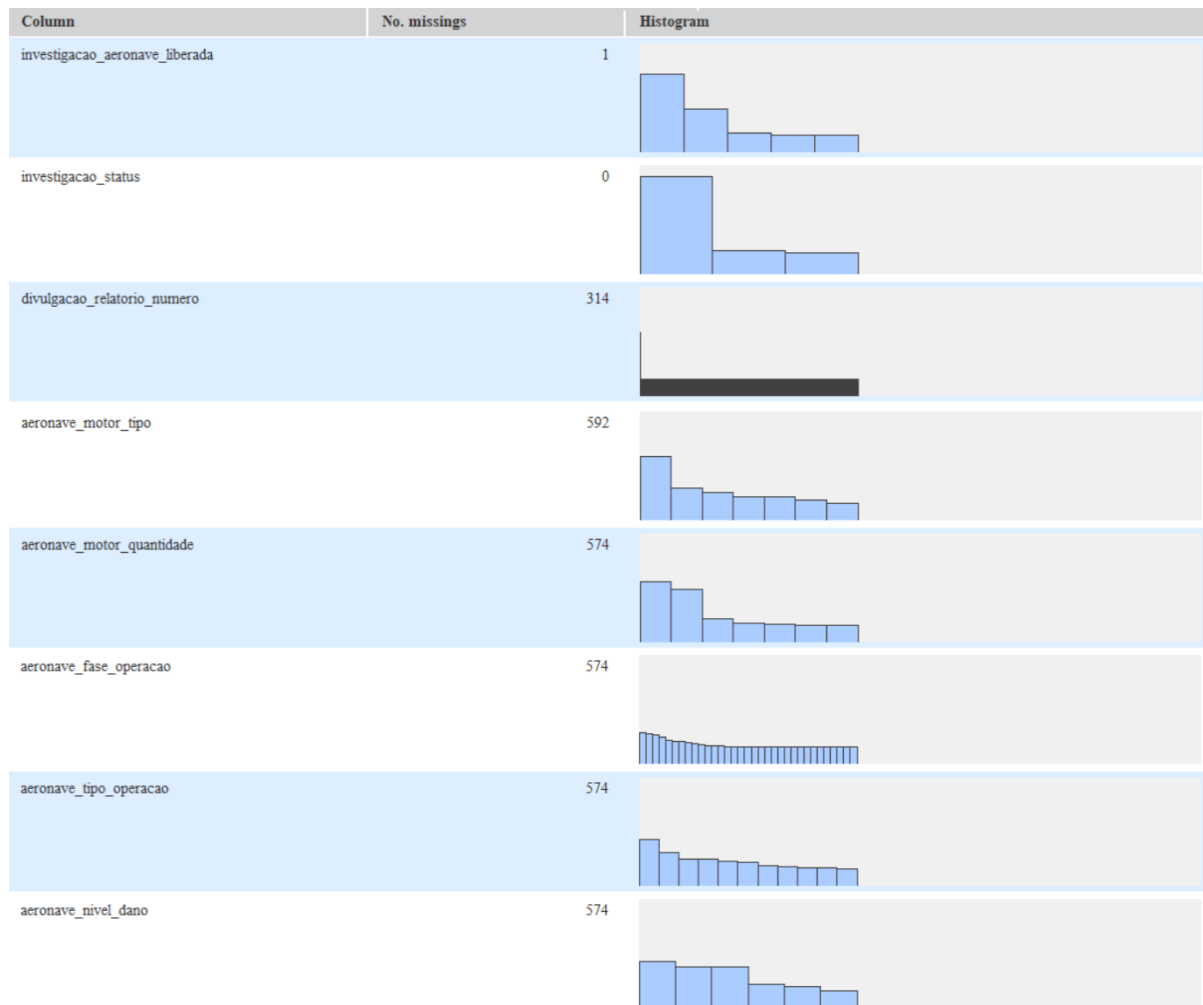
# Plota o histograma de cada campo
for col in df_ocorr_aeronauticas.columns:
    sns.displot(df_ocorr_aeronauticas[col])
    #df_ocorr_aeronauticas[col].hist()
    #plt.show()
```

Figura 14: Dados estatísticos para campos numéricos



Figura 15: Dados estatísticos para campos de texto





Após a análise estatística do dado foi realizada a consulta de valores nulos por ano e assim foi observado a existência de valores vazios para diversas colunas obtidas do dataset aeronave.csv, ao verificar a quantidade de “missing values” por ano, foi identificado que a grande maioria era referente à 2010:

**Tabela 3: Missing values por ano do *dataset* df\_ocorr\_aeronauticas**

ano	aeronave_tipo_v eiculo	aeronave_motor _tipo	aeronave_motor _quantidade	aeronave_registr o_categoria	aeronave_registr o_segmento	aeronave_fase_o peracao	aeronave_tipo_o peracao	aeronave_nivel_d ano	aeronave_fatalid ades_total
2010	574	574	574	574	574	574	574	574	574
2011	0	0	0	0	0	0	0	0	0
2012	0	0	0	0	0	0	0	0	0
2013	0	0	0	0	0	0	0	0	0
2014	0	0	0	0	0	0	0	0	0
2015	0	0	0	0	0	0	0	0	0
2016	0	0	0	0	0	0	0	0	0
2017	0	0	0	0	0	0	0	0	0
2018	0	0	0	0	0	0	0	0	0
2019	0	0	0	0	0	0	0	0	0
2020	0	0	0	0	0	0	0	0	0
2021	0	5	0	0	0	0	0	0	0
2022	0	13	0	0	0	0	0	0	0

Por fim, foi gerada a matriz de correlação linear a fim de verificar a relação entre as colunas de cada um dos databases. A matriz de correlação linear é utilizada para medir a intensidade e a direção da associação linear entre as colunas, ou seja, quanto uma coluna se altera em função de mudanças na outra.

Na primeira tentativa de extração da matriz, antes da aplicação de filtro para as colunas, foi obtido o resultado mostrado na **Figura 7**, exibida na sessão **3. Processamento/Tratamento de Dados**. Segue abaixo o código Python usado para gerar o gráfico de correlação linear:

**Figura 16: Código Python para gerar gráfico de correlação**

Gráfico de Correlação

```
[96] # Seleciona todas as colunas que são de texto
df_categorical = df_ocorr_aeronauticas_full[df_ocorr_aeronauticas_full.select_dtypes(include=['object']).columns]

# Conversão dos dados categóricos em numéricos com o método OrdinalEncoder
ordinal_encoder = OrdinalEncoder()
data_encoded = ordinal_encoder.fit_transform(df_categorical)
data_encoded = pd.DataFrame(data_encoded, columns=df_categorical.columns)

# Seleciona todas as colunas que são numéricas
df_numerical = df_ocorr_aeronauticas[df_ocorr_aeronauticas.select_dtypes(exclude=['object']).columns]

# Concatena as colunas que passaram pelo processo de conversão e as colunas numéricas para formar um único database
df_corr = pd.concat([data_encoded, df_numerical], axis=1)
df_corr = df_corr.drop('ocorrencia_pais', axis=1) # Retirando coluna país pois a mesma possui um único valor para todo o database

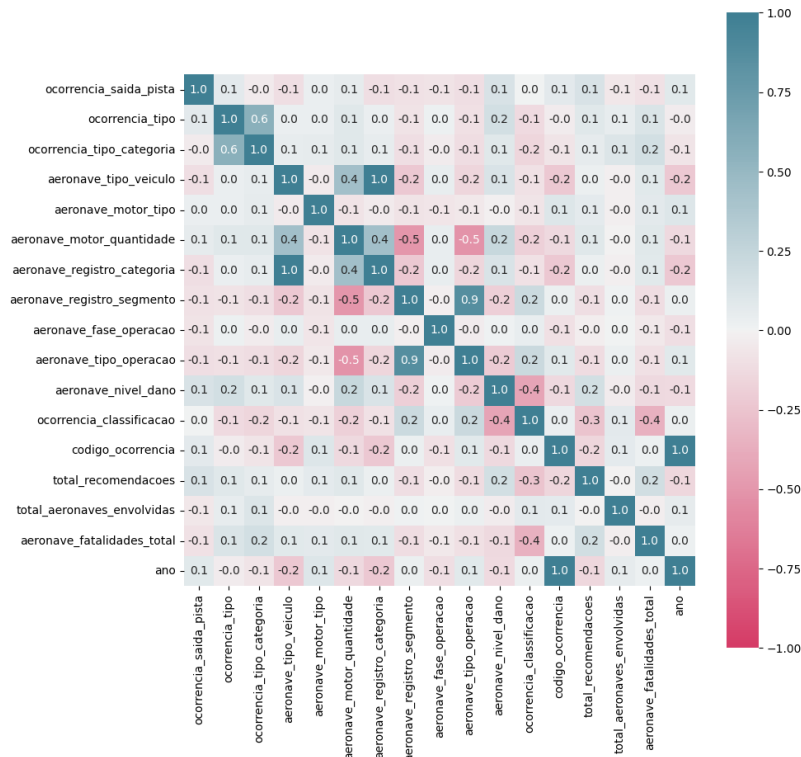
[97] # Calcular a matriz de correlação linear
corr = df_corr.corr(method='spearman')

# Aumentar tamanho da figura
fig, ax = plt.subplots(figsize=(20,20))
# Plotar um heatmap da matriz de correlação
axis_corr = sns.heatmap(corr, vmin=-1, vmax=1, center=0, cmap=sns.diverging_palette(1, 220, n=500),
square=True, annot=True, fmt=".1f")

plt.show()
```

Após aplicação do filtro de colunas, levando em consideração a correlação de cada uma das colunas à variável resposta “ocorrencia\_classificacao” o seguinte resultado foi obtido para a matriz de correlação linear:

**Figura 17: Matriz de correlação linear após filtro de colunas**



A matriz de correlação linear é útil para identificar relações fortes ou fracas entre as variáveis, e pode ser utilizada para a seleção de variáveis em análises estatísticas mais complexas, como regressões. Além disso, a matriz de correlação linear pode ser utilizada para detectar multicolinearidade, que é a presença de alta correlação entre duas ou mais variáveis independentes em um modelo estatístico.

## 5. Criação de Modelos de Machine Learning

Após as etapas de coleta de dados, processamento e análise exploratória do dataset, esse capítulo irá abordar a aplicação de modelos de machine learning e a escolha do mais adequado, utilizando métricas de avaliação de performance.

O objetivo do estudo é identificar qual o melhor rótulo para um determinado tipo de ocorrência, dado um conjunto de características referentes ao cenário em que a mesmo aconteceu, optou-se por aplicar algoritmos de classificação, utilizados em problemas de aprendizagem supervisionada, que consistem em prever uma variável-alvo a partir de um conjunto de variáveis preditoras.

Um modelo de classificação é uma técnica de aprendizado de máquina que permite prever a qual classe uma determinada amostra pertence, com base em características ou recursos específicos que podem ser extraídos dos dados. Esse modelo é comumente usado em problemas onde o objetivo é classificar novas amostras em uma ou mais categorias predefinidas.

Nesse estudo serão utilizados 3 algoritmos de classificação diferentes, são eles: Árvore de Decisão, Gradient Boosted e Randon Forest.

A Árvore de Decisão é um modelo de aprendizagem supervisionada que utiliza uma estrutura em forma de árvore para tomar decisões com base em características ou atributos dos dados. Uma das principais vantagens desse modelo é sua capacidade de lidar com dados heterogêneos e de grande volume, além de ser fácil de interpretar. No entanto, as árvores de decisão tendem a ser suscetíveis ao overfitting, ou seja, podem se ajustar demais aos dados de treinamento, prejudicando a generalização.

O modelo Gradient Boosted é um algoritmo de aprendizagem supervisionada que usa árvores de decisão como base. Uma das principais vantagens desse modelo é sua capacidade de lidar com dados complexos e não lineares, permitindo que seja usado em problemas de classificação e regressão. Além disso, ele tem boa capacidade de generalização, o que significa que é capaz de se adaptar bem a novos dados. Por outro lado, o Gradient Boosted pode ser bastante lento em relação a outros modelos, devido à sua abordagem iterativa.

Já o modelo Randon Forest é um conjunto de árvores de decisão independentes, onde cada árvore é construída a partir de um subconjunto aleatório dos dados e das variáveis disponíveis. Uma das principais vantagens desse modelo é sua capacidade de lidar com dados heterogêneos e de grande volume, além de ser capaz de lidar bem com overfitting. Além disso, o Randon Forest é capaz de fornecer informações sobre a importância de cada variável na tomada de decisão. No entanto, ele pode ser lento em relação a outros modelos, especialmente quando há muitas variáveis.

### 5.1. Métricas de Avaliação dos Modelos

A avaliação de um modelo de classificação de Machine Learning é essencial para medir sua eficácia em prever resultados e identificar possíveis problemas que precisam ser corrigidos. Existem várias métricas e ferramentas que podem ser utilizadas para avaliar a performance de um modelo de classificação, entre elas, a matriz de confusão, que é uma tabela que compara as previsões do modelo com os valores reais. Ela apresenta quatro valores: verdadeiros positivos (TP), verdadeiros negativos (TN), falsos positivos (FP) e falsos negativos (FN).

**Figura 18: Matriz de confusão**

		Valor Predito	
		Sim	Não
Real	Sim	Verdadeiro Positivo (TP)	Falso Negativo (FN)
	Não	Falso Positivo (FP)	Verdadeiro Negativo (TN)

Com base nas informações obtidas através da matriz de confusão é possível gerar outras métricas como:

**Acurácia:** é a proporção de previsões corretas em relação ao total de previsões realizadas. Essa é uma métrica simples e útil, mas pode não ser a melhor escolha em casos onde as classes estão desbalanceadas.

**Precisão:** é a proporção de verdadeiros positivos em relação a todas as previsões positivas feitas pelo modelo. Ela é útil quando se quer minimizar os falsos positivos.

**Recall:** é a proporção de verdadeiros positivos em relação a todos os valores reais positivos. É útil quando se quer minimizar os falsos negativos.

**F1 Score:** é uma média harmônica entre a precisão e o recall. É útil quando se quer ter um balanço entre essas duas métricas.



Para a avaliação dos modelos, a principal medida de avaliação que será utilizada é a acurácia, porém todas as métricas acima serão consideradas como critérios de comparação entre algoritmos.

## 5.2. Aplicação dos Modelos

Antes da aplicação do modelo de Machine Learning, foi efetuada a separação do dataset em dois novos, um contendo as variáveis preditoras (X) e outra contendo a variável-alvo (y), conforme mostrado na Figura 19.

**Figura 19: separação do dataset em dados de entrada e target**

```
# Separar os dados de entrada (features) e saída (target)
X = df_ocorr_aeronauticas.drop('ocorrencia_classificacao', axis=1)
y = df_ocorr_aeronauticas['ocorrencia_classificacao']
```

Como a database de ocorrências aeronáuticas possui muitas colunas de texto e em sua maioria os algoritmos de Machine Learning não suportam esse formato de dado, foi necessário utilizar um método de codificação para converter esses dados em dados numéricos. Para isso foi usado o método One-Hot Encoding da biblioteca Sklearn.

Na codificação one-hot, cada categoria do dado é transformada em uma nova coluna binária, indicando se a categoria está presente ou não em cada linha de dado. Por exemplo, se a coluna "aeronave\_categoria" tiver as categorias "AVIÃO", "HELICÓPTERO", "PLANADOR" e "BALÃO", serão criadas quatro novas colunas: "aeronave\_categoria\_aviao", "aeronave\_categoria\_helicoptero", "aeronave\_categoria\_planador" e "aeronave\_categoria\_balao", contendo apenas valores 0 ou 1.

**Tabela 4: Aplicação do One-Hot Encoding**

aeronave_registro_categoria	AVIÃO	HELICÓPTERO	PLANADOR	BALÃO
AVIÃO	1	0	0	0
HELICÓPTERO	0	1	0	0
AVIÃO	1	0	0	0
PLANADOR	0	0	1	0
BALÃO	0	0	0	1

Já na codificação ordinal, as categorias são transformadas em valores numéricos, geralmente de forma crescente ou decrescente. Por exemplo, na coluna "Tamanho" com as categorias "pequeno", "médio" e "grande", a codificação ordinal poderia transformá-las em 1, 2 e 3, respectivamente.

One-hot encoding foi escolhido pois as variáveis categóricas não possuem uma ordem ou relação de ordem entre elas, como ele cria uma nova coluna binária para cada categoria distinta, isso permite que o modelo capture a associação entre cada categoria individualmente, sem impor uma relação de ordem.

**Figura 20: Código Python aplicando o One-Hot Encoding**

```
# Aplicar One-Hot Encoding nas colunas categóricas
categorical_columns = ['ocorrencia_saida_pista', 'ocorrencia_tipo', 'ocorrencia_tipo_categoria',
                      'aeronave_tipo_veiculo', 'aeronave_motor_tipo', 'aeronave_motor_quantidade',
                      'aeronave_registro_categoria', 'aeronave_registro_segmento', 'aeronave_fase_operacao',
                      'aeronave_tipo_operacao', 'aeronave_nivel_dano']

ohe = OneHotEncoder()
X_categorical = ohe.fit_transform(X[categorical_columns])

# Concatenar as colunas numéricas e categóricas
X_numerical = X.drop(columns=categorical_columns)
X_encoded = sp.sparse.hstack([X_numerical, X_categorical])
```

Uma vez aplicada a codificação, foram iniciados os testes utilizando diferentes algoritmos, na imagem a seguir é mostrada a implementação do modelo de classificação Decision Tree, sem a aplicação de hiperparâmetros, o modelo obteve uma acurácia de 73%.

**Figura 21: Código Python de Decision Tree**

Execução do modelo de classificação Decision Tree

```
[ ] # Criar modelo de Árvore de Decisão
    dt_model = DecisionTreeClassifier()

    # Fazer a validação cruzada e obter as previsões
    y_pred_dt = cross_val_predict(dt_model, X_encoded, y, cv=10)

    # Aplicar o cross validation
    scores_dt = cross_val_score(dt_model, X_encoded, y, cv=10)

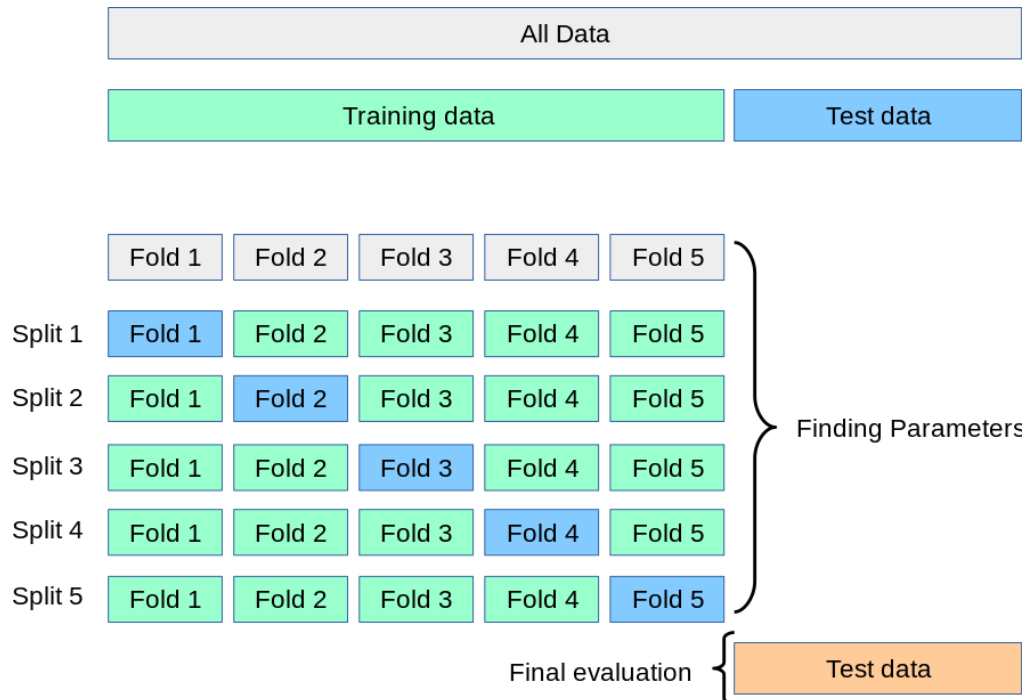
    # Imprimir os resultados da validação cruzada
    print("Accuracy: %0.2f (+/- %0.2f)" % (scores_dt.mean(), scores_dt.std() * 2))

Accuracy: 0.73 (+/- 0.24)
```

Para todas as implementações de modelos de Machine Learning, foram utilizadas técnicas de Cross Validation que consiste em dividir o conjunto de dados

em subconjuntos (chamados folds) e realizar múltiplas iterações de treinamento e teste. Tanto no método `cross_val_predict`, quanto no `cross_val_score`, foi utilizado o parâmetro `cv = 10` para dividir o dataset em 10 subconjuntos, sendo que em cada iteração 9 conjuntos foram utilizados para treinamento e 1 para teste, conforme exemplo mostrado na **Figura 22**.

**Figura 22: Método de Cross Validation**



Na execução do algoritmo de Gradient Boosted foi obtido uma acurácia de 85% com a implementação do modelo sem configuração dos hiperparâmetros:

**Figura 23: Código Python de Gradient Boosted**

Execução do modelo de classificação Gradient Boosted

```
[ ] # Definir o modelo de Gradient Boosted com hiperparâmetros padrão
    gb_model = GradientBoostingClassifier()

    # Fazer as previsões com Cross Validation
    y_pred_gb = cross_val_predict(gb_model, X_encoded, y, cv=10)

    # Calcular o score com Cross Validation
    scores_gb = cross_val_score(gb_model, X_encoded, y, cv=10, scoring='accuracy')

    # Imprimir os resultados da validação cruzada
    print("Accuracy: %0.2f (+/- %0.2f)" % (scores_gb.mean(), scores_gb.std() * 2))

Accuracy: 0.85 (+/- 0.11)
```

O melhor resultado de implementação de modelo sem a configuração de hiperparâmetro foi visualizado ao utilizar o Random Forest, onde a acurácia foi de 87%, conforme mostrado na imagem a seguir:

**Figura 24: Código Python de Random Forest**

Execução do modelo de classificação Random Forest

```
[ ] # Criar um modelo Random Forest com 100 árvores
    rf_model = RandomForestClassifier()

    # Fazer a validação cruzada do modelo e obter as previsões
    y_pred_rf = cross_val_predict(rf_model, X_encoded, y, cv=10)

    # Aplicar o cross validation
    scores_rf = cross_val_score(rf_model, X_encoded, y, cv=10)

    # Imprimir os resultados da validação cruzada
    print("Accuracy: %0.2f (+/- %0.2f)" % (scores_rf.mean(), scores_rf.std() * 2))

Accuracy: 0.87 (+/- 0.08)
```

## 5.2. Utilização de Hiperparâmetros para melhoria de performance dos Modelos

Após a aplicação dos algoritmos de Machine Learning utilizando a forma de implementação mais básica, foi feito o trabalho de alteração dos hiperparâmetros de cada um deles com o objetivo de obter o melhor resultado possível para a classificação.

Hiperparâmetros são parâmetros externos ao modelo de aprendizado de máquina que precisam ser definidos antes do treinamento do modelo. Esses parâmetros influenciam o processo de treinamento do modelo e afetam o desempenho e a capacidade de generalização do modelo.

Ao contrário dos parâmetros do modelo, que são aprendidos durante o treinamento, os hiperparâmetros são definidos manualmente e controlam aspectos do processo de treinamento, como a complexidade do modelo, a taxa de aprendizado, o número de estimadores, a profundidade máxima das árvores, entre outros.

Para encontrar os melhores valores de hiperparâmetros é possível testar as possibilidades manualmente ou realizar uma busca sistemática utilizando técnicas

como a busca em grade (grid search) ou busca aleatória (random search). Isso envolve testar diferentes combinações de valores de hiperparâmetros e avaliar o desempenho do modelo para cada combinação, a fim de selecionar aquela que produz o melhor resultado.

Devido a questões relacionadas a tempo de execução e performance do método, foi optado nesse projeto pela utilização do RandomizedSearchCV da biblioteca Scikit-learn.

Com os ajustes de hiperparâmetros foi obtida uma acurácia de 88% no algoritmo de Random Florest, conforme mostrado na figura a seguir:

**Figura 25: Código Python de Random Forest com ajuste de hiperparâmetros**

```
# Definir o espaço de hiperparâmetros
param_dist = {
    'n_estimators': randint(10, 500),
    'max_depth': [5, 8, 9, 10, 15, None],
    'min_samples_split': randint(2, 10),
    'min_samples_leaf': randint(1, 5),
    'max_features': [ 'sqrt', 'log2']
}

# Criar o modelo de Random Forest
rf_model = RandomForestClassifier()

# Realizar a busca aleatória
random_search = RandomizedSearchCV(rf_model, param_distributions=param_dist, cv= 10,
                                   n_iter=20, scoring='accuracy', random_state=10)
random_search.fit(X_encoded, y)

# Obter os melhores hiperparâmetros encontrados
best_params = random_search.best_params_

# Treinar o modelo com os melhores hiperparâmetros
rf_model_best = RandomForestClassifier(**best_params)

# Fazer a validação cruzada do modelo e obter as previsões
y_pred_rf = cross_val_predict(rf_model_best, X_encoded, y, cv=10)

# Aplicar o cross validation
scores_rf = cross_val_score(rf_model_best, X_encoded, y, cv=10)

# Imprimir os resultados da validação cruzada
print("Accuracy: %0.2f (+/- %0.2f)" % (scores_rf.mean(), scores_rf.std() * 2))
```

Accuracy: 0.88 (+/- 0.10)

Os hiperparâmetros encontrados pelo Random Search podem ser vistos na **Figura 26:**

**Figura 26: Valores de hiperparâmetros encontrados pelo Random Search para a execução do Random Forest**

```
# Imprimir os melhores parâmetros encontrados
print("Melhores Parâmetros:")
print(random_search.best_params_)

# Imprimir a melhor pontuação
print("Melhor Pontuação:")
print(random_search.best_score_)

Melhores Parâmetros:
{'max_depth': None, 'max_features': 'sqrt', 'min_samples_leaf': 3, 'min_samples_split': 7, 'n_estimators': 244}
Melhor Pontuação:
0.8805876216968012
```

O mesmo processo foi executado, com o Random Search, para os algoritmos `DecisionTreeClassifier` e `GradientBoostingClassifier` em ambos foi observado melhoras na execução dos modelos. Para a Árvore de Decisão houve um aumento da acurácia para 80%, já o Gradient Boosting teve uma melhora na acurácia para 87%.

Os melhores parâmetros identificados para o Decision Tree foram: {'criterion': 'gini', 'max\_depth': None, 'max\_features': 'sqrt', 'min\_samples\_leaf': 1, 'min\_samples\_split': 8, 'splitter': 'best'}

Para o classificador Gradient Boosting foram obtidos os seguintes parâmetros: {'learning\_rate': 0.01, 'loss': 'log\_loss', 'max\_depth': 10, 'max\_features': 'sqrt', 'min\_samples\_leaf': 4, 'min\_samples\_split': 5, 'n\_estimators': 190}

**Figura 27: Código Python de Decision Tree com ajuste de hiperparâmetros**

```
# Definir o espaço de hiperparâmetros
param_dist = {
    'max_depth': [None], #
    'min_samples_split': randint(2, 10),
    'min_samples_leaf': randint(1, 5),
    'max_features': ['sqrt', 'log2'],
    'criterion': ['gini', 'entropy', 'log_loss'],
    'splitter': ['best', 'random']
}

# Criar o modelo de Random Forest
dt_model = DecisionTreeClassifier()
```

```

# Realizar a busca aleatória
random_search = RandomizedSearchCV(dt_model, param_distributions=param_dist, cv= 10,
                                   n_iter=20, scoring='accuracy', random_state=10)
random_search.fit(X_encoded, y)

# Obter os melhores hiperparâmetros encontrados
best_params = random_search.best_params_

# Treinar o modelo com os melhores hiperparâmetros
dt_model_best = DecisionTreeClassifier(**best_params)

# Fazer a validação cruzada do modelo e obter as previsões
y_pred_dt = cross_val_predict(dt_model_best, X_encoded, y, cv=10)

# Aplicar o cross validation
scores_dt = cross_val_score(dt_model_best, X_encoded, y, cv=10)

# Imprimir os resultados da validação cruzada
print("Accuracy: %0.2f (+/- %0.2f)" % (scores_dt.mean(), scores_dt.std() * 2))

```

Accuracy: 0.80 (+/- 0.24)

**Figura 28: Código Python de Gradient Boosting com ajuste de hiperparâmetros**

```

# Definir o espaço de hiperparâmetros
param_dist = {
    'loss': ['log_loss'],
    'n_estimators': randint(50, 500),
    'max_depth': [3, 5, 10],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'max_features': ['sqrt', 'log2'],
    'learning_rate': [0.01, 0.1, 1]
}

# Criar o modelo de Random Forest
gb_model = GradientBoostingClassifier()

# Realizar a busca aleatória
random_search = RandomizedSearchCV(gb_model, param_distributions=param_dist, cv= 10,
                                   n_iter=20, scoring='accuracy', random_state=10)
random_search.fit(X_encoded, y)

# Obter os melhores hiperparâmetros encontrados
best_params = random_search.best_params_

# Treinar o modelo com os melhores hiperparâmetros
gb_model_best = GradientBoostingClassifier(**best_params)

# Fazer a validação cruzada do modelo e obter as previsões
y_pred_gb = cross_val_predict(gb_model_best, X_encoded, y, cv=10)

# Aplicar o cross validation
scores_gb = cross_val_score(gb_model_best, X_encoded, y, cv=10)

# Imprimir os resultados da validação cruzada
print("Accuracy: %0.2f (+/- %0.2f)" % (scores_gb.mean(), scores_gb.std() * 2))

```

Accuracy: 0.87 (+/- 0.08)

## 6. Interpretação dos Resultados

A etapa da interpretação dos resultados foi executada utilizando código Python para gerar a Matriz de Confusão e demais métricas descritas na sessão **5.1. Métricas de Avaliação dos Modelos**. Inicialmente foi necessário criar Labels para facilitar a interpretação das informações obtidas, depois foi construída uma sessão de código para plotar as informações de cada um dos modelos estudados.

**Figura 29: Código Python para gerar métricas de avaliação dos algoritmos de ML**  
Resultados

```
[ ] # Labels para os eixos x e y
    labels = ['ACIDENTE', 'INCIDENTE', 'INCIDENTE GRAVE']
```

Resultados do modelo de classificação Decision Tree

```
[ ] # Calcular a matriz de confusão
    cm_dt = confusion_matrix(y, y_pred_dt)

    cr_dt = classification_report(y, y_pred_dt)
    print(cr_dt)

    # Mostrar a matriz de confusão em forma de gráfico
    sns.heatmap(cm_dt, annot=True, cmap='Blues', fmt='g', xticklabels=labels, yticklabels=labels)

    # Adicionar rótulos para os eixos x e y
    plt.xlabel('Classe Predita')
    plt.ylabel('Classe Verdadeira')

    # Exibir o gráfico
    plt.show()
```

Resultados do modelo de classificação Gradient Boosted

```
[ ] # Calcular a matriz de confusão
    cm_gb = confusion_matrix(y, y_pred_gb)

    cr_gb = classification_report(y, y_pred_gb)
    print(cr_gb)

    # Mostrar a matriz de confusão em forma de gráfico
    sns.heatmap(cm_gb, annot=True, cmap='Blues', fmt='g', xticklabels=labels, yticklabels=labels)

    # Adicionar rótulos para os eixos x e y
    plt.xlabel('Classe Predita')
    plt.ylabel('Classe Verdadeira')

    # Exibir o gráfico
    plt.show()
```



### Resultados do modelo de classificação Random Forest

```
[ ] # Calcular a matriz de confusão
cm_rf = confusion_matrix(y, y_pred_rf)

cr_rf = classification_report(y, y_pred_rf)
print(cr_rf)

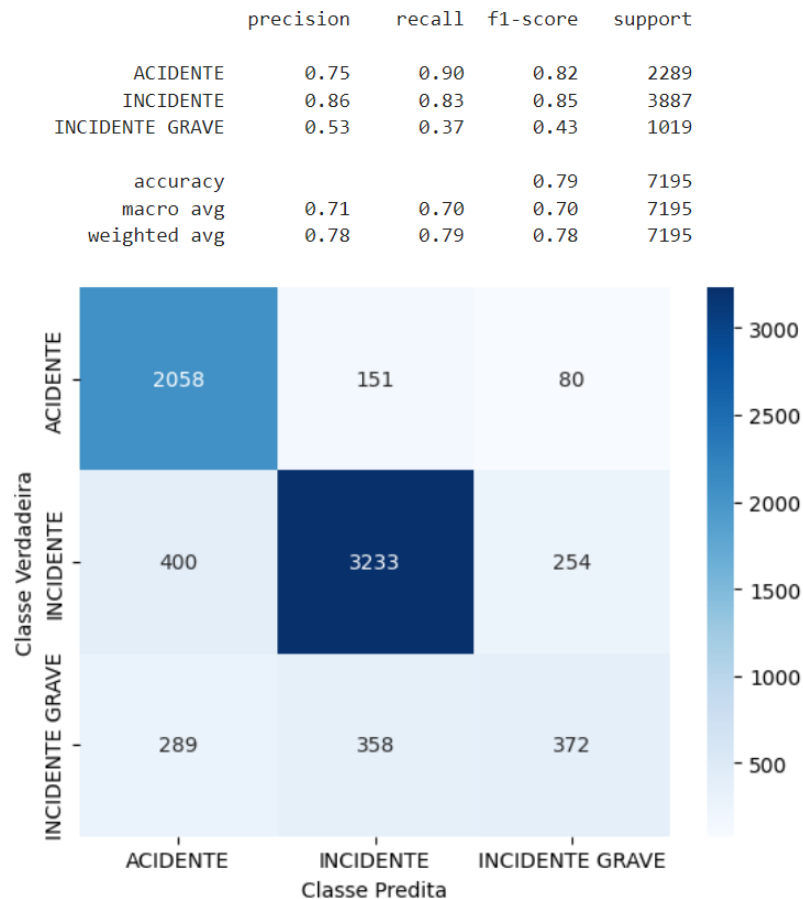
# Plotar a matriz de confusão como um heatmap
sns.heatmap(cm_rf, annot=True, cmap='Blues', fmt='g', xticklabels=labels, yticklabels=labels)

# Adicionar rótulos para os eixos x e y
plt.xlabel('Classe Predita')
plt.ylabel('Classe Verdadeira')

# Exibir o gráfico
plt.show()
```

Para tornar a visualização dos dados mais amigáveis, foram utilizadas as funções Heatmap da biblioteca Seaborn e Pyplot da biblioteca Matplotlib, também foi usado o método `classification_report` para criar relatório de texto mostrando as principais métricas de classificação. Os resultados obtidos para cada algoritmo podem ser visualizados nas **Figuras 30, 31 e 32**:

**Figura 30: Métricas para Decison Tree**

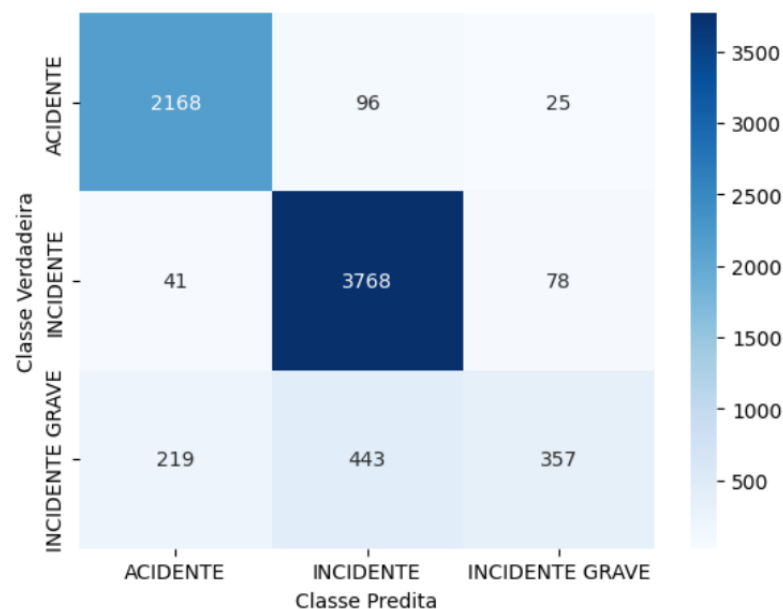


Apesar da acurácia de 80% no modelo de Decision Tree, através da Matriz de Confusão é possível perceber 647 registros foram classificados incorretamente com outras categorias em vez de INCIDENTE GRAVE, devido a isso, o parâmetro “recall” apresenta 37% para essa classificação, enquanto para ACIDENTE e INCIDENTE a porcentagem aumenta para 90% e 83% respectivamente.

Um cenário semelhante é observado também ao analisar as métricas do Gradient Boosting que apesar de obter uma acurácia de 87%, possui recall para INCIDENTE GRAVE igual a 35%.

**Figura 31: Métricas para Gradient Boosting**

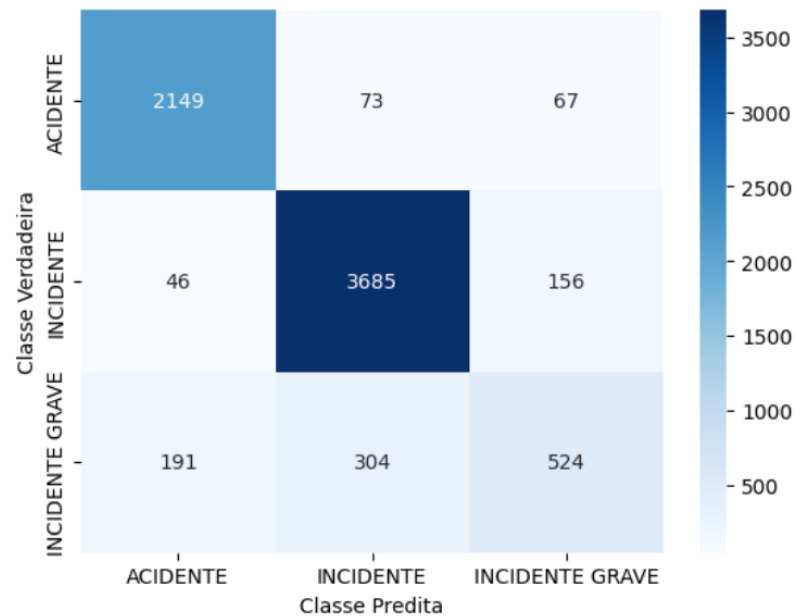
	precision	recall	f1-score	support
ACIDENTE	0.89	0.95	0.92	2289
INCIDENTE	0.87	0.97	0.92	3887
INCIDENTE GRAVE	0.78	0.35	0.48	1019
accuracy			0.87	7195
macro avg	0.85	0.76	0.77	7195
weighted avg	0.87	0.87	0.86	7195



O recall de INCIDENTE GRAVE apresenta um melhor resultado no algoritmo de Random Forest, onde o valor sobe para 51%, para ACIDENTE e INCIDENTE os valores correspondem a 94% e 95% de verdadeiros positivos.

**Figura 32: Métricas para Random Florest**











	precision	recall	f1-score	support
ACIDENTE	0.90	0.94	0.92	2289
INCIDENTE	0.91	0.95	0.93	3887
INCIDENTE GRAVE	0.70	0.51	0.59	1019
accuracy			0.88	7195
macro avg	0.84	0.80	0.81	7195
weighted avg	0.88	0.88	0.88	7195



O Random Forest é escolhido como o mais adequado para os dados avaliados pois além de apresentar uma acurácia superior, possui melhores resultados de recall, precisão e por consequência de f1-score.

## 7. Apresentação dos Resultados

Para a apresentação dos resultados obtidos, foi utilizado o modelo de Canvas proposto por Dourard.

<div><div>PREDICTION TASK</div><div></div></div> <div><p>Analisar as informações geradas pelo CENIPA sobre ocorrências aeronáuticas, a fim de classificar as mesmas como 'INCIDENTE', 'INCIDENTE GRAVE' ou 'ACIDENTE'</p></div>	<div><div>DECISIONS</div><div></div></div> <div><p>O resultado do modelo é a predição do nível da ocorrência analisada, possibilitando determinar ações de fiscalização, capacitação e prevenção de acidentes aéreos além de reduzir o tempo de trabalho dos analistas da CENIPA no processo manual de classificação.</p></div>	<div><div>VALUE PROPOSITION</div><div></div></div> <div><p>O objetivo é de ter todos todas as ocorrências aeronáuticas classificadas, podendo analisar e estudar a gravidade dos acidentes. Conhecendo a gravidade das ocorrências, a quantidade de ocorrências para cada classificação, dentre outros, pode-se identificar ações preventivas, como: fiscalização, educação, etc.</p></div>	<div><div>DATA COLLECTION</div><div></div></div> <div><p>Os novos dados poderão ser coletados no portal de dados abertos da Força Aérea Brasileira.</p></div>	<div><div>DATA SOURCES</div><div></div></div> <div><p>Os dados foram coletados do Portal Brasileiro de Dados Abertos e foram utilizados os dados da base de dados de ocorrências aeronáuticas do CENIPA.</p><p>As tabelas utilizadas no trabalho foram referentes a OCORRÊNCIA, OCORRÊNCIA_TIPO e AERONAVE.</p></div>
<div><div>IMPACT SIMULATION</div><div></div></div> <div><p>As simulações com os dados de ocorrência do período de 2010 a 2022 mostraram uma acurácia de classificação de 88% utilizando o modelo de classificação Randon Florest. Além da acurácia, foram analisadas outras métricas como precisão, recall, f1-score e suporte.</p></div>	<div><div>MAKING PREDICTIONS</div><div></div></div> <div><p>As predições são realizadas após o tratamento e o processamento dos dados, como a base de dados é atualizada diariamente, as predições podem ser executadas uma vez ao dia.</p></div>	<div><div>BUILDING MODELS</div><div></div></div> <div><p>Foram testados três algoritmos: Decision Tree Classifier e o Random Forest Classifier e o Gradient Boosting Classifier. O modelo foi construído com o algoritmo que teve o melhor desempenho na classificação, no caso o Random Forest.</p></div>		<div><div>FEATURES</div><div></div></div> <div><p>As variáveis preditoras X são:</p><p>total_recomendacoes, total_aeronaves_envolvidas, ocorrencia_saida_pista, ocorrencia_tipo, ocorrencia_tipo_categoria,, aeronave_tipo_veiculo, aeronave_motor_tipo, aeronave_motor_quantidade, aeronave_registro_categoria, aeronave_registro_segmento, aeronave_fase_operacao, aeronave_tipo_operacao, aeronave_nivel_dano, aeronave_fatalidades_total.</p><p>E a variável alvo Y é:</p><p>ocorrencia_classificacao.</p></div>
		<div><div>MONITORING</div><div></div></div> <div><p>A avaliação e o monitoramento do modelo foi realizada por meio da classificação das ocorrências, medindo a acurácia das classes, bem como pela medição do tempo gasto para classificação de <i>novas</i> registros pelo modelo.</p></div>		

## 8. Links

A seguir, estão os links do vídeo de apresentação e do repositório contendo os dados utilizados no projeto.

Link para o vídeo: <https://youtu.be/YPdmVKKV8H4>

Link para o repositório: <https://github.com/talitasandrade/TCCPUCMINAS>

## REFERÊNCIAS

ASSESSORIA DE COMUNICAÇÃO SOCIAL DA ANAC. **Mercado aéreo em 2019: maior número de passageiros transportados da série histórica.** ANAC, 2020. Disponível em: <<https://www.gov.br/anac/pt-br/noticias/2020/mercado-aereo-registra-maior-numero-de-passageiros-transportados-da-serie-historica>>. Acesso em: 10 de abr. de 2023.

**Frota de Aeronaves Civis Brasileiras em Tempo Real.** ANAC, 2023. Disponível em: <<https://app.powerbi.com/view?r=eyJrljoiMGQ2MjdiMWYtNzQxZS00Mzk2LWE0NmEtMzFiMTNmNzI1OTk3liwidCI6ImI1NzQ4ZjZILWI0YTQtNGlyYi1hYjJhLWVmOTUyMjM2ODM2NilsImMiOjR9>>. Acesso em: 23 de abr. de 2023.

**CENIPA - Ocorrências Aeronáuticas na Aviação Civil Brasileira.** Força Aérea Brasileira, 2022. Disponível em: <<https://dados.gov.br/dados/conjuntos-dados/ocorrencias-aeronauticas-da-aviacao-civil-brasileira>>. Acesso em: 10 de abr. de 2023.

**História do CENIPA.** Força Aérea Brasileira. Disponível em: <<https://www2.fab.mil.br/cenipa/index.php/historico>>. Acesso em: 10 de abr. de 2023.

**O que é investigação.** Força Aérea Brasileira. Disponível em: <<https://www2.fab.mil.br/cenipa/index.php/investigacoes>>. Acesso em: 11 de abr. de 2023.

**Cross-validation: evaluating estimator performance.** Disponível em: <[https://scikit-learn.org/stable/modules/cross\\_validation.html](https://scikit-learn.org/stable/modules/cross_validation.html)>. Acesso em: 25 de abr. de 2023.

**Boletim de Logística: A Importância do Transporte Aéreo para o Brasil.** Disponível em: <<https://ontl.epl.gov.br/wp-content/uploads/2022/02/Setor-Aereo-Brasileiro-v3.pdf>>. Acesso em: 10 de abr. de 2023.