

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

**Fernanda Santos Da Silva Velloso**

**ANÁLISE DE OCORRÊNCIAS AERONÁUTICAS**

Belo Horizonte  
2022

**Fernanda Santos Da Silva Velloso**

## **ANÁLISE 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

2022

## SUMÁRIO

<b>1. Introdução .....</b>	<b>4</b>
<b>1.1. Contextualização.....</b>	<b>4</b>
<b>1.2. O problema proposto .....</b>	<b>4</b>
<b>1.3. Objetivos .....</b>	<b>6</b>
<b>3. Processamento/Tratamento de Dados .....</b>	<b>11</b>
<b>4. Análise e Exploração dos Dados .....</b>	<b>15</b>
<b>5. Criação de Modelos de Machine Learning.....</b>	<b>25</b>
<b>6. Interpretação dos Resultados.....</b>	<b>34</b>
<b>7. Apresentação dos Resultados.....</b>	<b>35</b>
<b>8. Links .....</b>	<b>36</b>
<b>REFERÊNCIAS .....</b>	<b>12</b>
<b>APÊNDICE .....</b>	<b>37</b>

## **1. Introdução**

### **1.1. Contextualização**

Com o crescimento das operações aéreas, surge a necessidade de aprimoração das formas condução do transporte aéreo e investimento em infraestrutura e principalmente em procedimentos e ferramentas de segurança operacional. Tais ações visam manter a taxa de acidentes e incidentes dentro da meta estabelecida pelas agências reguladoras nacionais e internacionais

Com o intuito de mitigar essas ocorrências, em 1951 nasceu SIPAER - Sistema de Investigação e Prevenção de Acidentes e em 1965 passou a realizar investigações Aeronáuticas, que tem por missão “Promover a prevenção de acidentes aeronáuticos, preservando os recursos humanos e materiais, visando ao progresso da aviação brasileira”.

O CENIPA (Centro de Investigação e Prevenção de Acidentes Aeronáuticos) foi criado em 19 de novembro de 1971 e veio com o intuito de ser o órgão principal do SIPAER, o qual após a criação do CENIPA se tornou um sistema ao invés de serviço como era ainda considerado (BRASIL, 2017). Após sua criação, o próprio já criou o novo Comitê Nacional de Prevenção de Acidentes Aeronáuticos (CNPAA), com o objetivo de trazer tanto a comunidade aeronáutica nacional como a estrangeira, a fim de dialogar novas ideias quanto à prevenção de acidentes e de que forma poderiam aumentar cada vez mais a segurança operacional.

O acidente aeronáutico, pode ser definido como toda ocorrência relacionada com a operação de uma aeronave havida entre o período em que qualquer pessoa entra na aeronave com a intenção de realizar um voo até o momento em que todas as pessoas venham desembarcado, em consequência da qual: (1) qualquer pessoa tenha sofrido lesões graves ou morrido, exceto quando as lesões resultarem de causas naturais ou forem 13 auto ou por outrem infligidas; (2) a aeronave tenha sofrido danos ou falha estrutural: (I) afetando adversamente a resistência estrutural, desempenho ou características de voo; ou (II) exigindo substituição ou reparos importantes do componente afetado, ou (III) a aeronave tenha sido considerada desaparecida.

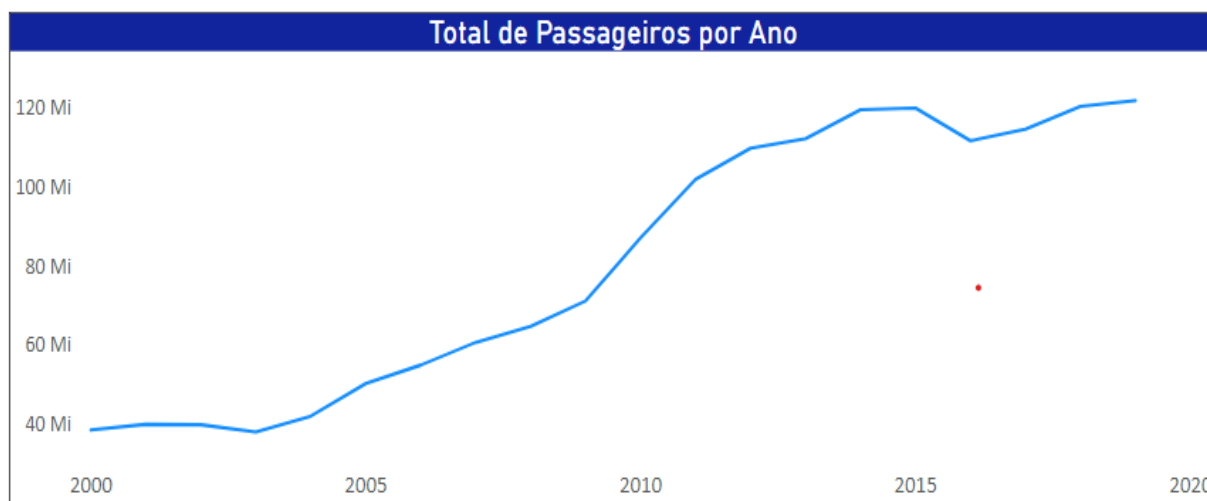
Há uma grande diferença de acidentes aeronáuticos para incidentes aeronáuticos, sendo que um acidente é mais delicado e de relevância bem maior para o meio aeronáutico. Um acidente é considerado desta forma: toda ocorrência que aconteça desde que o tripulante

ingresse na aeronave e com a intenção de realizar um voo e que pelo menos uma das situações venha a ocorrer na aeronave ou nos tripulantes, qualquer pessoa sofra alguma lesão grave ou morra, a aeronave sofra dano muito grande ou até mesmo falha estrutural e se a aeronave desaparecer ou ficar inacessível, acontecendo qualquer um dos três fatores citados já é considerado um acidente aeronáutico (COSTA FILHO, 2015, p.236).

Observando os dados disponibilizados pelo SIPAER, bem como diversos relatórios finais de investigações de acidentes, é possível perceber não apenas a similaridade entre parte dos acidentes, mas também a recorrência de alguns dos fatores contribuintes, fatores estes que foram considerados na criação do modelo deste trabalho.

### **(Why?)**

A análise dos dados de acidentes aéreos torna-se importante tanto para dar apoio à solução de diversos problemas como para prever possíveis ocorrências futuras, tendo em vista o crescimento e constante volume de tráfego, conforme pode ser verificado no gráfico abaixo baseado na base de dados estatística de transporte aéreo e operações aéreas da ANAC – Agencia Nacional de Aviação Civil.



### **(Who?)**

A base de dados de ocorrências aeronáuticas é gerenciada pelo Centro de Investigação e Prevenção de Acidentes Aeronáuticos (CENIPA). Dentre as informações disponíveis estão os dados sobre as aeronaves envolvidas, fatalidades, local, data, horário dos eventos e informações taxonômicas típicas das investigações de acidentes (AIG).

Essa base de dados é composta por informações preliminares provenientes do formulário CENIPA-05 (Ficha de Notificação de Ocorrências Aeronáuticas) e consolidada a partir dos relatórios de investigações publicados.

**(What?):**

Quais são as categorias de acidentes (Acidentes, Incidente e Incidente Grave), baseadas em variáveis da aviação.

**(Where?):**

Território brasileiro

**(When?):**

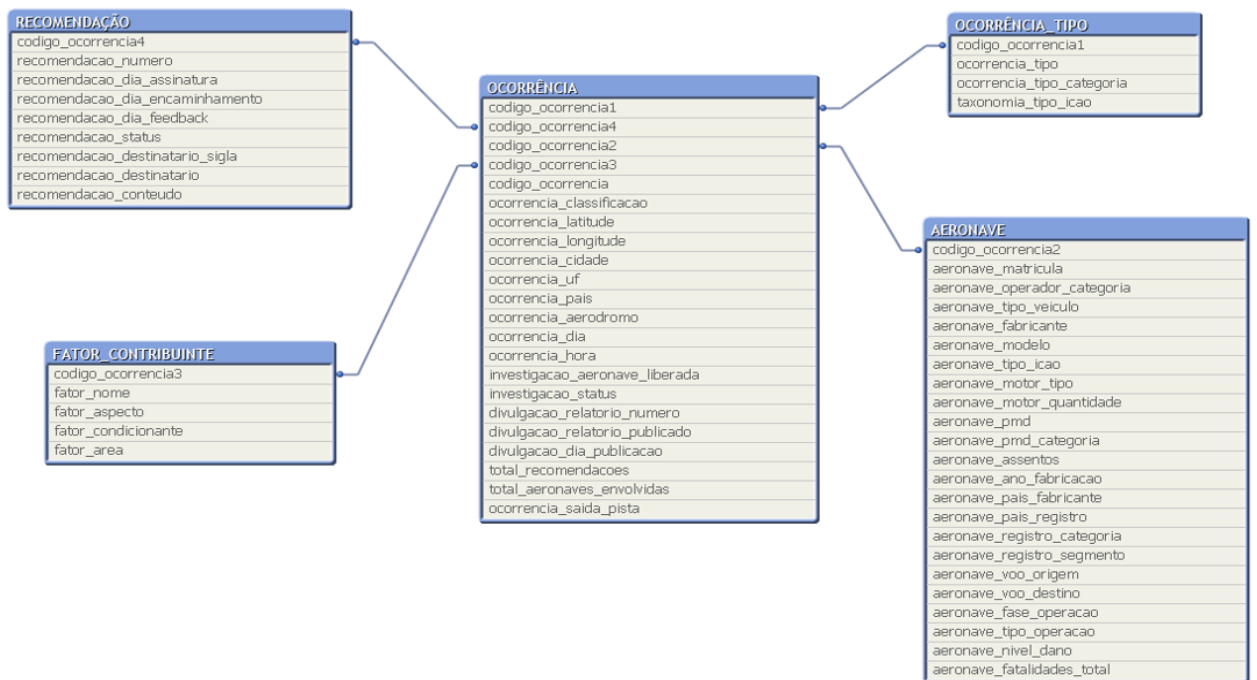
Constam nesta base de dados as ocorrências aeronáuticas notificadas ao CENIPA nos anos de 2012 a 2021 que ocorreram em solo brasileiro.

### **1.3. Objetivos**

Um dos principais objetivos deste trabalho é identificar e verificar os perigos e riscos que podem vir a acontecer no dia a dia da aviação através de métodos estatísticos, a existência de padrões nos perfis de aeronautas envolvidos em acidentes aeronáuticos envolvendo lapsos de julgamento de pilotagem, processo decisório, e a existência de correlações entre fatores contribuintes para operação da aviação.

## 2. Coleta de Dados

Este é o relacionamento de todas as bases disponibilizadas:



No entanto, foram utilizadas somente as seguintes colunas para o datasheet base de dados:

Base	Coluna	Descrição
Ocorrências	codigo_ocorrendia	Código da ocorrência, numeração de controle
Ocorrências	ocorrendia_classificacao	Classificação da ocorrência em Acidente, Incidente e Incidente Grave
Ocorrências	ocorrendia_cidade	Cidade em que ocorreu a ocorrência
Ocorrências	ocorrendia_uf	Estado em que ocorreu a ocorrência
Ocorrências	ocorrendia_dia	Dia da ocorrência
Aeronaves	codigo_ocorrendia2	Código da ocorrência, numeração de controle
Aeronaves	aeronave_matricula	Matricula da aeronave envolvida no acidente

Aeronaves	aeronave_operador_categoria	Categoria de habilitação do operador do voo
Aeronaves	aeronave_tipo_veiculo	Tipos de veículo: anfíbio, avião, balão, dirigível, girocôptero, helicóptero, hidroavião, motoplanador, planador, trike e ultraleve
Aeronaves	aeronave_fabricante	Fabricante da aeronave
Aeronaves	aeronave_modelo	conforme código ICAO [5] referente ao modelo da aeronave
Aeronaves	aeronave_tipo_icao	aeronaves que requerem um treinamento avançado e focado apenas na sua operação
Aeronaves	aeronave_motor_tipo	Tipos de motores da aeronave: Jato, pistão, turbo eixo, turboélice e sem tração
Aeronaves	aeronave_motor_quantidade	conforme quantidade de motores cadastrado no RAB
Aeronaves	aeronave_pmd	Peso máximo de decolagem
Aeronaves	aeronave_pmd_categoria	categoria, em relação à certificação de tipo de aeronaves regulamentada pelo RAB
Aeronaves	aeronave_assentos	Quantidade de assentos
Aeronaves	aeronave_ano_fabricacao	Ano de Fabricação da Aeronave
Aeronaves	aeronave_pais_fabricante	Pais de fabricação da aeronave
Aeronaves	aeronave_pais_registro	Pais de registro da aeronave



Aeronaves	aeronave_registro_categoria	conforme registro da aeronave cadastrado no RAB
Aeronaves	aeronave_registro_segimento	conforme registro da aeronave cadastrado no RAB
Aeronaves	aeronave_voo_origem	Local de origem do voo
Aeronaves	aeronave_voo_destino	Local de Destino do voo
Aeronaves	aeronave_fase_operacao	Fase de Operação conforme normatização da SAIPHER em vigor
Aeronaves	aeronave_tipo_operacao	Quanto ao tipo de operação: agrícola, especializada, instrução, não regular, policial, privada, regular e taxi aéreo
Aeronaves	aeronave_nivel_dano	Nível de ano da ocorrência
Aeronaves	aeronave_fatalidades_total	Quantidade de fatalidades envolvida no acidente
Fator Contribuinte	codigo_ocorrencia3	Código da ocorrência, numeração de controle
Fator Contribuinte	fator_nome	Nome do fator que ocasionou a ocorrências
Fator Contribuinte	fator_aspecto	Agrupamento do nome do fator
Fator Contribuinte	fator_condicionante	Condições para que houvesse a ocorrência
Fator Contribuinte	fator_area	Area do Fator nome que ocasionou a ocorrência
Ocorrência Tipo	codigo_ocorrencia1	Código da ocorrência, numeração de controle

Ocorrência Tipo	ocorrencia_tipo	Classificação da ocorrência quanto ao tipo do acontecimento
Ocorrência Tipo	ocorrencia_tipo_categoria	Categorização do tipo de ocorrência
Ocorrência Tipo	taxonomia_tipo_icao	categorias descritas pela OACI, através do CAST-ICAO Common Taxonomy Team (CICTT)

### 3. Processamento/Tratamento de Dados

O Tratamento de dados foi feito com Python, no ambiente Jupyter Notebook, utilizando a biblioteca “pandas”.

Primeiramente foram carregadas as bases de análise e feita limpeza e adequação dos dados conforme passos abaixo.

- Carregamento da base de dados Ocorrências:

```
#1.3 #Carregar a base de dados ocorrencia

#Leitura do arquivo ocorrencia.csv
df_ocorrencia = pd.read_csv("C:\\Users\\fefaj\\OneDrive\\Área de Trabalho\\
TCC Aprovado\\Desenvolvimento\\Base de dados\\ocorrencia.csv",
                             encoding = "UTF-8", sep = ";", decimal = ',')
```

- Em se tratando de base de dados quanto menos informações não necessárias melhor será o processamento. Como algumas colunas trazem informações que não serão utilizadas na análise foi feita a Limpeza das colunas desnecessárias:

```
#Exclusão das colunas que não serão utilizadas
df_ocorrencia = df_ocorrencia.drop(columns=['codigo_ocorrencia1',
'codigo_ocorrencia2',
'codigo_ocorrencia3',
'codigo_ocorrencia4',
'ocorrencia_latitude',
'ocorrencia_longitude',
'ocorrencia_pais',
'ocorrencia_aerodromo',
'ocorrencia_hora',
'investigacao_aeronave_liberada',
'investigacao_status',
'divulgacao_relatorio_numero',
'divulgacao_relatorio_publicado',
'divulgacao_dia_publicacao',
'total_recomendacoes',
'total_aeronaves_envolvidas',
'ocorrencia_saida_pista'])
```

Um dos passos da limpeza dos dados foi a verificação e limpeza de valores nulos que podem impactar na contagem geral de linhas:

```
In [14]: df_ocorrencia.isnull().sum()
```

```
Out[14]: codigo_ocorrencia      0
          ocorrencia_classificacao  0
          ocorrencia_cidade      0
          ocorrencia_uf          0
          dtype: int64
```

- Conversão da Coluna que contém a data da ocorrência para “ano”, visto que para geração de gráficos, esta informação por menorizada iria aumentar muito o eixo x, dificultando a visualização e para uma análise de uma forma macro deste trabalho, não é necessário saber esta informação com esse nível de exatidão:

```
#Transformado a coluna ocorrencia_dia para o ano da célula
df_ocorrencia['ocorrencia_dia'] = df_ocorrencia['ocorrencia_dia'].dt.year
```

Feito essa primeira limpeza dos dados foi carregada a base de dados aeronave:

```
#Carregar a base de dados aeronave
df_aeronave = pd.read_csv("C:\\Users\\fefaj\\OneDrive\\Área de Trabalho\\
TCC Aprovado\\Desenvolvimento\\Base de dados\\aeronave.csv",
                          encoding = "UTF-8", sep = ";", decimal = ',')
```

- Foi Alterado a coluna 'codigo\_ocorrencia2' para 'codigo\_ocorrencia' para ficar similar o dataframe ocorrência, e mantermos a chave primária com a mesma nomenclatura, além disso foi feita a exclusão de itens nulos como foi feita na base ocorrências:

```
#Alterar a coluna 'codigo_ocorrencia2' para 'codigo_ocorrencia' para ficar similar o dataframe ocorrencia
df_aeronave.rename(columns={'codigo_ocorrencia2': 'codigo_ocorrencia'}, inplace=True)
```

- Carregamento da base de dados fator\_Contribuente:

```
#Carregar a base de dados fator_Contribuente
df_fatorContribuinte = pd.read_csv("C:\\Users\\fefaj\\OneDrive\\Área de Trabalho\\
TCC Aprovado\\Desenvolvimento\\Base de dados\\fatorcontribuinte.csv")
```

- Para tratamento dessa base foi renomeada coluna como foi feito na base aeronaves e retirados os valores duplicados pela numeração do código de ocorrência:

```
df_fatorContribuinte["codigo_ocorrencia"].duplicated().sum()
```

```
3333
```

```
df_fatorContribuinte = df_fatorContribuinte.drop_duplicates(subset=['codigo_ocorrencia'])
```

```
df_fatorContribuinte["codigo_ocorrencia"].duplicated().sum()
```

```
0
```

Conforme as bases foram sendo tratadas, elas foram adicionadas a base ocorrências pela chave primária “código de ocorrência”:

```
df_basededados = pd.merge(df_basededados, df_fatorContribuinte, on='codigo_ocorrencia', how="left")
```

- Carregamento da base de dados ocorrencia\_tipo. Após isso foram feitos os mesmos passos de tratamento das bases anteriores, como retirada de valores duplicados e renomeação da coluna código de ocorrência:

```
#Carregar a base de dados ocorrencia tipo
```

```
df_ocorrenciatipo = pd.read_csv("C:\\Users\\fefaj\\OneDrive\\Área de Trabalho\\  
TCC Aprovado\\Desenvolvimento\\Base de dados\\ocorrenciatipo.csv",  
                                encoding = "UTF-8", sep = ";", decimal = ',')
```

- Durante todo o processo de análise e exploração dos dados foram feitas consultas à base de dados, para verificar quais colunas estavam presentes e formato das mesmas:

```
df_basededados.info ()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 6114 entries, 0 to 6113
Data columns (total 35 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   codigo_ocorrendia                    6114 non-null   int64
1   ocorrencia_classificacao             6114 non-null   object
2   ocorrencia_cidade                    6114 non-null   object
3   ocorrencia_uf                        6114 non-null   object
4   ocorrencia_ano                       6114 non-null   object
5   aeronave_matricula                   6114 non-null   object
6   aeronave_operador_categoria          6114 non-null   object
7   aeronave_tipo_veiculo                6114 non-null   object
8   aeronave_fabricante                  6114 non-null   object
9   aeronave_modelo                      6114 non-null   object
10  aeronave_tipo_icao                    6114 non-null   object
11  aeronave_motor_tipo                  6111 non-null   object
12  aeronave_motor_quantidade            6114 non-null   object
13  aeronave_pmd                         6114 non-null   int64
14  aeronave_pmd_categoria               6114 non-null   int64
15  aeronave_assentos                    5937 non-null   float64
16  aeronave_ano_fabricacao              5932 non-null   float64
17  aeronave_pais_fabricante              6114 non-null   object
18  aeronave_pais_registro                6114 non-null   object
19  aeronave_registro_categoria          6114 non-null   object
```

- Inserida coluna para contagem para facilitação de geração de alguns gráficos. Esta coluna tem o intuito de contar as linhas da base de dados, o que consequentemente conta o número de ocorrências;

```
#Inserindo coluna para contagem
df_basededados['Contagem'] = 1
```

Depois do tratamento feito em separado das bases, elas foram concatenadas em uma só, denominada df\_basedados e feito um tratamento geral. Após todo o tratamento da base ficou com 1140 linhas, 34 colunas. Esta foi a base utilizada para a exploração dos dados.

- Exclusão das colunas onde o código da ocorrência é inválido e de valores nulos, pois estas informações podem impactar na análise de forma negativa:

```
#Excluir colunas onde o código da ocorrência é inválido
df_basededados['codigo_ocorrendia'].dropna(axis = 0, inplace = True)
```

```
#exclusão de valores nulos de valores nulos no dataframe
df_basededados.dropna(inplace=True)
```

#### 4. Análise e Exploração dos Dados

Para esta etapa do trabalho foram feitos alguns ajustes adicionais na base, porque na primeira tentativa de geração de gráficos foi visualizado algumas inconsistências. É possível verificar durante todo o desenvolvimento que conforme forma notadas estas inconsistências algumas alterações fora sendo executadas.

Uma delas foram a substituição de caracteres identificados nos primeiros gráficos, como "\*\*\*\*" que foi substituído por 'NÃO IDENTIFICADO'.

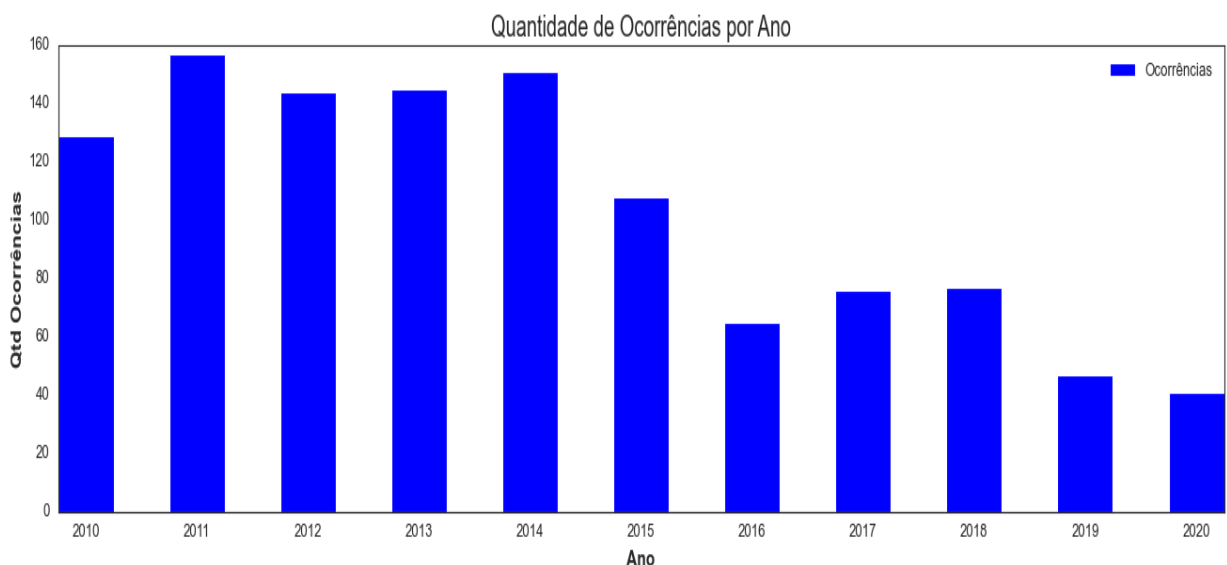
```
df_basededados = df_basededados.apply(lambda x: x.replace('****', 'NÃO IDENTIFICADO'))
df_basededados['Contagem'] = 1
```

- O primeiro objeto da análise foi a contagem de ocorrência por anos, a fim de identificar um aumento ou diminuição de acidentes ao longo do tempo:

```
#Quantidade de acidentes por ano
#Contagem das opções da coluna ano
ano_ocorrencias = Counter(df_basededados['ocorrencia_ano'])
ano_ocorrencias
```

```
Counter({'2010': 129,
        '2011': 157,
        '2012': 144,
        '2013': 145,
        '2014': 151,
        '2015': 108,
        '2016': 65,
        '2017': 76,
        '2018': 77,
        '2019': 47,
        '2020': 41})
```

E foi gerado um gráfico que exemplifica essa contagem:



O que se pode observar que a quantidade de acidente vem diminuindo consideravelmente ao longo dos anos, onde em 2019 e 2020 tivemos um dos menos índices de ocorrências. Claro que a pandemia foi um fator que pode ter afetado nesses números. Este decréscimo só poderá ser conformado após o fim da pandemia do COVID-19 quando tráfego aéreo voltará ao seu ritmo e quantidade habituais. É importante ressaltar que como este fluxo aéreo também aumenta a cada ano, deverá ser feita uma análise proporcional para comparação.

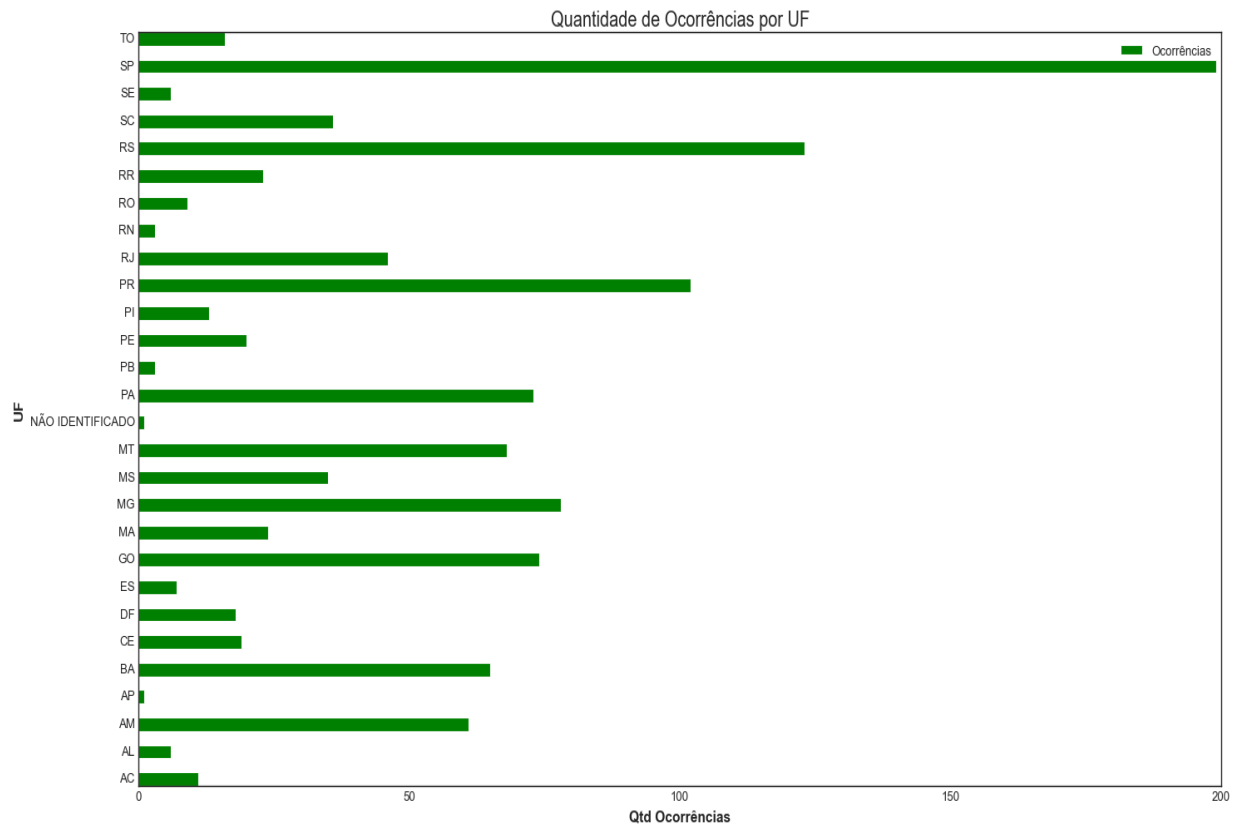
- Após a análise por anos foi feita análise por UF, a fim de identificar padrões de diferentes aeródromos, visto que cada localidade é gerida por entidades diferentes e também alguns aspectos geográficos podem influenciar nos acidentes. Outro aspecto importante é a densidade de voos de cada Estado que deve ser levado em consideração nesta análise

```
Ocorrenciaporuf = df_basededados.groupby('ocorrencia_uf')[['codigo_ocorrencia']].count ()
```

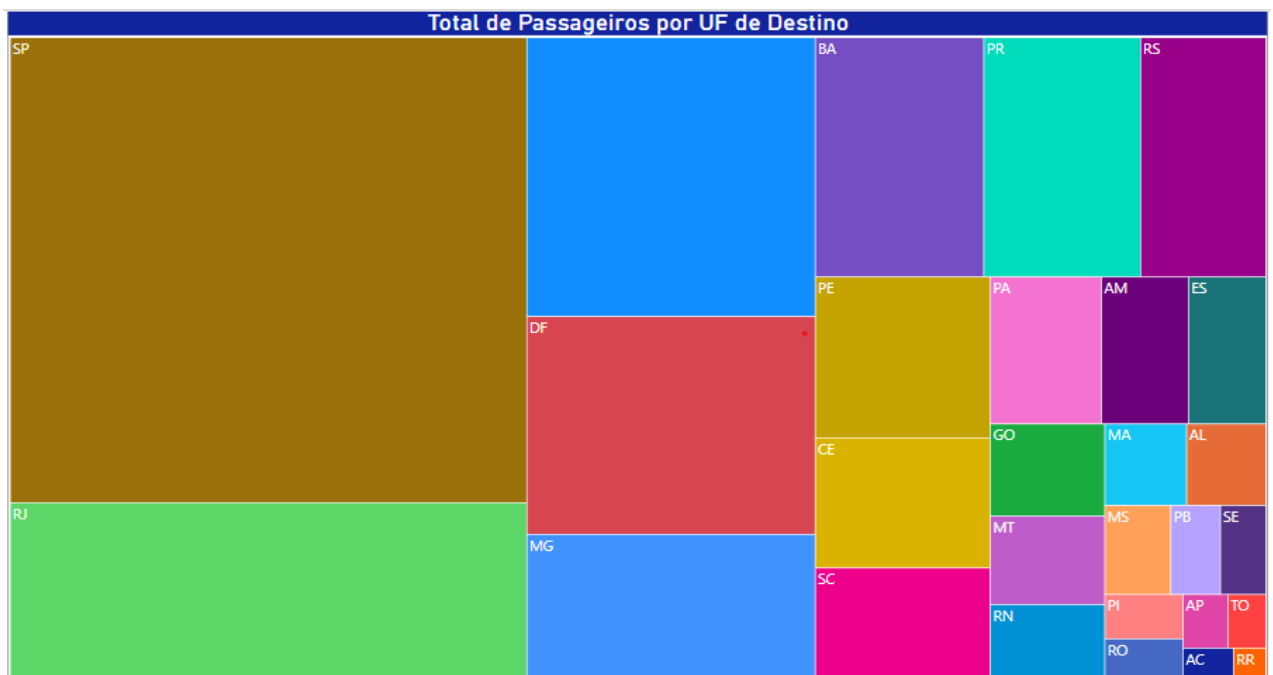
```
Ocorrenciaporuf[:30]
```

codigo_ocorrencia	
ocorrencia_uf	
AC	11
AL	6
AM	61
AP	1
BA	65
CE	19
DF	18
ES	7
GO	74
MA	24
MG	78





Como pode ser visto, o Estado de São Paulo lidera, com folga, a quantidade de ocorrência muito provavelmente devido a quantidade de decolagens e pousos de voos em seu território. Para comprovação esta hipótese foi utilizada a base da ANAC para contagem de passageiros de voos com destino a São Paulo, nos anos de 2000 a 2019:



Pode se verificar no gráfico que existem valores vazios, os quais não tem nomenclatura de UF, mas

mesmo assim é possível reafirmar a lógica de quanto mais voos maior a possibilidade de ocorrer acidentes aéreos.

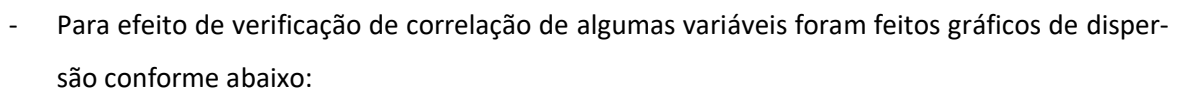
- Outro fator importante a ser considerado é que cada região tem uma predominância de Clima com suas diversidades meteorológicas e climáticas o que também pode influenciar os voos e impactando em acidentes.

Outra análise realizada foi a de ocorrências por tipo de aeronave, onde o tipo de veículo “aeronave” é o que mais apresenta incidências, já que este é o meio mais comum e frequente de voo utilizado no Brasil.

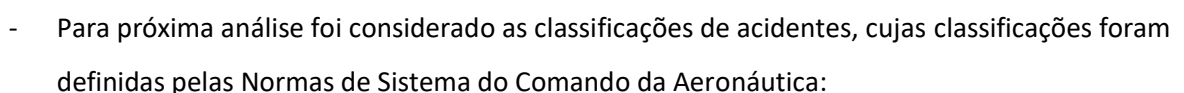
```
#Gráfico de Quantidade de ocorrências por tipo de veículo"
df_basededados.aeronave_tipo_veiculo.hist(bins=20)
plt.style.use('classic')
plt.xlabel("Tipo de Veículo")
plt.ylabel("Quantidade de ocorrências")
plt.title("Quantidade de ocorrências por tipo de veículo")
plt.show()
```



- No levantamento de ocorrências por fabricante é possível identificar que o fabricante Neiva Industria Aeronáutica obtém o maior quantitativo. Em pesquisas feitas sobre acidentes com essas aeronaves apontem falhas mecânicas no motor da aeronave, que revela a falta de combustão e problema na distribuição de lubrificante em mais de um acidente. Além disso, mais uma vez a análise pode ser influenciado pelo perímetro comparativo



```
from bokeh.sampledata.iris import flowers as dados
X = df_basededados['aeronave_tipo_veiculo']
Y = df_basededados['fator_condicionante']
plt.scatter(X,Y)
plt.show()
```



“ACIDENTE AERONÁUTICO (NSCA 3-1) É 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. • Nota-1 - Em observância ao Anexo 13 da OACI, as lesões decorrentes de um Acidente Aeronáutico que resultem em fatalidade até 30 dias da data da ocorrência são consideradas lesões fatais. • Nota-2 - Uma aeronave será considerada desaparecida quando as buscas oficiais forem encerradas e os destroços não forem encontrados.

**INCIDENTE AERONÁUTICO** • Incidente aeronáutico por definição, é toda ocorrência inclusive de tráfego aéreo associada a operação de uma aeronave, havendo intenções de voo, que não se chegue a se caracterizar como um acidente, mas que afete ou possa afetar a segurança da operação.

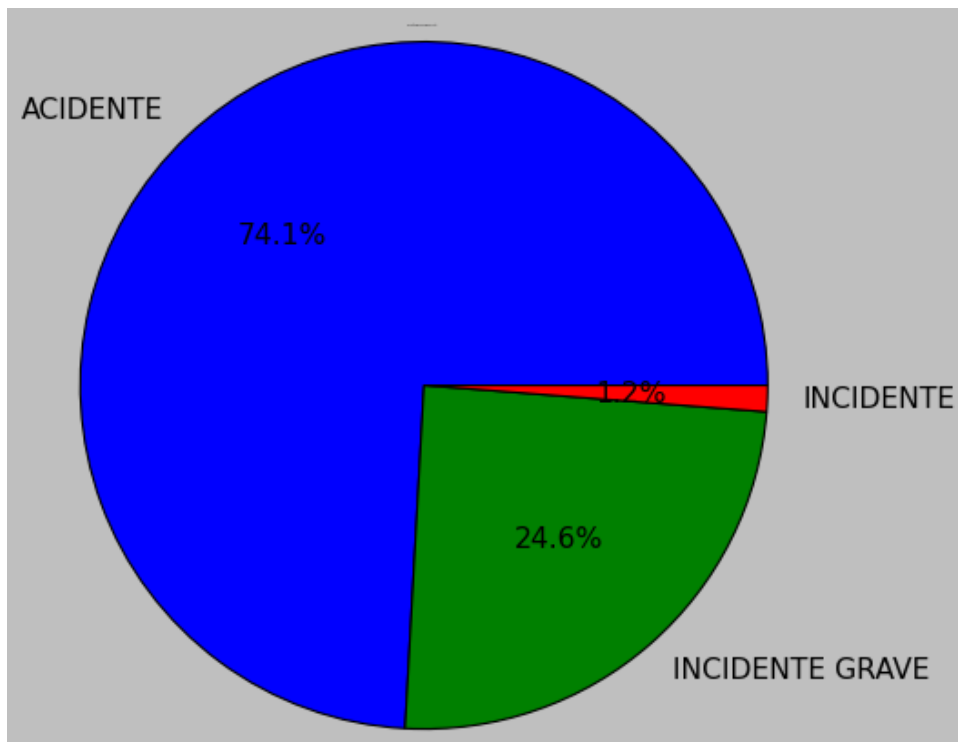
**INCIDENTE AERONÁUTICO GRAVE** Incidente ocorrido sob circunstâncias em que um acidente quase ocorreu. A diferença entre o incidente grave e o acidente está apenas nas Consequências. Dentre outras, as seguintes ocorrências caracterizam-se como incidente grave:

- a) fogo ou fumaça no compartimento de passageiros, de carga ou fogo no motor, ainda que tenha sido extinto com a utilização de extintores de incêndio;
- b) situações que exijam o uso emergencial de oxigênio por tripulante;
- c) falha estrutural da aeronave ou desintegração de motor em voo, que não configurem um acidente;
- d) quase colisão em voo que requereu a realização de uma manobra evasiva;
- e) CFIT marginalmente evitado;
- f) decolagem interrompida em pista fechada ou ocupada por outra aeronave;
- g) decolagem de pista ocupada por outra aeronave, sem separação segura; Incidente aeronáutico por definição, é toda ocorrência inclusive de tráfego aéreo associada a operação de uma aeronave,

havendo intenções de voo, que não se chegue a se caracterizar como um acidente, mas que afete ou possa afetar a segurança da operação.”

Sendo assim, os acidentes são os mais impactantes para investigação, mas os incidentes são um indicativo para possíveis futuros acidentes e frutos de recomendações.

```
plt.style.use('classic')
plt.pie(ocorrenciaclassificacao.values(), labels = ocorrenciaclassificacao.keys(),
       autopct = '%1.1f%%', textprops={'fontsize': 14})
plt.title("Ocorrências Por Classificação", fontsize=1)
plt.axis("image")
plt.show()
```



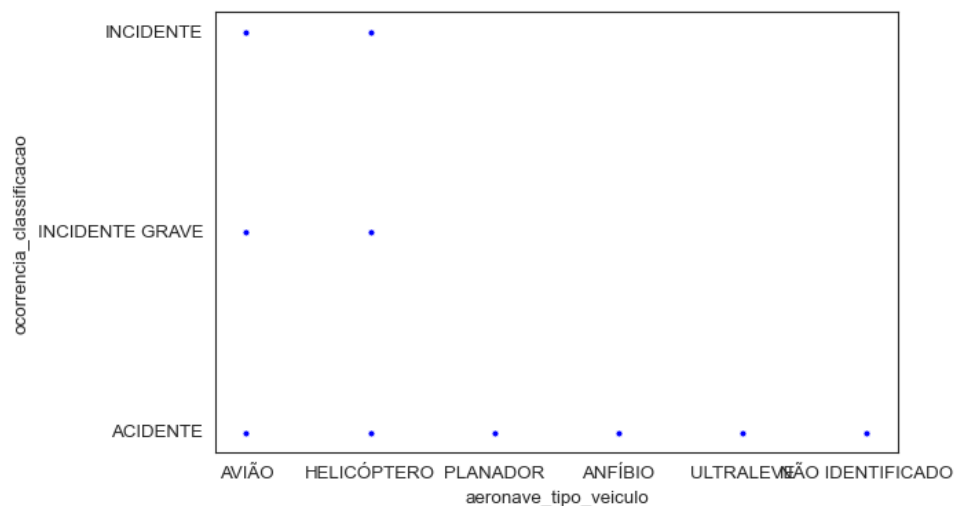
Os acidentes permeiam quase a totalidade das ocorrências.

- Mais uma vez foi utilizado o gráfico de dispersão para verificação de correlação entre fatores, neste é possível observar que para alguns tipos de aeronave, como planador e anfíbio existem somente acidentes sem incidentes ou incidentes graves

```
import seaborn as sns
import matplotlib.pyplot as plt

# Criando o ambiente do gráfico
sns.set_style("white")
plt.figure(figsize=(8, 5))

# Gráfico de Dispersão
g = sns.scatterplot(x="aeronave_tipo_veiculo", y="ocorrencia_classificacao",
                  data=df_basededados)
plt.show()
```

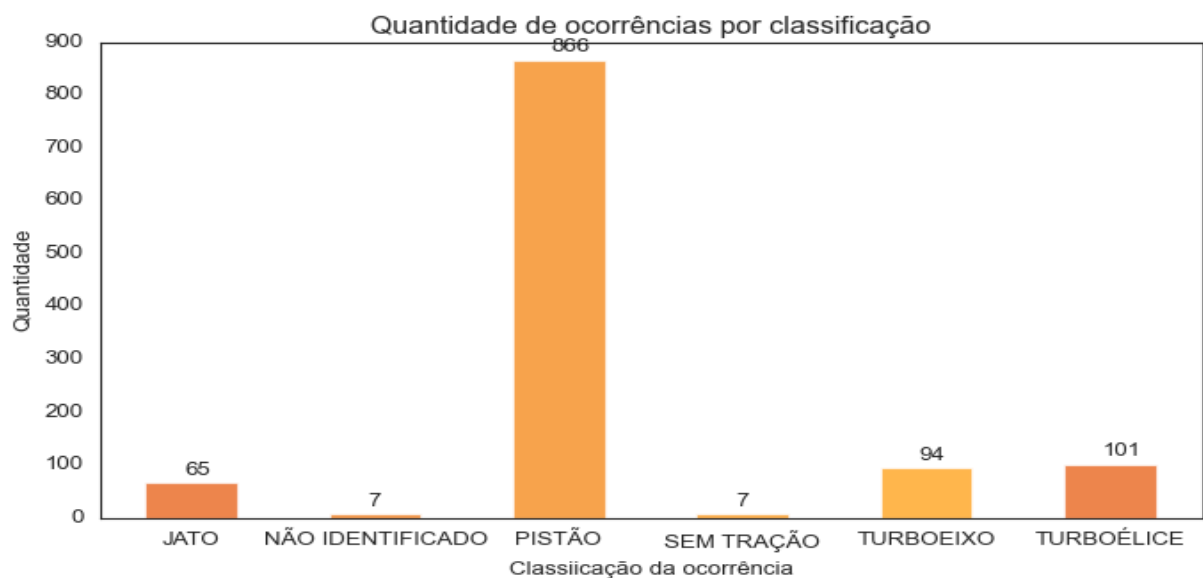


- Outro fator relevante analisado é tipo de motor da aeronave:

```
classificacao = df_basededados.groupby('aeronave_motor_tipo')['Contagem'].sum().plot(kind='bar',
    figsize=(10,5), color=random.choice(colors), alpha=.7)

for p in classificacao.patches:
    b=p.get_bbox()
    classificacao.annotate("{:.0f}".format(b.y1 + b.y0), ((b.x0 + b.x1)/2 - 0.04, b.y1 + 15))

plt.xticks(rotation=0)
plt.xlabel('Tipo do motor da Aeronave')
plt.ylabel('Quantidade')
plt.title('Quantidade de ocorrências por Tipo de Motor')
plt.show()
```



É deduzido do gráfico que os tipos de motores pistão tem elevada ocorrência de panes, em pesquisas é possível entender que esse motivo é pela ausência de sensores de temperatura na maioria das aeronaves antigas.

- Na análise feita das quantidades de motores X quantidade de ocorrências é possível visualizar que os Aviões Monomotores são os que tem mais frequência de acidentes:

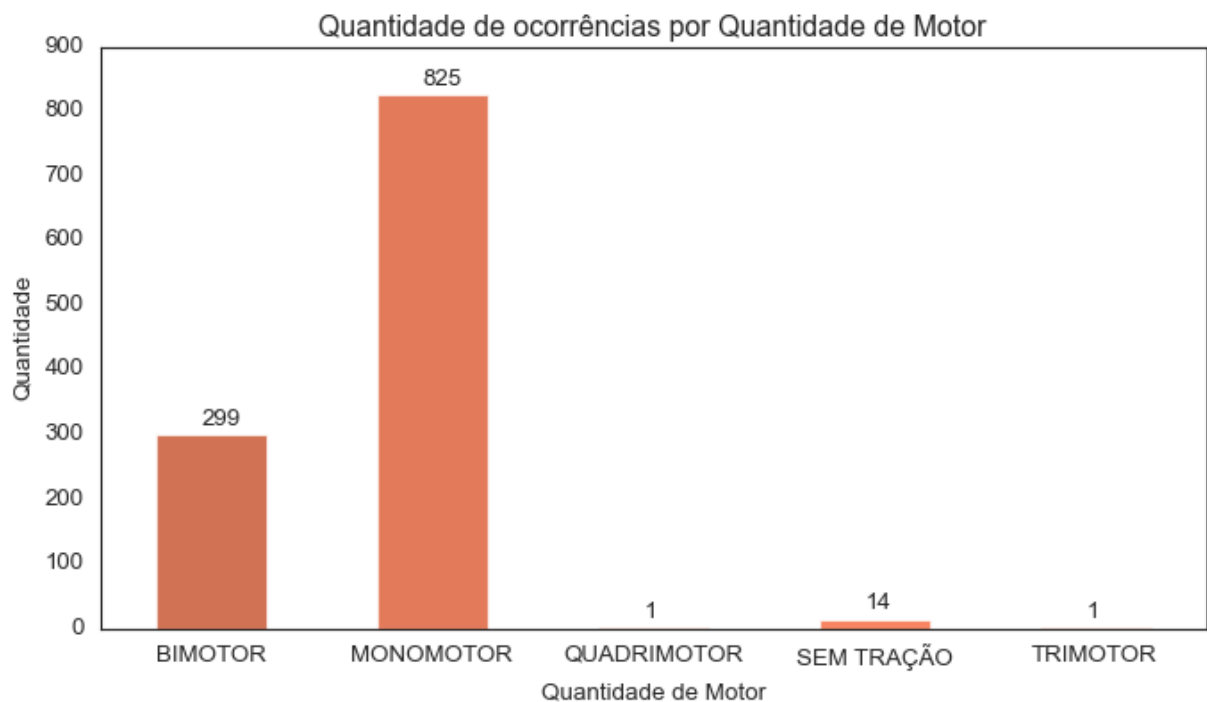
```

classificacao = df_basededados.groupby('aeronave_motor_quantidade')['Contagem'].sum().plot(kind='bar',
figsize=(10,5), color=random.choice(colors), alpha=.7)

for p in classificacao.patches:
    b=p.get_bbox()
    classificacao.annotate("{:.0f}".format(b.y1 + b.y0), ((b.x0 + b.x1)/2 - 0.04, b.y1 + 15))

plt.xticks(rotation=0)
plt.xlabel('Quantidade de Motor')
plt.ylabel('Quantidade')
plt.title('Quantidade de ocorrências por Quantidade de Motor')
plt.show()

```



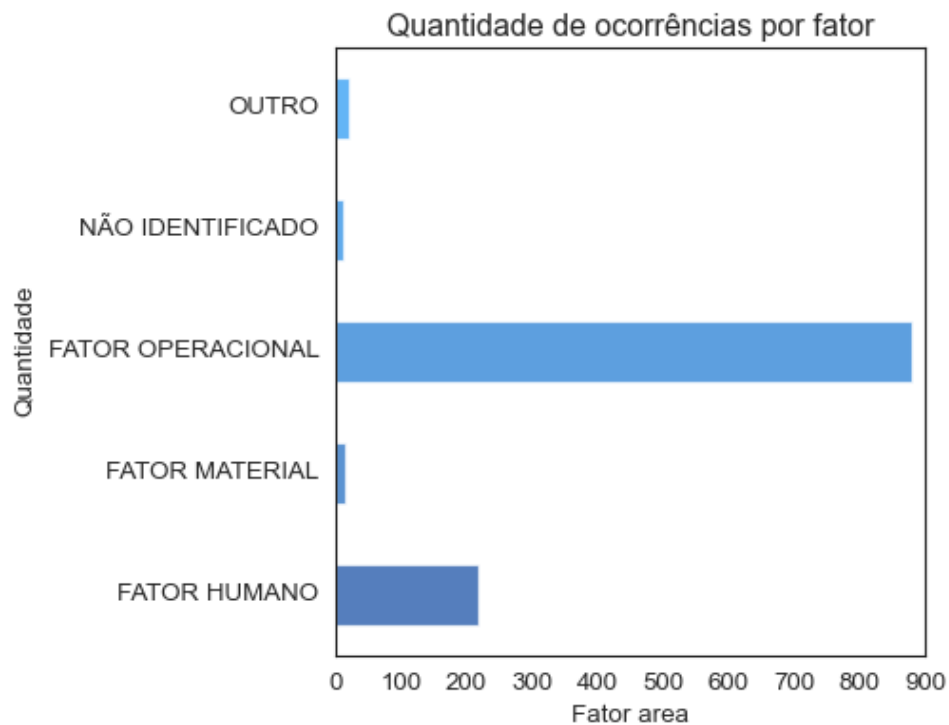
É muito difícil argumentar e provar estatisticamente que aviões bimotores são mais seguros que monomotor, ou vice e versa. Um documento produzido pela Associação de Operadores e Pilotos de **aeronaves** (AOPA), indicou que os números de acidentes com aeronaves monomotores eram superiores aos bimotores, pelo fato de que são a maioria. Por sua vez, para cada dez acidentes, apenas um gerava fatalidade. Enquanto as aeronaves bimotoras causam a morte de 50% dos acidentes, embora sejam em menor quantidade, lamentavelmente as falhas de motor em voo contribuíram em 40% dos acidentes no Brasil.

- Passando-se para análise de aspectos além das aeronaves, foi feita estratificação desses fatores e conclui-se que os acidentes ocorrem em sua maioria devido a fatores operacionais, que diz respeito a relação homem-máquina e ao seu desempenho durante a atividade de voo, manutenção, planejamento. Os demais fatores estão explicados na figura do método investigativo da SAIPHER.

```
fatorarea = df_basededados.groupby('fator_area')['Contagem'].sum().plot(kind='barh',
figsize=(5,5), color=random.choice(colors), alpha=.7)

for p in fatorarea .patches:
    b=p.get_bbox()

plt.xticks(rotation=0)
plt.xlabel('Fator area')
plt.ylabel('Quantidade')
plt.title('Quantidade de ocorrências por fator')
plt.show()
```





## 5. Criação de Modelos de Machine Learning

O objetivo da criação dos modelos de Machine Learning é correlacionar os fatores analisados com as categorias de ocorrências (incidente, acidente e incidente grave) optou-se por utilizar algoritmos de classificação, que são cálculos preditivos usados para atribuir dados a categorias predefinidas, analisando um conjunto de dados de treinamento, O que se encaixa perfeitamente a análise pois temos 3 classificações associadas a diversas características/variáveis de observações passadas. A classificação é uma sub-categoria de aprendizagem supervisionada. Classificação é o processo de tomar algum tipo de entrada e atribuir um rótulo a ela.

A aprendizagem supervisionada é útil nos casos em que uma propriedade (rótulo) está disponível para um determinado conjunto de dados (conjunto de treinamento). Ou seja, os dados de entradas e saídas são vetores conhecidos, por consequência, o algoritmo define como uma forma de prever o rótulo de saída com base na entrada das informações declaradas.

Nesse trabalho foram utilizados 5 tipos de algoritmos de classificação, são eles: Árvore de Decisão, Regressão Logística, Naïve Bayes, Gradiente Descendente, KNN (K - Nearest Neighbors) e Randon Forest.

Antes de criação dos modelos, foram necessárias algumas adequações na base de dados. A primeira destas foi a delimitação de colunas que correspondem a fatores associados, foi criado um novo dataframe para este novo perímetro. É evidente que quanto mais fatores maior e aceitabilidade do modelo, mas para este estudo foram considerados somente alguns deles, conforme abaixo:

```
#Escolha dos atributos para um novo dataset dos modelos de ,machine learninh
df_set = df_basededados[['ocorrencia_classificacao',
                          'aeronave_tipo_veiculo',
                          'fator_aspecto',
                          'aeronave_motor_quantidade', 'fator_area', 'fator_condicionante']].copy()

df_set.head()
```

	ocorrencia_classificacao	aeronave_tipo_veiculo	fator_aspecto	aeronave_motor_quantidade	fator_area	fator_condicionante
3	ACIDENTE	AVIÃO	DESEMPENHO DO SER HUMANO	MONOMOTOR	FATOR OPERACIONAL	OPERAÇÃO DA AERONAVE
7	INCIDENTE GRAVE	AVIÃO	DESEMPENHO DO SER HUMANO	BIMOTOR	FATOR OPERACIONAL	OPERAÇÃO DA AERONAVE
8	INCIDENTE GRAVE	AVIÃO	DESEMPENHO DO SER HUMANO	BIMOTOR	FATOR OPERACIONAL	MANUTENÇÃO DA AERONAVE
11	ACIDENTE	HELICÓPTERO	DESEMPENHO DO SER HUMANO	MONOMOTOR	FATOR OPERACIONAL	OPERAÇÃO DA AERONAVE
13	ACIDENTE	AVIÃO	DESEMPENHO DO SER HUMANO	MONOMOTOR	FATOR OPERACIONAL	OPERAÇÃO DA AERONAVE

- Outra adequação foi a exclusão de linhas com valores "NÃO IDENTIFICADO". Estes valores para análise de dados não foram tão impactantes, mas para desenvolvimento do

modelo de Machine learning poderia impactar por se tratar de uma string de difícil conversão para valores binários.

```
#retirandoos valoresde linha com NÃO IDENTIFICADO
filtro1 = df_set [df_set['ocorrencia_classificacao'] == 'NÃO IDENTIFICADO' ].index
df_set.drop(filtro1 , inplace=True)
```

```
#retirandoos valoresde linha com NÃO IDENTIFICADO
filtro2 = df_set [df_set['aeronave_tipo_veiculo'] == 'NÃO IDENTIFICADO' ].index
df_set.drop(filtro2 , inplace=True)
```

```
#retirandoos valoresde linha com NÃO IDENTIFICADO
filtro3 = df_set [df_set['fator_aspecto'] == 'NÃO IDENTIFICADO' ].index
df_set.drop(filtro3 , inplace=True)
```

```
#retirandoos valoresde linha com NÃO IDENTIFICADO
filtro4 = df_set [df_set['fator_area'] == 'NÃO IDENTIFICADO' ].index
df_set.drop(filtro4 , inplace=True)
```

```
#retirandoos valoresde linha com NÃO IDENTIFICADO
filtro5 = df_set [df_set['fator_condicionante'] == 'NÃO IDENTIFICADO' ].index
df_set.drop(filtro5 , inplace=True)
```

Os modelos de machine learning trabalham basicamente com a Matemática, por isso, não existe a possibilidade de o modelo trabalhar diretamente com dados categóricos em formato de string (texto), como estão na base de dados, é necessário codificar. Codificação, é a transformação de variáveis categóricas em numéricas (ou binárias). Por isso foi feita a transformação destes dados não numéricos em dados numéricos

```
#Identificação dos valores da coluna aeronave_tipo_veiculo
df_set['aeronave_tipo_veiculo'].unique()
```

```
array(['AVIÃO', 'HELICÓPTERO', 'PLANADOR', 'ANFÍBIO', 'ULTRALEVE'],
      dtype=object)
```

```
#Transformação de valores categóricos em valores inteiros coluna aeronave_tipo_veiculo
ajuste_aeronave_tipo_veiculo = {'AVIÃO': 0, 'HELICÓPTERO': 1, 'PLANADOR': 2, 'ANFÍBIO': 3, 'ULTRALEVE':4}
df_set['aeronave_tipo_veiculo'] = df_set['aeronave_tipo_veiculo'].map(ajuste_aeronave_tipo_veiculo )
```

```
#Identificação dos valores da coluna fator_aspecto
df_set['fator_aspecto'].unique()
```

```
array(['DESEMPENHO DO SER HUMANO', 'ASPECTO PSICOLÓGICO'], dtype=object)
```

```
#Transformação de valores categóricos em valores inteiros coluna fator_aspecto
ajuste_fator_aspecto = {'DESEMPENHO DO SER HUMANO': 0, 'ASPECTO PSICOLÓGICO': 1,
                        'ASPECTO MÉDICO': 2, 'ASPECTO DE FABRICAÇÃO': 3,
                        'ELEMENTOS RELACIONADOS AO AMBIENTE OPERACIONAL': 4, 'OUTRO': 5,
                        'INFRAESTRUTURA AEROPORTUÁRIA': 6, 'ERGONOMIA': 7, 'ASPECTO DE PROJETO': 8,
                        'ASPECTO DE MANUSEIO DO MATERIAL': 9}

df_set['fator_aspecto'] = df_set['fator_aspecto'].map(ajuste_fator_aspecto)
```

```
#Identificação dos valores da coluna aeronave_motor_quantidade
df_set['aeronave_motor_quantidade'].unique()
```

```
array(['MONOMOTOR', 'BIMOTOR', 'SEM TRACÇÃO', 'QUADRIMOTOR', 'TRIMOTOR'],
      dtype=object)
```

```
#Transformação de valores categóricos em valores inteiros coluna aeronave_motor_quantidade
ajuste_aeronave_motor_quantidade = {'MONOMOTOR': 0, 'BIMOTOR': 1, 'SEM TRACÇÃO': 2, 'QUADRIMOTOR': 3, 'TRIMOTOR': 4}
df_set['aeronave_motor_quantidade'] = df_set['aeronave_motor_quantidade'].map(ajuste_aeronave_motor_quantidade)
```

```
#Transformação de valores categóricos em valores inteiros coluna fator_area
ajuste_fator_area = {'FATOR OPERACIONAL': 0, 'FATOR HUMANO': 1, 'FATOR MATERIAL': 2, 'OUTRO': 3 }
df_set['fator_area'] = df_set['fator_area'].map(ajuste_fator_area)

#Identificação dos valores da coluna fator_condicionante
df_set['fator_condicionante'].unique()

array(['OPERAÇÃO DA AERONAVE', 'MANUTENÇÃO DA AERONAVE', 'INDIVIDUAL',
      'PSICOSSOCIAL', 'ORGANIZACIONAL',
      'PRESTAÇÃO DE SERVIÇOS DE TRÁFEGO AÉREO'], dtype=object)

#Identificação dos valores da coluna fator_condicionante
df_set['fator_condicionante'].unique()

array(['OPERAÇÃO DA AERONAVE', 'MANUTENÇÃO DA AERONAVE', 'INDIVIDUAL',
      'PSICOSSOCIAL', 'ORGANIZACIONAL',
      'PRESTAÇÃO DE SERVIÇOS DE TRÁFEGO AÉREO'], dtype=object)

#Transformação de valores categóricos em valores inteiros coluna fator_condicionante
ajuste_fator_condicionante = {'OPERAÇÃO DA AERONAVE': 0, 'MANUTENÇÃO DA AERONAVE': 1, 'INDIVIDUAL': 2,
                              'PSICOSSOCIAL': 3, 'ORGANIZACIONAL': 4,
                              'PRESTAÇÃO DE SERVIÇOS DE TRÁFEGO AÉREO': 5}
df_set['fator_condicionante'] = df_set['fator_condicionante'].map(ajuste_fator_condicionante)
```

O dataframe será dividido em dois: um onde consta apenas os atributos que serão analisados e outra com a categorização do tipo de ocorrência. O dataframe que contém todos os atributos será chamado de `X_train` e o que contém o resultado será chamado de `y_train`.

```
#Divisão para as bases de treinamento
X_train = df_set.drop(['ocorrencia_classificacao'], axis = 1)
y_train = df_set.ocorrencia_classificacao
```

```
X_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1012 entries, 3 to 5709
Data columns (total 5 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   aeronave_tipo_veiculo                 1012 non-null   int64
1   fator_aspecto                         1012 non-null   int64
2   aeronave_motor_quantidade             1012 non-null   int64
3   fator_area                           1012 non-null   int64
4   fator_condicionante                   1012 non-null   int64
dtypes: int64(5)
memory usage: 47.4 KB
```

```
#Tipo da serie y_train
type(y_train)
```

```
pandas.core.series.Series
```

Após a divisão entre bases de treinamento e de teste utilizando a função “`train_test_split`”. Para a divisão de percentual de dados para treinamento e para teste, será utilizado o padrão da função, que é de 75% para treinamento e 25% para teste.

```
#Criação das bases de teste e treinamento
#75% para treinamento e 25% para teste
xtreinamento, xteste, ytreinamento, yteste = train_test_split(X_train, y_train, random_state = 0)
```

```
#Informação do dataframe de treinamento
xtreinamento.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 759 entries, 2869 to 3174
Data columns (total 5 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   aeronave_tipo_veiculo                 759 non-null    int64
1   fator_aspecto                        759 non-null    int64
2   aeronave_motor_quantidade            759 non-null    int64
3   fator_area                           759 non-null    int64
4   fator_condicionante                  759 non-null    int64
dtypes: int64(5)
memory usage: 35.6 KB
```

```
#Informação do dataframe de teste
xteste.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 253 entries, 3343 to 1835
Data columns (total 5 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   aeronave_tipo_veiculo                 253 non-null    int64
1   fator_aspecto                        253 non-null    int64
2   aeronave_motor_quantidade            253 non-null    int64
3   fator_area                           253 non-null    int64
4   fator_condicionante                  253 non-null    int64
dtypes: int64(5)
memory usage: 11.9 KB
```

```
print (ytreinamento)
```

```
2869      ACIDENTE
8      INCIDENTE GRAVE
2313      ACIDENTE
4793      INCIDENTE GRAVE
1372      ACIDENTE
...
3973      ACIDENTE
934      INCIDENTE GRAVE
2906      ACIDENTE
2606      ACIDENTE
3174      INCIDENTE GRAVE
Name: ocorrencia_classificacao, Length: 759, dtype: object
```

O Modelo de Machine Learning pode ser avaliado pelas seguintes métricas de avaliação do modelo são:

**Precisão** - Pode-se definir que foi a capacidade de evitar falsos positivos na segmentação da nuvem de pontos.

**Revocação (Recall)** - É a proporção entre as segmentações corretas e o total de segmentações realizadas. Essa métrica indica a quão boa foi a segmentação na escolha dos pontos corretos da nuvem, ou seja, os pontos que deveriam ser VP foram rotulados como tal assim como os VN.

**Acurácia (Accuracy)**

Corresponde a proporção entre os pontos segmentados corretamente, sendo eles regiões de interesse, com a soma destes mais os pontos definidos como falso positivos e falsos negativos.

**Medida F (F Measure, F1)**

É a média harmônica entre precisão e revocação. Uma vez que seu valor está alto significa que a acurácia que obtivemos é relevante, ou seja, os valores de VP, VN, FP, FN aferidos não apresentam grandes distorções. Também pode-se interpretar como uma medida de confiabilidade da acurácia.

A elaboração para todos será semelhante, começando pela importação do respectivo algoritmo, realizando o treinamento por meio da função “fit” nas duas bases de treinamento e registrando sua acurácia por meio da função score. Em seguida, será utilizada a função “predict”, que retorna uma matriz de previsões para cada instância de dados no conjunto de testes na  $x_{teste}$  e sua saída será comparada com a série  $y_{teste}$ , tendo suas medidas de avaliação sendo geradas por meio das funções “accuracy\_score” e “classification\_report”.

#### 1. Árvore de decisão

A vantagem principal das Árvores de Decisão é a tomada de decisões levando em consideração as variáveis mais relevantes, além de uma melhor compreensão para a maioria das pessoas e a grande desvantagem é ser muito simples para relações complexas, como utilizamos poucos

fatores para classificação final, este foi um dos modelos escolhidos para realização do estudo

```
#Criação do modelo utilizando a Árvore de decisão
#xtreinamento, xteste, ytreinamento, yteste
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
ocorrencia_classificacao_tree = DecisionTreeClassifier(random_state=0)
ocorrencia_classificacao_tree = ocorrencia_classificacao_tree.fit(xtreinamento, ytreinamento)
print("Acurácia: ", ocorrencia_classificacao_tree.score(xtreinamento, ytreinamento))
Train_predict = ocorrencia_classificacao_tree.predict(xteste)
print("Acurácia de previsão: ", accuracy_score(yteste, Train_predict))
print(classification_report(yteste, Train_predict))
```

```
Acurácia: 0.7720685111989459
Acurácia de previsão: 0.7351778656126482
```

	precision	recall	f1-score	support
ACIDENTE	0.76	0.98	0.86	185
INCIDENTE	0.00	0.00	0.00	3
INCIDENTE GRAVE	0.42	0.08	0.13	65
accuracy			0.74	253
macro avg	0.39	0.35	0.33	253
weighted avg	0.66	0.74	0.66	253

## 2. Naïve Bayes

Este algoritmo é usado principalmente em classificação de texto e com os problemas que têm múltiplas classes. Bayes realiza uma classificação probabilística de observações, caracterizando-as em classes pré-definidas. A resolução de problemas relacionados a texto é muito bem resolvida com a utilização do Naive Bayes. Classificação de textos, filtragem de SPAM e análise de sentimento em redes sociais são algumas das muitas aplicações para o algoritmo. Além disso, o algoritmo é muito robusto para previsões em tempo real, ainda mais por precisar de poucos dados para realizar a classificação. Entretanto, caso haja necessidade de correlacionar fatores, o Naive Bayes tende a falhar na predição. Fato este que pode ser comprovado comparando-se o método utilizado anterior, em que a acurácia do método árvore de decisão foi superior para os dados desta análise.

```
#Criação do modelo utilizando Naïve Bayes
from sklearn.naive_bayes import GaussianNB
nb = GaussianNB()
nb = nb.fit(xtreinamento, ytreinamento)
print("Acurácia: ", nb.score(xtreinamento, ytreinamento))
tp_nb = nb.predict(xteste)
print("Acurácia de previsão: ", accuracy_score(yteste, tp_nb))
print(classification_report(yteste, tp_nb))
```

Acurácia: 0.45586297760210803

Acurácia de previsão: 0.4505928853754941

	precision	recall	f1-score	support
ACIDENTE	0.82	0.34	0.48	185
INCIDENTE	0.00	0.00	0.00	3
INCIDENTE GRAVE	0.29	0.78	0.43	65
accuracy			0.45	253
macro avg	0.37	0.38	0.30	253
weighted avg	0.67	0.45	0.46	253

### 3. Gradiente Descendente

```
#Criação do modelo utilizando Gradiente Descendente
from sklearn.linear_model import SGDClassifier
sgd = SGDClassifier()
sgd = sgd.fit(xtreinamento, ytreinamento)
print("Acurácia: ", sgd.score(xtreinamento, ytreinamento))
tp_sgd = sgd.predict(xteste)
print("Acurácia de previsão: ", accuracy_score(yteste, tp_sgd))
print(classification_report(yteste, tp_sgd))
```

Acurácia: 0.7523056653491436

Acurácia de previsão: 0.7351778656126482

	precision	recall	f1-score	support
ACIDENTE	0.73	1.00	0.85	185
INCIDENTE	0.00	0.00	0.00	3
INCIDENTE GRAVE	1.00	0.02	0.03	65
accuracy			0.74	253
macro avg	0.58	0.34	0.29	253
weighted avg	0.79	0.74	0.63	253

Em termos simples, o gradiente descendente estocástico não usa todos os dados, mas sim uma fração dos dados. Esse subconjunto dos dados é obtido com amostragem dos dados originais. A amostragem é o componente estocástico do algoritmo. Com isso, o traço da função objetivo deixa de ser suave, o que impossibilita o uso de critérios de parada baseados em diferenças de valores consecutivos. Então o mais comum é executar o algoritmo até exceder o número máximo de iterações. O número máximo de iterações deve ser escolhido de forma a garantir suficiente proximidade com o ponto de ótimo da função. Diante das medidas de avaliação deste modelo, a árvore de decisão ainda se mostra mais adequada para análise.

#### 4. KNN (K - Nearest Neighbors)

Este método compara os valores de output com os demais para determinar quais são semelhantes e quais são diferentes, funciona bem para valores numéricos e categóricos, é recomendado para bases de dados não muito grandes, que é o caso deste estudo.

```
#Criação do modelo utilizando KNN (K - Nearest Neighbors)
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()
knn = knn.fit(xtreinamento, ytreinamento)
print("Acurácia: ", knn.score(xtreinamento, ytreinamento))
tp_knn = knn.predict(xteste)
print("Acurácia de previsão: ", accuracy_score(yteste, tp_knn))
print(classification_report(yteste, tp_knn))
```

Acurácia: 0.758893280632411

Acurácia de previsão: 0.7351778656126482

	precision	recall	f1-score	support
ACIDENTE	0.74	0.99	0.85	185
INCIDENTE	0.00	0.00	0.00	3
INCIDENTE GRAVE	0.50	0.03	0.06	65
accuracy			0.74	253
macro avg	0.41	0.34	0.30	253
weighted avg	0.67	0.74	0.63	253

#### 5. Random Forest

Em português, Random Forest significa floresta aleatória. Este nome explica muito bem o funcionamento do algoritmo, que irá criar muitas árvores de decisão, de maneira aleatória, formando o que podemos enxergar como uma floresta, onde cada árvore será utilizada na escolha do resultado final. Tende a apresentar modelos de alta qualidade, motivo pelo qual foi escolhido. Pode se observar que as métricas e valores foram muito próximos ao método de árvore de decisão.



```
#Criação do modelo utilizando Random Forest
from sklearn.ensemble import RandomForestClassifier
rfm = RandomForestClassifier()
rfm = rfm.fit(xtreinamento, ytreinamento)
print("Acurácia: ", rfm.score(xtreinamento, ytreinamento))
tp_rfm = rfm.predict(xteste)
print("Acurácia de previsão: ", accuracy_score(yteste, tp_rfm))
print(classification_report(yteste, tp_rfm))
```

Acurácia: 0.7720685111989459

Acurácia de previsão: 0.7430830039525692

	precision	recall	f1-score	support
ACIDENTE	0.76	0.98	0.86	185
INCIDENTE	0.00	0.00	0.00	3
INCIDENTE GRAVE	0.50	0.11	0.18	65
accuracy			0.74	253
macro avg	0.42	0.36	0.34	253
weighted avg	0.68	0.74	0.67	253

## 6. Interpretação dos Resultados

Algoritmo	Acurácia de previsão	Precisão	Revocação	F1-Score
Árvore de Decisão	0,77	0,76	0,98	0,86
Naïve Bayes	0,46	0,82	0,34	0,48
Gradiente Descendente	0,75	0,73	1	0,85
KNN	0,75	0,74	0,99	0,85
Randon Forest	0,77	0,76	0,98	0,86

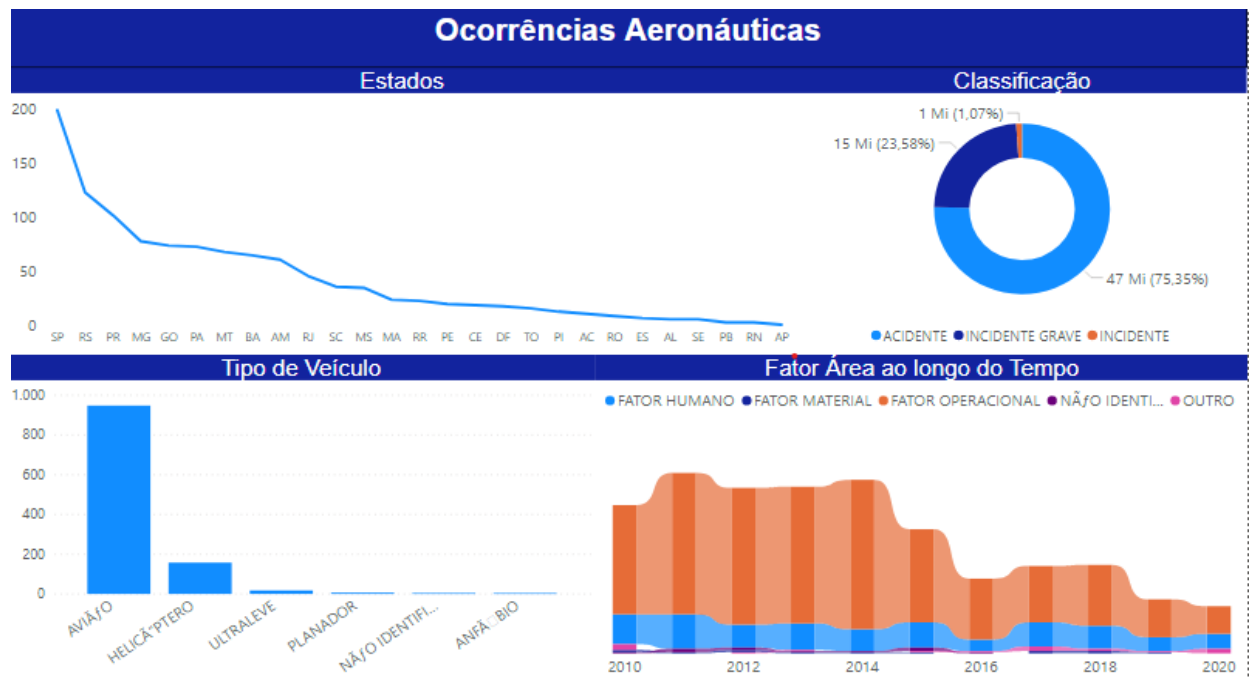
Como interpretação do resultado baseado nos valores apresentados com os modelos de machine Learning, é possível verificar que os modelos com a melhor performance para estes dados foram a árvore de decisão e o Randon Forest.

Considerando os resultados obtidos com a árvore de decisão, a probabilidade de ocorrência do tipo acidente é de 76 %. E analisando a acurácia de previsão, temos o valor de 0,77 ou seja, de todas as amostras, o algoritmo acerta em 77% das vezes, afirmando que na maioria das vezes com dados informados a ocorrência será do tipo acidente.

Com isso, pode-se concluir que a análise de dados do histórico de acidentes é fundamentalmente importante para recomendações de segurança específicas e objetivas para os a fim de corrigir as deficiências detectadas. Estes dados servem de embasamento para as recomendações futuras que contribuem para evitar a recorrência de casos.

Em se tratando de aviação existem diversos fatores que podem afetar o sucesso do voo, sendo a análise de fatos passados uma ferramenta muito útil.

## 7. Apresentação dos Resultados



## 8. Links

Link para o vídeo: <https://youtu.be/p4VXCWqG1KM>

Link para o repositório: <https://github.com/Fernandassv/TCC-PUC-Minas>

## APÊNDICE

```
#1.Processamento de tratamento de dados #Importação da biblioteca pandas
import pandas as pd
import numpy as np
import datetime
import matplotlib.pyplot as plt
from sklearn.datasets import make_classification
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
import scipy.stats as stats
import random
#Carregamento do pacote Sklearn
import sklearn.metrics as m
from sklearn import preprocessing
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import RandomizedSearchCV
```

In [135]:

```
#1.3 #Carregar a base de dados ocorrencia #leitura do arquivo
ocorrencia.csv
df_ocorrencia = pd.read_csv("C:\\Users\\fefaj\\OneDrive\\Área de
Trabalho\\TCC Aprovado\\Desenvolvimento\\Base de dados\\ocorrencia.csv",
encoding = "UTF-8", sep = ";", decimal = ',')
```

In [136]:

```
df_ocorrencia.info()
<class 'pandas.core.frame.DataFrame'>RangeIndex: 6114 entries, 0 to 6113
Data columns (total 22 columns): # Column Non-Null Count
Dtype ---
codigo_ocorrencia 6114 non-null int64 1
codigo_ocorrencia1 6114 non-null
int64 2
codigo_ocorrencia2 6114 non-null int64 4
codigo_ocorrencia3 6114 non-null int64 5
codigo_ocorrencia4 6114 non-null object 6
ocorrencia_classificacao 6114 non-null object 7
ocorrencia_latitude 4552 non-null
object 8
ocorrencia_longitude 6114 non-null object 9
ocorrencia_cidade 6114 non-null object 10
ocorrencia_uf 6114 non-null object 11
ocorrencia_pais 6114 non-null
object 12
ocorrencia_aerodromo 6114 non-null object 13
ocorrencia_dia 6114 non-null object 14
ocorrencia_hora 5827 non-null object 15
investigacao_aeronave_liberada 5773 non-null object 16
investigacao_status 5319 non-null
object 17
divulgacao_relatorio_numero 6114 non-null object 18
divulgacao_relatorio_publicado 1577 non-null object 19
total_recomendacoes 6114 non-null
int64 20
total_aeronaves_envolvidas 6114 non-null
int64 21
ocorrencia_saida_pista 6114 non-null
object
dtypes:
int64(7), object(15)
memory usage: 1.0+ MB
```

In [137]:

```
#Exclusão das colunas que não serão utilizadas
df_ocorrencia = df_ocorrencia.drop(columns=['codigo_ocorrencia1', 'codigo_ocorrencia2', 'codigo_ocorrencia3', 'codigo_ocorrencia4', 'ocorrencia_latitude', 'ocorrencia_longitude', 'ocorrencia_pais', 'ocorrencia_aerodromo', 'ocorrencia_hora', 'investigacao_aeronave_liberada', 'investigacao_status', 'divulgacao_relatorio_numero', 'divulgacao'])
```

```
_relatorio_publicado', 'divulgacao_dia_publicacao', 'total_recomendacoes', 'total_aeronaves_envolvidas', 'ocorrencia_saida_pista']])
```

In [138]:

```
#Verificar os valores nulos df_ocorrencia.isnull().sum()
```

Out[138]:

```
codigo_ocorrencia      0 ocorrencia_classificacao      0 ocorrencia_cidade
0 ocorrencia_uf          0 ocorrencia_dia              0 dtype: int64
```

In [139]:

```
#verificar o tipos das colunas df_ocorrencia.dtypes
```

Out[139]:

```
codigo_ocorrencia      int64 ocorrencia_classificacao
object ocorrencia_cidade      object ocorrencia_uf
object ocorrencia_dia          object dtype: object
```

In [140]:

```
#converter a coluna ocorrencia_dia para data df_ocorrencia['ocorrencia_dia'] =
pd.DataFrame({'ocorrencia_dia':
pd.to_datetime(df_ocorrencia['ocorrencia_dia'])})
```

In [141]:

```
#verificar o tipos das colunas df_ocorrencia.dtypes
```

Out[141]:

```
codigo_ocorrencia      int64 ocorrencia_classificacao
object ocorrencia_cidade      object ocorrencia_uf
object ocorrencia_dia      datetime64[ns] dtype: object
```

In [142]:

```
#Transformado a coluna ocorrencia_dia para o ano da
célula df_ocorrencia['ocorrencia_dia']
df_ocorrencia['ocorrencia_dia'].dt.year
```

In [143]:

```
#Alterar a coluna 'ocorrencia_dia' para 'ocorrencia_ano'
df_ocorrencia.rename(columns={'ocorrencia_dia': 'ocorrencia_ano'},
inplace=True)
```

In [144]:

```
df_ocorrencia['ocorrencia_ano'] = df_ocorrencia['ocorrencia_ano'].astype(str)
```

In [145]:

```
#Carregar a base de dados aeronave df_aeronave =
pd.read_csv("C:\\Users\\fefaj\\OneDrive\\Área de Trabalho\\TCC
Aprovado\\Desenvolvimento\\Base de dados\\aeronave.csv",
encoding = "UTF-8", sep = ";", decimal = ',')
```

In [146]:

```
df_aeronave.info()
<class 'pandas.core.frame.DataFrame'> RangeIndex: 6188 entries, 0 to 6187 Data
columns (total 23 columns): # Column Non-Null Count
Dtype ---
codigo_ocorrencia2 6188 non-null int64 1 aeronave_matricula
6188 non-null object 2 aeronave_operador_categoria 6188 non-null
object 3 aeronave_tipo_veiculo 6188 non-null object 4
```

```

aeronave_fabricante      6188 non-null   object    5   aeronave_modelo
6188 non-null   object    6   aeronave_tipo_icao      6188 non-null
object    7   aeronave_motor_tipo      6185 non-null   object    8
aeronave_motor_quantidade  6188 non-null   object    9   aeronave_pmd
6188 non-null   int64    10  aeronave_pmd_categoria  6188 non-null   int64
11  aeronave_assentos      6010 non-null   float64   12
aeronave_ano_fabricacao   6005 non-null   float64   13
aeronave_pais_fabricante  6188 non-null   object    14
aeronave_pais_registro    6188 non-null   object    15
aeronave_registro_categoria 6188 non-null   object    16
aeronave_registro_segmento 6188 non-null   object    17  aeronave_voo_origem
6187 non-null   object    18  aeronave_voo_destino    6187 non-null
object    19  aeronave_fase_operacao  6188 non-null   object    20
aeronave_tipo_operacao    6188 non-null   object    21  aeronave_nivel_dano
6188 non-null   object    22  aeronave_fatalidades_total 6188 non-null   int64
dtypes: float64(2), int64(4), object(17)memory usage: 1.1+ MB

```

In [147]:

```

#Alterar a coluna 'codigo_ocorrencia2' para 'codigo_ocorrencia' para ficar
similar o dataframe ocorrenciadf_aeronave.rename(columns={'codigo_oco
rrencia2': 'codigo_ocorrencia'}, inplace=True)

```

In [148]:

```

df_aeronave.info()
<class 'pandas.core.frame.DataFrame'>RangeIndex: 6188 entries, 0 to 6187Data
columns (total 23 columns): #   Column                               Non-Null Count
Dtype --- -----
codigo_ocorrencia      6188 non-null   int64    1   aeronave_matricula
6188 non-null   object    2   aeronave_operador_categoria 6188 non-null
object    3   aeronave_tipo_veiculo      6188 non-null   object    4
aeronave_fabricante    6188 non-null   object    5   aeronave_modelo
6188 non-null   object    6   aeronave_tipo_icao      6188 non-null
object    7   aeronave_motor_tipo      6185 non-null   object    8
aeronave_motor_quantidade  6188 non-null   object    9   aeronave_pmd
6188 non-null   int64    10  aeronave_pmd_categoria  6188 non-null   int64
11  aeronave_assentos      6010 non-null   float64   12
aeronave_ano_fabricacao  6005 non-null   float64   13
aeronave_pais_fabricante 6188 non-null   object    14
aeronave_pais_registro  6188 non-null   object    15
aeronave_registro_categoria 6188 non-null   object    16
aeronave_registro_segmento 6188 non-null   object    17  aeronave_voo_origem
6187 non-null   object    18  aeronave_voo_destino    6187 non-null
object    19  aeronave_fase_operacao  6188 non-null   object    20
aeronave_tipo_operacao  6188 non-null   object    21  aeronave_nivel_dano
6188 non-null   object    22  aeronave_fatalidades_total 6188 non-null   int64
dtypes: float64(2), int64(4), object(17)memory usage: 1.1+ MB

```

In [149]:

```
df_aeronave["codigo_ocorrencia"].duplicated().sum()
```

74

Out[149]:

In [150]:

```
#Removendo os valores duplicados da coluna 'codigo_ocorrencia'df_aeronave =
df_aeronave.drop_duplicates(subset=['codigo_ocorrencia'])
```

In [151]:

```
df_aeronave["codigo_ocorrencia"].duplicated().sum()
```

Out[151]:

0

In [152]:

```
df_basededados = pd.merge(df_ocorrencia, df_aeronave, on='codigo_ocorrencia',
how="left")
```

In [153]:

```
df_basededados.info()
```

```
<class 'pandas.core.frame.DataFrame'>Int64Index: 6114 entries, 0 to 6113Data
columns (total 27 columns): #    Column                                Non-Null Count
Dtype --- -----
codigo_ocorrencia            6114 non-null    int64    1
ocorrencia_classificacao     6114 non-null    object    2  ocorrencia_cidade
6114 non-null    object    3  ocorrencia_uf          6114 non-null
object    4  ocorrencia_ano          6114 non-null    object    5
aeronave_matricula           6114 non-null    object    6
aeronave_operador_categoria  6114 non-null    object    7  aeronave_tipo_veiculo
6114 non-null    object    8  aeronave_fabricante     6114 non-null
object    9  aeronave_modelo           6114 non-null    object    10
aeronave_tipo_icao            6114 non-null    object    11  aeronave_motor_tipo
6111 non-null    object    12  aeronave_motor_quantidade 6114 non-null
object    13  aeronave_pmd                6114 non-null    int64    14
aeronave_pmd_categoria       6114 non-null    int64    15  aeronave_assentos
5937 non-null    float64  16  aeronave_ano_fabricacao   5932 non-null
float64  17  aeronave_pais_fabricante  6114 non-null    object    18
aeronave_pais_registro       6114 non-null    object    19
aeronave_registro_categoria  6114 non-null    object    20
aeronave_registro_segmento   6114 non-null    object    21  aeronave_voo_origem
6113 non-null    object    22  aeronave_voo_destino      6113 non-null
object    23  aeronave_fase_operacao      6114 non-null    object    24
aeronave_tipo_operacao       6114 non-null    object    25  aeronave_nivel_dano
6114 non-null    object    26  aeronave_fatalidades_total 6114 non-null    int64
dtypes: float64(2), int64(4), object(21)memory usage: 1.3+ MB
```

In [154]:

```
#Carregar a base de dados fator_Contribuinte df_fatorContribuinte =
pd.read_csv("C:\\Users\\fefaj\\OneDrive\\Área de Trabalho\\TCC
Aprovado\\Desenvolvimento\\Base de dados\\fatorcontribuinte.csv")
```

In [155]:

```
df_fatorContribuinte.info()
```



```
<class 'pandas.core.frame.DataFrame'>RangeIndex: 4485 entries, 0 to 4484Data
columns (total 5 columns): #    Column                Non-Null Count  Dtype ---
-----
-----
0    codigo_ocorrencia3    4485 non-null  object
1    fator_nome            4485 non-null  object
2    fator_condicionante    4485 non-null  object
3    fator_area            4485 non-null  object
4    fator_area            4485 non-null  object
dtypes: int64(1), object(4)memory usage: 175.3+ KB
```

In [156]:

```
#Alterar a coluna 'codigo_ocorrencia3' para 'codigo_ocorrencia' para ficar
similar
o
dataframe
ocorrencia3df_fatorContribuinte.rename(columns={'codigo_ocorrencia3':
'codigo_ocorrencia'}, inplace=True)
```

In [157]:

```
df_fatorContribuinte.info()
<class 'pandas.core.frame.DataFrame'>RangeIndex: 4485 entries, 0 to 4484Data
columns (total 5 columns): #    Column                Non-Null Count  Dtype ---
-----
-----
0    codigo_ocorrencia    4485 non-null  object
1    fator_nome            4485 non-null  object
2    fator_condicionante    4485 non-null  object
3    fator_area            4485 non-null  object
4    fator_area            4485 non-null  object
dtypes: int64(1), object(4)memory usage: 175.3+ KB
```

In [158]:

```
#Verificação
de
Valores
duplicadosdf_fatorContribuinte["codigo_ocorrencia"].duplicated().sum()
```

Out[158]:

3333

In [159]:

```
#Exclusão
de
Valores
duplicadosdf_fatorContribuinte
df_fatorContribuinte.drop_duplicates(subset=['codigo_ocorrencia'])
```

In [160]:

```
#Verificação
de
Valores
duplicadosdf_fatorContribuinte["codigo_ocorrencia"].duplicated().sum()
```

Out[160]:

0

In [161]:

```
df_basededados = pd.merge(df_basededados, df_fatorContribuinte,
on='codigo_ocorrencia', how="left")
```

In [162]:

```
df_basededados.info()
<class 'pandas.core.frame.DataFrame'>Int64Index: 6114 entries, 0 to 6113Data
columns (total 31 columns): #    Column                Non-Null Count  Dtype ---
-----
-----
0
codigo_ocorrencia    6114 non-null  int64    1
ocorrencia_classificacao    6114 non-null  object    2
ocorrencia_cidade    6114 non-null  object    3
ocorrencia_uf    6114 non-null
ocorrencia_ano    6114 non-null  object    5
```

```

aeronave_matricula          6114 non-null  object  6
aeronave_operador_categoria  6114 non-null  object  7  aeronave_tipo_veiculo
6114 non-null  object  8  aeronave_fabricante          6114 non-null
object  9  aeronave_modelo          6114 non-null  object  10
aeronave_tipo_icao          6114 non-null  object  11  aeronave_motor_tipo
6111 non-null  object  12  aeronave_motor_quantidade  6114 non-null
object  13  aeronave_pmd          6114 non-null  int64  14
aeronave_pmd_categoria      6114 non-null  int64  15  aeronave_assentos
5937 non-null  float64 16  aeronave_ano_fabricacao  5932 non-null
float64 17  aeronave_pais_fabricante  6114 non-null  object  18
aeronave_pais_registro      6114 non-null  object  19
aeronave_registro_categoria  6114 non-null  object  20
aeronave_registro_segmento  6114 non-null  object  21  aeronave_voo_origem
6113 non-null  object  22  aeronave_voo_destino          6113 non-null
object  23  aeronave_fase_operacao  6114 non-null  object  24
aeronave_tipo_operacao      6114 non-null  object  25  aeronave_nivel_dano
6114 non-null  object  26  aeronave_fatalidades_total  6114 non-null  int64
27  fator_nome          1152 non-null  object  28  fator_aspecto
1152 non-null  object  29  fator_condicionante          1152 non-null
object  30  fator_area          1152 non-null  object dtypes:
float64(2), int64(4), object(25)memory usage: 1.5+ MB

```

In [163]:

```

#Carregar a base de dados ocorrencia tipo df_ocorrenciatipo =
pd.read_csv("C:\\Users\\fefaj\\OneDrive\\Área de Trabalho\\TCC
Aprovado\\Desenvolvimento\\Base de dados\\ocorrenciatipo.csv",
encoding = "UTF-8", sep = ";", decimal = ',')

```

In [164]:

```

df_ocorrenciatipo.info()
<class 'pandas.core.frame.DataFrame'>RangeIndex: 6283 entries, 0 to 6282Data
columns (total 4 columns): #    Column                               Non-Null Count
Dtype ---  -----
codigo_ocorrencia1          6283 non-null  int64  1  ocorrencia_tipo
6283 non-null  object  2  ocorrencia_tipo_categoria  6283 non-null  object  3
taxonomia_tipo_icao          6283 non-null  objectdtypes: int64(1),
object(3)memory usage: 196.5+ KB

```

In [165]:

```

#Alterar a coluna 'codigo_ocorrencia1' para 'codigo_ocorrencia' para ficar
similar o dataframe
ocorrenciadf_ocorrenciatipo.rename(columns={'codigo_ocorrencia1':
'codigo_ocorrencia'}, inplace=True)

```

In [166]:

```

df_ocorrenciatipo.info()
<class 'pandas.core.frame.DataFrame'>RangeIndex: 6283 entries, 0 to 6282Data
columns (total 4 columns): #    Column                               Non-Null Count
Dtype ---  -----
codigo_ocorrencia          6283 non-null  int64  1  ocorrencia_tipo

```

```
6283 non-null    object 2    ocorrencia_tipo_categoria 6283 non-null    object 3
taxonomia_tipo_icao      6283 non-null    objectdtypes: int64(1),
object(3)memory usage: 196.5+ KB
```

In [167]:

```
df_ocorrenciatipo["codigo_ocorrencia"].duplicated().sum()
```

Out[167]:

```
169
```

In [168]:

```
df_ocorrenciatipo
df_ocorrenciatipo.drop_duplicates(subset=['codigo_ocorrencia'])
```

In [169]:

```
df_ocorrenciatipo["codigo_ocorrencia"].duplicated().sum()
```

Out[169]:

```
0
```

In [170]:

```
df_basededados = pd.merge(df_basededados, df_ocorrenciatipo,
on='codigo_ocorrencia', how="left")
```

In [171]:

```
df_basededados.info ()
```

```
<class 'pandas.core.frame.DataFrame'>Int64Index: 6114 entries, 0 to 6113Data
columns (total 34 columns): #    Column                                Non-Null Count
Dtype --- -----
codigo_ocorrencia      6114 non-null    int64    1
ocorrencia_classificacao  6114 non-null    object    2
6114 non-null    object 3    ocorrencia_uf                6114 non-null
object 4    ocorrencia_ano                6114 non-null    object    5
aeronave_matricula      6114 non-null    object    6
aeronave_operador_categoria  6114 non-null    object    7
6114 non-null    object 8    aeronave_fabricante            6114 non-null
object 9    aeronave_modelo                6114 non-null    object    10
aeronave_tipo_icao        6114 non-null    object    11
6111 non-null    object 12    aeronave_motor_quantidade      6114 non-null
object 13    aeronave_pmd                6114 non-null    int64    14
aeronave_pmd_categoria    6114 non-null    int64    15
5937 non-null    float64 16    aeronave_ano_fabricacao        5932 non-null
float64 17    aeronave_pais_fabricante      6114 non-null    object    18
aeronave_pais_registro    6114 non-null    object    19
aeronave_registro_categoria  6114 non-null    object    20
aeronave_registro_segmento  6114 non-null    object    21
6113 non-null    object 22    aeronave_voo_destino            6113 non-null
object 23    aeronave_fase_operacao          6114 non-null    object    24
aeronave_tipo_operacao    6114 non-null    object    25
6114 non-null    object 26    aeronave_fatalidades_total      6114 non-null    int64
27 fator_nome                1152 non-null    object    28 fator_aspecto
1152 non-null    object 29    fator_condicionante            1152 non-null
object 30    fator_area                1152 non-null    object    31
```

```

ocorrencia_tipo          6114 non-null   object   32
ocorrencia_tipo_categoria 6114 non-null   object   33  taxonomia_tipo_icao
6114 non-null   object dtypes: float64(2), int64(4), object(28)memory usage:
1.6+ MB

```

In [172]:

```
#Inserindo coluna para contagemdf_basededados['Contagem'] = 1
```

In [173]:

```

df_basededados.info ()
<class 'pandas.core.frame.DataFrame'>Int64Index: 6114 entries, 0 to 6113Data
columns (total 35 columns): #    Column                                Non-Null Count
Dtype --- -----
codigo_ocorrencia          6114 non-null   int64    1
ocorrencia_classificacao   6114 non-null   object    2  ocorrencia_cidade
6114 non-null   object    3  ocorrencia_uf          6114 non-null
object    4  ocorrencia_ano          6114 non-null   object    5
aeronave_matricula         6114 non-null   object    6
aeronave_operador_categoria 6114 non-null   object    7  aeronave_tipo_veiculo
6114 non-null   object    8  aeronave_fabricante     6114 non-null
object    9  aeronave_modelo          6114 non-null   object   10
aeronave_tipo_icao          6114 non-null   object   11  aeronave_motor_tipo
6111 non-null   object   12  aeronave_motor_quantidade 6114 non-null
object   13  aeronave_pmd              6114 non-null   int64    14
aeronave_pmd_categoria     6114 non-null   int64    15  aeronave_assentos
5937 non-null   float64  16  aeronave_ano_fabricacao   5932 non-null
float64  17  aeronave_pais_fabricante 6114 non-null   object   18
aeronave_pais_registro     6114 non-null   object   19
aeronave_registro_categoria 6114 non-null   object   20
aeronave_registro_segmento 6114 non-null   object   21  aeronave_voo_origem
6113 non-null   object   22  aeronave_voo_destino     6113 non-null
object   23  aeronave_fase_operacao     6114 non-null   object   24
aeronave_tipo_operacao     6114 non-null   object   25  aeronave_nivel_dano
6114 non-null   object   26  aeronave_fatalidades_total 6114 non-null   int64
27  fator_nome              1152 non-null   object   28  fator_aspecto
1152 non-null   object   29  fator_condicionante     1152 non-null
object   30  fator_area              1152 non-null   object   31
ocorrencia_tipo          6114 non-null   object   32
ocorrencia_tipo_categoria 6114 non-null   object   33  taxonomia_tipo_icao
6114 non-null   object   34  Contagem              6114 non-null   int64
dtypes: float64(2), int64(5), object(28)memory usage: 1.7+ MB

```

In [174]:

```
#Excluir      colunas      onde      o      código      da      ocorrência      é
inválidodf_basededados['codigo_ocorrencia'].dropna(axis = 0, inplace = True)
```

In [175]:

```
#exclusão      de      valores      nulos      de      valores      nulos      no
dataframedf_basededados.dropna(inplace=True)
```

In [176]:

```
df_basededados.info ()
<class 'pandas.core.frame.DataFrame'>Int64Index: 1140 entries, 3 to 5709Data
columns (total 35 columns): #    Column                                Non-Null Count
Dtype --- -----
codigo_ocorrencia          1140 non-null    int64    1
ocorrencia_classificacao   1140 non-null    object    2  ocorrencia_cidade
1140 non-null    object    3  ocorrencia_uf          1140 non-null
object    4  ocorrencia_ano          1140 non-null    object    5
aeronave_matricula         1140 non-null    object    6
aeronave_operador_categoria 1140 non-null    object    7  aeronave_tipo_veiculo
1140 non-null    object    8  aeronave_fabricante     1140 non-null
object    9  aeronave_modelo          1140 non-null    object    10
aeronave_tipo_icao          1140 non-null    object    11  aeronave_motor_tipo
1140 non-null    object    12  aeronave_motor_quantidade 1140 non-null
object    13  aeronave_pmd              1140 non-null    int64    14
aeronave_pmd_categoria     1140 non-null    int64    15  aeronave_assentos
1140 non-null    float64 16  aeronave_ano_fabricacao   1140 non-null
float64 17  aeronave_pais_fabricante  1140 non-null    object    18
aeronave_pais_registro     1140 non-null    object    19
aeronave_registro_categoria 1140 non-null    object    20
aeronave_registro_segmento 1140 non-null    object    21  aeronave_voo_origem
1140 non-null    object    22  aeronave_voo_destino     1140 non-null
object    23  aeronave_fase_operacao     1140 non-null    object    24
aeronave_tipo_operacao     1140 non-null    object    25  aeronave_nivel_dano
1140 non-null    object    26  aeronave_fatalidades_total 1140 non-null    int64
27  fator_nome              1140 non-null    object    28  fator_aspecto
1140 non-null    object    29  fator_condicionante      1140 non-null
object    30  fator_area              1140 non-null    object    31
ocorrencia_tipo            1140 non-null    object    32
ocorrencia_tipo_categoria   1140 non-null    object    33  taxonomia_tipo_icao
1140 non-null    object    34  Contagem                1140 non-null    int64
dtypes: float64(2), int64(5), object(28)memory usage: 320.6+ KB
```

In [177]:

```
#2. Análise e exploração de dados#2.1Gerar gráficos#Importação da função Counter
e da biblioteca matplotlib.pyplotfrom collections import Counterimport
matplotlib.pyplot as plt
```

In [178]:

```
df_basededados = df_basededados.apply(lambda x: x.replace('***','NÃO
IDENTIFICADO'))df_basededados['Contagem'] = 1
```

In [179]:

```
#Quantidade de acidentes por ano#Contagem das opções da coluna anoano_ocorrencias
= Counter(df_basededados['ocorrencia_ano'])ano_ocorrencias
```

Out[179]:

```
Counter({'2010': 129,          '2011': 157,          '2012': 144,
'2013': 145,          '2014': 151,          '2015': 108,          '2016': 65,
'2017': 76,          '2018': 77,          '2019': 47,          '2020': 41})
```

In [180]:

```
#identificação de valores nulos no dataframe df_basededados.isnull().sum()
```

Out[180]:

```

codigo_ocorrencia      0ocorrencia_classificacao
0ocorrencia_cidade      0ocorrencia_uf
0ocorrencia_ano          0aeronave_matricula
0aeronave_operador_categoria 0aeronave_tipo_veiculo
0aeronave_fabricante     0aeronave_modelo
0aeronave_tipo_icao        0aeronave_motor_tipo
0aeronave_motor_quantidade 0aeronave_pmd
0aeronave_pmd_categoria   0aeronave_assentos
0aeronave_ano_fabricacao  0aeronave_pais_fabricante
0aeronave_pais_registro   0aeronave_registro_categoria
0aeronave_registro_segmento 0aeronave_voo_origem
0aeronave_voo_destino     0aeronave_fase_operacao
0aeronave_tipo_operacao   0aeronave_nivel_dano
0aeronave_fatalidades_total 0fator_nome              0fator_aspecto
0fator_condicionante      0fator_area
0ocorrencia_tipo          0ocorrencia_tipo_categoria
0taxonomia_tipo_icao       0Contagem              0dtype: int64

```

In [181]:

```
Ocorrenciaporano = df_basededados.groupby('ocorrencia_ano')[['codigo_ocorrencia']].count ()
```

In [182]:

```
Ocorrenciaporano[:30]
```

Out[182]:

codigo\_ocorrencia

ocorrencia\_ano

2010	129
2011	157
2012	144
2013	145
2014	151
2015	108
2016	65
2017	76
2018	77
2019	47

2020

41

In [183]:

```
#Plotar gráfico de Quantidade de Ocorrências por Ano
Ocorrenciaporano.plot(
    kind='bar',      figsize=(20, 5),      color='blue')plt.style.use("seaborn-
white")plt.axis      ('auto')plt.xlabel('Ano',      fontsize      =      14,
fontweight='bold')plt.ylabel('Qtd      Ocorrências',      fontsize      =      14      ,
fontweight='bold')plt.legend(['Ocorrências'],      fontsize      =
12)plt.title("Quantidade      de      Ocorrências      por      Ano",      fontsize      =
18)plt.yticks(fontsize=12)plt.xticks(fontsize=12, rotation=360)plt.show()
```

In [184]:

```
Ocorrenciaporuf
df_basededados.groupby('ocorrendia_uf')[['codigo_ocorrendia']].count ()
```

In [185]:

```
Ocorrenciaporuf[:30]
```

Out[185]:

codigo\_ocorrendia

ocorrendia_uf	
AC	11
AL	6
AM	61
AP	1
BA	65
CE	19
DF	18
ES	7
GO	74
MA	24
MG	78
MS	35
MT	68
NÃO IDENTIFICADO	1
PA	73

PB	3
PE	20
PI	13
PR	102
RJ	46
RN	3
RO	9
RR	23
RS	123
SC	36
SE	6
SP	199
TO	16

In [186]:

```
df_basededados.info()
<class 'pandas.core.frame.DataFrame'>Int64Index: 1140 entries, 3 to 5709Data
columns (total 35 columns): #    Column                                Non-Null Count
Dtype ---  -----
codigo_ocorrencia          1140 non-null    int64    1
ocorrencia_classificacao   1140 non-null    object    2
1140 non-null    object    3  ocorrencia_uf          1140 non-null
object    4  ocorrencia_ano          1140 non-null    object    5
aeronave_matricula         1140 non-null    object    6
aeronave_operador_categoria 1140 non-null    object    7
1140 non-null    object    8  aeronave_fabricante     1140 non-null
object    9  aeronave_modelo          1140 non-null    object    10
aeronave_tipo_icao          1140 non-null    object    11
1140 non-null    object    12  aeronave_motor_quantidade 1140 non-null
object    13  aeronave_pmd             1140 non-null    int64    14
aeronave_pmd_categoria     1140 non-null    int64    15
1140 non-null    float64  16  aeronave_ano_fabricacao   1140 non-null
float64  17  aeronave_pais_fabricante   1140 non-null    object    18
aeronave_pais_registro     1140 non-null    object    19
aeronave_registro_categoria 1140 non-null    object    20
aeronave_registro_segmento 1140 non-null    object    21
1140 non-null    object    22  aeronave_voo_destino     1140 non-null
object    23  aeronave_fase_operacao    1140 non-null    object    24
aeronave_tipo_veiculo     1140 non-null    object    25
aeronave_motor_tipo       1140 non-null    object    26
aeronave_assentos         1140 non-null    object    27
aeronave_voo_origem       1140 non-null    object    28
aeronave_destino           1140 non-null    object    29
aeronave_categoria         1140 non-null    object    30
aeronave_fabricante        1140 non-null    object    31
aeronave_modelo            1140 non-null    object    32
aeronave_operador          1140 non-null    object    33
aeronave_registro          1140 non-null    object    34
aeronave_tipo              1140 non-null    object    35
```



```

aeronave_tipo_operacao      1140 non-null object 25 aeronave_nivel_dano
1140 non-null object 26 aeronave_fatalidades_total 1140 non-null int64
27 fator_nome                1140 non-null object 28 fator_aspecto
1140 non-null object 29 fator_condicionante        1140 non-null
object 30 fator_area          1140 non-null object 31
ocorrencia_tipo             1140 non-null object 32
ocorrencia_tipo_categoria   1140 non-null object 33 taxonomia_tipo_icao
1140 non-null object 34 Contagem                    1140 non-null int64
dtypes: float64(2), int64(5), object(28)memory usage: 320.6+ KB

```

In [187]:

```
tipoaeronave = Counter(df_basededados['ocorrencia_uf'])tipoaeronave
```

Out[187]:

```

Counter({'MT': 68,          'AM': 61,          'PR': 102,          'SP': 199,
'PA': 73,          'TO': 16,          'SC': 36,          'GO': 74,          'BA':
65,          'DF': 18,          'MG': 78,          'AC': 11,          'RS': 123,
'RR': 23,          'MS': 35,          'PE': 20,          'RJ': 46,          'AL':
6,          'RO': 9,          'MA': 24,          'PI': 13,          'CE': 19,
'SE': 6,          'ES': 7,          'RN': 3,          'AP': 1,          'PB': 3,
'NÃO IDENTIFICADO': 1})

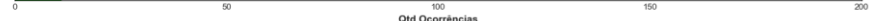
```

In [188]:

```

Ocorrenciaporuf.plot          (kind='barh',          figsize=(21,          11),
color='green')plt.style.use("seaborn-pastel")plt.axis ('auto')plt.xlabel('Qtd
Ocorrências', fontsize = 14, fontweight='bold')plt.ylabel('UF', fontsize = 14 ,
fontweight='bold')plt.legend(['Ocorrências'],          fontsize
=
12)plt.title("Quantidade de Ocorrências por UF",          fontsize
=
18)plt.yticks(fontsize=12)plt.xticks(fontsize=12, rotation=360)plt.show()

```



In [189]:

```

#Quantidade de acidentes por tipo de aeronave#Contagem das opções da coluna tipo
de
aeronavetipoaeronave
=
Counter(df_basededados['aeronave_tipo_veiculo'])tipoaeronave

```

Out[189]:

```

Counter({'AVIÃO': 948,          'HELICÓPTERO': 158,          'PLANADOR': 7,
'ANFÍBIO': 4,          'ULTRALEVE': 18,          'NÃO IDENTIFICADO': 5})


```

In [190]:

```

#Gráfico de Quantidade de ocorrências por tipo de
veículo"df_basededados.aeronave_tipo_veiculo.hist(bins=20)plt.style.use('class
ic')plt.xlabel("Tipo de Veículo")plt.ylabel("Quantidade de
ocorrências")plt.title("Quantidade de ocorrências por tipo de
veículo")plt.show()

```



In [191]:


```

aeronavefabricante
=
df_basededados.groupby('aeronave_fabricante')[['codigo_ocorrencia']].count ()

```

In [192]:

```
aeronavefabricante.plot(kind='barh', figsize=(21, 11),
color='green')plt.style.use("seaborn-pastel")plt.axis('auto')plt.xlabel('Qtd
Ocorrências', fontsize = 14, fontweight='bold')plt.ylabel('UF', fontsize = 14 ,
fontweight='bold')plt.legend(['Ocorrências'], fontsize
= 12)plt.title("Quantidade de Ocorrências por UF", fontsize =
18)plt.yticks(fontsize=12)plt.xticks(fontsize=12, rotation=360)plt.show()
```




In [193]:

```
import seaborn as sns
```

In [194]:

```
from bokeh.sampledata.iris import flowers as dadosX =
df_basededados['aeronave_tipo_veiculo']Y =
df_basededados['fator_condicionante']plt.scatter(X,Y)plt.show()
```



In [195]:


```
#Quatidade de acidentes por classificação#Contagem das opções da coluna
anoocorrenciaclassificacao =
Counter(df_basededados['ocorrencia_classificacao'])ocorrenciaclassificacao
```

Out[195]:

```
Counter({'ACIDENTE': 845, 'INCIDENTE GRAVE': 281, 'INCIDENTE': 14})
```


In [196]:

```
plt.style.use('classic')plt.pie(ocorrenciaclassificacao.values(), labels =
ocorrenciaclassificacao.keys(), autopct = '%1.1f%%',
textprops={'fontsize': 14})plt.title("Ocorrências Por Classificação",
fontsize=1)plt.axis("image")plt.show()
```



In [197]:

```
import seaborn as snsimport matplotlib.pyplot as plt # Criando o ambiente do
gráfico sns.set_style("white")plt.figure(figsize=(8, 5)) # Gráfico de Dispersão
= sns.scatterplot(x="aeronave_tipo_veiculo", y="ocorrencia_classificacao",
data=df_basededados)plt.show()
```



In [198]:

```
df_basededados.info
```

Out[198]:

```
<bound method DataFrame.info of          codigo_ocorrencia
ocorrencia_classificacao      ocorrencia_cidade  \3          39527
ACIDENTE      LUCAS DO RIO VERDE      7          39707      INCIDENTE
GRAVE          CANUTAMA      8          39156      INCIDENTE
GRAVE          CASCAVEL      11          40069
ACIDENTE          SÃO PAULO      13          39507
ACIDENTE      PRIMAVERA DO LESTE      ...          ...
...          ...      5640          79619
ACIDENTE          FEIJÓ      5649          79627
ACIDENTE          BIRITIBA-MIRIM      5676          79671
ACIDENTE      SÃO MIGUEL DO IGUAÇU      5697          79692
```

ACIDENTE	CATANDUVA	5709	79713
ACIDENTE SANTA VITÓRIA DO PALMAR	ocorrencia_uf	ocorrencia_ano	
aeronave_matricula \3	MT	2010	PTURT 7
AM 2010	PTEVH 8	PR	2010
PPPTO 11	SP	2010	PPMAW 13
MT 2010	PTUTM ...	...	...
... 5640	AC	2020	PTRPU 5649
SP 2020	PUPAFJ 5676	PR	2020
PUMLU 5697	SP	2020	PPHRH 5709
RS 2020	PTUXD	aeronave_operador_categoria	
aeronave_tipo_veiculo \3	NÃO IDENTIFICADO	AVIÃO	NÃO
7	NÃO IDENTIFICADO	AVIÃO 8	NÃO
IDENTIFICADO	AVIÃO 11	NÃO IDENTIFICADO	
HELICÓPTERO 13	NÃO IDENTIFICADO	AVIÃO ...	
...	... 5640	NÃO IDENTIFICADO	
AVIÃO 5649	NÃO IDENTIFICADO	ULTRALEVE 5676	
NÃO IDENTIFICADO	ULTRALEVE 5697	NÃO IDENTIFICADO	
AVIÃO 5709	NÃO IDENTIFICADO	AVIÃO	
aeronave_fabricante aeronave_modelo ...	aeronave_nivel_dano \3	NEIVA	
INDUSTRIA AERONAUTICA	EMB-202 ...	SUBSTANCIAL 7	
EMBRAER EMB-810C ...	LEVE 8	AEROSPATIALE AND	
ALENIA ATR-72-212A ...	LEVE 11		
HELIBRAS AS 350 B3 ...	DESTRUÍDA 13	NEIVA INDUSTRIA	
AERONAUTICA EMB-202A ...	SUBSTANCIAL ...		
...	... 5640	NEIVA INDUSTRIA	
AERONAUTICA EMB-720D ...	SUBSTANCIAL 5649	NÃO	
IDENTIFICADO ADVENTURE ...	SUBSTANCIAL 5676	NÃO	
IDENTIFICADO J250 ...	SUBSTANCIAL 5697	NEIVA INDUSTRIA	
AERONAUTICA 56-C-1 ...	SUBSTANCIAL 5709		
EMBRAER EMB-202 ...	SUBSTANCIAL		
aeronave_fatalidades_total	fator_nome \3		
0 APLICAÇÃO DE COMANDOS 7	0	INDISCIPLINA	
DE VOO 8	0 MANUTENÇÃO DA AERONAVE	11	
0 APLICAÇÃO DE COMANDOS 13	0	PESSOAL DE	
APOIO ...	...	5640	
0 PLANEJAMENTO DE VOO 5649	0	OUTRO	
FATOR 5676	0 MANUTENÇÃO DA AERONAVE	5697	
0 APLICAÇÃO DE COMANDOS 5709	0 APLICAÇÃO DE		
COMANDOS	fator_aspecto	fator_condicionante	
fator_area \3	DESEMPENHO DO SER HUMANO	OPERAÇÃO DA AERONAVE	FATOR
OPERACIONAL 7	DESEMPENHO DO SER HUMANO	OPERAÇÃO DA AERONAVE	FATOR
OPERACIONAL 8	DESEMPENHO DO SER HUMANO	MANUTENÇÃO DA AERONAVE	FATOR
OPERACIONAL 11	DESEMPENHO DO SER HUMANO	OPERAÇÃO DA AERONAVE	FATOR
OPERACIONAL 13	DESEMPENHO DO SER HUMANO	OPERAÇÃO DA AERONAVE	FATOR
OPERACIONAL ...	...	...	
... 5640	DESEMPENHO DO SER HUMANO	OPERAÇÃO DA AERONAVE	FATOR

```

OPERACIONAL 5649 OUTRO NÃO IDENTIFICADO
OUTRO 5676 DESEMPENHO DO SER HUMANO MANUTENÇÃO DA AERONAVE FATOR
OPERACIONAL 5697 DESEMPENHO DO SER HUMANO OPERAÇÃO DA AERONAVE FATOR
OPERACIONAL 5709 DESEMPENHO DO SER HUMANO OPERAÇÃO DA AERONAVE FATOR
OPERACIONAL ocorrencia_tipo \3 OPERAÇÃO A BAIXA
ALTITUDE 7 COM TREM DE POUSO 8 FOGO EM VOO
11 PERDA DE CONTROLE NO SOLO 13 FALHA DO MOTOR EM VOO ...
... 5640 COLISÃO COM FAUNA 5649 FALHA DO MOTOR EM VOO 5676
FALHA DO MOTOR EM VOO 5697 POUSO LONGO 5709
PANE SECA ocorrencia_tipo_categoria
taxonomia_tipo_icao \3 OPERAÇÃO A BAIXA ALTITUDE
LALT 7 FALHA OU MAU FUNCIONAMENTO DE SISTEMA / COMPON...
SCF-NP 8 FOGO/FUMAÇA (SEM IMPACTO) | FOGO EM VOO
F-NI 11 PERDA DE CONTROLE NO SOLO
LOC-G 13 FALHA OU MAU FUNCIONAMENTO DO MOTOR | FALHA DO...
SCF-PP ...
... 5640 COLISÃO COM FAUNA
WILD 5649 FALHA OU MAU FUNCIONAMENTO DO MOTOR | FALHA DO...
SCF-PP 5676 FALHA OU MAU FUNCIONAMENTO DO MOTOR | FALHA DO...
SCF-PP 5697 EXCURSÃO DE PISTA | POUSO LONGO
RE 5709 COMBUSTÍVEL | PANE SECA
FUEL Contagem 3 1 7 1 8 1 11
1 13 1 ... 5640 1 5649 1 5676 1
5697 1 5709 1 [1140 rows x 35 columns]>

```

In [199]:

```

#Cores para serem usadas nos gráficos colors =
[['#0D47A1', '#1565C0', '#1976D2', '#1E88E5', '#2196F3'],
 ['#311B92', '#512DA8', '#673AB7', '#9575CD', '#B39DDB'],
 ['#1B5E20', '#388E3C', '#4CAF50', '#81C784', '#66BB6A'],
 ['#E65100', '#EF6C00', '#F57C00', '#FB8C00', '#FF9800'],
 ['#3E2723', '#4E342E', '#5D4037', '#6D4C41', '#795548'],
 ['#BF360C', '#D84315', '#E64A19', '#F4511E', '#FF5722'],
 ['#880E4F', '#AD1457', '#C2185B', '#D81B60', '#E91E63']]

```

In [335]:

```

classificacao =
df_basededados.groupby('aeronave_motor_tipo')['Contagem'].sum().plot(kind='bar',
    figsize=(10,5), color=random.choice(colors), alpha=.7) for p in
classificacao.patches:
    b=p.get_bbox()
classificacao.annotate("{:.0f}".format(b.y1 + b.y0), ((b.x0 + b.x1)/2 - 0.04,
b.y1 + 15)) plt.xticks(rotation=0)plt.xlabel('Tipo do motor da
Aeronave')plt.ylabel('Quantidade')plt.title('Quantidade de ocorrências por Tipo
de Motor')plt.show()

```

Tipo do motor da Aeronave

In [336]:

```

classificacao =
df_basededados.groupby('aeronave_motor_quantidade')['Contagem'].sum().plot(kin

```

```
d='bar',      figsize=(10,5), color=random.choice(colors), alpha=.7) for p in
classificacao.patches:
classificacao.annotate("{:.0f}".format(b.y1 + b.y0), ((b.x0 + b.x1)/2 - 0.04,
b.y1 + 15)) plt.xticks(rotation=0)plt.xlabel('Quantidade de
Motor')plt.ylabel('Quantidade')plt.title('Quantidade de ocorrências por
Quantidade de Motor')plt.show()
```

Quantidade de Motor

In [316]:

```
plotagem_data = df_basededados.copy()plotagem_data['ocorrencia_ano'] =
plotagem_data['ocorrencia_ano'].apply(lambda x:
x.replace(x,x[0:4]))plotagem_data.groupby('ocorrencia_ano')['Contagem'].sum().
plot(color='purple', figsize=(12,5), grid=True) plt.title('Quantidade de
Ocorrênciaao longo dos Anos \n Total: ' + str(df_basededados['Contagem'].sum())
+ ' ocorrências')plt.xlabel('Ano Referência')plt.ylabel('Quantidade de
Ocorrências')plt.show()
```

Ano Referência

In [340]:

```
fatorarea =
df_basededados.groupby('fator_area')['Contagem'].sum().plot(kind='barh',figsiz
e=(5,5), color=random.choice(colors), alpha=.7) for p in fatorarea .patches:
b=p.get_bbox() plt.xticks(rotation=0)plt.xlabel('Fator
area')plt.ylabel('Quantidade')plt.title('Quantidade de ocorrências por
fator')plt.show()
```

Fator area

In [330]:

```
# convertendo para CSV file e salvando no computador para manipulação de dados
no Power BI
df_basededados.to_csv(r'C:\Users\fefaj\OneDrive\Área de
Trabalho.csv')
```

In [203]:

```
#Importação da função train_test_splitfrom sklearn.model_selection import
train_test_split
```

In [262]:

```
#Escolha dos atributos para um novo dataset dos modelos de ,machine
Learninhd_set = df_basededados[['ocorrencia_classificacao',
'aeronave_tipo_veiculo', 'fator_aspecto',
'aeronave_motor_quantidade', 'fator_area',
'fator_condicionante']].copy()df_set.head()
```

Out[262]:

	ocorren- cia_classifica- cao	aero- nave_tipo_vei- culo	fator_aspecto	aero- nave_mo- tor quanti- dade	fator_area	fator_condici- onante
			DESEMPENHO			
3	ACIDENTE	AVIÃO	DO SER HU- MANO	MONOMO- TOR	FATOR OPE- RACIONAL	OPERAÇÃO DA AERONAVE

7	INCIDENTE GRAVE	AVIÃO	DESEMPENHO DO SER HUMANO	BIMOTOR	FATOR OPERACIONAL	OPERAÇÃO DA AERONAVE
8	INCIDENTE GRAVE	AVIÃO	DESEMPENHO DO SER HUMANO	BIMOTOR	FATOR OPERACIONAL	MANUTENÇÃO DA AERONAVE
11	ACIDENTE	HELICÓPTERO	DESEMPENHO DO SER HUMANO	MONOMOTOR	FATOR OPERACIONAL	OPERAÇÃO DA AERONAVE
13	ACIDENTE	AVIÃO	DESEMPENHO DO SER HUMANO	MONOMOTOR	FATOR OPERACIONAL	OPERAÇÃO DA AERONAVE

In [263]:

```
df_set.info()
<class 'pandas.core.frame.DataFrame'>Int64Index: 1140 entries, 3 to 5709
Data columns (total 6 columns): #   Column                               Non-Null Count
Dtype ---
ocorrencia_classificacao    1140 non-null object 1
aeronave_tipo_veiculo       1140 non-null object 2
fator_aspecto               1140 non-null object 3
aeronave_motor_quantidade   1140 non-null object 4
fator_area                 1140 non-null object 5
fator_condicionante         1140 non-null object 6
objectdtypes: object(6)memory usage: 62.3+ KB
```

In [266]:

```
#retirandoos valoresde linha com NÃO IDENTIFICADO filtro1 = df_set
[df_set['ocorrencia_classificacao'] == 'NÃO IDENTIFICADO']
.indexdf_set.drop(filtro1 , inplace=True)
```

In [267]:

```
#retirandoos valoresde linha com NÃO IDENTIFICADO filtro2 = df_set
[df_set['aeronave_tipo_veiculo'] == 'NÃO IDENTIFICADO']
.indexdf_set.drop(filtro2 , inplace=True)
```

In [268]:

```
#retirandoos valoresde linha com NÃO IDENTIFICADO filtro3 = df_set
[df_set['fator_aspecto'] == 'NÃO IDENTIFICADO' ].indexdf_set.drop(filtro3 ,
inplace=True)
```

In [269]:

```
#retirandoos valoresde linha com NÃO IDENTIFICADO filtro4 = df_set
[df_set['fator_area'] == 'NÃO IDENTIFICADO' ].indexdf_set.drop(filtro4 ,
inplace=True)
```

In [270]:

```
#retirandoos valoresde linha com NÃO IDENTIFICADO filtro5 = df_set
[df_set['fator_condicionante'] == 'NÃO IDENTIFICADO' ].indexdf_set.drop(filtro5
, inplace=True)
```

In [271]:

```

df_set['ocorrencia_classificacao'].unique()
Out[271]:
array(['ACIDENTE', 'INCIDENTE GRAVE', 'INCIDENTE'], dtype=object)

#Identificação dos valores da
aeronave_tipo_veiculo df_set['aeronave_tipo_veiculo'].unique()
Out[272]:
array(['AVIÃO', 'HELICÓPTERO', 'PLANADOR', 'ANFÍBIO', 'ULTRALEVE'],
dtype=object)

#Transformação de valores categóricos em valores inteiros coluna
aeronave_tipo_veiculo ajuste_aeronave_tipo_veiculo = {'AVIÃO': 0, 'HELICÓPTERO':
1, 'PLANADOR': 2, 'ANFÍBIO': 3, 'ULTRALEVE': 4} df_set['aeronave_tipo_veiculo'] =
df_set['aeronave_tipo_veiculo'].map(ajuste_aeronave_tipo_veiculo )
In [273]:

#Identificação dos valores da
aeronave_tipo_veiculo df_set['aeronave_tipo_veiculo'].unique()
Out[274]:
array([0, 1, 2, 3, 4], dtype=int64)

#Identificação dos valores da
fator_aspecto df_set['fator_aspecto'].unique()
Out[275]:
array(['DESEMPENHO DO SER HUMANO', 'ASPECTO PSICOLÓGICO'], dtype=object)

#Transformação de valores categóricos em valores inteiros coluna
fator_aspecto ajuste_fator_aspecto = {'DESEMPENHO DO SER HUMANO': 0, 'ASPECTO
PSICOLÓGICO': 1,
'ASPECTO MÉDICO': 2, 'ASPECTO DE FABRICAÇÃO':
3,
'ELEMENTOS RELACIONADOS AO AMBIENTE OPERACIONAL': 4, 'OUTRO':
5,
'INFRAESTRUTURA AEROPORTUÁRIA': 6, 'ERGONOMIA': 7, 'ASPECTO
DE PROJETO': 8,
'ASPECTO DE MANUSEIO DO MATERIAL': 9}
df_set['fator_aspecto'] = df_set['fator_aspecto'].map(ajuste_fator_aspecto)
In [277]:

df_set['fator_aspecto'].unique()
Out[277]:
array([0, 1], dtype=int64)

#Identificação dos valores da
aeronave_motor_quantidade df_set['aeronave_motor_quantidade'].unique()
Out[278]:
array(['MONOMOTOR', 'BIMOTOR', 'SEM TRAÇÃO', 'QUADRIMOTOR', 'TRIMOTOR'],
dtype=object)

#Transformação de valores categóricos em valores inteiros coluna
aeronave_motor_quantidade ajuste_aeronave_motor_quantidade = {'MONOMOTOR': 0,
'BIMOTOR': 1, 'SEM TRAÇÃO': 2, 'QUADRIMOTOR': 3, 'TRIMOTOR': 4}

```

```

df_set['aeronave_motor_quantidade'] =
df_set['aeronave_motor_quantidade'].map(ajuste_aeronave_motor_quantidade)
In [280]:
#Identificação dos valores da coluna fator_area
df_set['fator_area'].unique()
Out[280]:
array(['FATOR OPERACIONAL', 'FATOR HUMANO'], dtype=object)
In [281]:
#Transformação de valores categóricos em valores inteiros coluna
fator_area
ajuste_fator_area = {'FATOR OPERACIONAL': 0, 'FATOR HUMANO': 1, 'FATOR
MATERIAL': 2, 'OUTRO': 3}
df_set['fator_area'] =
df_set['fator_area'].map(ajuste_fator_area)
In [282]:
#Identificação dos valores da coluna
fator_condicionante
df_set['fator_condicionante'].unique()
Out[282]:
array(['OPERAÇÃO DA AERONAVE', 'MANUTENÇÃO DA AERONAVE', 'INDIVIDUAL',
'PSICOSSOCIAL', 'ORGANIZACIONAL', 'PRESTAÇÃO DE SERVIÇOS DE TRÁFEGO
AÉREO'], dtype=object)
In [283]:
#Transformação de valores categóricos em valores inteiros coluna
fator_condicionante
ajuste_fator_condicionante = {'OPERAÇÃO DA AERONAVE': 0,
'MANUTENÇÃO DA AERONAVE': 1, 'INDIVIDUAL': 2, 'PSICOSSOCIAL': 3,
'ORGANIZACIONAL': 4, 'PRESTAÇÃO DE SERVIÇOS DE TRÁFEGO AÉREO':
5}
df_set['fator_condicionante'] =
df_set['fator_condicionante'].map(ajuste_fator_condicionante)
In [284]:
#Divisão para as bases de treinamento
X_train =
df_set.drop(['ocorrencia_classificacao'], axis = 1)
y_train =
df_set.ocorrencia_classificacao
In [285]:
X_train.info()
<class 'pandas.core.frame.DataFrame'>Int64Index: 1012 entries, 3 to 5709Data
columns (total 5 columns): # Column Non-Null Count
Dtype---
aeronave_tipo_veiculo 1012 non-null int64 1 fator_aspecto
1012 non-null int64 2 aeronave_motor_quantidade 1012 non-null int64 3
fator_area 1012 non-null int64 4 fator_condicionante
1012 non-null int64
dtypes: int64(5)memory usage: 47.4 KB
In [286]:
#Tipo da serie y_train
type(y_train)
Out[286]:
pandas.core.series.Series
In [287]:
#Criação das bases de teste e treinamento#75% para treinamento e 25% para
teste
treinamento, xteste, ytreinamento, yteste = train_test_split(X_train,
y_train, random_state = 0)

```



In [288]:

```
#Informação do dataframe de treinamento
xtreinamento.info()
<class 'pandas.core.frame.DataFrame'>Int64Index: 759 entries, 2869 to 3174
Data columns (total 5 columns): #   Column                               Non-Null Count
Dtype---   -----
aeronave_tipo_veiculo    759 non-null    int64 1   fator_aspecto
759 non-null    int64 2   aeronave_motor_quantidade 759 non-null    int64 3
fator_area              759 non-null    int64 4   fator_condicionante
759 non-null    int64
dtypes: int64(5)
memory usage: 35.6 KB
```

In [289]:

```
#Informação do dataframe de teste
xteste.info()
<class 'pandas.core.frame.DataFrame'>Int64Index: 253 entries, 3343 to 1835
Data columns (total 5 columns): #   Column                               Non-Null Count
Dtype---   -----
aeronave_tipo_veiculo    253 non-null    int64 1   fator_aspecto
253 non-null    int64 2   aeronave_motor_quantidade 253 non-null    int64 3
fator_area              253 non-null    int64 4   fator_condicionante
253 non-null    int64
dtypes: int64(5)
memory usage: 11.9 KB
```

In [290]:

```
print (ytreinamento)
2869          ACIDENTE8          INCIDENTE GRAVE2313          ACIDENTE4793
INCIDENTE GRAVE1372          ACIDENTE          ...          3973
ACIDENTE934          INCIDENTE GRAVE2906          ACIDENTE2606
ACIDENTE3174          INCIDENTE GRAVE
Name: ocorrencia_classificacao, Length: 759,
dtype: object
```

In [291]:

```
#Importação das funções para as medidas de avaliação dos algoritmos
from sklearn.metrics import accuracy_score, classification_report
```

In [292]:

```
#Criação do modelo utilizando a Árvore de decisão
xtreinamento, xteste, ytreinamento, yteste
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
ocorrencia_classificacao_tree = DecisionTreeClassifier(random_state=0)
ocorrencia_classificacao_tree.fit(xtreinamento, ytreinamento)
print("Acurácia:", ocorrencia_classificacao_tree.score(xtreinamento, ytreinamento))
Train_predict = ocorrencia_classificacao_tree.predict(xteste)
print("Acurácia de previsão: ", accuracy_score(yteste, Train_predict))
print(classification_report(yteste, Train_predict))
Acurácia: 0.7720685111989459
Acurácia de previsão: 0.7351778656126482
precision    recall  f1-score   support
ACIDENTE      0.76      0.98
0.86         185      INCIDENTE      0.00      0.00      0.00
3INCIDENTE GRAVE      0.42      0.08      0.13         65      accuracy
0.74         253      macro avg      0.39      0.35      0.33         253
weighted avg      0.66      0.74      0.66         253
```

In [293]:

```
#Criação do modelo utilizando a Regressão Logística
from sklearn.linear_model
import LogisticRegression
lr = LogisticRegression()
lr = lr.fit(xtreinamento, ytreinamento)
print("Acurácia: ", lr.score(xtreinamento, ytreinamento))
tp_lr = lr.predict(xteste)
print("Acurácia de previsão: ", accuracy_score(yteste, tp_lr))
print(classification_report(yteste, tp_lr))
```

Acurácia: 0.758893280632411 Acurácia de previsão: 0.7351778656126482

precision	recall	f1-score	support	ACIDENTE	0.75	0.98
0.85	185	INCIDENTE	0.00	0.00	0.00	
3	INCIDENTE GRAVE	0.44	0.06	0.11	65	accuracy
0.74	253	macro avg	0.40	0.35	0.32	253
weighted avg	0.66	0.74	0.65	253		

C:\Users\fefaj\anaconda3\lib\site-packages\sklearn\metrics\\_classification.py:1221: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

In [294]:

```
#Criação do modelo utilizando Naïve Bayes
from sklearn.naive_bayes
import GaussianNB
nb = GaussianNB()
nb = nb.fit(xtreinamento, ytreinamento)
print("Acurácia: ", nb.score(xtreinamento, ytreinamento))
tp_nb = nb.predict(xteste)
print("Acurácia de previsão: ", accuracy_score(yteste, tp_nb))
print(classification_report(yteste, tp_nb))
```

Acurácia: 0.45586297760210803 Acurácia de previsão: 0.4505928853754941

precision	recall	f1-score	support	ACIDENTE	0.82	0.34
0.48	185	INCIDENTE	0.00	0.00	0.00	
3	INCIDENTE GRAVE	0.29	0.78	0.43	65	accuracy
0.45	253	macro avg	0.37	0.38	0.30	253
weighted avg	0.67	0.45	0.46	253		

In [295]:

```
#Criação do modelo utilizando Gradiente Descendente
from sklearn.linear_model
import SGDClassifier
sgd = SGDClassifier()
sgd = sgd.fit(xtreinamento, ytreinamento)
print("Acurácia: ", sgd.score(xtreinamento, ytreinamento))
tp_sgd = sgd.predict(xteste)
print("Acurácia de previsão: ", accuracy_score(yteste, tp_sgd))
print(classification_report(yteste, tp_sgd))
```

Acurácia: 0.7523056653491436 Acurácia de previsão: 0.7351778656126482

precision	recall	f1-score	support	ACIDENTE	0.73	1.00
0.85	185	INCIDENTE	0.00	0.00	0.00	
3	INCIDENTE GRAVE	1.00	0.02	0.03	65	accuracy
0.74	253	macro avg	0.58	0.34	0.29	253
weighted avg	0.79	0.74	0.63	253		

C:\Users\fefaj\anaconda3\lib\site-packages\sklearn\metrics\\_classification.py:1221: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

In [296]:

```
#Criação do modelo utilizando KNN (K - Nearest Neighbors)from sklearn.neighbors
import KNeighborsClassifierknn = KNeighborsClassifier()knn =
knn.fit(xtreinamento, ytreinamento)print("Acurácia: ", knn.score(xtreinamento,
ytreinamento))tp_knn = knn.predict(xteste)print("Acurácia de previsão: ",
accuracy_score(yteste, tp_knn))print(classification_report(yteste, tp_knn))
Acurácia: 0.758893280632411Acurácia de previsão: 0.7351778656126482
precision    recall  f1-score   support

ACIDENTE      0.74      0.99
0.85         185      INCIDENTE      0.00      0.00      0.00
3INCIDENTE GRAVE      0.50      0.03      0.06      65      accuracy
0.74         253      macro avg      0.41      0.34      0.30      253
weighted avg      0.67      0.74      0.63      253
C:\Users\fefaj\anaconda3\lib\site-
packages\sklearn\metrics\_classification.py:1221: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
_warn_prf(average, modifier, msg_start, len(result))
```

In [331]:

```
#Criação do modelo utilizando Random Forest from sklearn.ensemble import
RandomForestClassifierrfm = RandomForestClassifier()rfm = rfm.fit(xtreinamento,
ytreinamento)print("Acurácia: ", rfm.score(xtreinamento, ytreinamento))tp_rfm =
rfm.predict(xteste)print("Acurácia de previsão: ", accuracy_score(yteste,
tp_rfm))print(classification_report(yteste, tp_rfm))
Acurácia: 0.7720685111989459Acurácia de previsão: 0.7430830039525692
precision    recall  f1-score   support

ACIDENTE      0.76      0.98
0.86         185      INCIDENTE      0.00      0.00      0.00
3INCIDENTE GRAVE      0.50      0.11      0.18      65      accuracy
0.74         253      macro avg      0.42      0.36      0.34      253
weighted avg      0.68      0.74      0.67      253
```