



PONTIFÍCIA UNIVERSIDADE CATÓLICA DE
MINAS GERAIS

Pós-graduação Lato Sensu em Ciência de Dados
e Big Data

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

Aluna: Talita Santos Andrade
31/05/2023

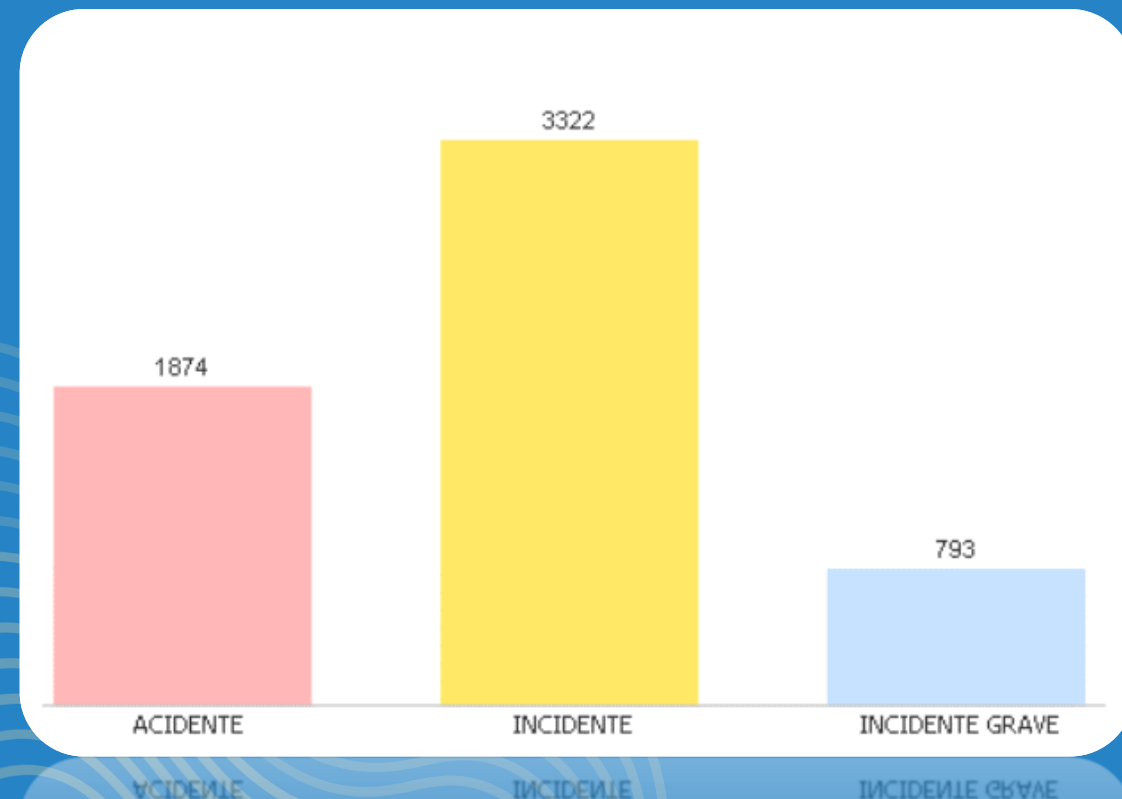
INTRODUÇÃO

- O transporte aéreo é de suma importância para o desenvolvimento do Brasil, que possui um dos mais fortes sistemas dessa categoria do mundo.
- 20.680 aeronaves registradas no Registro Aeronáutico Brasileiro (RAB)
- 5.984 ocorrências aeronáuticas na aviação civil brasileira entre os anos de 2012 a 2023.



OBJETIVO

- Classificar as ocorrências aeronáuticas como 'INCIDENTE', 'INCIDENTE GRAVE' e 'ACIDENTE' de acordo com as informações registradas pelo CENIPA na base de dados de ocorrências aeronáuticas brasileiras nos anos de 2010 a 2022.



COLETA DE DADOS



gov.br Governo Federal

Órgãos do Governo Acesso à Informação Legislação Acessibilidade Entrar

Dados Abertos

Conjunto de Dados > CENIPA - Ocorrências ...

CENIPA - Ocorrências Aeronáuticas na Aviação Civil Brasileira

+ Seguir Avaliar

Atualizado -

ESCALA DE SATISFAÇÃO 0 / 10

0 - RESPOSTAS

Organização

Força Aérea Brasileira

Força Aérea Brasileira (FAB) é o ramo aéreo das Forças Armadas do Brasil e um dos três serviços uniformizados nacionais. A FAB foi formada quando os ramos aéreos do Exército e da Marinha foram fundidos em uma força militar única. Ambos os ramos de ar transferiram seus equipamentos, instalações e pessoal para a nova força armada.

6 conjunto

+ Seguir Contato

Descrição

▼ Importação de Bibliotecas

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

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

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

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

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

COLETA DE DADOS

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6769 entries, 0 to 6768
Data columns (total 22 columns):
```

#	Column	Non-Null Count	Dtype
0	codigo_ocorrencia	6769 non-null	int64
1	codigo_ocorrencia1	6769 non-null	int64
2	codigo_ocorrencia2	6769 non-null	int64
3	codigo_ocorrencia3	6769 non-null	int64
4	codigo_ocorrencia4	6769 non-null	int64
5	ocorrencia_classificacao	6769 non-null	object
6	ocorrencia_latitude	5135 non-null	object
7	ocorrencia_longitude	5135 non-null	object
8	ocorrencia_cidade	6769 non-null	object
9	ocorrencia_uf	6769 non-null	object
10	ocorrencia_pais	6769 non-null	object
11	ocorrencia_aerodromo	6769 non-null	object
12	ocorrencia_dia	6769 non-null	object
13	ocorrencia_hora	6767 non-null	object
14	investigacao_aeronave_liberada	6531 non-null	object
15	investigacao_status	6428 non-null	object
16	divulgacao_relatorio_numero	5987 non-null	object
17	divulgacao_relatorio_publicado	6769 non-null	object
18	divulgacao_dia_publicacao	1781 non-null	object
19	total_recomendacoes	6769 non-null	int64
20	total_aeronaves_envolvidas	6769 non-null	int64
21	ocorrencia_saida_pista	6769 non-null	object

```
dtypes: int64(7), object(15)
memory usage: 1.1+ MB
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6339 entries, 0 to 6338
Data columns (total 23 columns):
```

#	Column	Non-Null Count	Dtype
0	codigo_ocorrencia2	6339 non-null	int64
1	aeronave_matricula	6306 non-null	object
2	aeronave_operador_categoria	6306 non-null	object
3	aeronave_tipo_veiculo	6306 non-null	object
4	aeronave_fabricante	6306 non-null	object
5	aeronave_modelo	6306 non-null	object
6	aeronave_tipo_icao	6306 non-null	object
7	aeronave_motor_tipo	6289 non-null	object
8	aeronave_motor_quantidade	6306 non-null	object
9	aeronave_pmd	6306 non-null	float64
10	aeronave_pmd_categoria	6306 non-null	float64
11	aeronave_assentos	6064 non-null	float64
12	aeronave_ano_fabricacao	6078 non-null	float64
13	aeronave_pais_fabricante	6306 non-null	object
14	aeronave_pais_registro	6306 non-null	object
15	aeronave_registro_categoria	6306 non-null	object
16	aeronave_registro_segmento	6306 non-null	object
17	aeronave_voo_origem	6305 non-null	object
18	aeronave_voo_destino	6305 non-null	object
19	aeronave_fase_operacao	6306 non-null	object
20	aeronave_tipo_operacao	6306 non-null	object
21	aeronave_nivel_dano	6306 non-null	object
22	aeronave_fatalidades_total	6306 non-null	float64

```
dtypes: float64(5), int64(1), object(17)
memory usage: 1.1+ MB
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7100 entries, 0 to 7099
Data columns (total 4 columns):
```

#	Column	Non-Null Count	Dtype
0	codigo_ocorrencia1	7100 non-null	int64
1	ocorrencia_tipo	7099 non-null	object
2	ocorrencia_tipo_categoria	7099 non-null	object
3	taxonomia_tipo_icao	7099 non-null	object

```
dtypes: int64(1), object(3)
memory usage: 222.0+ KB
```

OCORRÊNCIA TIPO
codigo_ocorrencia1
ocorrencia_tipo
ocorrencia_tipo_categoria
taxonomia_tipo_icao

OCORRÊNCIA
codigo_ocorrencia1
codigo_ocorrencia4
codigo_ocorrencia2
codigo_ocorrencia3
codigo_ocorrencia
ocorrencia_classificacao
ocorrencia_latitude
ocorrencia_longitude
ocorrencia_cidade
ocorrencia_uf
ocorrencia_pais
ocorrencia_aerodromo
ocorrencia_dia
ocorrencia_hora
investigacao_aeronave_liberada
investigacao_status
divulgacao_relatorio_numero
divulgacao_relatorio_publicado
divulgacao_dia_publicacao
total_recomendacoes
total_aeronaves_envolvidas
ocorrencia_saida_pista

AERONAVE
codigo_ocorrencia2
aeronave_matricula
aeronave_operador_categoria
aeronave_tipo_veiculo
aeronave_fabricante
aeronave_modelo
aeronave_tipo_icao
aeronave_motor_tipo
aeronave_motor_quantidade
aeronave_pmd
aeronave_pmd_categoria
aeronave_assentos
aeronave_ano_fabricacao
aeronave_pais_fabricante
aeronave_pais_registro
aeronave_registro_categoria
aeronave_registro_segmento
aeronave_voo_origem
aeronave_voo_destino
aeronave_fase_operacao
aeronave_tipo_operacao
aeronave_nivel_dano
aeronave_fatalidades_total

PROCESSAMENTO DE DADOS

- Análise das informações dos datasets;
- Transformação e tratamento dos dados em informações úteis para a análise e tomada de decisão.

Normalização de colunas numéricas

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

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

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

PROCESSAMENTO DE DADOS

- Dataset original possui muitos campos nulos ou preenchidos com valores ***
- Alguns campos contendo mais de 1000 registros nulos:

divulgacao_dia_publicacao	5271
ocorrencia_latitude	1661
ocorrencia_longitude	1661

Substituição de valores nulos ou '***'

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

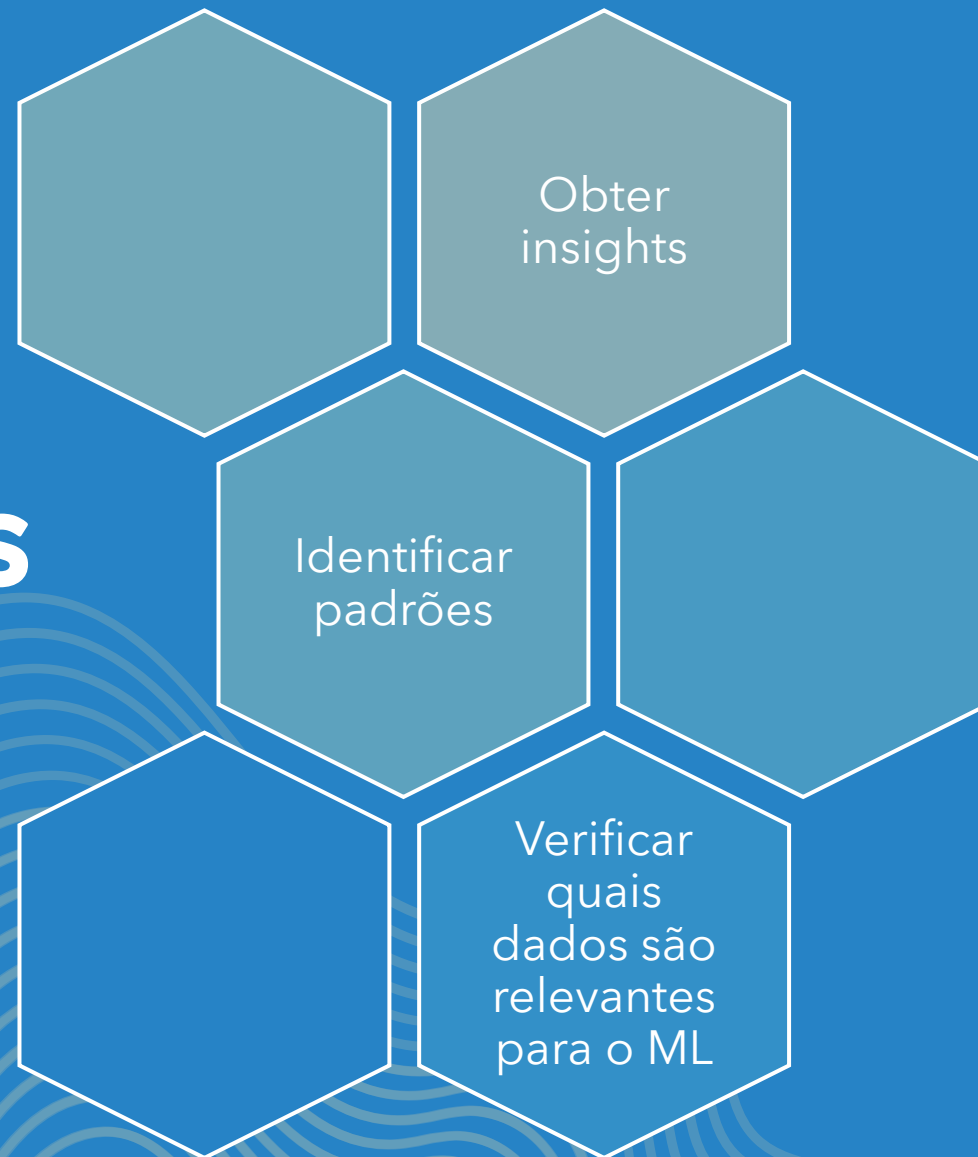
# Substituição de valores NULOS de colunas com informação do tipo texto por valor 'NÃO INFORMADO'
df_ocorr_aeronauticas[df_ocorr_aeronauticas.select_dtypes(
    include=['object']).columns] = df_ocorr_aeronauticas.select_dtypes(
    include=['object']).fillna('NÃO INFORMADO')

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

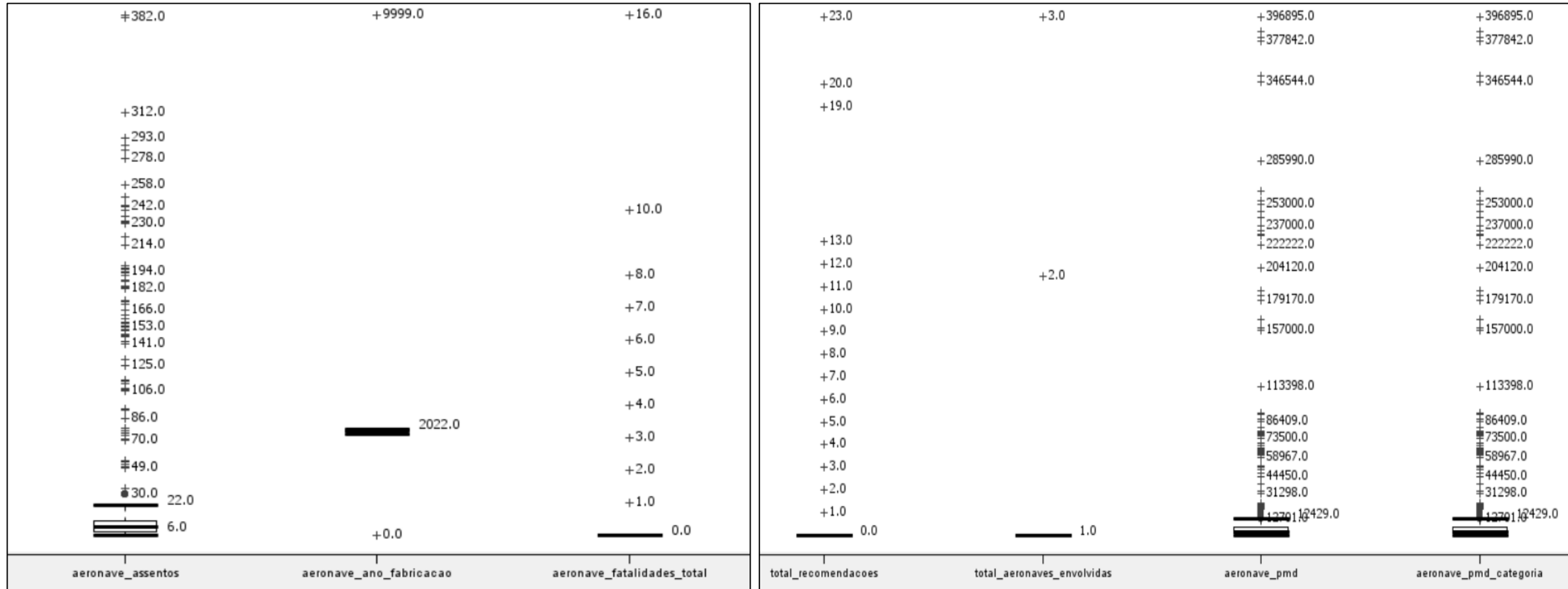
```
count = df_ocorr_aeronauticas.apply(lambda x: x[x=='***'].count())
print(count)
```

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

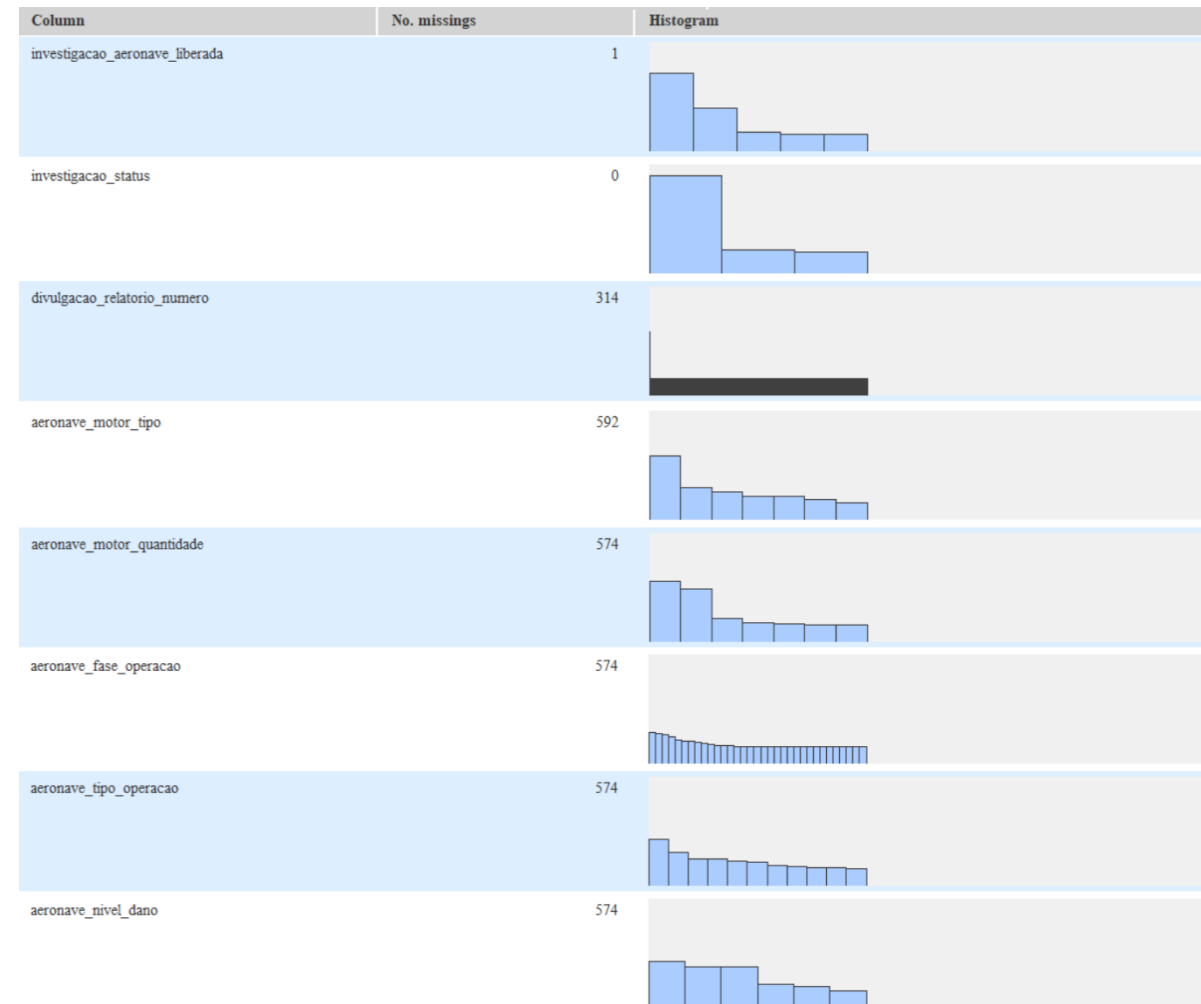
ANÁLISE E EXPLORAÇÃO DOS DADOS



ANÁLISE E EXPLORAÇÃO



ANÁLISE E EXPLORAÇÃO



ANÁLISE E EXPLORAÇÃO

Gráfico de Correlação

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

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

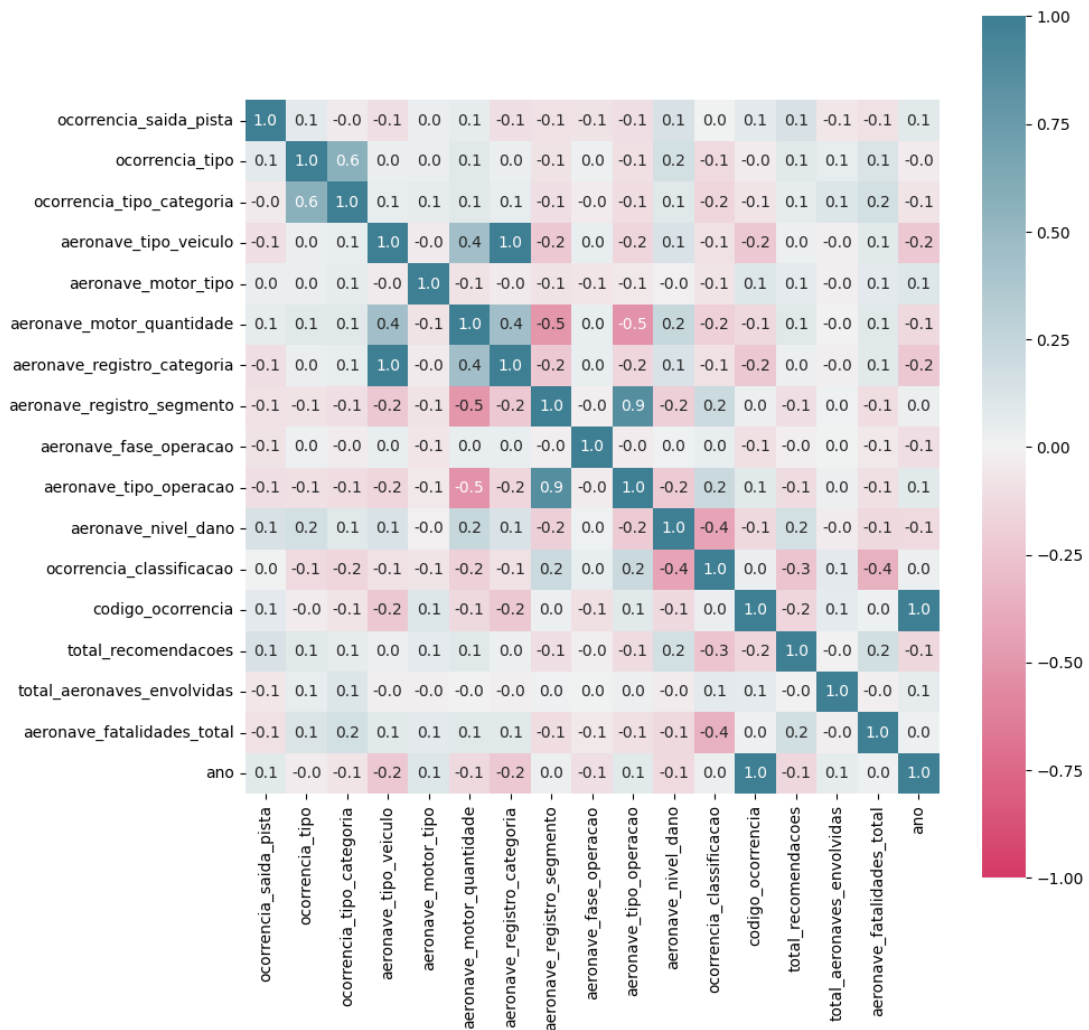
# Seleciona todas as colunas que são numéricas
df_numerical = df_ocorr_aeronauticas[df_ocorr_aeronauticas.select_dtypes(exclude=['object']).columns]

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

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

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

plt.show()
```





MODELOS DE MACHINE LEARNING

Utilizados 3 algoritmos de classificação diferentes:

Árvore de Decisão


Gradient Boosting


Random Florest


MODELOS DE MACHINE LEARNING

Métricas de avaliação:

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

 **Acurácia**
Proporção de previsões corretas em relação ao total de previsões realizadas.

 **Recall**
Proporção de verdadeiros positivos em relação a todos os valores reais positivos.

 **F1 Score**
Média harmônica entre a precisão e o recall.

MODELOS DE MACHINE LEARNING

Códigos sem alteração de hiperparâmetros:

Execução do modelo de classificação Decision Tree

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

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

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

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

Accuracy: 0.73 (+/- 0.24)

Execução do modelo de classificação Gradient Boosted

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

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

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

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

Accuracy: 0.85 (+/- 0.11)

Execução do modelo de classificação Random Forest

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

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

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

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

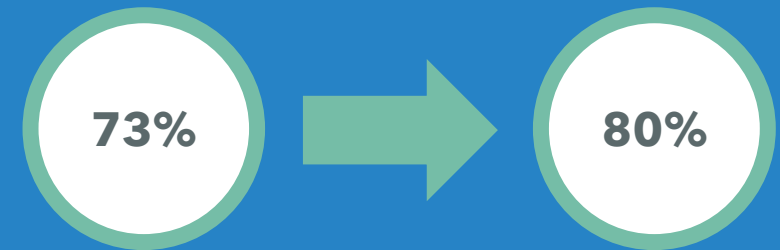
Accuracy: 0.87 (+/- 0.08)

Utilização de Cross
Validation

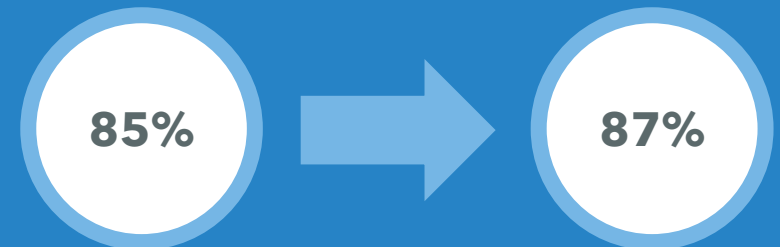
ALTERAÇÃO DE HIPERPARÂMETRO

- Melhora na performance dos modelos;
- Utilização da técnica de Random Search (RandomizedSearchCV da biblioteca Scikit-learn);
- Melhoria da acurácia após o processo.

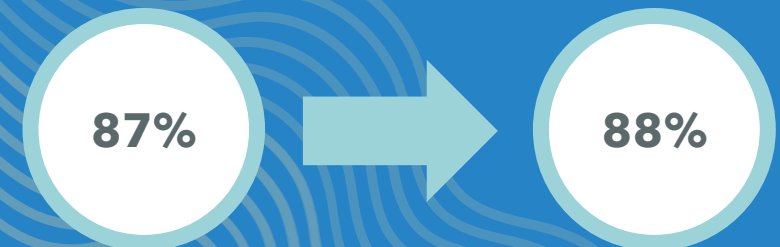
Decision Tree



Gradient Boosting



Random Florest



ALTERAÇÃO DE HIPERPARÂMETRO

Códigos com alteração de hiperparâmetros:

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

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

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

random_search.fit(X_encoded, y)

# Obter os melhores hiperparâmetros encontrados
best_params = random_search.best_params_
```

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

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

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

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

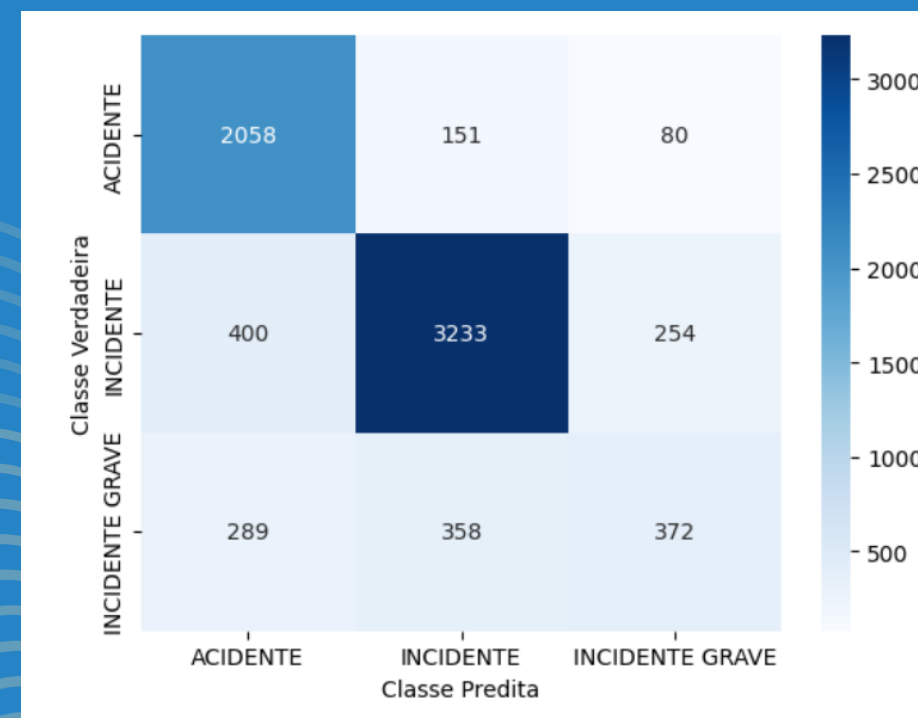
Accuracy: 0.88 (+/- 0.10)

Utilização de Cross
Validation

RESULTADOS

Decision Tree

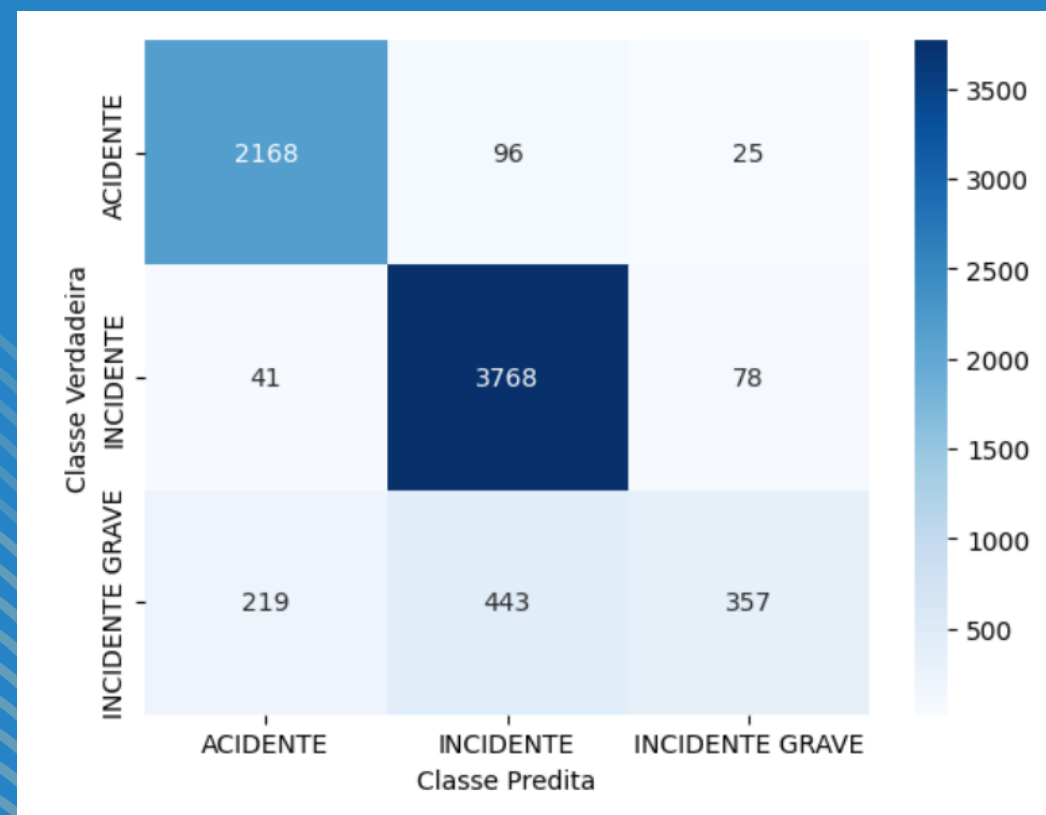
	precision	recall	f1-score	support
ACIDENTE	0.75	0.90	0.82	2289
INCIDENTE	0.86	0.83	0.85	3887
INCIDENTE GRAVE	0.53	0.37	0.43	1019
accuracy			0.79	7195
macro avg	0.71	0.70	0.70	7195
weighted avg	0.78	0.79	0.78	7195



RESULTADOS

Gradient Boosting

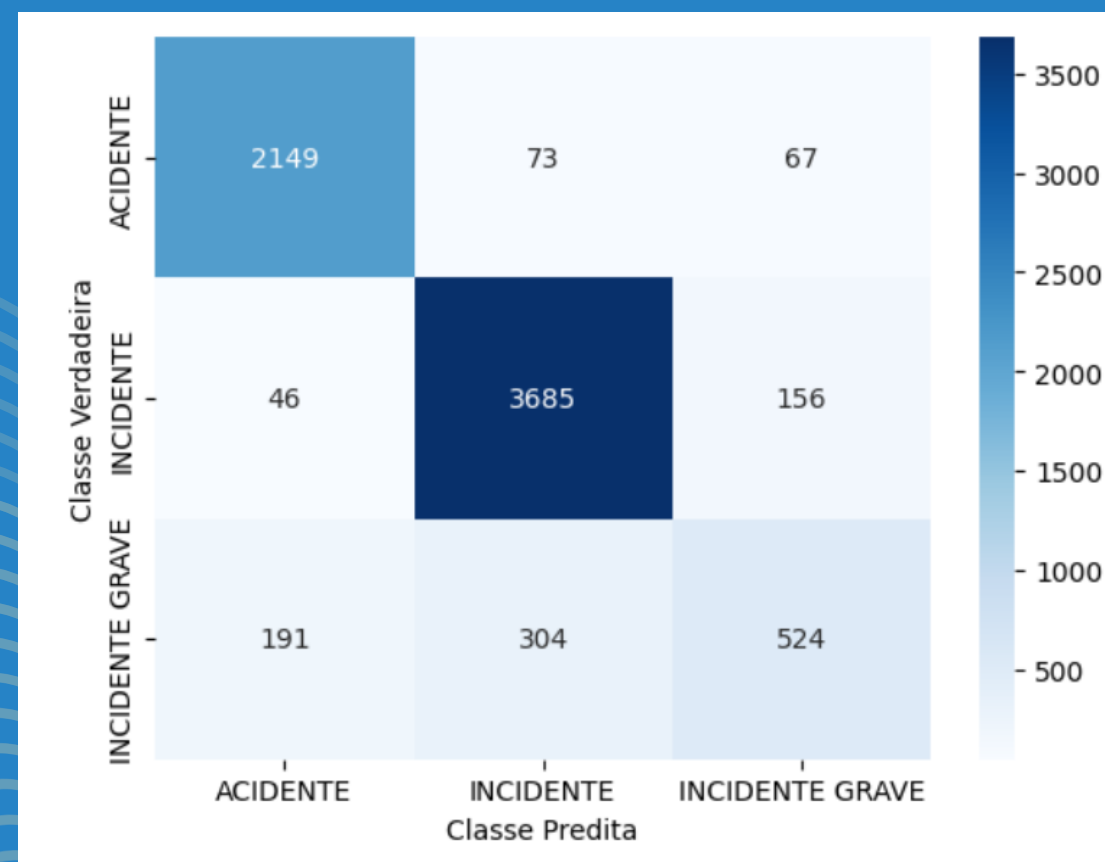
	precision	recall	f1-score	support
ACIDENTE	0.89	0.95	0.92	2289
INCIDENTE	0.87	0.97	0.92	3887
INCIDENTE GRAVE	0.78	0.35	0.48	1019
accuracy			0.87	7195
macro avg	0.85	0.76	0.77	7195
weighted avg	0.87	0.87	0.86	7195



RESULTADOS

Random Florest

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



CONSIDERAÇÕES FINAIS

Códigos com alteração de hiperparâmetros:

Método escolhido: Random Forest

Acurácia: 88%

F1 Score:
Acidente = 92%
Incidente = 93%
Incidente grave = 59%



PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS

Pós-graduação Lato Sensu em Ciência de Dados e Big Data

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

Aluna: Talita Santos Andrade

31/05/2023