Aircraft Purchase Risk Analysis

Overview

The aim of this project is to analyze which aircraft are the lowest risk for a company to start as a new business endeavor using actionable insights derived from In-depth data analysis.

Business Problem

To determine the safest aircraft with the lowest risk for a company to purchase when starting a new business endeavor. This will be decision will be guided by actionable insights from analysis

Data Understanding

The dataset contains aviation accident data, with 31 columns detailing various aspects of each incident.

It includes information on:

- Aircraft details: Make, model, number of engines, engine type, amateur-built.
- Accident Details: Date, location (city/state/country), phase of flight, weather conditions, and accident severity.
- Operational Factors: Purpose of flight , regulatory information (FAR description), and air carrier details.
- Injury & Damage Information: Number of fatalities, serious injuries, minor injuries, and uninjured passengers.

To gain these insights begin by:

1. Importing required libraries
2. Loading the dataset
3. Displaying an overview

```python
#import the pandas and numpy libraries and give them aliases
import pandas as pd
import numpy as np

# import libraries for visualization for later on
import matplotlib.pyplot as plt
import seaborn as sns

#load the dataset and display the first 5 rows
df = pd.read_csv('AviationData.csv', encoding= 'latin-1')
pd.set_option('display.max_columns', None) #displays all columns
```

```
#create a copy of the dataframe to preserve the original
df_AviationData = df.copy()

#preview the dataframe
df_AviationData.head()
```

```
<ipython-input-148-fc9b934311cf>:2: DtypeWarning: Columns (6,7,28)
have mixed types. Specify dtype option on import or set
low_memory=False.
  df = pd.read_csv('AviationData.csv', encoding= 'latin-1')
```

{"type":"dataframe","variable_name":"df_AviationData"}

Display summary and summary statistics of the data

- This helps get an overview of the data

using the .describe() and the .info()

```
# display a summary statistics including all columns(objects)
df_AviationData.describe(include='object')
```

{"type":"dataframe"}

```
#display the summary statistics of numerical columns
df_AviationData.describe()
```

{"summary":"{\n \"name\": \"df_AviationData\",\n \"rows\": 8,\n
\"fields\": [\n   {\n   \"column\": \"Number.of.Engines\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
29275.352572590105,\n   \"min\": 0.0,\n   \"max\": 82805.0,\
n   \"num_unique_values\": 6,\n   \"samples\": [\n
82805.0,\n   1.1465853511261397,\n   8.0\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Total.Fatal.Injuries\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
27378.479009832645,\n   \"min\": 0.0,\n   \"max\": 77488.0,\
n   \"num_unique_values\": 5,\n   \"samples\": [\n
0.6478551517654346,\n   349.0,\n   5.485960107558412\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"Total.Serious.Injuries\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
26995.889138086313,\n   \"min\": 0.0,\n   \"max\": 76379.0,\
n   \"num_unique_values\": 5,\n   \"samples\": [\n
0.27988059545162935,\n   161.0,\n   1.544083645233758\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"Total.Minor.Injuries\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
27189.05531192526,\n   \"min\": 0.0,\n   \"max\": 76956.0,\n
\"num_unique_values\": 5,\n   \"samples\": [\n
0.3570611778158948,\n   380.0,\n   2.2356253196561946\n

],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Total.Uninjured\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 29300.669351650497,\n   \"min\": 0.0,\n   \"max\": 82977.0,\n   \"num_unique_values\": 7,\n   \"samples\": [\n   82977.0,\n   5.325439579642552,\n   2.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe"}

```
# display the dataframe summary
df_AviationData.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 88889 entries, 0 to 88888
Data columns (total 31 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Event.Id                88889 non-null  object
 1   Investigation.Type      88889 non-null  object
 2   Accident.Number         88889 non-null  object
 3   Event.Date              88889 non-null  object
 4   Location                88837 non-null  object
 5   Country                 88663 non-null  object
 6   Latitude                34382 non-null  object
 7   Longitude               34373 non-null  object
 8   Airport.Code            50132 non-null  object
 9   Airport.Name            52704 non-null  object
 10  Injury.Severity         87889 non-null  object
 11  Aircraft.damage         85695 non-null  object
 12  Aircraft.Category       32287 non-null  object
 13  Registration.Number     87507 non-null  object
 14  Make                    88826 non-null  object
 15  Model                   88797 non-null  object
 16  Amateur.Built           88787 non-null  object
 17  Number.of.Engines       82805 non-null  float64
 18  Engine.Type             81793 non-null  object
 19  FAR.Description         32023 non-null  object
 20  Schedule                12582 non-null  object
 21  Purpose.of.flight       82697 non-null  object
 22  Air.carrier             16648 non-null  object
 23  Total.Fatal.Injuries    77488 non-null  float64
 24  Total.Serious.Injuries  76379 non-null  float64
 25  Total.Minor.Injuries    76956 non-null  float64
 26  Total.Uninjured         82977 non-null  float64
 27  Weather.Condition       84397 non-null  object
 28  Broad.phase.of.flight   61724 non-null  object
 29  Report.Status           82505 non-null  object
 30  Publication.Date        75118 non-null  object
dtypes: float64(5), object(26)
memory usage: 21.0+ MB
```

Data Preparation

Data cleaning

In this step after I will further explore and understood the data. Then I will be clean it so that what is left are the relevant columns with clean rows

1.Select relevant features

- Begin by selecting the columns that can be used to solve the business problem.These are the ones I shall focus on.
- By focusing on these i will likely uncover patterns and trends early on and avoid noise . It also makes the analysis process easier to interprate.
- During analysis if some features turn out to not be as relevant then they will be dropped

1.1 Dropping columns

- I start by drop irrelevant columns after seeing the summary. These are columns I am sure wont help in my analysis
- I can later on choose to drop other columns after interacting with the data further
- When i have the relevant relevant columns. I go ahead and begin by focusing on features that directly address my business needs
- Later on I can later on decide if I should explore indirect features.

```
#drop columns related to identification,naming and others
key_features = df_AviationData.drop(columns = ['Event.Id',
'Investigation.Type', 'Accident.Number', 'Event.Date',
                        'Airport.Code','Latitude', 'Longitude',
'Airport.Name', 'FAR.Description', 'Registration.Number',
                        'Report.Status', 'Publication.Date']).copy()

#display columns
key_features.columns

Index(['Location', 'Country', 'Injury.Severity', 'Aircraft.damage',
       'Aircraft.Category', 'Make', 'Model', 'Amateur.Built',
       'Number.of.Engines', 'Engine.Type', 'Schedule',
'Purpose.of.flight',
       'Air.carrier', 'Total.Fatal.Injuries',
'Total.Serious.Injuries',
       'Total.Minor.Injuries', 'Total.Uninjured', 'Weather.Condition',
       'Broad.phase.of.flight'],
      dtype='object')
```

1.2 Type of columns

- Identifying what type of columns I have are they:

1. Categorical
2. Numerical
3. Columns dealing with time
4. Columns containing written text

```
# display a general overview of all columns including categorical with the highest mode and frequency
df_AviationData.describe(include='object')
```

{"type":"dataframe"}

```
#display general information
key_features.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 88889 entries, 0 to 88888
Data columns (total 19 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   Location                88837 non-null  object 
 1   Country                 88663 non-null  object 
 2   Injury.Severity         87889 non-null  object 
 3   Aircraft.damage         85695 non-null  object 
 4   Aircraft.Category       32287 non-null  object 
 5   Make                    88826 non-null  object 
 6   Model                   88797 non-null  object 
 7   Amateur.Built           88787 non-null  object 
 8   Number.of.Engines       82805 non-null  float64
 9   Engine.Type             81793 non-null  object 
 10  Schedule                12582 non-null  object 
 11  Purpose.of.flight       82697 non-null  object 
 12  Air.carrier             16648 non-null  object 
 13  Total.Fatal.Injuries    77488 non-null  float64
 14  Total.Serious.Injuries  76379 non-null  float64
 15  Total.Minor.Injuries    76956 non-null  float64
 16  Total.Uninjured         82977 non-null  float64
 17  Weather.Condition       84397 non-null  object 
 18  Broad.phase.of.flight   61724 non-null  object 
dtypes: float64(5), object(14)
memory usage: 12.9+ MB
```

2.Deal with missing values

```
# display total number of rows
print(len(key_features))
#check the summary of missing values in each column
key_features.isna().sum()
```

```
88889
```

```
Location                      52
Country                      226
Injury.Severity             1000
Aircraft.damage             3194
Aircraft.Category          56602
Make                          63
Model                         92
Amateur.Built                102
Number.of.Engines           6084
Engine.Type                 7096
Schedule                   76307
Purpose.of.flight           6192
Air.carrier                72241
Total.Fatal.Injuries       11401
Total.Serious.Injuries     12510
Total.Minor.Injuries       11933
Total.Uninjured             5912
Weather.Condition           4492
Broad.phase.of.flight      27165
dtype: int64
```

```
#drop columns with more than 50% missing value
key_features.drop(key_features.columns[key_features.isna().mean()
>0.5], axis=1, inplace = True)

#confirm row is dropped
key_features.isna().sum()
```

```
Location                      52
Country                      226
Injury.Severity             1000
Aircraft.damage             3194
Make                          63
Model                         92
Amateur.Built                102
Number.of.Engines           6084
Engine.Type                 7096
Purpose.of.flight           6192
Total.Fatal.Injuries       11401
Total.Serious.Injuries     12510
Total.Minor.Injuries       11933
Total.Uninjured             5912
Weather.Condition           4492
Broad.phase.of.flight      27165
dtype: int64
```

3. Imputing

For categorical data impute with the mode while for numerical columns I impute with median since it is the safest as mean can be affected by outliers if they exist

```
#Impute the mode for the columns containing categories
key_features['Injury.Severity'] =
key_features['Injury.Severity'].fillna(key_features['Injury.Severity']
.mode()[0])
key_features['Aircraft.damage'] =
key_features['Aircraft.damage'].fillna(key_features['Aircraft.damage']
.mode()[0])
key_features['Make'] =
key_features['Make'].fillna(key_features['Make'].mode()[0])
key_features['Model'] =
key_features['Model'].fillna(key_features['Model'].mode()[0])
key_features['Amateur.Built'] =
key_features['Amateur.Built'].fillna(key_features['Amateur.Built'].mod
e()[0])
key_features['Engine.Type'] =
key_features['Engine.Type'].fillna(key_features['Engine.Type'].mode()
[0])
key_features['Purpose.of.flight'] =
key_features['Purpose.of.flight'].fillna(key_features['Purpose.of.flig
ht'].mode()[0])
key_features['Weather.Condition'] =
key_features['Weather.Condition'].fillna(key_features['Weather.Conditi
on'].mode()[0])
key_features['Broad.phase.of.flight'] =
key_features['Broad.phase.of.flight'].fillna(key_features['Broad.phase
.of.flight'].mode()[0])

#Impute the median for the columns containing numeric data
key_features['Number.of.Engines'] =
key_features['Number.of.Engines'].fillna(key_features['Number.of.Engin
es'].median())
key_features['Total.Fatal.Injuries'] =
key_features['Total.Fatal.Injuries'].fillna(key_features['Total.Fatal.
Injuries'].median())
key_features['Total.Serious.Injuries'] =
key_features['Total.Serious.Injuries'].fillna(key_features['Total.Seri
ous.Injuries'].median())
key_features['Total.Minor.Injuries'] =
key_features['Total.Minor.Injuries'].fillna(key_features['Total.Minor.
Injuries'].median())
key_features['Total.Uninjured'] =
key_features['Total.Uninjured'].fillna(key_features['Total.Uninjured']
.median())

#check for any missing values
key_features.isna().sum()
```

```
Location                52
Country                226
Injury.Severity          0
```

```
Aircraft.damage           0
Make                      0
Model                     0
Amateur.Built             0
Number.of.Engines         0
Engine.Type               0
Purpose.of.flight         0
Total.Fatal.Injuries      0
Total.Serious.Injuries    0
Total.Minor.Injuries      0
Total.Uninjured           0
Weather.Condition         0
Broad.phase.of.flight     0
dtype: int64
```

4. Deal with duplicate values in the dataframe

```
#check for duplicates
display(key_features.duplicated().sum())
```

```
np.int64(563)
```

```
#drop duplicates
key_features = key_features.drop_duplicates()
#confirm duplicates are dropped
key_features.duplicated().sum()
```

```
np.int64(0)
```

Validate that the datatypes are correct

- Categories columns should have object datatype
- Numerical columns should have float or int

```
# validate datatypes categories are objects while numerics are float
key_features.dtypes
```

```
Location                   object
Country                    object
Injury.Severity            object
Aircraft.damage            object
Make                       object
Model                      object
Amateur.Built              object
Number.of.Engines         float64
Engine.Type                object
Purpose.of.flight          object
Total.Fatal.Injuries      float64
Total.Serious.Injuries    float64
Total.Minor.Injuries      float64
Total.Uninjured           float64
```

```
Weather.Condition          object
Broad.phase.of.flight      object
dtype: object
```

5. Ensure consistent formatting in the Categorical columns

```
#check for consistent formatting
print(key_features['Injury.Severity'].value_counts())
print(key_features['Aircraft.damage'].unique())
print(key_features['Make'].unique())
print(key_features['Model'].unique())
print(key_features['Amateur.Built'].unique())
print(key_features['Engine.Type'].unique())
print(key_features['Purpose.of.flight'].unique())
print(key_features['Weather.Condition'].unique())
```

```
Injury.Severity
Non-Fatal     67813
Fatal(1)       6163
Fatal          5262
Fatal(2)       3704
Incident       2215
              ...
Fatal(96)         1
Fatal(89)         1
Fatal(199)        1
Fatal(114)        1
Fatal(57)         1
Name: count, Length: 109, dtype: int64
['Destroyed' 'Substantial' 'Minor' 'Unknown']
['Stinson' 'Piper' 'Cessna' ... 'JAMES R DERNOVSEK' 'ORLICAN S R O'
 'ROYSE RALPH L']
['108-3' 'PA24-180' '172M' ... 'ROTORWAY EXEC 162-F' 'KITFOX S5'
 'M-8 EAGLE']
['No' 'Yes']
['Reciprocating' 'Turbo Fan' 'Turbo Shaft' 'Unknown' 'Turbo Prop'
 'Turbo Jet' 'Electric' 'Hybrid Rocket' 'Geared Turbofan' 'LR' 'NONE'
 'UNK']
['Personal' 'Business' 'Instructional' 'Unknown' 'Ferry'
 'Executive/corporate' 'Aerial Observation' 'Aerial Application'
 'Public Aircraft' 'Skydiving' 'Other Work Use' 'Positioning'
 'Flight Test' 'Air Race/show' 'Air Drop' 'Public Aircraft - Federal'
 'Glider Tow' 'Public Aircraft - Local' 'External Load'
 'Public Aircraft - State' 'Banner Tow' 'Firefighting' 'Air Race show'
 'PUBS' 'ASHO' 'PUBL']
['UNK' 'IMC' 'VMC' 'Unk']
```

```
# #perform formatting on unconsistent feature
# key_features['Weather.Condition']=
key_features['Weather.Condition'].str.title()
```

```python
# Apply title case to all string columns in the DataFrame
for column in key_features.select_dtypes(include='object').columns:
    key_features[column] = key_features[column].str.title()

# validate the changes for one column
key_features['Weather.Condition'].unique()
```

```
array(['Unk', 'Imc', 'Vmc'], dtype=object)
```

6. Validate that the data is clean

```python
#display overview
key_features.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 88326 entries, 0 to 88888
Data columns (total 16 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Location                88274 non-null  object
 1   Country                 88100 non-null  object
 2   Injury.Severity         88326 non-null  object
 3   Aircraft.damage         88326 non-null  object
 4   Make                    88326 non-null  object
 5   Model                   88326 non-null  object
 6   Amateur.Built           88326 non-null  object
 7   Number.of.Engines       88326 non-null  float64
 8   Engine.Type             88326 non-null  object
 9   Purpose.of.flight       88326 non-null  object
 10  Total.Fatal.Injuries    88326 non-null  float64
 11  Total.Serious.Injuries  88326 non-null  float64
 12  Total.Minor.Injuries    88326 non-null  float64
 13  Total.Uninjured         88326 non-null  float64
 14  Weather.Condition       88326 non-null  object
 15  Broad.phase.of.flight   88326 non-null  object
dtypes: float64(5), object(11)
memory usage: 11.5+ MB
```

Data Analysis

Here I identify key insights from the cleaned dataset to help the company make data-driven decision about which aircraft to buy

Create new feature to combine multiple features

Now i can focus on the two columns which were Total.Injuries and Total.Uninjured.

```python
#combine the 3 injury columns
key_features['Total.Injuries'] = key_features['Total.Fatal.Injuries']
+ key_features['Total.Serious.Injuries'] +
```

```
key_features['Total.Minor.Injuries']

#drop those 3 columns and focus only on total injuries column
key_features = key_features.drop(columns = ['Total.Fatal.Injuries',
'Total.Serious.Injuries', 'Total.Minor.Injuries'])
```

Create columns to calculate the Injury rate and the Uninjured rate

- However using the raw values like this may not give fair comparison especially if some aircraft characteristics are more common in the dataset which could potentially lead to more accidents but that doesn't necessarily mean that they are more dangerous than than the other.
- E.G., Model A can have 10 injuries and 14 uninjured (Total= 24)while model B can have 5 injuries and 3 uninjured (Total = 8) when we compute the percentage we find that model A has an injury rate of 42% while model B has 63% making model A which has higher injuries safer
- `Since the injury rate and uninjured rate are complementary then i can only focus on creating one feature. E.g., if injury rate is 60% then it means uninjured rate is 40%. I will then use in my this feature in my analysis to find aircraft with lowest risk of injury`

```
#crete column to storing the percentage of injury
key_features['Injury.Rate(%)'] = (key_features['Total.Injuries'] /
(key_features['Total.Injuries']+ key_features['Total.Uninjured'])) *
100

#display overview of the 3 columns
key_features[['Total.Injuries', 'Total.Uninjured',
'Injury.Rate(%)']].head()
```

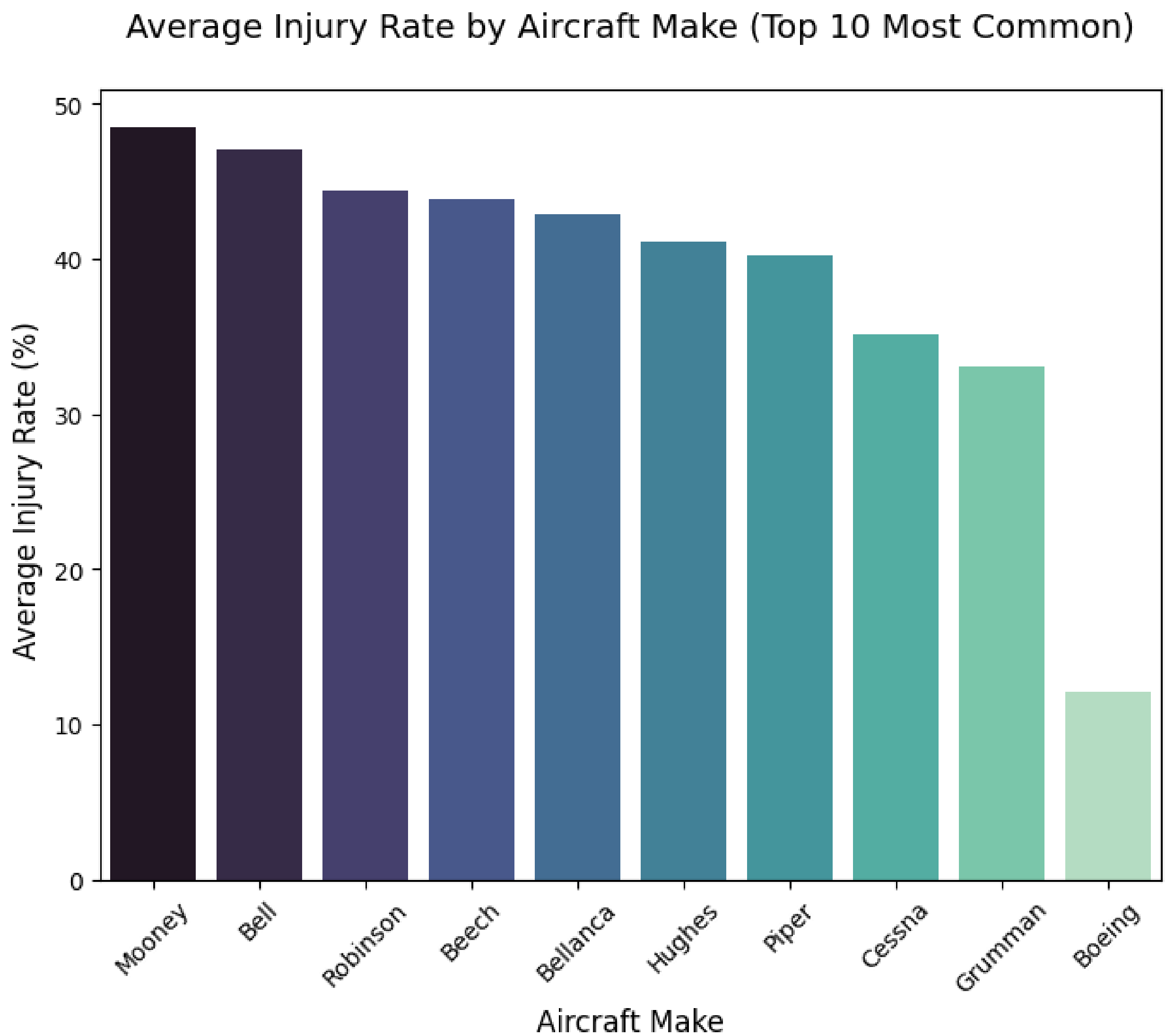

```
{"summary":"{\n  \"name\": \"key_features[['Total\",\n  \"rows\": 5,\n
\"fields\": [\n    {\n      \"column\": \"Total.Injuries\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.8366600265340756,\n        \"min\": 2.0,\n        \"max\": 4.0,\n
\"num_unique_values\": 3,\n        \"samples\": [\n          2.0,\n
4.0,\n          3.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Total.Uninjured\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0.44721359549995804,\n        \"min\":
0.0,\n        \"max\": 1.0,\n        \"num_unique_values\": 2,\n
\"samples\": [\n          1.0,\n          0.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Injury.Rate(%)\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
11.180339887498949,\n        \"min\": 75.0,\n        \"max\": 100.0,\n
\"num_unique_values\": 2,\n        \"samples\": [\n          75.0,\n
```

```
100.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\n}","type":"dataframe"}
```


Visualization

1. Average Injury Rate by Aircraft Make

```
# Calculate the average Injury Rate for top 10 makes
make_injury_rate = (

key_features[key_features['Make'].isin(key_features['Make'].value_coun
ts().head(10).index)]
    .groupby('Make')['Injury.Rate(%)']
    .mean()
    .sort_values(ascending=False)
)

# Plot
plt.figure(figsize=(8, 6))
sns.barplot(x=make_injury_rate.index, y=make_injury_rate.values,
            palette = sns.color_palette("mako",
n_colors=len(make_injury_rate))
            , hue=make_injury_rate.index)

plt.title("Average Injury Rate by Aircraft Make (Top 10 Most Common)\
n", fontsize=14)
plt.xlabel("Aircraft Make", fontsize=12)
plt.ylabel("Average Injury Rate (%) ", fontsize=12)
plt.xticks(rotation=45)
plt.show();
```

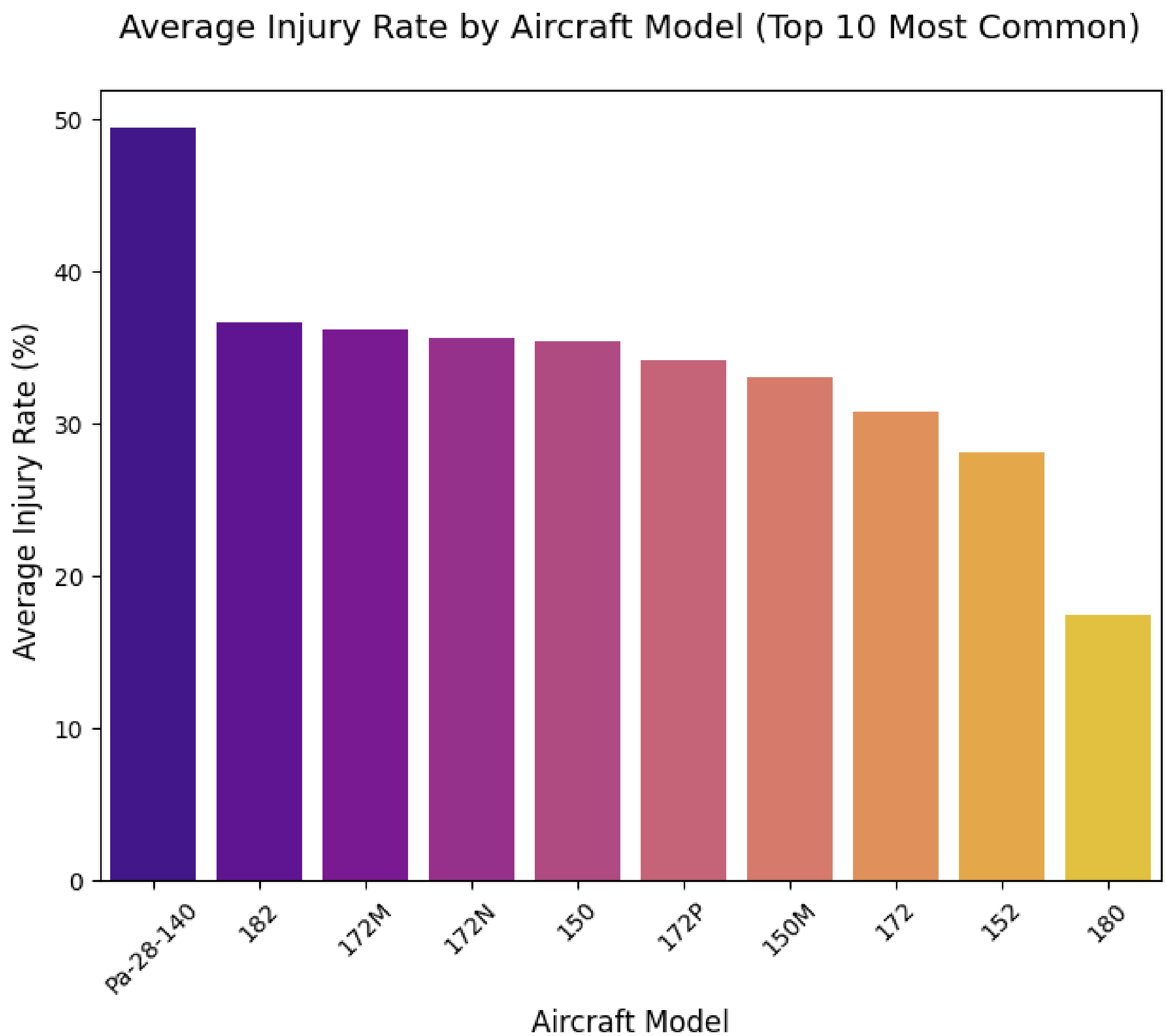


Insight -Aircraft make Boeing has consistently lower injury rates, indicating better safety performance and less prone to accidents.

2. Average Injury Rate by Aircraft Model

```
# Calculate the average Injury Rate for only these top 10 models
model_injury_rate = (

key_features[key_features['Model'].isin(key_features['Model'].value_co
unts().head(10).index)]
    .groupby('Model')['Injury.Rate(%)']
    .mean()
    .sort_values(ascending=False)
)

# Plot
plt.figure(figsize=(8, 6))
```

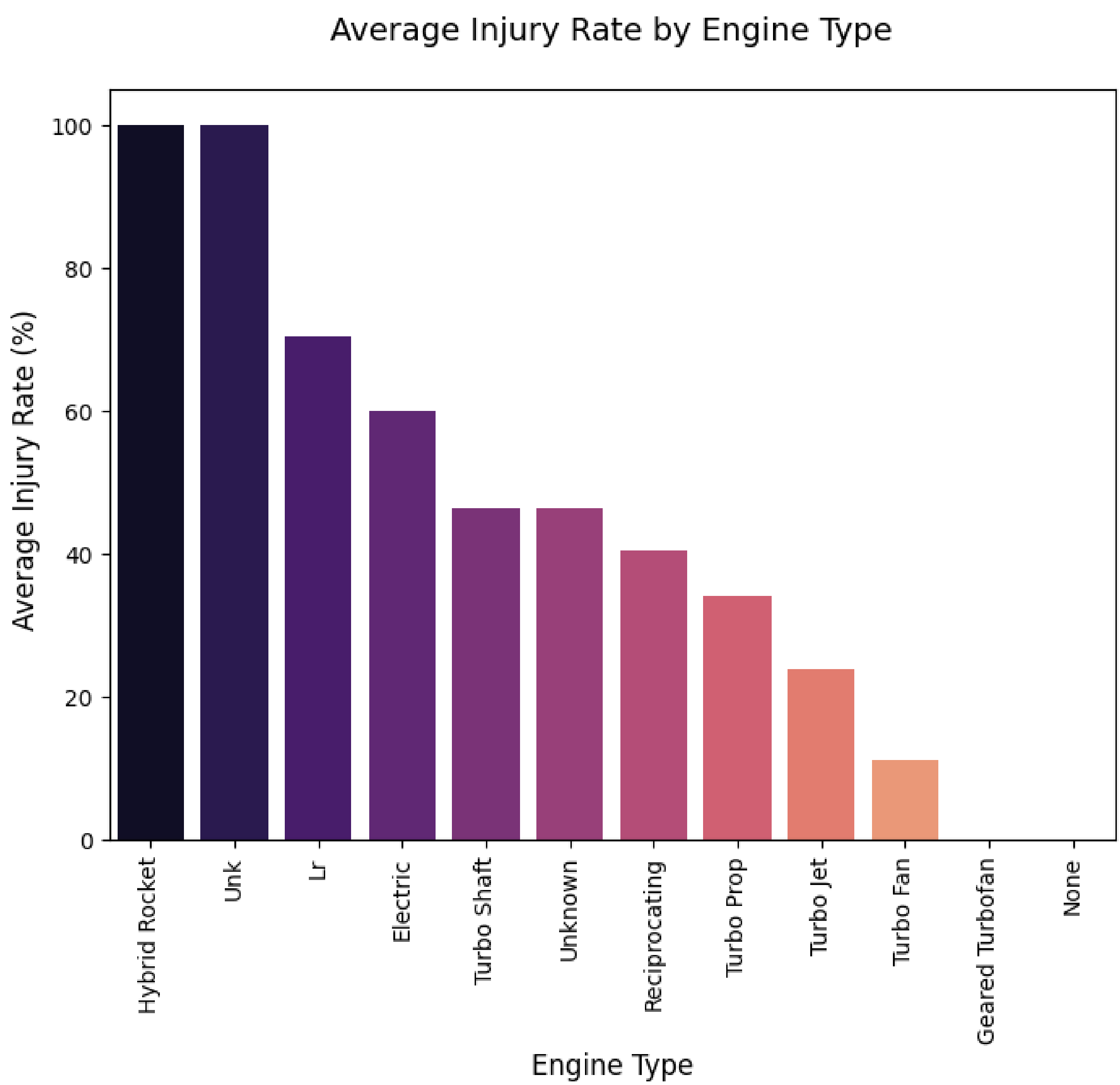
```
sns.barplot(x=model_injury_rate.index, y=model_injury_rate.values,
            palette = sns.color_palette("plasma",
n_colors=len(model_injury_rate))
            , hue=model_injury_rate.index)

plt.title("Average Injury Rate by Aircraft Model (Top 10 Most Common)\
n", fontsize=14)
plt.xlabel("Aircraft Model", fontsize=12)
plt.ylabel("Average Injury Rate (%) ", fontsize=12)
plt.xticks(rotation=45)
plt.show()
```



Insight -Aircraft model 180 and 152 have consistently lower injury rates, indicating better safety performance.

3. Average Injury Rate by Engine Type

```
# Calculate the average Injury Rate for each engine type
engine_injury_rate = (
    key_features.groupby('Engine.Type')['Injury.Rate(%)']
    .mean()
    .sort_values(ascending=False)
)

# Plot
plt.figure(figsize=(8, 6))
sns.barplot(x=engine_injury_rate.index, y=engine_injury_rate.values,
            palette = sns.color_palette("magma",
n_colors=len(engine_injury_rate))
            , hue=engine_injury_rate.index)

plt.title("Average Injury Rate by Engine Type\n", fontsize=14)
plt.xlabel("Engine Type", fontsize=12)
plt.ylabel("Average Injury Rate (%) ", fontsize=12)
plt.xticks(rotation=90)
plt.show()
```

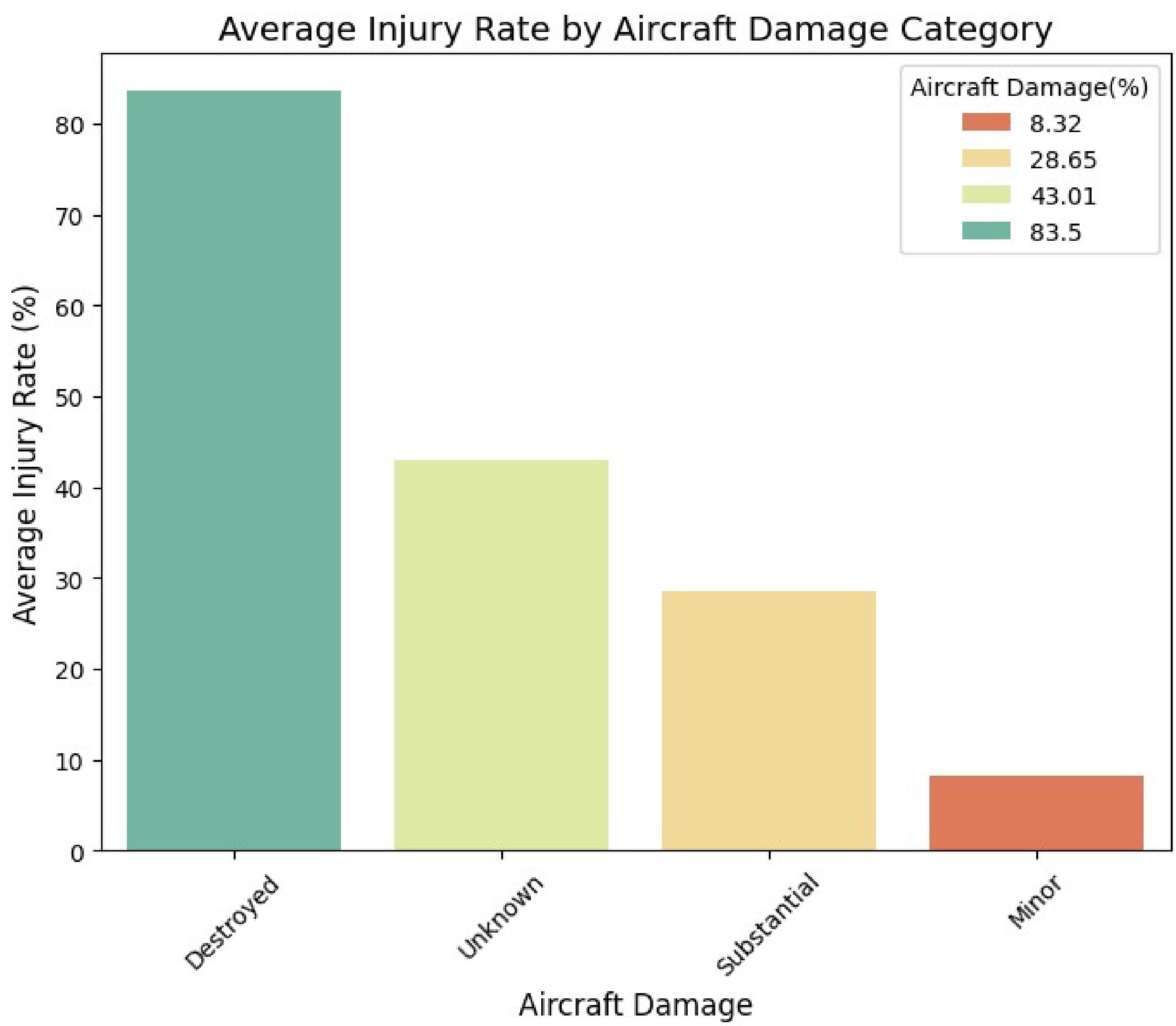


Insight -Most powerful engines have the highest injury rate. best to consider is Turbo Fan or Turbo Jet

4. Average Injury Rate by Aircraft Damage Category

```
# Calculate the average Injury Rate for each Aircraft Damage category
damage_injury_rate = (
    key_features.groupby("Aircraft.damage")["Injury.Rate(%)"]
    .mean().round(2)
    .sort_values(ascending=False)
)

# Plot
plt.figure(figsize=(8, 6))
```

```
sns.barplot(
    x=damage_injury_rate.index,
    y=damage_injury_rate.values,
   palette = sns.color_palette("Spectral",
n_colors=len(damage_injury_rate)),
     hue=damage_injury_rate)

plt.legend(title="Aircraft Damage(%)")
plt.title("Average Injury Rate by Aircraft Damage Category",
fontsize=14)
plt.xlabel("Aircraft Damage", fontsize=12)
plt.ylabel("Average Injury Rate (%)", fontsize=12)
plt.xticks(rotation=45)
plt.show();
```

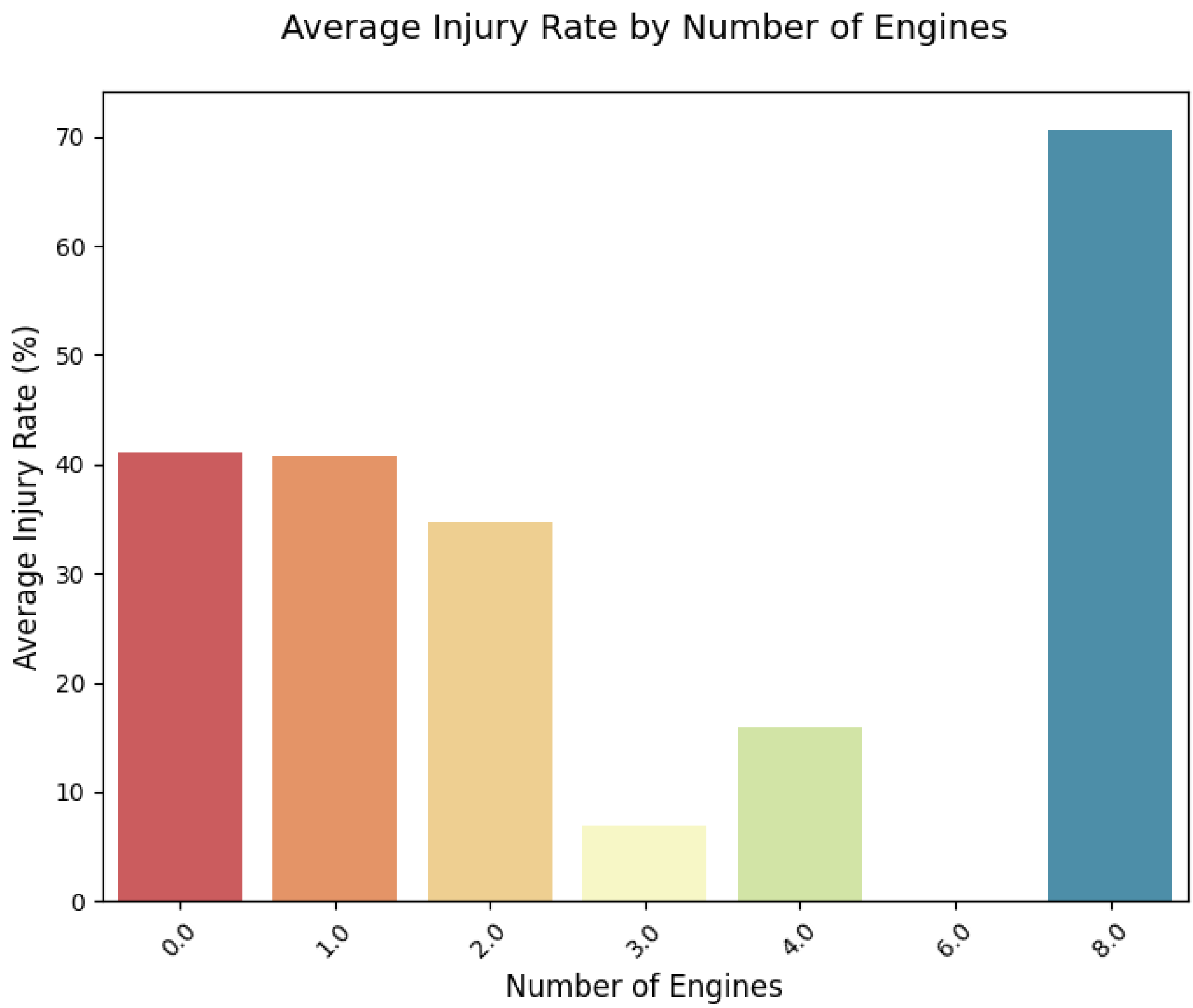


Insight -The more severe the damage e.g., Destroyed, the higher the likelihood of severe injuries.

5. Average Injury Rate by Number of Engines

```
# Calculate the average Injury Rate for each Number of Engines
category
num_engines_injury_rate = (
    key_features.groupby("Number.of.Engines")["Injury.Rate(%)"]
    .mean()
    .round(2)
    .sort_values(ascending=False)
)

# Plot (Vertical Bar Chart)
plt.figure(figsize=(8, 6))
sns.barplot(
    x=num_engines_injury_rate.index,  # Number of Engines on the x-
axis
    y=num_engines_injury_rate.values,  # Injury Rate on the y-axis
    palette=sns.color_palette("Spectral",
n_colors=len(num_engines_injury_rate))  # Gradient color palette
)

# Add title and labels
plt.title("Average Injury Rate by Number of Engines\n", fontsize=14)
plt.xlabel("Number of Engines", fontsize=12)
plt.ylabel("Average Injury Rate (%)", fontsize=12)

# Rotate x-axis labels if needed
plt.xticks(rotation=45)
plt.show()
```

```
<ipython-input-179-3518298609e3>:11: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(
```



Insight -The number of engines doesn't seem to have a relationship with the average injury rate therefore can't consider this factor

Business Recommendations

This analysis leads to **three Business recommendations**:

1. For a safer fleet, it is advisable to prioritize the **Boeing** aircraft, as they appear to be less prone to severe accidents
2. The aircraft models **180** and **152** have shown consistently lower injury rates. Opting for these models could significantly mitigate the risk of injury
3. Aircraft with more powerful engines tend to have higher injury rates. To minimize risk, focus on aircraft powered by **Turbo Fan** or **Turbo Jet engines**, which typically offer a better balance of performance and safety

Conclusion

Based on the analysis, selecting aircraft from manufacturers like **Boeing**, focusing on models such as the **180** and **152,** and choosing aircraft with **Turbo Fan** or **Turbo Jet engines**, will likely reduce the overall risk of accidents and injuries.