DUVERSEAU Jeff Lwear

Projet Phase 1

Data Science

Overview

Ce projet s'inscrit dans le cadre du programme de Data Sciences de Bootcamp Akademi, et marque la fin de la première phase de la formation. Il a pour objectif de tester ma compréhension et ma maîtrise des notions abordées durant cette phase.

Le projet couvre les thématiques suivantes :

- Pandas : manipulation et nettoyage de données
- Data Visualization : représentation graphique des données pour en faciliter l'analyse
- Exploring Statistical Data : exploration et interprétation de données statistiques
- Python Data Structures : utilisation des structures de données fondamentales en Python

À travers ce projet, j'ai appliqué les compétences acquises pour analyser un jeu de données réel, en mettant l'accent sur la rigueur, l'interprétation des résultats et la qualité des visualisations.

Business Problem

L'entreprise se développe dans de nouveaux secteurs afin de diversifier son portefeuille. Elle souhaite notamment acquérir et exploiter des avions pour des entreprises commerciales et privées, mais ignore les risques potentiels liés à l'aviation. Je suis chargé d'identifier les avions présentant le moins de risques pour l'entreprise lors du lancement de cette nouvelle activité. Je dois ensuite traduiremes conclusions en informations exploitables que le responsable de la nouvelle division aéronautique pourra utiliser pour l'aider à choisir les avions à acquérir.

Début

Pour commencer notre analyse, nous avons importé les librairies nécessaires, notamment **pandas**, afin de manipuler efficacement les données.

```
# Importons les librairies
import pandas as pd
import numpy as np
from numbers import Number
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import warnings
warnings.filterwarnings('ignore')
```

Nous avons ensuite chargé l'ensemble de données en créant un DataFrame nommé **df** à partir du fichier **Aviation_Data.csv**. Cette étape nous a permis d'importer un grand volume de données, composé de 88 889 enregistrements répartis sur 31 colonnes.

```
# Lecture du fichier
df = pd.read_csv("AviationData.csv", encoding = "mac_roman ")
```

Data Understanding

En examinant la structure générale du DataFrame, nous avons constaté que la majorité des colonnes contiennent des données de type object (texte), tandis que cinq colonnes sont de type float64 (numérique). Parmi les colonnes texte, certaines, telles que Event.Date et Publication.Date, doivent être converties au format datetime pour faciliter les analyses temporelles.

```
df.dtypes      # Types de données
```

```
Event.Id                object
Investigation.Type      object
Accident.Number         object
Event.Date              object
Location                object
Country                 object
Latitude                object
Longitude               object
Airport.Code            object
Airport.Name            object
Injury.Severity         object
Aircraft.damage         object
Aircraft.Category       object
Registration.Number     object
Make                    object
Model                   object
Amateur.Built           object
Number.of.Engines      float64
Engine.Type             object
FAR.Description         object
Schedule                object
Purpose.of.flight       object
Air.carrier             object
Total.Fatal.Injuries   float64
Total.Serious.Injuries float64
Total.Minor.Injuries   float64
Total.Uninjured        float64
Weather.Condition       object
Broad.phase.of.flight   object
Report.Status           object
Publication.Date        object
dtype: object
```

Nous avons également effectué un aperçu des premières et dernières lignes du DataFrame pour mieux comprendre le contenu et la qualité des données. Cette étape a révélé des informations importantes telles que les types de vol, les lieux des accidents, ainsi que les blessures associées.

```
df.head(10)      # Aperçu des 10 premières lignes
```

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latitude	Longitude	Airport.Code	Airport.Name	...	Purpose.of.flight	Air.carrier	Total.Fatal.Injuries	Total.Serious.Inj
0	20001218X45444	Accident	SEA87LA080	1948-10-24	MOOSE CREEK, ID	United States	NaN	NaN	NaN	NaN	...	Personal	NaN	2.0	
1	20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPORT, CA	United States	NaN	NaN	NaN	NaN	...	Personal	NaN	4.0	
2	20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville, VA	United States	36.922223	-81.878056	NaN	NaN	...	Personal	NaN	3.0	
3	20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA, CA	United States	NaN	NaN	NaN	NaN	...	Personal	NaN	2.0	
4	20041105X01764	Accident	CHI79FA064	1979-08-02	Canton, OH	United States	NaN	NaN	NaN	NaN	...	Personal	NaN	1.0	
5	20170710X52551	Accident	NYC79AA106	1979-09-17	BOSTON, MA	United States	42.445277	-70.758333	NaN	NaN	...	NaN	Air Canada	NaN	
6	20001218X45446	Accident	CHI81LA106	1981-08-01	COTTON, MN	United States	NaN	NaN	NaN	NaN	...	Personal	NaN	4.0	
7	20020909X01562	Accident	SEA82DA022	1982-01-01	PULLMAN, WA	United States	NaN	NaN	NaN	BLACKBURN AG STRIP	...	Personal	NaN	0.0	
8	20020909X01561	Accident	NYC82DA015	1982-01-01	EAST HANOVER, NJ	United States	NaN	NaN	N58	HANOVER	...	Business	NaN	0.0	
9	20020909X01560	Accident	MIA82DA029	1982-01-01	JACKSONVILLE, FL	United States	NaN	NaN	JAX	JACKSONVILLE INTL	...	Personal	NaN	0.0	

10 rows × 31 columns

```
df.tail(10)      # Aperçu des 10 dernières lignes
```

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latitude	Longitude	Airport.Code	Airport.Name	...	Purpose.of.flight	Air.carrier	Total.Fatal.Injuries	Total.Serious.Inju
88879	20221219106472	Accident	DCA23LA096	2022-12-18	Kahului, HI	United States	NaN	NaN	NaN	NaN	...	NaN	HAWAIIAN AIRLINES INC	0.0	
88880	20221219106477	Accident	WPR23LA071	2022-12-18	San Manual, AZ	United States	NaN	NaN	NaN	NaN	...	Personal	Chandler Air Service	0.0	
88881	20221221106483	Accident	CEN23LA067	2022-12-21	Auburn Hills, MI	United States	NaN	NaN	NaN	NaN	...	Personal	Pilot	0.0	
88882	20221222106486	Accident	CEN23LA068	2022-12-21	Reserve, LA	United States	NaN	NaN	NaN	NaN	...	Instructional	NaN	0.0	
88883	20221228106502	Accident	GAA23WA046	2022-12-22	Brasnorte,	Brazil	NaN	NaN	NaN	NaN	...	NaN	NaN	1.0	
88884	20221227106491	Accident	ERA23LA093	2022-12-26	Annapolis, MD	United States	NaN	NaN	NaN	NaN	...	Personal	NaN	0.0	
88885	20221227106494	Accident	ERA23LA095	2022-12-26	Hampton, NH	United States	NaN	NaN	NaN	NaN	...	NaN	NaN	0.0	
88886	20221227106497	Accident	WPR23LA075	2022-12-26	Payson, AZ	United States	341525N	1112021W	PAN	PAYSON	...	Personal	NaN	0.0	
88887	20221227106498	Accident	WPR23LA076	2022-12-26	Morgan, UT	United States	NaN	NaN	NaN	NaN	...	Personal	MC CESSNA 210N LLC	0.0	
88888	20221230106513	Accident	ERA23LA097	2022-12-29	Athens, GA	United States	NaN	NaN	NaN	NaN	...	Personal	NaN	0.0	

10 rows × 31 columns

L’exploration approfondie via la fonction .isna() a montré que plusieurs colonnes contiennent un nombre significatif de valeurs manquantes. Par exemple, les coordonnées géographiques (Latitude, Longitude) sont souvent absentes, ainsi que certaines données spécifiques aux avions ou aux phases de vol.

```
#Vérification du nombre de valeurs manquantes dans chaque colonne
df.isna().sum()
```

```
Event.Id                    0
Investigation.Type          0
Accident.Number             0
Event.Date                  0
Location                   52
Country                   226
Latitude                54507
Longitude               54516
Airport.Code            38757
Airport.Name            36185
Injury.Severity          1000
Aircraft.damage          3194
Aircraft.Category       56602
Registration.Number      1382
Make                       63
Model                      92
Amateur.Built             102
Number.of.Engines        6084
Engine.Type              7096
FAR.Description         56866
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
Report.Status            6384
Publication.Date        13771
dtype: int64
```

La vérification des valeurs manquantes a confirmé ce constat, mettant en lumière l’importance de traiter ces données manquantes dans les étapes ultérieures afin de garantir la fiabilité de l’analyse.

Cette phase de compréhension des données a ainsi posé les bases pour orienter les traitements, les nettoyages et les analyses futures, en tenant compte des particularités et des limites du dataset.

Data Cleanning

Dans le cadre de notre analyse des accidents aériens, la première étape essentielle a été de préparer proprement les données. Nous avons commencé par examiner la colonne Event.Id, qui représente l'identifiant unique de chaque événement. En théorie, chaque ligne du dataset devrait correspondre à un événement unique. Pourtant, en comptant le nombre de valeurs uniques dans cette colonne, nous avons constaté qu’il y en avait 87 951, alors que le dataset contient 88 889 lignes. Cela signalait la présence de doublons.

```
# Vérifions s’il y a des doublons dans la colonne Event.Id.
df['Event.Id'].nunique() # affichons les valeurs uniques de la colonne Event.Id
```

Pour confirmer cela, nous avons extrait toutes les lignes présentant des doublons sur cette colonne. Effectivement, certains événements étaient répertoriés plusieurs fois, souvent avec des différences mineures sur d'autres colonnes.

```
df[df.duplicated(subset='Event.Id', keep=False)]
```

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latitude	Longitude	Airport.Code	Airport.Name	...	Purpose.of.flight	Air.carrier	Total.Fatal.Injuries	Total.Seric
117	20020917X01908	Accident	DCA82AA012B	1982-01-19	ROCKPORT, TX	United States	NaN	NaN	RKP	ARANSAS COUNTY AIRPORT	...	Personal	NaN	3.0	
118	20020917X01908	Accident	DCA82AA012A	1982-01-19	ROCKPORT, TX	United States	NaN	NaN	RKP	ARANSAS COUNTY AIRPORT	...	Executive/corporate	NaN	3.0	
153	20020917X02259	Accident	LAX82FA049A	1982-01-23	VICTORVILLE, CA	United States	NaN	NaN	NaN	NaN	...	Personal	NaN	2.0	
158	20020917X02400	Accident	MIA82FA038B	1982-01-23	NEWPORT RICHEY, FL	United States	NaN	NaN	NaN	NaN	...	Personal	NaN	0.0	
159	20020917X02400	Accident	MIA82FA038A	1982-01-23	NEWPORT RICHEY, FL	United States	NaN	NaN	NaN	NaN	...	Personal	NaN	0.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
88796	20221121106336	Accident	WPR23LA041	2022-11-18	Las Vegas, NV	United States	361239N	1151140W	VGT	NORTH LAS VEGAS	...	Instructional	702 HELICOPTER INC	0.0	
88797	20221122106340	Incident	DCA23WA071	2022-11-18	Marrakech,	Morocco	NaN	NaN	NaN	NaN	...	NaN	British Airways	0.0	
88798	20221122106340	Incident	DCA23WA071	2022-11-18	Marrakech,	Morocco	NaN	NaN	NaN	NaN	...	NaN	Valair Private Jets	0.0	
88813	20221123106354	Accident	WPR23LA045	2022-11-22	San Diego, CA	United States	323414N	1165825W	SDM	Brown Field Municipal Airport	...	Instructional	HeliStream Inc.	0.0	
88814	20221123106354	Accident	WPR23LA045	2022-11-22	San Diego, CA	United States	323414N	1165825W	SDM	Brown Field Municipal Airport	...	Public Aircraft - Federal	U.S. Navy	0.0	

1874 rows × 31 columns

Afin de garantir l'unicité de nos données, nous avons supprimé ces doublons, en ne conservant que la première occurrence de chaque identifiant.

```
df = df.drop_duplicates(subset='Event.Id', keep='first')
```

Une fois les doublons éliminés, nous en avons profité pour corriger le type de la colonne Event.Date, qui était initialement interprétée comme une chaîne de caractères (object). La conversion vers un format de date (datetime) nous a permis de faciliter les manipulations futures sur cette dimension temporelle.

```
df['Event.Date'] = pd.to_datetime(df['Event.Date'], errors='coerce')
```

```
print(df['Event.Date'].dtype)  # Doit afficher: datetime64[ns]
print(df['Event.Date'].head()) # Affiche les premières dates
print("Valeurs non converties (NaT) :", df['Event.Date'].isna().sum())
```

```
datetime64[ns]
0   1948-10-24
1   1962-07-19
2   1974-08-30
3   1977-06-19
4   1979-08-02
Name: Event.Date, dtype: datetime64[ns]
Valeurs non converties (NaT) : 0
```

Ensuite, pour simplifier l'analyse chronologique, nous avons extrait l'année de chaque événement et l'avons stockée dans une nouvelle colonne appelée Event.Year. Grâce à cela, nous avons pu constater que notre jeu de données couvre une large période, allant de 1948 à 2022.

```
#Définissons une colonne « Année » et affichons les années uniques
df["Event.Year"] = df.loc[:,("Event.Date")].dt.year
df["Event.Year"].unique()
```

```
array([1948, 1962, 1974, 1977, 1979, 1981, 1982, 1983, 1984, 1985, 1986,
       1987, 1988, 1989, 1990, 1991, 1992, 1993, 1994, 1995, 1996, 1997,
       1998, 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008,
       2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019,
       2020, 2021, 2022])
```

Notre dataset couvre une période allant de l'année 1948 jusqu'à l'année 2022.

Nous avons porté notre attention sur les colonnes de type texte. Une problématique courante dans les données textuelles est l'incohérence de casse : une même valeur peut apparaître tantôt en majuscules, tantôt en minuscules. Pour prévenir ce type d'erreurs, nous avons uniformisé toutes les chaînes de caractères en minuscules.

Uniformisons la casse dans les colonnes de type texte (Object) pour éviter les doublons invisibles causés par des majuscules/minuscules incohérentes.

```
# Identifions les colonnes de type object
cat_cols = df.select_dtypes(include='object').columns

# Appliquons str.lower() à toutes ces colonnes
for col in cat_cols:
    df[col] = df[col].str.lower()
```

Nous avons analysé le taux de valeurs manquantes dans chaque colonne. Pour conserver la qualité et la fiabilité de notre analyse, nous avons décidé de supprimer toutes les colonnes qui comportaient plus de 40 % de données manquantes. Ce filtrage nous a permis de réduire notre jeu de données à ses colonnes les plus exploitables, sans compromettre son intégrité analytique.

Supprimons les colonnes avec plus de 40 % de valeurs manquantes, cela nous permet de conserver uniquement les variables les plus exploitables.

```
# Seuil de tolérance : 40% de valeurs manquantes
seuil = 0.4

# Calcul du pourcentage de valeurs manquantes pour chaque colonne
missing_ratio = df.isnull().mean()

# Filtrage des colonnes à conserver
columns_to_keep = missing_ratio[missing_ratio <= seuil].index

# Nouveau DataFrame avec uniquement les colonnes retenues
df = df[columns_to_keep]
```

Après avoir nettoyé les doublons et structuré les données temporelles, il devenait essentiel de rationaliser les colonnes disponibles. Certaines d'entre elles, bien que potentiellement informatives dans un contexte administratif ou réglementaire, ne servaient pas notre objectif analytique. C'est ainsi que nous avons retiré des variables telles que : Accident.Number, Event.Date, ou encore

Location, car elles n'apportaient ni valeur ajoutée à l'analyse statistique, ni leviers explicatifs concrets.

Certaines colonnes semblent peu pertinentes pour mon analyse, donc je les supprime afin de simplifier la gestion des données.

```
columns_to_drop = [
    'Accident.Number',
    'Registration.Number',
    'Publication.Date',
    'Report.Status',
    'Event.Id',
    'Event.Date',
    'Location',
    'Country',
    'Injury.Severity',


]
df = df.drop(columns=columns_to_drop)
```

En étudiant la répartition annuelle des événements via la variable Event.Year, une observation cruciale s'est dégagée : la majorité des enregistrements débute réellement à partir de 1982. Avant cette date, les cas sont rares, voire anecdotiques. Pour assurer une meilleure représentativité et éviter des biais statistiques liés à des années trop peu documentées, nous avons donc limité l'analyse aux événements survenus à partir de 1982.

```
df = df[df['Event.Year'] >= 1982]
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 87944 entries, 7 to 88888
Data columns (total 15 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Investigation.Type      87944 non-null  object
 1   Aircraft.damage         84841 non-null  object
 2   Make                    87881 non-null  object
 3   Model                   87852 non-null  object
 4   Amateur.Built           87844 non-null  object
 5   Number.of.Engines       81918 non-null  float64
 6   Engine.Type             80902 non-null  object
 7   Purpose.of.flight       81823 non-null  object
 8   Total.Fatal.Injuries    76678 non-null  float64
 9   Total.Serious.Injuries  75624 non-null  float64
 10  Total.Minor.Injuries    76186 non-null  float64
 11  Total.Uninjured         82082 non-null  float64
 12  Weather.Condition       83471 non-null  object
 13  Broad.phase.of.flight   60830 non-null  object
 14  Event.Year              87944 non-null  int32
dtypes: float64(5), int32(1), object(9)
memory usage: 10.4+ MB
```

Vint ensuite le moment d’aborder les valeurs manquantes, qui sont fréquentes dans des bases de données aussi vastes et hétérogènes. Nous avons commencé par les colonnes relatives aux blessures (Total.Fatal.Injuries, Total.Serious.Injuries, etc.). Ici, le contexte est déterminant : une cellule vide dans ce type de colonne ne signifie probablement pas "inconnu", mais plutôt "aucune blessure". Sur cette base, nous avons décidé de remplacer toutes les valeurs manquantes par 0, ce qui donne une lecture bien plus fidèle à la réalité des événements.

Dans le contexte de ces colonnes (nombre de blessés par gravité et de personnes indemnes dans un accident d’aviation), une valeur manquante ne signifie pas "valeur inconnue", mais très probablement "aucune personne de cette catégorie". Donc, nous allons remplacer les valeurs manquantes par 0.

```
df[['Total.Fatal.Injuries', 'Total.Serious.Injuries', 'Total.Minor.Injuries', 'Total.Uninjured']] = \
    df[['Total.Fatal.Injuries', 'Total.Serious.Injuries', 'Total.Minor.Injuries', 'Total.Uninjured']].fillna(0)
```

D’autres colonnes ont exigé un traitement plus spécifique. Par exemple, Number.of.Engines, bien qu’affichée comme numérique, ne pouvait pas recevoir une valeur nulle en cas de données manquantes, car cela reviendrait à supposer qu’un avion a pu voler sans moteur. Une hypothèse évidemment incorrecte. Étant donné que seules quelques lignes étaient concernées, nous avons choisi la méthode la plus fiable : supprimer uniquement ces entrées.

Bien que la colonne Number.of.Engines soit de type float, on ne peut pas remplacer les valeurs manquantes (NaN) par 0, car cela indiquerait que l'appareil n'a pas de moteur, ce qui serait faux ou très peu probable. Une telle substitution pourrait biaiser fortement les résultats lors des analyses, notamment celles basées sur la performance ou la gravité des accidents en lien avec le nombre de moteurs.

De plus, comme cette colonne contient très peu de valeurs manquantes, la solution la plus fiable est de supprimer uniquement les lignes concernées.

```
# Suppression des lignes où Number.of.Engines est manquant
df = df.dropna(subset=['Number.of.Engines'])
```

Nous nous sommes intéressés à la colonne Weather.Condition, qui indique les conditions météorologiques au moment de l’accident. Celle-ci comprenait des catégories comme vmc (vol à vue), imc (vol aux instruments), mais aussi unk (inconnu) et des valeurs manquantes (NaN). Dans un souci de clarté et de cohérence, nous avons pris la décision de regrouper toutes les valeurs manquantes sous la catégorie unk, afin de ne pas perdre d’informations exploitables et d’uniformiser les données "inconnues".

```
#Verifions les differentes valeurs de la colonne Weather.Condition
print(df['Weather.Condition'].unique())
```

```
['vmc' 'imc' 'unk' nan]
```

La colonne Weather.Condition contient 4 types de valeurs :

vmc : Visual Meteorological Conditions (conditions de vol à vue) imc : Instrument Meteorological Conditions (conditions de vol aux instruments)

unk : indique que la condition météorologique est inconnue

NaN (valeurs manquantes) : absence d'information

Choix de traitement Étant donné que unk est déjà utilisé pour signifier « inconnu », nous allons remplacer toutes les valeurs manquantes (NaN) par unk, afin de :

Unifier les valeurs inconnues sous un même libellé clair,

Éviter de perdre des lignes avec des données potentiellement utiles.

```
df['Weather.Condition'] = df['Weather.Condition'].fillna('unk')
```

Nous avons examiné Broad.phase.of.flight, qui décrit la phase de vol au moment de l'incident (décollage, croisière, atterrissage, etc.). Un constat important s'est imposé : plus de 26 % des lignes ne comportaient pas cette information. Remplacer ces valeurs par Unknown aurait eu pour effet de gonfler artificiellement cette catégorie, ce qui risquait de fausser les analyses statistiques. Supprimer ces lignes aurait signifié perdre un quart du jeu de données, ce qui n'était pas envisageable non plus. Nous avons donc choisi de laisser les valeurs manquantes en l'état, et d'adapter leur prise en compte selon les analyses futures.

```
#Verifions les valeurs de la colonne Broad.phase.of.flight
print(df['Broad.phase.of.flight'].unique())
```

```
['takeoff' 'landing' 'cruise' 'approach' 'taxi' 'unknown' 'descent'
 'maneuvering' 'climb' 'standing' 'go-around' 'other' nan]
```

verifions le pourcentage de valeurs manquantes dans la colonne Broad.phase.of.flight.

```
# Calcul du pourcentage de valeurs manquantes
missing_percentage = df['Broad.phase.of.flight'].isna().mean() * 100
missing_percentage
```

```
26.834151224395125
```

Le travail de préparation des données touchait presque à sa fin, mais une dernière série de vérifications s'imposait. De nombreuses colonnes catégorielles (de type object) subsistaient dans le jeu de données, et certaines comportaient encore des valeurs manquantes.

Plutôt que de risquer de perdre des lignes potentiellement riches en informations, j'ai fait le choix d'harmoniser les valeurs absentes dans ces colonnes. Pour cela, toutes les valeurs NaN (non renseignées) ont été remplacées par la chaîne 'unknown', à l'exception de la colonne Broad.phase.of.flight, pour laquelle une stratégie particulière avait déjà été définie.

```
# Liste des colonnes de type object sauf 'Broad.phase.of.flight'
cols_to_fill = [col for col in df.select_dtypes(include='object').columns if col != 'Broad.phase.of.flight']

# Remplacer les NaN par 'unknown' uniquement dans ces colonnes
df[cols_to_fill] = df[cols_to_fill].fillna('unknown')
```

Pour capitaliser ce travail minutieux et préparer le terrain pour une future analyse visuelle (notamment sous Power BI), la version nettoyée du dataset a été exportée sous forme de fichier CSV. Ce fichier constitue désormais une base fiable, structurée, sans valeurs manquantes injustifiées ni incohérences de type. Il est prêt à être utilisé pour des tableaux de bord, des visualisations interactives ou même des modèles prédictifs.

```
#gadons cette version pour dresser un rapport Power BI
df.to_csv("C:\\Users\\djilw\\Bootcamp_Akademi_Phase1_Projet\\Cleanning_Version.csv", index=False)
```

Conclusion de la phase de nettoyage :

Ce long processus de transformation a permis de passer d'un jeu de données brut, désordonné et hétérogène, à un jeu épuré, cohérent et exploitable, tout en conservant un maximum d'informations utiles. Chaque décision a été prise avec rigueur, en tenant compte du sens des données, et non par automatisme.

Analysis and Results

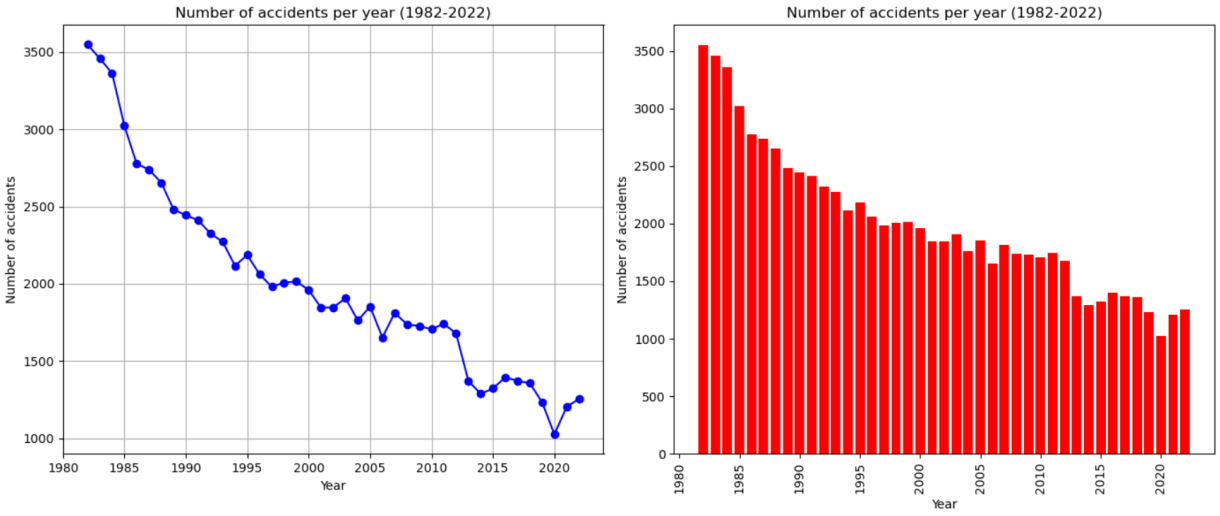
Après avoir consolidé et nettoyé notre jeu de données, nous avons amorcé une exploration chronologique des accidents d'aviation pour identifier les évolutions majeures, les tendances structurelles et les périodes potentiellement critiques.

Évolution du nombre d'accidents dans le temps

Nous avons commencé par observer la fréquence annuelle des accidents entre 1982 et 2022. Deux visualisations complémentaires ont été produites :

- Un graphique linéaire pour suivre visuellement la tendance,
- Un histogramme pour comparer d'un coup d'œil les volumes par année.

```python
# Comptage du nombre d'accidents par année
accidents_per_year = df['Event.Year'].value_counts().sort_index()

# Taille de la figure
plt.figure(figsize=(14, 6))

# 1. Graphique en ligne avec plt.plot
plt.subplot(1, 2, 1)
plt.plot(accidents_per_year.index, accidents_per_year.values, marker='o', linestyle='-', color='blue')
plt.title("Number of accidents per year (1982-2022)")
plt.xlabel("Year")
plt.ylabel("Number of accidents")
plt.grid(True)

# 2. Graphique en barres avec plt.bar
plt.subplot(1, 2, 2)
plt.bar(accidents_per_year.index, accidents_per_year.values, color='red')
plt.title("Number of accidents per year (1982-2022)")
plt.xlabel("Year")
plt.ylabel("Number of accidents")
plt.xticks(rotation=90)

plt.tight_layout()
plt.show()
```

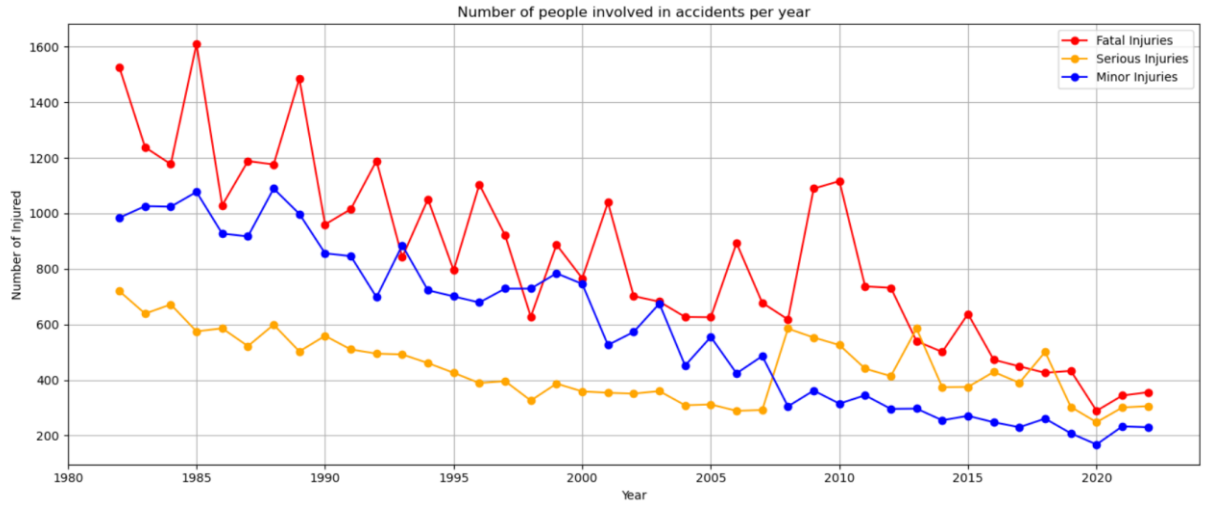


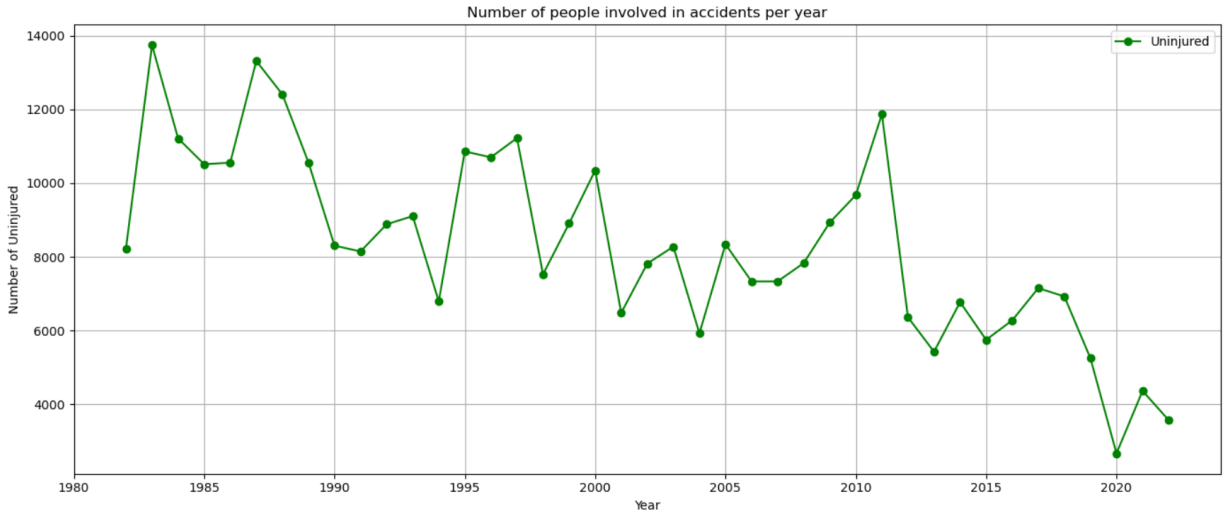
Commentaire : Les deux graphiques confirment une baisse significative et régulière du nombre d'accidents depuis les années 1982.

Nombre de personnes touchées par an (blessures et indemnes)

Afin de mieux comprendre l'ampleur humaine des accidents, nous avons ensuite analysé le nombre de personnes impliquées dans ces événements, en les répartissant par type de conséquence :

Blessures mortelles ; Blessures graves ; Blessures mineures ; Personnes indemnes.





Commentaire :

- ❖ **Tendances des blessures (graphique multi-lignes)**

Trois courbes distinctes ont permis d'examiner l'évolution :

- En rouge, les blessés mortels,
- En orange, les blessés graves,
- En bleu, les blessés légers.

Là encore, une tendance à la baisse claire se dessine, en particulier pour les décès. On note une chute remarquable du nombre de décès au cours des dix dernières années, ce qui constitue un indicateur fort des gains en sécurité et réactivité des services de secours.

De manière générale, toutes les formes de blessures connaissent une régression progressive, bien que certaines années présentent des pics isolés.

❖ Évolution des indemnes (graphique spécifique)

La courbe représentant les passagers indemnes nous renseigne indirectement sur la résilience accrue des systèmes de sécurité, mais aussi sur la capacité à sauver des vies même lors d'accidents.

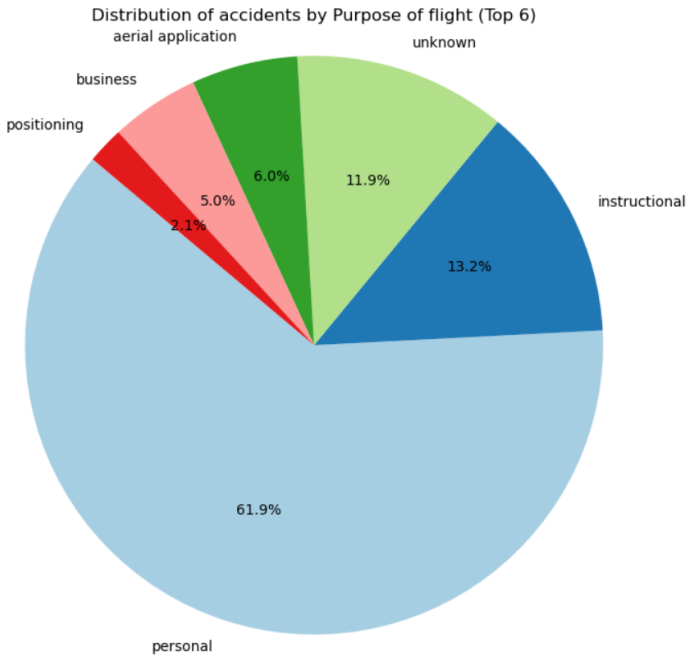
Constat positif : Le nombre de survivants tend à rester élevé dans les années récentes, ce qui renforce le constat global d'une meilleure maîtrise des situations critiques, grâce à l'ingénierie, la formation et la coordination des secours.

Répartition des accidents selon l'objectif de vol

L'objectif du vol joue un rôle fondamental dans la nature et la fréquence des accidents. Afin d'en dégager les tendances majeures, nous avons examiné les six finalités de vol les plus représentées dans notre jeu de données.

❖ Répartition globale (Top 6)

Un graphique circulaire a été utilisé pour illustrer la distribution des accidents selon le type de vol.



Commentaire :

Les résultats montrent clairement que deux catégories se détachent du lot :

- Les vols personnels dominent largement, représentant la part la plus importante des incidents.
- Les vols d'instruction (ou pédagogiques) arrivent en seconde position.

Évolution annuelle des victimes selon l'objectif du vol

Pour aller plus loin, nous avons analysé de façon détaillée l'évolution du nombre de personnes blessées ou indemnes dans les accidents, en fonction de l'objectif du vol. Nous avons notamment comparé deux types de vols bien distincts :

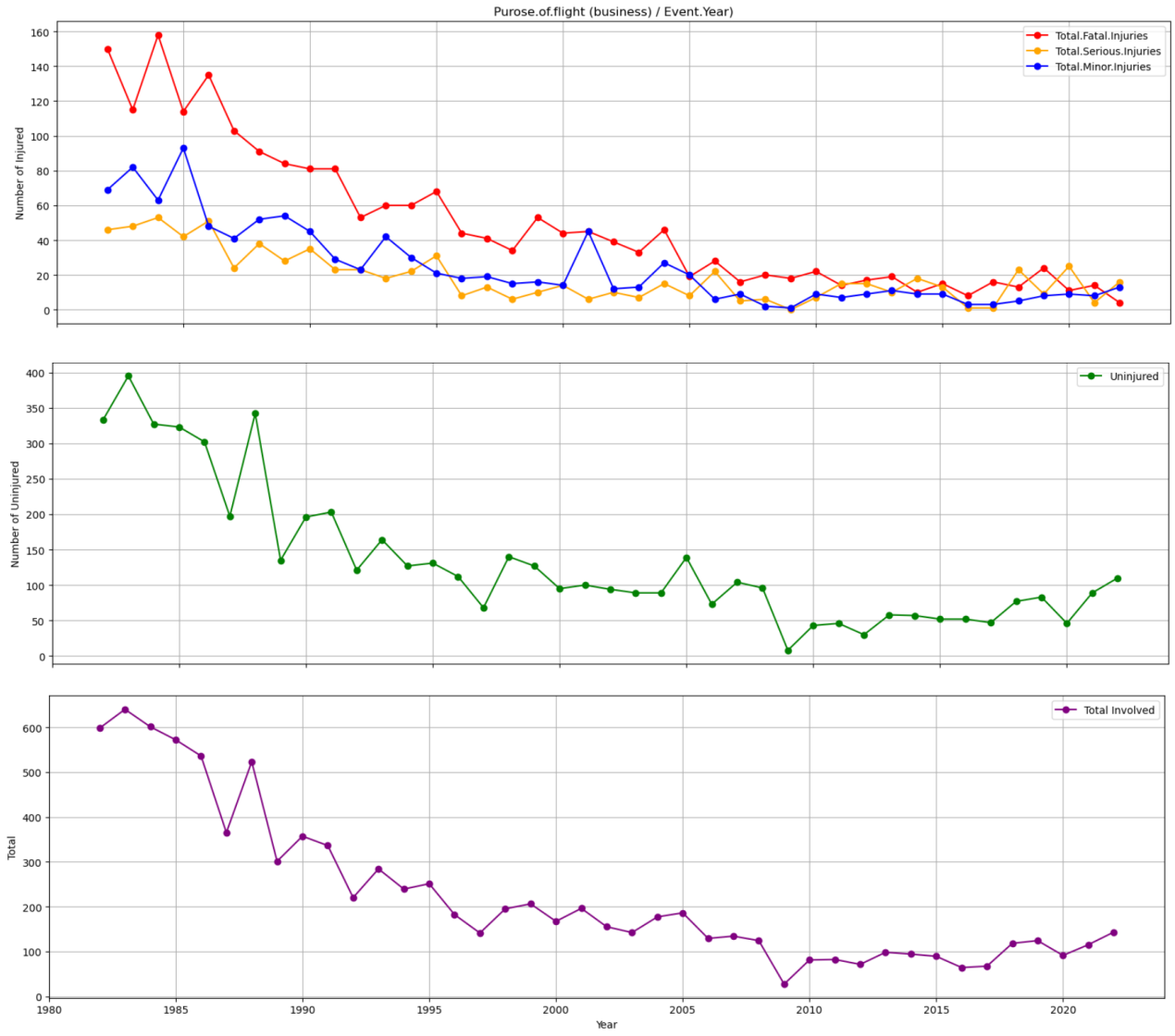
- Les vols à but professionnel (business),
- Les vols à but personnel (personal).

❖ Vols à but professionnel (Business)

Trois graphiques ont été générés :

1. Blessures mortelles, graves et légères
2. Nombre de personnes indemnes
3. Total de personnes impliquées par an

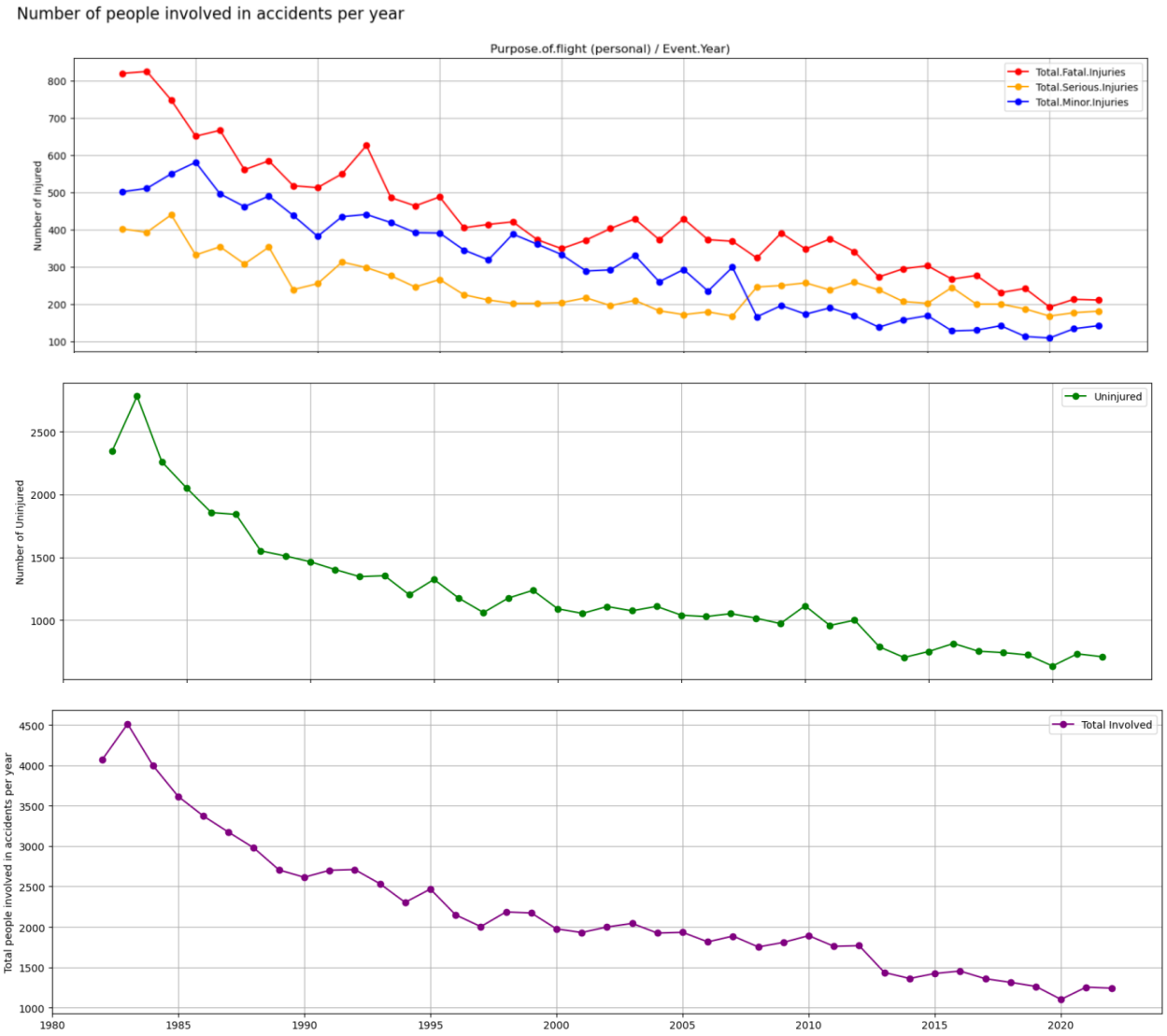
Number of people involved in accidents per year



Commentaire :

Les données révèlent que les accidents dans le cadre de vols professionnels sont globalement moins fréquents et touchent un nombre plus limité de personnes. On note également une baisse continue du nombre de victimes au fil des années.

❖ **Vols à but personnel (Personal)**



Commentaire :

L'analyse des vols personnels montre :

- Des pics de blessés bien plus marqués certaines années,
- Une fréquence plus élevée des accidents, impliquant parfois un plus grand nombre de personnes.

Néanmoins, la tendance générale reste à la baisse, avec une réduction significative des blessures mortelles au cours des dix dernières années.

Répartition des fatalités selon la phase de vol

L'analyse des phases de vol dans lesquelles se produisent les accidents permet d'identifier les moments les plus critiques dans le déroulement d'un vol. Pour cela, nous avons regroupé et comparé le nombre total de blessés mortels (Total.Fatal.Injuries) par grande phase de vol (Broad.phase.of.flight).

❖ Top 5 des phases les plus meurtrières

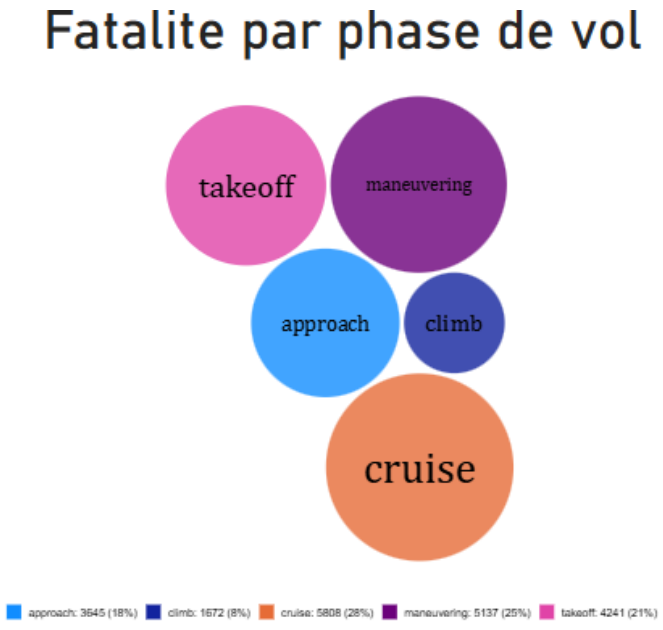
Affichons les top 5 phases de vol associées au plus grand nombre de décès

```
]: # Regrouper par phase de vol et sommer les fatalités
fatalities_by_phase = df.groupby('Broad.phase.of.flight')['Total.Fatal.Injuries'].sum().reset_index()

# Trier et garder les 5 phases les plus mortelles
top5_fatalities = fatalities_by_phase.sort_values(by='Total.Fatal.Injuries', ascending=False).head(5)

# Afficher le DataFrame
print(top5_fatalities)
```

```
  Broad.phase.of.flight  Total.Fatal.Injuries
2                cruise                  5808
6           maneuvering                  5137
9              takeoff                  4241
0              approach                  3645
1                climb                  1672
```

❖ Visualisation : Bubble Chart des phases les plus mortelles

Pour illustrer ces résultats de manière plus visuelle et intuitive, un **graphe en bulles** a été généré. Chaque bulle représente une phase de vol, et sa taille est proportionnelle au **nombre de décès** enregistrés.



Commentaire :

Ce graphique montre clairement que :

- La phase Cruise est associée au plus grand volume de fatalités, représentée par la plus grosse bulle.
- Les phases Maneuvering et Takeoff sont également très critiques.

Détermination des Make_Models les moins risqués : approche en deux temps

L'identification des avions les moins risqués repose sur une analyse en deux dimensions complémentaires. En effet, se baser sur un seul critère peut conduire à des conclusions biaisées. Voici les deux axes d'analyse retenus :

1. **Analyse selon le nombre total de personnes indemnes**

Dans un premier temps, nous groupons les données par Make_Model et classons les aéronefs selon le nombre total de personnes indemnes lors des accidents. L'hypothèse ici est que plus un avion compte de personnes sorties indemnes, plus il est potentiellement sûr. Nous allons ensuite calculer un ratio de risque comme suit :

Risk_Ratio = Nombre total de blessés / (Nombre total de blessés + Nombre total d'indemnes)

Ce ratio permet d'estimer, en moyenne, la probabilité qu'un accident entraîne des blessures plutôt qu'aucune.

Avantage : Cette approche met en avant les avions qui protègent le mieux leurs occupants.

Limite : Certains avions peuvent apparaître comme peu risqués simplement parce qu'ils sont rarement utilisés ou impliqués dans des accidents mineurs.

2. **Analyse selon la fréquence d'implication dans des accidents**

Dans un second temps, nous regardons les modèles d'avions les plus fréquemment impliqués dans des accidents (c'est-à-dire avec un grand nombre d'occurrences). Cela permet d'évaluer la robustesse statistique des données.

Cependant, un grand nombre d'occurrences ne signifie pas nécessairement un risque élevé. En réalité, cela peut simplement refléter une forte popularité ou une large utilisation du modèle. Par exemple, les modèles Cessna sont très présents dans les bases de données d'accidents, car ils sont massivement utilisés dans l'aviation légère.

Ainsi, pour ces modèles à haute fréquence, nous calculons aussi un ratio :

Risk_Ratio = Nombre total de blessés / Nombre total d'indemnes

Cela permet de déterminer si, à fréquence d'usage élevée, un modèle reste globalement sûr ou présente un risque relatif plus élevé.

- ❖ **Première approche : Identifier les Make_Model les plus protecteurs**

Étape 1 : Mesurer les personnes indemnes

Nous avons d'abord regroupé les données par **Make_Model** (constructeur + modèle) pour comptabiliser :

- Le **nombre total de personnes sorties indemnes**,
- Le **nombre total de personnes blessées** (toutes gravités confondues),
- Le **nombre d'accidents impliquant ce modèle**.

L'objectif ici était de repérer les modèles d'avions qui, en cas d'accident, protègent efficacement les passagers.

```
# Étape 1 : Créer une nouvelle colonne 'Total.Injuries' dans le DataFrame d'origine
df['Total.Injuries'] = df[['Total.Serious.Injuries', 'Total.Minor.Injuries', 'Total.Fatal.Injuries']].sum(axis=1)

# Étape 2 : Regrouper par Make et Model, et agréger
summary_df = df.groupby(['Make', 'Model']).agg(
    Total_Uninjured=('Total.Uninjured', 'sum'),
    Total_Injuries=('Total.Injuries', 'sum'),
    Occurrences=('Make', 'size')  # ou 'size', selon ta logique
).reset_index()

# Étape 3 : Trier par Total_Uninjured décroissant
summary_df = summary_df.sort_values(by='Total_Uninjured', ascending=False)

# Étape 4 : Afficher les 10 premiers
print(summary_df.head(10))
```

```
                  Make     Model  Total_Uninjured  Total_Injuries  \
2891             boeing       737             9504             536
10604  mcdonnell douglas  dc-10-10             5840             458
3074             boeing       767             4225              70
3032             boeing   747-422             3770              52
2842             boeing   727-200             3630              55
3030             boeing   747-400             3492               9
10608  mcdonnell douglas  dc-10-30             3168              69
3061             boeing   757-232             3139              48
2936             boeing   737-300             3135              87
10637  mcdonnell douglas   dc-9-82             3009             174

       Occurrences
2891           122
10604           35
3074            38
3032            12
2842            36
3030            13
10608           15
3061            22
2936            34
10637           28
```

Étape 2 : Calcul du Risk Ratio

Pour chaque Make_Model, nous avons calculé un Risk_Ratio, qui mesure le risque moyen de blessure par rapport à l'ensemble des personnes impliquées :

$$\text{Risk Ratio} = \frac{\text{Blessés totaux}}{\text{Blessés totaux} + \text{Indemnes totaux}}$$

Ce ratio permet d'estimer la proportion de blessés parmi les passagers impliqués, quelle que soit la gravité de l'accident.

Étape 3 : Focus sur les 10 meilleurs résultats

Nous avons sélectionné les 10 modèles avec le plus grand nombre de personnes indemnes, puis trié ces 10 selon leur Risk Ratio croissant (du moins risqué au plus risqué).

Nous avons ensuite enrichi l'analyse avec :

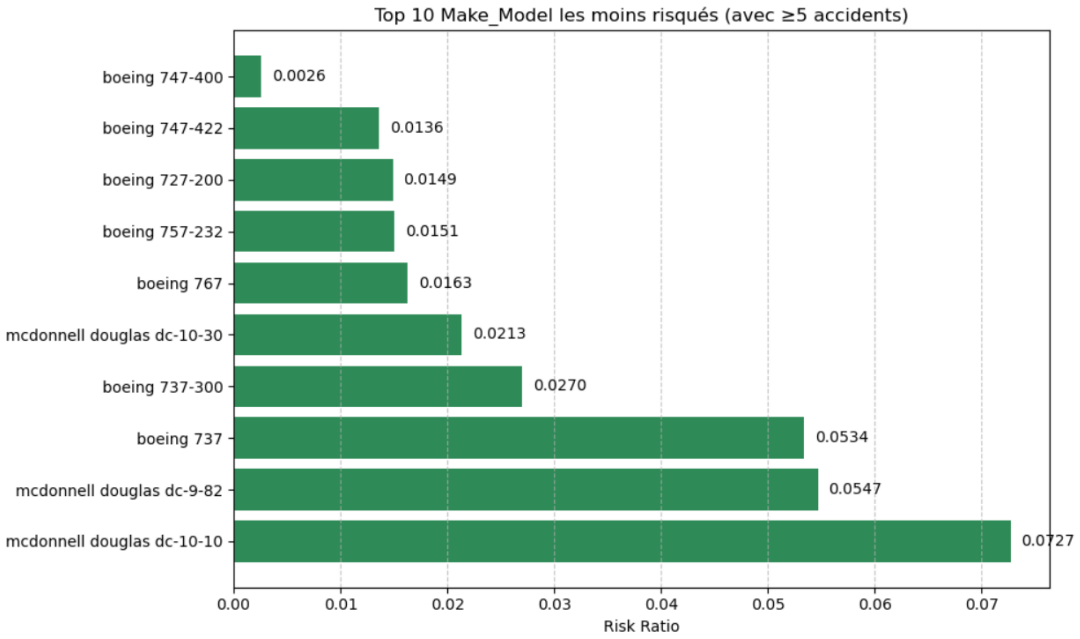
- Le nombre moyen de blessés par accident,
- Le nombre moyen de personnes indemnes par accident.

	Make	Model	Total_Uninjured	Total_Injuries	\
3030	boeing	747-400	3492	9	
3032	boeing	747-422	3770	52	
2842	boeing	727-200	3630	55	
3061	boeing	757-232	3139	48	
3074	boeing	767	4225	70	
10608	mcdonnell douglas	dc-10-30	3168	69	
2936	boeing	737-300	3135	87	
2891	boeing	737	9504	536	
10637	mcdonnell douglas	dc-9-82	3009	174	
10604	mcdonnell douglas	dc-10-10	5840	458	

	Occurrences	Injured_per_Accident	Uninjured_per_Accident	Risk_Ratio
3030	13	0.692308	268.615385	0.002571
3032	12	4.333333	314.166667	0.013605
2842	36	1.527778	100.833333	0.014925
3061	22	2.181818	142.681818	0.015061
3074	38	1.842105	111.184211	0.016298
10608	15	4.600000	211.200000	0.021316
2936	34	2.558824	92.205882	0.027002
2891	122	4.393443	77.901639	0.053386
10637	28	6.214286	107.464286	0.054665
10604	35	13.085714	166.857143	0.072721

❖ **Visualisation des résultats**

Pour illustrer nos analyses de manière claire et comparative, nous avons réalisé plusieurs diagrammes en barres, mettant en relation les différents Make_Model avec leur Risk Ratio calculé.



Commentaire :

- ❖ Boeing 747-400
  - Modèle le plus sûr avec un Risk Ratio de 0,26 %
  - Très faible taux de blessés : moins d'1 blessé pour 100 passagers
  - 13 occurrences, ce qui donne une base statistique fiable
  - Allie forte capacité (≈ 270 indemnes/accident) et haute sécurité
- ❖ Boeing 747-422
  - Deuxième place avec un Risk Ratio d'environ 1,36 %
  - Appartient à la même famille que le 747-400
  - Encore plus de survivants moyens par accident, malgré un taux de blessés légèrement plus élevé
  - Confirme la fiabilité générale de la série 747
- ❖ Boeing 727-200
  - Troisième modèle le plus sûr (Risk Ratio ≈ 1,49 %)
  - 36 occurrences, la plus large base parmi les trois
  - Bonne constance de sécurité : peu de blessés (≈ 1,5) et plus de 100 indemnes en moyenne.

Étape 4: Extraction des données techniques

Pour chaque Make_Model du top 10, nous avons extrait les caractéristiques suivantes :

- Nombre de moteurs (Number.of.Engines)
- Type de moteur (Engine.Type)
- Construction artisanale ou professionnelle (Amateur.Built)

Note: Nous avons utilisé le mode pour extraire la valeur la plus fréquente lorsqu'un même modèle d'avion apparaît avec plusieurs configurations. Cela permet d'identifier la configuration la plus représentative (type de moteur, nombre de moteurs, etc.) sans être influencé par des variations ou erreurs isolées.

En effet, un même Make_Model peut apparaître dans le jeu de données avec de légères variations dans les colonnes comme :

Engine.Type (ex : "turbo tan", "turbo jet"),

Number.of.Engines (ex : parfois 2, parfois 4 si erreurs de saisie),

Amateur.Built (ex : "yes", "no").

Plutôt que de prendre la première valeur ou la moyenne (non pertinente pour du texte), le mode permet de retenir celle qui revient le plus souvent, ce qui reflète la configuration la plus typique ou la plus probable pour ce modèle d'avion.

C'est donc une méthode fiable pour résumer les caractéristiques dominantes d'un type d'appareil.

	Make_Model	Number.of.Engines	Engine.Type	Amateur.Built
0	boeing 727-200	3	turbo fan	no
1	boeing 737	2	unknown	no
2	boeing 737-300	2	turbo fan	no
3	boeing 747-400	4	turbo fan	no
4	boeing 747-422	4	turbo fan	no
5	boeing 757-232	2	turbo fan	no
6	boeing 767	2	turbo fan	no
7	mcdonnell douglas dc-10-10	3	turbo fan	no
8	mcdonnell douglas dc-10-30	3	turbo fan	no
9	mcdonnell douglas dc-9-82	2	turbo fan	no

Commentaire: Nombre de moteurs Tous les modèles sont équipés de 2 à 4 moteurs, ce qui confirme leur conception comme avions multimoteurs. Avoir plusieurs moteurs améliore la sécurité en vol, notamment en cas de défaillance partielle.

Type de moteur La quasi-totalité des avions utilisent des moteurs de type turbo fan, à l'exception d'un cas de type "unknown" (inconnu). Les turbo fans sont largement reconnus pour leur efficacité à haute altitude, leur fiabilité et leur performance sur les vols commerciaux.

Construction par des amateurs Tous ces modèles ont Amateur.Built = 'no', ce qui signifie qu'ils ont été construits par des professionnels qualifiés, selon des normes industrielles rigoureuses. Cela réduit considérablement les risques d'erreurs de fabrication ou d'entretien.

- **Deuxième approche : Les modèles les plus fréquents dans les accidents**

Objectif

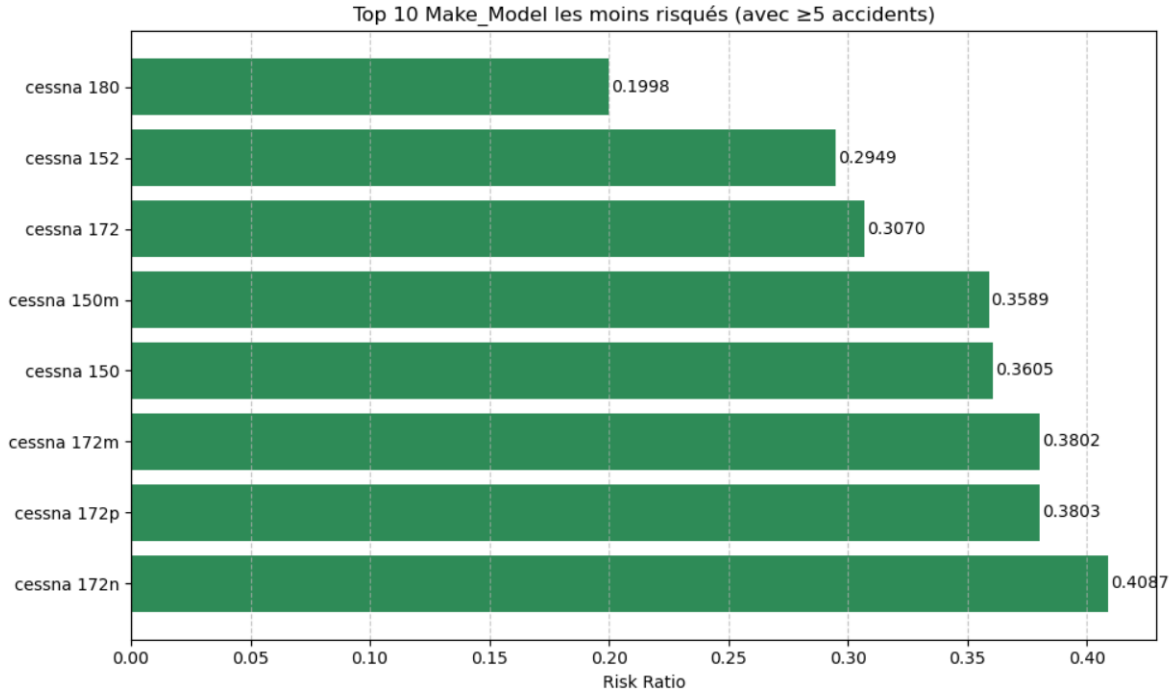
Évaluer la robustesse statistique de notre analyse en examinant les modèles les plus souvent impliqués dans des accidents.

En procédant de la même manière que la première approche on a les résultats suivant :

	Make	Model	Total_Uninjured	Total_Injured	Occurrences	\
4197	cessna	180	973	243	611	
4091	cessna	152	2236	935	2285	
4115	cessna	172	2165	959	1652	
4090	cessna	150m	543	304	574	
4064	cessna	150	738	416	795	
4162	cessna	172m	1037	636	770	
4167	cessna	172p	836	513	667	
4164	cessna	172n	1308	904	1125	

	Injured_per_Accident	Uninjured_per_Accident	Risk_Ratio
4197	0.397709	1.592471	0.199836
4091	0.409190	0.978556	0.294860
4115	0.580508	1.310533	0.306978
4090	0.529617	0.945993	0.358914
4064	0.523270	0.928302	0.360485
4162	0.825974	1.346753	0.380155
4167	0.769115	1.253373	0.380282
4164	0.803556	1.162667	0.408680

❖ **Visualisation des résultats**



	Make_Model	Number.of.Engines	Engine.Type	Amateur.Built
0	cessna 150	1	reciprocating	no
1	cessna 150m	1	reciprocating	no
2	cessna 152	1	reciprocating	no
3	cessna 172	1	reciprocating	no
4	cessna 172m	1	reciprocating	no
5	cessna 172n	1	reciprocating	no
6	cessna 172p	1	reciprocating	no
7	cessna 180	1	reciprocating	no
8	cessna 182	1	reciprocating	no
9	piper pa-28-140	1	reciprocating	no

Remarque :

Les 10 Make_Models les moins risqués selon le critère du nombre d'occurrences d'accidents sont tous :

Monomoteurs (1 seul moteur),

Équipés de moteurs à pistons (reciprocating),

Construits de manière professionnelle (pas d'amateur).

Cela reflète principalement leur forte présence dans l'aviation légère et leur usage fréquent en formation, ce qui explique leur surreprésentation dans les statistiques.

Comparaison avec les modèles les plus sûrs selon le nombre de personnes indemnes À l'inverse, les Make_Models identifiés comme les plus fiables selon le critère du nombre de personnes indemnes sont :

Multimoteurs (2 à 4 moteurs),

Dotés de moteurs turbo fan, reconnus pour leur puissance, fiabilité et performance en vol commercial,

Professionnellement construits, adaptés aux standards de l'aviation civile.

Conclusion

Les modèles évalués selon le nombre de personnes indemnes apparaissent plus fiables sur le plan technique, grâce à leur motorisation plus sûre et redondante. En revanche, les modèles fréquents dans les accidents ne sont pas nécessairement dangereux, mais souvent surutilisés dans des contextes à risque (formation, vols courts, aviation légère).

- **Les Top10 Make_Models les plus risques**

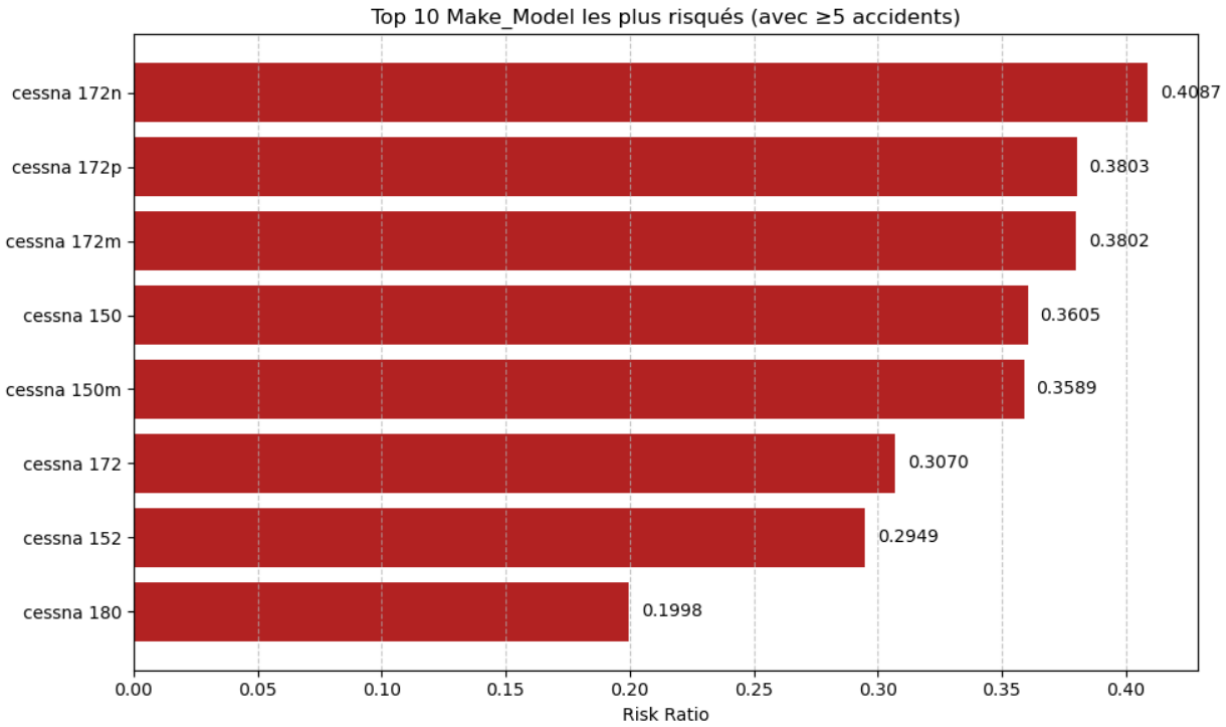
Nous avons appliqué la même méthodologie, mais en inversant notre angle d'analyse. Cette fois, notre objectif était d'identifier les modèles d'avions les plus risqués, en nous concentrant sur le nombre total de blessés recensés par modèle puis calculer le Risk_Ratio.

Résultats :

	Make	Model	Total_Uninjured	Total_Injuries	Occurrences	\
12358	piper	pa-28-140	826.0	957.0	912	
12375	piper	pa-28-181	605.0	639.0	505	
2302	bell	206b	560.0	564.0	487	
4164	cessna	172n	1308.0	904.0	1125	
4167	cessna	172p	836.0	513.0	667	
4162	cessna	172m	1037.0	636.0	770	
4115	cessna	172	2165.0	959.0	1652	
4091	cessna	152	2236.0	935.0	2285	
2891	boeing	737	9504.0	536.0	122	

	Injured_per_Accident	Uninjured_per_Accident	Risk_Ratio
12358	1.049342	0.905702	0.536736
12375	1.265347	1.198020	0.513666
2302	1.158111	1.149897	0.501779
4164	0.803556	1.162667	0.408680
4167	0.769115	1.253373	0.380282
4162	0.825974	1.346753	0.380155
4115	0.580508	1.310533	0.306978
4091	0.409190	0.978556	0.294860
2891	4.393443	77.901639	0.053386

❖ **Visualisation :**



Analyse des Top 10 Make_Models les plus risqués Les résultats montrent que ces 10 modèles d’avion sont principalement :

De marque Cessna (à l’exception d’un Piper),

Équipés d’un seul moteur (Number.of.Engines = 1.0),

Utilisant un moteur à combustion interne classique (Engine.Type = reciprocating),

Non construits par des amateurs (Amateur.Built = no), donc conformes aux standards professionnels.

Interprétation Monomoteur = plus de vulnérabilité L'absence de moteur de secours augmente le risque en cas de panne.

Moteurs à pistons = performance limitée Moins puissants et moins fiables sur longues distances ou en haute altitude, surtout s’ils sont mal entretenus.

Usage fréquent en formation = surreprésentation Ces modèles sont très utilisés par des pilotes en formation, ce qui explique en partie leur taux d'accidents élevé, non pas par défaut de conception, mais par contexte d’utilisation.

- **Business Recommendation 1**

Phases critiques à sécuriser en priorité Phases : Cruise, Maneuvering, Takeoff, Landing, Approach. Ces cinq phases concentrent la majorité des accidents mortels. Pourquoi ?

Ce sont les phases les plus exigeantes en termes de charge de travail, de pilotage manuel et de facteurs externes (vents, visibilité, erreurs humaines).

Recommandations :

Intensifier la formation des pilotes pour ces phases (simulateurs réalistes, entraînement en conditions extrêmes).

Renforcer l'automatisation et les aides à la décision (alertes automatisées, meilleure intégration des données météo).

Surveiller en temps réel la performance des pilotes et systèmes durant ces phases via des systèmes embarqués.

- **Business Recommendation 2**

Modèles d'aéronefs à privilégier (meilleure sécurité observée) Modèles recommandés : Pour les vols commerciaux, optez pour des modèles comme: boeing 747-400; boeing 757-232; boeing 767; mcdonnell douglas dc-10-30 et boeing 737-300, ils sont multimoteurs, propulsés par des turbo fans, et non construits par des amateurs.

Ces caractéristiques sont des indicateurs fiables de sécurité, car elles reflètent des standards de conception professionnelle, de redondance mécanique et de performance moteur.

Pour l'apprentissage ou les petits trajets privés, les Cessna 152/172 restent de bonnes options, à condition que l'entretien soit rigoureux et que les vols se déroulent dans des conditions météorologiques favorables.

Ces choix s'appuient directement sur les données empiriques observées dans les 10 dernières années.

- **Business Recommendation 3**

Modèles à éviter pour les vols à haut risque ou longues distances : piper pa-28-140; piper pa-28-181; bell 206b; cessna 172n; cessna 172p

Ces modèles sont moins adaptés aux vols complexes ou exigeants, en particulier :

Vols en montagne, longues distances ou conditions météorologiques difficiles.

Conclusion

L’analyse montre que la sécurité aérienne a considérablement progressé depuis 1982, avec une baisse notable du nombre d’accidents et de blessés. Cette amélioration est le fruit de plusieurs facteurs : avancées technologiques, renforcement des réglementations, amélioration de la formation des pilotes, meilleure gestion des risques et maintenance plus rigoureuse des aéronefs.

Cependant, des zones de vigilance subsistent, notamment en fonction du type de vol, de la phase de vol, du type d’appareil et des conditions météorologiques. L'analyse approfondie permet ainsi de dégager des recommandations utiles pour renforcer encore la sécurité.