EOkeyo / aviator

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

🛫 Data Science Project: Aviation Risk Assessment This Jupyter Notebook documents the **process**, **analysis**, and **key findings** for the *Aviation Risk Assessment* project aimed at identifying the **lowest-risk aircraft** for a company entering the **commercial and private aviation** sectors.

Apache-2.0 license

0 stars 0 forks 0 watching Branches Activity Tags

Public repository

main 1 Branch 0 Tags Go to file t Go to file Add file Code

EOkeyo presentation ✓ 688730b · 6 minutes ago

notebookproject	presentation	6 minutes ago
.gitignore	Initial commit	yesterday
LICENSE	Initial commit	yesterday
README.md	README changes	41 minutes ago

README Apache-2.0 license

🛫 Data Science Project: Aviation Risk Assessments

▶️ View the Interactive Analysis (HTML)

Open aviation_analysis.html

🔗 Direct Links to Interactive Figures (Hosted)

- Events by Year and Severity
- Fatal Accident Rate Over Time
- Accident Severity Distribution
- Top 15 Manufacturers
- Accidents by Flight Category
- Weather Conditions vs Accident Severity
- Hierarchical View (Sunburst)
- Geographic Distribution

This Jupyter Notebook documents the **process**, **analysis**, and **key findings** for the *Aviation Risk Assessment* project aimed at identifying the **lowest-risk aircraft** for a company entering the **commercial and private aviation** sectors..

📦 Project Deliverables

- 🗣️ **Non-technical presentation** for the Head of the Aviation Division

- 📓 **Jupyter Notebook** (this file) for a data science audience
- 💻 **GitHub repository** with documented source code
- 📊 **Interactive Dashboard** for risk visualization and model insights

1. 🧭 Business Understanding

1.1 🎯 Project Goals

The primary goal is to determine which aircraft present the **lowest risk** for the company's new aviation venture covering both **commercial** and **private enterprises**.

The final output must include **three concrete, actionable business recommendations** for the Head of the Aviation Division.

1.2 ⚙️ Business Problem

The company is expanding into **new industries** to diversify its portfolio specifically purchasing and operating aircraft for **commercial and private aviation**.

However, the company **lacks knowledge about potential risks** associated with aircraft operations.

This analysis will deliver **data-driven insights** to guide **risk mitigation** and **aircraft selection decisions**.

1.3 👥 Target Audience and Communication

🗣️ Non-Technical Presentation

Audience: Head of the Aviation Division and business stakeholders

Focus:

- Clear, non-technical explanations (technical methods can be mentioned with brief context)
- A compelling narrative supported by **simple, well-formatted visualizations**

💻 Jupyter Notebook (This Document)

Audience: Data Science and Analytics Teams

Focus:

- Technical documentation with **Python + Markdown integration**
- Emphasis on **project goals, data, methods, and results**
- Organized, modular, and **easy to skim** for quick understanding

2. 📊 Data Understanding and Preparation

2.1 🗂️ The Data Source

The dataset is sourced from the **National Transportation Safety Board (NTSB)** and contains **civil aviation accident data** spanning **1962 to 2023**.

It includes records from both **United States airspace** and **international waters**, providing a comprehensive foundation for analyzing aviation risk patterns and identifying low-risk aircraft types.

2.2 🧾 Initial Load and Validation

Cell 1: Imports and Basic Settings

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from datetime import datetime
import warnings
warnings.filterwarnings('ignore')

sns.set_theme(style='darkgrid')
plt.rcParams.update({'figure.dpi': 120})
```

Cell 2: Load Dataset

```
file_path = 'aircraft_data.csv'
df = pd.read_csv(file_path, encoding='latin-1', low_memory=False)
print("Loaded rows:", len(df))
df.head()
```

Cell 3: Validate Columns

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

print("Columns in file:", len(df.columns))
missing = set(expected_columns) - set(df.columns)
extra = set(df.columns) - set(expected_columns)
print("Missing columns:", missing)
print("Extra columns:", extra)
```

2.3 🧹 Data Cleaning, Standardization, and Derivations

The data preparation focuses on **standardization**, **handling missing values**, and **creating key fields** necessary for risk analysis, utilizing pandas **vectorized operations** (idiomatic and performant pandas).

🕓 Date Conversion

- `Event.Date` and `Publication.Date` are converted to **datetime** objects.

✂️ Whitespace & Missing Values

- Object columns are **stripped of whitespace**.
- Missing values (empty strings) are replaced with **NaN**, then filled with `'Unknown'`.

🧮 Injury & Total People

- Injury-related columns are ensured to be **numeric**, with nulls filled as **0**.

- A new column `Total.People` is derived for comprehensive passenger and crew accounting.

🔠 Standardization

- `Make` and `Model` fields are **converted to uppercase** for consistent grouping and comparison.

🧩 Derived Feature: Accident.Severity

A new categorical feature is created based on injury counts:

- **Fatal** → if `Total.Fatal.Injuries > 0`
- **Serious** → if `Total.Fatal.Injuries = 0` and `Total.Serious.Injuries > 0`
- **Minor** → otherwise

🛬 Derived Feature: Flight.Category

The detailed `Purpose.of.flight` field is **simplified** into key categories relevant to the business goal:

- `Commercial`
- `Business`
- `Personal`
- `Training`
- `Cargo/Ferry`
- `Other`

🌦️ Derived Feature: Weather.Category

The `Weather.Condition` field is simplified into broader analytical groups:

- **Good (VMC)**
- **Poor (IMC)**
- **Unknown**

Cell 4: Basic Cleaning & Standardization

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

# Trim whitespace for text columns
df = df.apply(lambda col: col.str.strip() if col.dtype == 'object' else col)

# Replace empty strings in object columns with NaN, then fill 'Unknown'
obj_cols = df.select_dtypes(include=['object']).columns
df[obj_cols] = df[obj_cols].replace({'': np.nan})
df[obj_cols] = df[obj_cols].fillna('Unknown')

# Ensure numeric injury fields exist and are numeric
injury_cols = ['Total.Fatal.Injuries','Total.Serious.Injuries','Total.Minor.Injuries','Total.Uninjured']
for c in injury_cols:
    if c not in df.columns:
        df[c] = 0
    df[c] = pd.to_numeric(df[c], errors='coerce').fillna(0).astype(int)

# Derived columns
df['Total.People'] = df[injury_cols].sum(axis=1)
df['Make'] = df['Make'].astype(str).str.upper()
df['Model'] = df['Model'].astype(str).str.upper()

print("Cleaned. Rows:", len(df))
df.head()
```

Cell 5: Accident Severity & Flight Category

```
# Severity
df['Accident.Severity'] = 'Minor'
df.loc[df['Total.Fatal.Injuries'] > 0, 'Accident.Severity'] = 'Fatal'
df.loc[(df['Total.Fatal.Injuries'] == 0) & (df['Total.Serious.Injuries'] > 0), 'Accident.Severity'] = 'Serious'

# Simplify Purpose.of.flight into Flight.Category
def categorize_purpose(p):
    s = str(p).upper()
    if 'PERSONAL' in s or 'RECREAT' in s or 'PLEASURE' in s:
        return 'Personal'
    if 'BUSINESS' in s or 'CORPORATE' in s:
        return 'Business'
    if any(x in s for x in ['INSTRUCTION','TRAINING']):
        return 'Training'
    if any(x in s for x in ['AIRLINE','SCHEDULED','COMMUTER','PART 121','PART 135']):
        return 'Commercial'
    if 'CARGO' in s or 'FERRY' in s:
        return 'Cargo/Ferry'
    if s == 'UNKNOWN':
        return 'Unknown'
    return 'Other'

df['Flight.Category'] = df['Purpose.of.flight'].apply(categorize_purpose)
df[['Purpose.of.flight','Flight.Category']].head(6)
```

Cell 6: Weather Category Simplification

```
def categorize_weather(w):
    s = str(w).upper()
    if any(x in s for x in ['VMC','CLEAR','FAIR']):
        return 'Good (VMC)'
    if any(x in s for x in ['IMC','CLOUD','RAIN','FOG','SNOW','STORM','THUNDER']):
        return 'Poor (IMC)'
    return 'Unknown'

df['Weather.Category'] = df['Weather.Condition'].apply(categorize_weather)
df[['Weather.Condition','Weather.Category']].head(6)
```

3. 📈 Data Analysis and Visualization

The analysis focuses on identifying **aircraft (Make/Model)** with the **lowest fatal accident rate** to inform the company's **purchasing decision**.

To ensure **statistical relevance**, the data is **filtered** to include only aircraft **Makes and Models** with a **significant number of recorded accidents** before calculating the **fatal accident rate**.

3.1 🔍 Exploratory Summaries

Cell 7: Exploratory Summaries

```
print("Top 10 Makes:")
display(df['Make'].value_counts().head(10))

print("\nFlight Category Distribution:")
display(df['Flight.Category'].value_counts())

print("\nAccident Severity counts:")
```

```
display(df['Accident.Severity'].value_counts())

print("\nYear range:", df['Event.Date'].dt.year.min(), "to", df['Event.Date'].dt.year.max())
```

🧠 Interpretation

- The data covers a long period (**1962–2023**).
- General aviation manufacturers (**CESSNA, PIPER, BEECH, BELL**) dominate the accident count, suggesting a focus on **lower-risk, light aircraft** is appropriate for the company's "start" phase.
- The **"Personal" flight category** accounts for the vast majority of accidents.

3.2 ⚖️ Key Metric: Fatal Accident Rate by Make (Filter: ≥ 50 Accidents)

We calculate the **fatal accident rate per Make** to identify manufacturers associated with **lower risk**.

$$\text{Fatal Accident Rate} = \frac{\text{Number of Fatal Accidents}}{\text{Total Accidents}} \times 100$$

Cell 8: Make-Level Fatal Accident Rate

```
if 'Event.Id' not in df.columns:
    df['Event.Id'] = df.index.astype(str)

make_stats = df.groupby('Make').agg(
    Total_Accidents=('Event.Id','count'),
    Fatal_Accidents=('Accident.Severity', lambda s: (s=='Fatal').sum())
).sort_values('Total_Accidents', ascending=False)

make_stats['Fatal_Accident_Rate'] = (make_stats['Fatal_Accidents'] / make_stats['Total_Accidents'] * 100).round(2)
make_stats_filtered = make_stats[make_stats['Total_Accidents'] >= 50].sort_values('Fatal_Accident_Rate')
make_stats_filtered.head(15)
```

🧠 Interpretation

- Filtering for Makes with at least **50 accidents** provides a more reliable comparison.
- Makes like **MCDONNELL DOUGLAS** and **AIRBUS** show very low fatal accident rates (**0–2%**), likely due to stringent commercial regulations and maintenance.
- Traditional light aircraft manufacturers like **CESSNA** and **PIPER** have higher rates (**8–10.5%**), but still provide a lower barrier to entry for the new division.

3.3 🛩️ Key Metric: Fatal Accident Rate by Model (Common Light Aircraft Makes)

Given the potential focus on **light aircraft** (*Cessna, Piper, Beech*), we dive deeper into the **models** for a more granular **risk assessment**.

Cell 9: Model-Level for Common Manufacturers

```
top_makes = ['CESSNA','PIPER','BEECH','BOEING','AIRBUS']
model_stats = df[df['Make'].isin(top_makes)].groupby(['Make','Model']).agg(
    Total_Accidents=('Event.Id','count'),
    Fatal_Accidents=('Accident.Severity', lambda s: (s=='Fatal').sum())
)
model_stats['Fatal_Accident_Rate'] = (model_stats['Fatal_Accidents'] / model_stats['Total_Accidents'] * 100).round(2)
model_stats_filtered = model_stats[model_stats['Total_Accidents'] >= 20].sort_values('Fatal_Accident_Rate')
model_stats_filtered.head(20)
```

🧠 Interpretation

- Within the most common light aircraft manufacturers:
  - **Cessna 172 models**, popular for training and private use, have some of the **lowest fatal accident rates (4.8–6.6%)**.
  - **Larger, more complex, or older models** (e.g., **BEECH B35**, **CESSNA 185**) tend to have higher fatal rates.
  - Selecting models with **proven lower rates** (like the **C-172 series**) minimizes initial operational risk.

3.4 ⏳ Risk Over Time: Fatal Accident Rate by Year

Understanding the **trend of the fatal accident rate over time** is crucial for **long-term strategy**.

Cell 10: Yearly Accident Counts and Fatal Rate

```
yearly = df.groupby(df['Event.Date'].dt.year).agg(
    Total_Accidents=('Event.Id','count'),
    Fatal_Accidents=('Accident.Severity', lambda s: (s=='Fatal').sum())
).dropna().sort_index()

yearly['Fatal_Rate'] = (yearly['Fatal_Accidents'] / yearly['Total_Accidents'] * 100).round(2)
yearly.tail(15)
```

🧠 Interpretation

- The **total number of accidents** has generally **decreased** over the long term.
- The **Fatal Accident Rate** (as a percentage of total accidents) also shows a general — though sometimes **volatile — downward trend** in recent decades.
- This indicates an overall **improvement in safety and risk management practices**.

3.5 📊 Interactive Visualizations

The following cells generate several **interactive visualizations**.

The key visuals selected for the **final non-technical presentation** must **directly support the three business recommendations**.

Cell 11: Interactive Visualizations with Plotly

```
import plotly.graph_objects as go
import plotly.express as px
from plotly.subplots import make_subplots

# Ensure Event.Id exists (create from index if missing)
if 'Event.Id' not in df.columns:
    df['Event.Id'] = df.index.astype(str)

# 1. Interactive Events by Year with Severity Breakdown
yearly_severity = df.groupby([df['Event.Date'].dt.year, 'Accident.Severity']).size().unstack(fill_value=0)
yearly_severity = yearly_severity.dropna()

fig1 = go.Figure()
for severity in yearly_severity.columns:
    fig1.add_trace(go.Bar(
        x=yearly_severity.index,
        y=yearly_severity[severity],
        name=severity,
        hovertemplate='<b>Year:</b> %{x}<br><b>Count:</b> %{y}<br><b>Severity:</b> ' + severity + '<extra></extra>'
    ))

fig1.update_layout(
    title='Interactive Events by Year and Severity',
    xaxis_title='Year',
    yaxis_title='Number of Accidents',
```

```
    barmode='stack',
    hovermode='x unified',
    height=500,
    legend=dict(orientation="h", yanchor="bottom", y=1.02, xanchor="right", x=1)
)
fig1.show()

# 2. Interactive Fatal Rate Over Time with Hover Details
yearly_detailed = df.groupby(df['Event.Date'].dt.year).agg(
    Total_Accidents=('Event.Id','count'),
    Fatal_Accidents=('Accident.Severity', lambda s: (s=='Fatal').sum()),
    Serious_Accidents=('Accident.Severity', lambda s: (s=='Serious').sum()),
    Minor_Accidents=('Accident.Severity', lambda s: (s=='Minor').sum())
).dropna().sort_index()

yearly_detailed['Fatal_Rate'] = (yearly_detailed['Fatal_Accidents'] / yearly_detailed['Total_Accidents'] * 100).round(2)

fig2 = go.Figure()
fig2.add_trace(go.Scatter(
    x=yearly_detailed.index,
    y=yearly_detailed['Fatal_Rate'],
    mode='lines+markers',
    name='Fatal Rate',
    line=dict(color='red', width=2),
    marker=dict(size=8),
    hovertemplate='<b>Year:</b> %{x}<br><b>Fatal Rate:</b> %{y}%<extra></extra>'
))

fig2.update_layout(
    title='Fatal Accident Rate (%) Over Time',
    xaxis_title='Year',
    yaxis_title='Fatal Rate (%)',
    hovermode='x unified',
    height=450
)
fig2.show()

# 3. Interactive Pie Chart - Accident Severity Distribution
severity_counts = df['Accident.Severity'].value_counts()

fig3 = go.Figure(data=[go.Pie(
    labels=severity_counts.index,
    values=severity_counts.values,
    hole=0.3,
    hovertemplate='<b>%{label}</b><br>Count: %{value}<br>Percentage: %{percent}<extra></extra>'
)])

fig3.update_layout(
    title='Accident Severity Distribution',
    height=500
)
fig3.show()

# 4. Interactive Manufacturer Analysis (Top 15 by accident count)
top_15_makes = df['Make'].value_counts().head(15).index
make_severity = df[df['Make'].isin(top_15_makes)].groupby(['Make', 'Accident.Severity']).size().unstack(fill_value=0)

fig4 = go.Figure()
for severity in make_severity.columns:
    fig4.add_trace(go.Bar(
        name=severity,
        x=make_severity.index,
        y=make_severity[severity],
        hovertemplate='<b>Make:</b> %{x}<br><b>Count:</b> %{y}<br><b>Severity:</b> ' + severity + '<extra></extra>'
    ))

fig4.update_layout(
    title='Top 15 Manufacturers by Accident Count and Severity',
    xaxis_title='Manufacturer',
    yaxis_title='Number of Accidents',
```

```
    barmode='stack',
    height=500,
    xaxis_tickangle=-45,
    legend=dict(orientation="h", yanchor="bottom", y=1.02, xanchor="right", x=1)
)
fig4.show()

# 5. Interactive Flight Category Distribution
flight_cat_counts = df['Flight.Category'].value_counts()

fig5 = px.bar(
    x=flight_cat_counts.index,
    y=flight_cat_counts.values,
    labels={'x': 'Flight Category', 'y': 'Number of Accidents'},
    title='Accidents by Flight Category',
    color=flight_cat_counts.values,
    color_continuous_scale='Viridis'
)

fig5.update_traces(
    hovertemplate='<b>Category:</b> %{x}<br><b>Count:</b> %{y}<extra></extra>'
)

fig5.update_layout(height=450, showlegend=False)
fig5.show()

# 6. Interactive Weather Conditions
weather_severity = df.groupby(['Weather.Category', 'Accident.Severity']).size().unstack(fill_value=0)

fig6 = go.Figure()
for severity in weather_severity.columns:
    fig6.add_trace(go.Bar(
        name=severity,
        x=weather_severity.index,
        y=weather_severity[severity],
        hovertemplate='<b>Weather:</b> %{x}<br><b>Count:</b> %{y}<br><b>Severity:</b> ' + severity + '<extra></extra>'
    ))

fig6.update_layout(
    title='Weather Conditions vs Accident Severity',
    xaxis_title='Weather Category',
    yaxis_title='Number of Accidents',
    barmode='group',
    height=450,
    legend=dict(orientation="h", yanchor="bottom", y=1.02, xanchor="right", x=1)
)
fig6.show()

# 7. Interactive Sunburst - Hierarchical View
sunburst_data = df[df['Make'].isin(top_15_makes)].copy()
sunburst_data = sunburst_data[sunburst_data['Total.Fatal.Injuries'] >= 0]

fig7 = px.sunburst(
    sunburst_data,
    path=['Make', 'Accident.Severity'],
    title='Hierarchical View: Manufacturer → Accident Severity',
    height=600
)

fig7.update_traces(
    hovertemplate='<b>%{label}</b><br>Count: %{value}<extra></extra>'
)
fig7.show()

# 8. Interactive Scatter: Geographic Distribution (if coordinates available)
# Convert Latitude and Longitude to numeric, coercing errors to NaN
df['Latitude'] = pd.to_numeric(df['Latitude'], errors='coerce')
df['Longitude'] = pd.to_numeric(df['Longitude'], errors='coerce')

# Filter for valid coordinates (not NaN and within valid ranges)
```

```
geo_data = df[
    (df['Latitude'].notna()) &
    (df['Longitude'].notna()) &
    (df['Latitude'].between(-90, 90)) &
    (df['Longitude'].between(-180, 180))
].copy()

if len(geo_data) > 0:
    sample_geo = geo_data.sample(min(5000, len(geo_data)))

    fig8 = px.scatter_mapbox(
        sample_geo,
        lat='Latitude',
        lon='Longitude',
        color='Accident.Severity',
        hover_data=['Make', 'Model', 'Event.Date', 'Country'],
        title=f'Geographic Distribution of Accidents (Sample of {len(sample_geo):,} points)',
        mapbox_style='carto-positron',
        zoom=2,
        height=600
    )
    fig8.show()
    print(f"📍 Mapped {len(sample_geo):,} accidents out of {len(df):,} total records")
else:
    print("⚠️  No valid geographic coordinates available for mapping")

print("✅ All interactive visualizations created!")
print("💡 Tips:")
print("   - Hover over elements for detailed information")
print("   - Click legend items to show/hide categories")
print("   - Use the toolbar to zoom, pan, and save images")
print("   - Double-click to reset view")
```

Cell 12: Export Cleaned Data

```
out_fn = 'aircraft_data_cleaned.csv'
cols = ['Event.Date','Year','Make','Model','Aircraft.Category','Registration.Number',
        'Flight.Category','Weather.Category','Accident.Severity',
        'Total.Fatal.Injuries','Total.Serious.Injuries','Total.Minor.Injuries',
        'Total.Uninjured','Total.People','Latitude','Longitude','Country','Airport.Name']
# keep only columns that exist
cols = [c for c in cols if c in df.columns]
df[cols].to_csv(out_fn, index=False)
print("Saved cleaned file:", out_fn)
```

Cell: Check and Install Plotly

```
import sys
import subprocess

def check_and_install_plotly():
    """Check if Plotly is installed, install if not"""
    try:
        import plotly
        print(f"✅ Plotly is already installed!")
        print(f"   Version: {plotly.__version__}")
        return True
    except ImportError:
        print("❌ Plotly is not installed.")
        print("📦 Installing Plotly now...")
        try:
            subprocess.check_call([sys.executable, "-m", "pip", "install", "plotly"])
            import plotly
            print(f"✅ Plotly successfully installed!")
            print(f"   Version: {plotly.__version__}")
            return True
        except Exception as e:
```

```
            print(f"❌ Error installing Plotly: {e}")
            print("\n💡 Try manually running: !pip install plotly")
            return False

# Run the check
if check_and_install_plotly():
    print("\n✨ You're ready to create interactive visualizations!")
else:
    print("\n⚠️  Please install Plotly manually and try again.")
```

4. 🏁 Conclusion and Recommendations

Based on the analysis, the following **concrete business recommendations** are provided to the **Head of the Aviation Division** to minimize initial risk for the new enterprise.

4.1 🛩️ Business Recommendation 1: Prioritize Proven, Low-Risk Light Aircraft

Finding:

Models like the **CESSNA 172** and **PIPER PA-28** series exhibit **low fatal accident rates (4–7%)** among the most common light aircraft, suitable for both *Private* and *Training* operations (a major segment of the current accident data).

Larger commercial aircraft (e.g., **AIRBUS**) show **0–2% fatal rates** but represent a significantly higher initial investment and operational complexity.

Recommendation:

Focus initial procurement on **well-established, low fatal-rate light aircraft models** like the *Cessna 172 series*.

This approach minimizes **per-flight risk** while allowing the company to **build operational expertise** in a proven segment.

4.2 🌧️ Business Recommendation 2: Implement Strict Safety Protocols for Weather Conditions

Finding:

The data, through the Weather.Category breakdown (Visualization 6), will likely show a significant number of accidents occurring in *'Poor (IMC)'* conditions or when the weather is *'Unknown'*, even though the majority of flights occur in *'Good (VMC)'* weather.

Recommendation:

Mandate and enforce **enhanced safety protocols**, **training**, and **go/no-go decision tools** for flights operating in or near **Instrument Meteorological Conditions (IMC)**.

Investing in **reliable weather intelligence** and **pilot training** for adverse conditions is critical to reduce the likelihood of **serious and fatal accidents**.

4.3 🧰 Business Recommendation 3: Adopt a Proactive Maintenance and Inspection Strategy

Finding:

While not directly analyzed as a Make or Model risk, the overall trend of the **Fatal Accident Rate Over Time (Visualization 2)** shows a general long-term **decrease in fatal rates**.

This improvement is strongly correlated with **industry-wide advancements** in technology, maintenance standards, and regulatory oversight.

Recommendation:

Commit to a **maintenance program** that significantly **exceeds minimum regulatory requirements**, utilizing **predictive analytics** (as a next step) and prioritizing **immediate fleet upgrades** for known mechanical failure points.

Proactive maintenance is a key operational lever to capitalize on the industry's improving safety trend and **differentiate the company as a low-risk operator**.

5. 📦 Next Steps and Deliverables

5.1 🧭 Next Steps

- **Cost-Benefit Analysis:**
  Cross-reference the identified low-risk aircraft (Make/Model) with **acquisition and operating costs** to finalize **purchasing options**.
- **In-Depth Causal Factor Analysis:**
  Use the original NTSB data fields (Broad.phase.of.flight, FAR.Description) to determine **why accidents occur** in the recommended aircraft/models and tailor **pilot training** and **operational procedures**.
- **Geographic Risk Analysis:**
  Analyze **accident density** in **target operating regions** (if the company has specific geographic plans).

5.2 📁 Deliverable Summary

Deliverable	Audience	Format	Location
Non-Technical Presentation	Business Stakeholders	PDF / Live Slide Deck	Canvas, GitHub (`presentation.pdf`)
Jupyter Notebook	Data Science Audience	`.ipynb` / PDF	Canvas, GitHub (`.ipynb` file)
GitHub Repository	Public / Recruiters	Cloud-Hosted Directory	GitHub Link
Interactive Dashboard	Stakeholders / Explorers	Cloud-Hosted Views (Tableau)	Linked in GitHub `README.md`

📊 Summary

This project provides a comprehensive data-driven analysis of aviation risk assessment, identifying the lowest-risk aircraft for a company entering the commercial and private aviation sectors. The analysis leverages NTSB data from 1962-2023 and delivers actionable recommendations focused on:

1. Proven, low-risk light aircraft models (Cessna 172, Piper PA-28)
2. Enhanced weather safety protocols

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Deployments 13

github-pages 11 hours ago

+ 12 deployments

Languages

- Jupyter Notebook 63.5%
- HTML 36.5%