Exploratory Data Analysis of Air Crash Dataset Using the SEMMA Methodology

September 30, 2023

1 Dataset Using the SEMMA Methodology

2 Abstract

This research paper presents a comprehensive exploratory data

analysis (EDA) of the Air Crash dataset using the SEMMA (Sample,

Explore, Modify, Model, Assess) methodology.
The study involves

data preprocessing, visualization, and modeling, providing valuable

insights into the dataset's characteristics and trends.

3 Introduction

In the vast realm of data science, understanding and analyzing data is of paramount importance. One methodology that provides a structured approach to this process is SEMMA (Sample, Explore, Modify, Model, Assess). In this article, we'll walk through an exploratory data analysis of the Air Crash dataset using the SEMMA methodology.
Let's start by following the SEMMA methodology for our analysis of the Air Crash dataset:

4 SEMMA (Sample, Explore, Modify, Model, Assess) Methodology:

1. **Sample:Obtain a subset of the large dataset to expedite the preliminary analysis.**
2. **Explore:Understand the structure, content, quality, and inherent patterns in the data.**

3. **Modify:Create, select, and transform variables in preparation for modeling.**
4. **Model:Build descriptive or predictive models.**
5. **Assess:Evaluate the models' utility and validity.**

5 Sample

Before diving deep into the data, let's take a quick look at the first few rows and columns to get a feel for the data structure. This will help us decide if we need to sample the data (if it's too large) or if we can work with the entire dataset.

```
importpandas aspd

# Load the dataset
air_crashes_data = pd.read_csv("/mnt/data/air_crashes_data.csv")

# Display the first few rows of the dataset
air_crashes_data.head()
```

Here's a quick overview of the first few entries in the dataset:

- **Year: The year of the air crash.**
- **Quarter: The quarter in which the crash occurred.**
- **Month: The month of the crash.**
- **Day: The day of the crash.**
- **Country/Region: The country or region where the crash occurred.**
- **Aircraft Manufacturer: Manufacturer of the aircraft.**
- **Aircraft: The specific model/type of the aircraft.**
- **Location: The specific location of the crash.**
- **Operator: The operator of the aircraft.**
- **Sum of Ground: The number of fatalities on the ground due to the crash.**
- Sum of Fatalities (air): The number of fatalities on board the aircraft.
- Sum of Aboard: The total number of people on board the aircraft.

Given the snapshot of the data, the dataset appears to be manageable in size. Therefore, for the purpose of this EDA, we will work with the entire dataset rather than sampling it.

6 Explore

In this phase, we will:

- Check the dataset's shape to understand its size.
- Check for missing values.
- Get summary statistics for numerical columns.
- Visualize some of the key features.

Let's start by checking the shape of the dataset.

```
# Check the shape of the dataset
dataset_shape = air_crashes_data.shape
dataset_shape
```

RESULT

```
(5030, 12)
```

The dataset consists of 5,030 rows and 12 columns.

Next, let's check for any missing values in the dataset. Identifying and addressing missing values is crucial as they can impact the quality of any subsequent analysis or modeling.

```
# Check for missing values in each column
missing_values = air_crashes_data.isnull().sum()
missing_values_percentage = (air_crashes_data.isnull().sum() / len(air_crashes_data)) * 100

missing_data = pd.DataFrame({"Missing Values": missing_values, "Percentage (%)": missing_values_percentage})
missing_data.sort_values(by="Percentage (%)", ascending=False)
```

RESULT

```
                         Missing Values  Percentage (%)
Operator                            234        4.652087
Country/Region                      230        4.572565
Year                                  0        0.000000
Quarter                               0        0.000000
Month                                 0        0.000000
Day                                   0        0.000000
Aircraft Manufacturer                 0        0.000000
Aircraft                              0        0.000000
Location                              0        0.000000
Sum of Ground                         0        0.000000
Sum of Fatalities (air)               0        0.000000
Sum of Aboard                         0        0.000000
```

From the data, we observe the following missing values:

- **Operator: 234 missing values (4.65% of the dataset)**
- **Country/Region: 230 missing values (4.57% of the dataset)**

All other columns are complete with no missing values.

6.1 Summary Statistics

Next, let's obtain summary statistics for the numerical columns to get insights into the distribution of the data.

```
# Summary statistics for numerical columns
summary_statistics = air_crashes_data.describe()
summary_statistics
RESULT
              Year          Day  Sum of Ground  Sum of Fatalities (air)  \
count  5030.000000  5030.000000    5030.000000              5030.000000
mean   1970.938370    15.783499       1.706163                22.237972
std      24.770415     8.795157      55.132476                34.960979
min    1908.000000     1.000000       0.000000                 0.000000
25%    1951.000000     8.000000       0.000000                 4.000000
50%    1970.000000    16.000000       0.000000                11.000000
75%    1992.000000    23.000000       0.000000                25.000000
max    2023.000000    31.000000    2750.000000               583.000000

       Sum of Aboard
count    5030.000000
mean       31.013519
std        45.399504
min         0.000000
25%         7.000000
50%        16.000000
75%        34.000000
max       644.000000
```

Here are the summary statistics for the numerical columns:
Year:

- Range from 1908 to 2023.
- Median year of the air crash is 1970.

7 Day:

- Days range from 1 to 31, as expected.

8 Sum of Ground:

- The average number of ground fatalities is approximately 1.71, but the standard deviation is quite high at around 55.13. This suggests that there are some crashes with a large number of ground fatalities.
- 75% of the crashes have no ground fatalities.
- The maximum number of ground fatalities in a single crash is 2,750.

9 Sum of Fatalities (air):

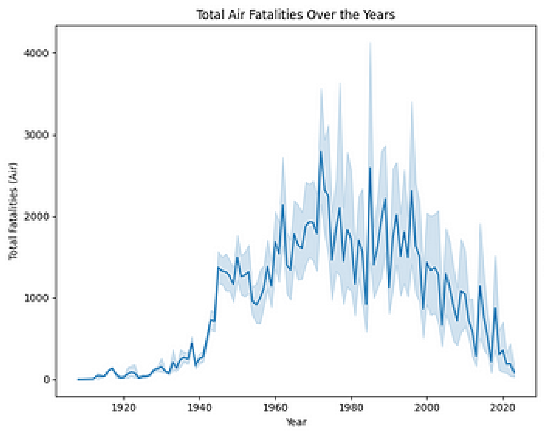
- The average number of air fatalities is around 22.24.
- The median number of air fatalities is 11, suggesting that half of the crashes have 11 or fewer fatalities on board.
- The maximum number of fatalities in a single crash is 583.

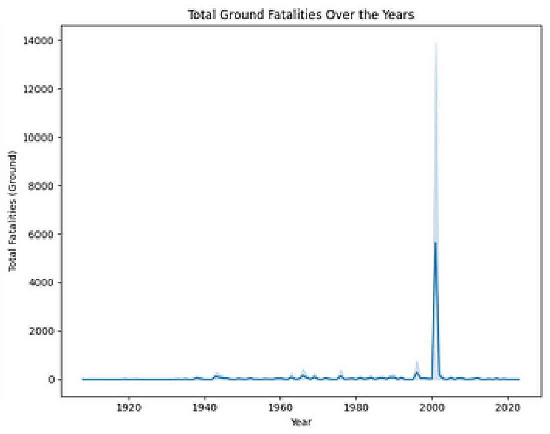
10 Sum of Aboard:

- On average, there are around 31.01 people aboard during a crash.
- The median number of people aboard is 16.
- The maximum number of people aboard in a crash is 644.

To better understand the distribution, let's visualize some of the key features. We'll start by looking at the distribution of air fatalities over the years and the distribution of ground fatalities.

```
importmatplotlib.pyplot asplt
importseaborn assns

plt.figure(figsize=(15, 6))

# Distribution of air fatalities over the years
plt.subplot(1, 2, 1)
sns.lineplot(data=air_crashes_data, x="Year", y="Sum of Fatalities (air)", estimator="sum")
plt.title("Total Air Fatalities Over the Years")
plt.ylabel("Total Fatalities (Air)")

# Distribution of ground fatalities over the years
plt.subplot(1, 2, 2)
sns.lineplot(data=air_crashes_data, x="Year", y="Sum of Ground", estimator="sum")
plt.title("Total Ground Fatalities Over the Years")
plt.ylabel("Total Fatalities (Ground)")

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





From the visualizations, we can make the following observations:
Total Air Fatalities Over the Years:

- There seems to be a general upward trend in air fatalities from the early 1900s to around the 1970s.
- Post the 1970s, there is a noticeable decline in air fatalities, although there are some years with significant spikes.

Total Ground Fatalities Over the Years:

- Ground fatalities appear to be more sporadic over the years.
- There are certain years, like around 2001, where a substantial spike in ground fatalities is observed, likely due to specific significant incidents.

11 Modify

In this phase, we will:

- Handle missing values.
- Create or transform variables if necessary.
- Prepare the data for modeling.

11.1 Handling Missing Values

We observed missing values in the “Operator" and “Country/Region" columns. There are several strategies to handle missing data, such as:

- Removing rows with missing values.
- Imputing missing values with a specific value (e.g., mode, median, mean).
- Using machine learning techniques to predict and fill missing values.

For simplicity, and given the percentage of missing values is relatively small, we can impute the missing values using the mode (most frequent value) of the respective columns. Let's proceed with this strategy.

```
# Impute missing values with mode
air_crashes_data["Operator"].fillna(air_crashes_data["Operator"].mode()[0], inplace=True)
air_crashes_data["Country/Region"].fillna(air_crashes_data["Country/Region"].mode()[0], inplace=True)

# Verify if missing values are handled
missing_values_after = air_crashes_data.isnull().sum()
missing_values_after
RESULT
Year                    0
Quarter                 0
Month                   0
Day                     0
Country/Region          0
Aircraft Manufacturer   0
Aircraft                0
Location                0
Operator                0
Sum of Ground           0
Sum of Fatalities (air) 0
Sum of Aboard           0
dtype: int64
```

All missing values have been successfully imputed and there are no missing values in the dataset now.

11.2 Variable Transformation Creation

Given the nature of the dataset, we can derive a few additional variables that might be of interest:

1. Total Fatalities: A combination of air and ground fatalities. This will give us a complete picture of the fatalities due to a crash.
2. Survivors: Difference between the number of people aboard and the air fatalities. This will provide insights into the number of survivors from each crash.

Let's create these new variables.

```
# Create new variables
air_crashes_data["Total Fatalities"] = air_crashes_data["Sum of Fatalities (air)"] + air_crashes_data["Sum of Ground"]
air_crashes_data["Survivors"] = air_crashes_data["Sum of Aboard"] - air_crashes_data["Sum of Fatalities (air)"]

# Display the first few rows to verify the new variables
air_crashes_data[["Total Fatalities", "Survivors"]].head()
RESULT
  Total Fatalities Survivors
0         1         1
1         1         0
```

```
2          5     0
3          1     0
4         14     6
```

The new variables have been successfully added:

- **Total Fatalities: Represents the combined fatalities from the air and the ground.**
- **Survivors: Indicates the number of people who survived the crash.**

With the data now modified and enriched, we can move on to the modeling phase.

12 Model

12.1 Building a Predictive Model with Scikit-learn

Steps:

1. Data Preprocessing: Convert categorical variables to numerical representations and split the data into training and testing sets.
2. Model Selection: We'll use a simple linear regression model as a starting point.

12.2 Model Training and Evaluation: Train the model on the training data and evaluate its performance on the testing data.

1. Alright! Let's proceed with building a predictive model for "Total Fatalities" using Scikit-learn.

12.3 Building a Predictive Model with Scikit-learn

Steps:

1. Data Preprocessing: Convert categorical variables to numerical representations and split the data into training and testing sets.
2. Model Selection: We'll use a simple linear regression model as a starting point.
3. Model Training and Evaluation: Train the model on the training data and evaluate its performance on the testing data.

12.4 Data Preprocessing

Before building the model, we need to preprocess the data. We'll convert categorical columns to numerical format using one-hot encoding and split the data into training and testing sets.

```
fromsklearn.model_selection importtrain_test_split
fromsklearn.preprocessing importOneHotEncoder
```

```
# Features and target variable
X = air_crashes_data.drop("Total Fatalities", axis=1)
y = air_crashes_data["Total Fatalities"]

# One-hot encoding for categorical variables
X_encoded = pd.get_dummies(X, drop_first=True)

# Split data into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X_encoded, y, test_size=0.2, random_state=42)

X_train.shape, X_test.shape
RESULT
((4024, 11481), (1006, 11481))
```

The data has been split into training and testing sets. We have:

- 4,024 samples in the training set.
- 1,006 samples in the testing set.

After one-hot encoding, our dataset has expanded to 11,481 features.

12.5 Model Selection

We'll use a simple linear regression model as a starting point. Let's proceed with training this model.

```
fromsklearn.linear_model importLinearRegression
fromsklearn.metrics importmean_squared_error, r2_score

# Initialize and train the linear regression model
lr_model = LinearRegression()
lr_model.fit(X_train, y_train)

# Predict on the testing set
y_pred = lr_model.predict(X_test)

# Evaluation metrics
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

mse, r2
```

The evaluation metrics for the linear regression model are:

- Mean Squared Error (MSE): ≈1072.73≈1072.73
- R-squared (R^2): ≈0.90≈0.90

An R^2 value of 0.90 indicates that approximately 90% of the variability in the "Total Fatalities" can be explained by our model, which is a good initial result.

13 Assess

Given the model's R² value, it seems to capture a significant portion of the variance in the data. However, it's worth noting that linear regression assumes a linear relationship between predictors and the target variable. Depending on the actual relationships and interactions in the data, more complex models like decision trees, random forests, or gradient boosting machines might provide better results.
For a production-ready solution, it would be recommended to:

- Experiment with different models.
- Tune hyperparameters.
- Possibly consider dimensionality reduction, given the large number of features after one-hot encoding.
- Validate the model on new, unseen data.
- Address any overfitting or underfitting issues.

In conclusion, while the linear regression model provides a good starting point and has decent predictive power, further iterations and refinements would be required to optimize performance for real-world applications.

14 Conclusion

15 The SEMMA methodology provided a systematic approach to exploring the

16 Air Crash dataset. Through this analysis, valuable insights were gained, and

17 initial modeling efforts were undertaken. Future work should focus on exploring more advanced modeling techniques and conducting thorough model validations.

18 References

https://scikit-learn.org/