Your First Exploratory Data Analysis (EDA) Project

2025-10-03

Contents

1 Project: Investigating the Titanic Disaster

Our Goal: We are going to analyze data from the passengers on the Titanic. Our mission is to explore the data to find patterns and clues about who was more likely to survive the famous shipwreck.

2 Our Tools

- **Python**: Our programming language.
- **Pandas**: A powerful library, like a magical, super-smart spreadsheet for Python. It helps us organize, clean, and manipulate data.
- **NumPy**: A library that is the master of mathematical calculations, especially with lists of numbers. Pandas actually uses it behind the scenes!
- **Matplotlib**: Our artist. This library helps us draw charts and graphs to visualize our findings.

3 Part 1: Setting Up Your Workshop

Before a detective starts investigating, they need their tools ready. We'll do the same.

3.1 Get a Notebook

The best way to do data analysis is in a "notebook" environment. It lets you write code and see the results immediately. I highly recommend using Google Colab. It's free, online, and has everything we need pre-installed. Just go to `https://colab.research.google.com`.

3.2 Install Libraries (If not using Colab)

If you are running Python on your own computer, you might need to install these libraries. Open your terminal or command prompt and type:

```
pip install pandas numpy matplotlib
```

4 Part 2: Getting the Evidence (The Dataset)

Every investigation needs evidence. Our evidence is the dataset.

4.1 Dataset Name

Titanic - Machine Learning from Disaster

4.2 What it is

A list of passengers on the Titanic, with details like their age, gender, ticket class, and whether they survived.

4.3 Download Link

Click here to download the Titanic dataset from Kaggle: `https://www.kaggle.com/c/titanic/data`

You might need to create a free Kaggle account to download it. This is a great website for finding fun datasets for future projects!

After downloading, save the file as `titanic.csv` in a place you can easily find. If you're using Google Colab, you can upload it directly to your ses-

sion.

A CSV (Comma-Separated Values) file is just a simple text file that acts like a spreadsheet. Each row is a new line, and each column is separated by a comma.

5 Part 3: The Investigation - Step-by-Step

Let's start our investigation! We will go through each step, explaining the code and the "why" behind it.

5.1 Step 1: Importing Our Tools

First, we need to tell our Python notebook that we want to use our special tools (Pandas, NumPy, and Matplotlib).

```
# Import the pandas library and give it a nickname 'pd'
# We use nicknames to make our code shorter and easier to type.
import pandas as pd

# Import the numpy library and give it a nickname 'np'
import numpy as np

# Import the pyplot module from matplotlib and give it a nickname '
# Pyplot is the part of matplotlib that helps us create plots.
import matplotlib.pyplot as plt

# This is a special command for Jupyter/Colab notebooks to make sur
%matplotlib inline
```

Why are we doing this?

Imagine you have a toolbox. You can't use a hammer until you take it out of the box. The import command is like taking our tools out of the toolbox so we can use them in our code.

5.2 Step 2: Loading the Data

Now, let's load our `titanic.csv` file into our magical Pandas spreadsheet, which we call a DataFrame.

```
# We use the read_csv function from pandas (pd) to load our file.
# We store the data in a variable called 'df' (short for DataFrame)
df = pd.read_csv('titanic.csv')
```

Why are we doing this?

Our data is sitting in a file. We need to bring it into our Python environment to work with it. The `pd.read_csv()` function does exactly that. The variable `df` now holds all our data.

5.3 Step 3: The First Look (Initial Reconnaissance)

A detective first scans the crime scene for a general overview. We'll do the same with our data.

5.3.1 A) Peeking at the Data

Let's see the first few rows to get a feel for what the data looks like.

```
# .head() shows us the first 5 rows of our data
df.head()
```

Explanation: `head()` is like opening a book to the first page. You see the column titles (PassengerId, Survived, Pclass, etc.) and the first few passengers' details.

Let's also look at the last few rows.

```
# .tail() shows us the last 5 rows
df.tail()
```

Explanation: `tail()` shows us the end of the file, which confirms everything loaded correctly.

5.3.2 B) Getting the Size of our Data

How many passengers (rows) and details about them (columns) do we have?

```
# .shape gives us the dimensions of our data (number of rows, numbe
df.shape
```

Explanation: This will output something like (891, 12), which means we have information on 891 passengers and 12 columns (features) for each one.

5.3.3 C) Getting a Full Summary

Let's get a more technical summary of our dataset.

```
# .info() provides a concise summary of the DataFrame.
df.info()
```

Explanation: `info()` is super useful! It tells us:

- The name of each column.
- The "Non-Null Count," which is how many cells in that column are not empty.
- The "Dtype" or data type (like int64 for numbers, float64 for numbers with decimals, and object for text).

Detective's Insight from `.info()`: Look closely! The `Age` column has only 714 non-null values, but we have 891 passengers. This means some passengers' ages are missing! The `Cabin` column has even more missing values (only 204 are filled). `Embarked` is missing just two. This is a crucial clue!

5.3.4 D) Getting a Mathematical Report Card

Let's get a quick statistical summary for all the columns that have numbers.

```
# .describe() gives us statistical details like mean, standard devi
df.describe()
```

Explanation: `.describe()` gives us a report card for our number columns.

- **count**: The number of non-empty values.
- **mean**: The average value (e.g., the average age was about 29.7 years).
- **std**: Standard deviation (how spread out the values are).
- **min**: The smallest value (e.g., the youngest passenger was about 0.42 years old).

- **25%, 50%, 75%**: These are percentiles. 50% is the median (the middle value).
- **max**: The largest value.

5.4 Step 4: Data Cleaning (The Cleanup Crew)

Our crime scene has some messes (missing data) that we need to clean up before we can analyze it properly.

5.4.1 A) Finding the Missing Values

Let's officially count the missing values in each column.

```
# .isnull() creates a table of True/False values (True if the cell
# .sum() then counts all the 'True' values in each column.
df.isnull().sum()
```

Explanation: This confirms our earlier suspicion. We have:

- Age: 177 missing values.
- Cabin: 687 missing values (this is a lot!).
- Embarked: 2 missing values.

5.4.2 B) Fixing the Missing Values (Data Imputation & Transformation)

We can't just ignore missing data. We need a strategy.

Strategy for 'Cabin': This column is missing too much data (687 out of 891). It's probably not useful for our analysis. Let's remove it completely. This is a data transformation step called dropping a column.

```
# .drop() removes rows or columns.
# 'Cabin' is the column to remove.
# axis=1 tells pandas that we want to remove a COLUMN, not a row.
# inplace=True means we are making the change permanent in our 'df'
df.drop('Cabin', axis=1, inplace=True)
```

Strategy for 'Age': Age seems like a very important clue (were children saved first?). We don't want to drop the column. Instead, let's fill the missing spots with a sensible value. A good idea is to use the median age. The median is the middle value, and it's better than the average (mean) if there are some very old or very young people that could skew the average.

```
# We calculate the median of the 'Age' column.
median_age = df['Age'].median()

# We use .fillna() to fill all the empty spots (NA) in the 'Age' co
df['Age'].fillna(median_age, inplace=True)
```

Strategy for 'Embarked': This column tells us where a passenger got on the ship. Only 2 values are missing. The best strategy here is to fill them with the most common port of embarkation. This is called the mode.

```
# .mode() finds the most frequent value. [0] is used to get the val
most_common_port = df['Embarked'].mode()[0]

# We fill the missing 'Embarked' spots with this value.
df['Embarked'].fillna(most_common_port, inplace=True)
```

5.4.3 C) Final Check!

Let's run our check again to make sure all missing values are gone.

```
# Let's check for any remaining missing values.
df.isnull().sum()
```

Success! The output should show 0 for all columns. Our data is now clean.

5.5 Step 5: Feature Engineering (Creating New Clues)

Sometimes, we can create new, more useful clues from the ones we already have. This is called Feature Engineering.

Let's look at the `SibSp` (number of siblings/spouses aboard) and `Parch` (number of parents/children aboard) columns. We can combine them to create a `FamilySize` column!

```
# We create a new column called 'FamilySize'.
# We add the values from 'SibSp', 'Parch', and add 1 (for the passe
df['FamilySize'] = df['SibSp'] + df['Parch'] + 1
```

Now let's remove columns we don't need for our analysis. `PassengerId`, `Name`, `Ticket`, `SibSp`, and `Parch` are not very useful for finding general patterns.

```
# We drop multiple columns at once by providing a list of names.
df.drop(['PassengerId', 'Name', 'Ticket', 'SibSp', 'Parch'], axis=1

# Let's look at our new, cleaner DataFrame!
df.head()
```

Explanation: Our DataFrame is now simpler and more focused. We have the important information and even a brand new clue, `FamilySize`. This is data manipulation.

5.6 Step 6: Data Visualization (Telling the Story with Pictures)

Numbers are great, but pictures tell a story much better. Let's use Matplotlib to draw some charts.

5.6.1 A) Who Survived? (Univariate Analysis)

Let's start with the most important question: How many people survived versus how many did not? (This is called Univariate Analysis because we are looking at just one variable: `Survived`).

```
# .value_counts() counts how many times each value appears in the '
# 0 = Did not survive, 1 = Survived
survival_counts = df['Survived'].value_counts()
print(survival_counts)

# Let's make a bar chart
plt.figure(figsize=(6,6))
plt.bar(survival_counts.index, survival_counts.values, tick_label=[
plt.title('Survival Count')
plt.ylabel('Number of Passengers')
plt.show()
```

Detective's Insight: The chart clearly shows that many more people did not survive than those who did.

5.6.2 B) Does Ticket Class Matter for Survival? (Bivariate Analysis)

Let's see if there's a connection between the passenger's class (`Pclass`) and their survival. (This is Bivariate Analysis because we are comparing two variables).

```
# We use pandas crosstab to create a table that shows survival coun
class_survival = pd.crosstab(df['Pclass'], df['Survived'])
print(class_survival)

# Let's plot this.
class_survival.plot(kind='bar')
plt.title('Survival Count by Passenger Class')
plt.xlabel('Passenger Class')
plt.ylabel('Number of Passengers')
plt.xticks(rotation=0, ticks=[0,1,2], labels=['1st Class', '2nd Cla
plt.legend(['Did Not Survive', 'Survived'])
plt.show()
```

Detective's Insight: Wow! This is a huge clue.

- In 1st Class, more people survived than died.
- In 3rd Class, far more people died than survived.

This strongly suggests that your ticket class had a big impact on your chances of survival.

5.6.3 C) Does Gender Matter for Survival?

What about gender? Was the "women and children first" rule followed?

```
# Create a crosstab for Sex and Survived
gender_survival = pd.crosstab(df['Sex'], df['Survived'])
print(gender_survival)

# Plot the results
gender_survival.plot(kind='bar')
plt.title('Survival Count by Gender')
plt.xlabel('Gender')
plt.ylabel('Number of Passengers')
plt.xticks(rotation=0)
plt.legend(['Did Not Survive', 'Survived'])
plt.show()
```

Detective's Insight: Absolutely incredible. A much larger proportion of females survived compared to males. This is another very strong clue.

5.6.4 D) What about Age?

Let's look at the age distribution of passengers who survived versus those who didn't. A histogram is great for this.

```
# Create a figure with two subplots side-by-side
fig, ax = plt.subplots(1, 2, figsize=(14, 6))

# Plot histogram for those who did not survive
ax[0].hist(df[df['Survived']==0]['Age'], bins=20, color='red')
ax[0].set_title('Age Distribution of Non-Survivors')
ax[0].set_xlabel('Age')
ax[0].set_ylabel('Number of Passengers')

# Plot histogram for those who survived
```

```
ax[1].hist(df[df['Survived']==1]['Age'], bins=20, color='green')
ax[1].set_title('Age Distribution of Survivors')
ax[1].set_xlabel('Age')

plt.show()
```

Detective's Insight:

Look at the green "Survivors" chart on the right. There's a big spike for very young children (ages 0-5). This suggests that many children were indeed saved.

For adults, the number of non-survivors is higher across most age groups, especially for people in their 20s and 30s.

6 Part 4: The Detective's Final Report (Summary of Findings)

After our detailed investigation, we can summarize our key findings:

- Survival was not random. There were clear patterns that influenced who survived the Titanic disaster.
- **Social Class was a Major Factor**: Passengers in First Class had a significantly higher chance of survival than those in Second Class, and especially those in Third Class.
- **Gender Played a Crucial Role**: Females had a much higher survival rate than males, suggesting the "women and children first" protocol was followed to some extent.

- **Age Mattered**: A large number of very young children were among the survivors, indicating they were given priority. A large number of passengers in their 20s and 30s (the largest age group on the ship) did not survive.

7 Conclusion

Congratulations! You have just completed your first full Exploratory Data Analysis project.

You started with a raw data file, just like a real data analyst. You then:

- Loaded and inspected the data.
- Cleaned it by handling missing values.
- Transformed it by dropping unnecessary columns.
- Performed feature engineering by creating a new `FamilySize` column.
- Analyzed and visualized the data to uncover hidden stories and insights.

EDA is all about being curious, asking questions, and using your tools to make the data tell its story. You can now take these skills and apply them to any dataset you find interesting!