Day29_EDA_WITH_LLM

June 27, 2025

1 LLM-Powered Exploratory Data Analysis (EDA) using Python, Pandas, Gradio & Mistral

1.1 Objective

In this notebook, we will: - Perform manual EDA using Python and seaborn - Identify missing values and distributions - Use a Large Language Model (LLM) to generate intelligent EDA insights - Build a Gradio-based AI-powered web app for EDA automation

---

1.2 Install Required Libraries

We will use: - `pandas` for data handling - `matplotlib` and `seaborn` for visualizations - `gradio` for the web interface - `ollama` for AI-generated insights using Mistral LLM

[2]:
```
# Install requried packages if dont have
! pip install pandas seaborn matplotlib gradio ollama
```

```
Requirement already satisfied: pandas in c:\users\lenovo\anaconda3\lib\site-
packages (2.2.3)
Requirement already satisfied: seaborn in c:\users\lenovo\anaconda3\lib\site-
packages (0.13.2)
Requirement already satisfied: matplotlib in c:\users\lenovo\anaconda3\lib\site-
packages (3.10.0)
Requirement already satisfied: gradio in c:\users\lenovo\anaconda3\lib\site-
packages (5.34.2)
Requirement already satisfied: ollama in c:\users\lenovo\anaconda3\lib\site-
packages (0.5.1)
Requirement already satisfied: numpy>=1.26.0 in
c:\users\lenovo\anaconda3\lib\site-packages (from pandas) (2.1.3)
Requirement already satisfied: python-dateutil>=2.8.2 in
c:\users\lenovo\anaconda3\lib\site-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in
c:\users\lenovo\anaconda3\lib\site-packages (from pandas) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in
c:\users\lenovo\anaconda3\lib\site-packages (from pandas) (2025.2)
Requirement already satisfied: contourpy>=1.0.1 in
c:\users\lenovo\anaconda3\lib\site-packages (from matplotlib) (1.3.1)
```

```
Requirement already satisfied: cycler>=0.10 in
c:\users\lenovo\anaconda3\lib\site-packages (from matplotlib) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in
c:\users\lenovo\anaconda3\lib\site-packages (from matplotlib) (4.55.3)
Requirement already satisfied: kiwisolver>=1.3.1 in
c:\users\lenovo\anaconda3\lib\site-packages (from matplotlib) (1.4.8)
Requirement already satisfied: packaging>=20.0 in
c:\users\lenovo\anaconda3\lib\site-packages (from matplotlib) (24.2)
Requirement already satisfied: pillow>=8 in c:\users\lenovo\anaconda3\lib\site-
packages (from matplotlib) (11.1.0)
Requirement already satisfied: pyparsing>=2.3.1 in
c:\users\lenovo\anaconda3\lib\site-packages (from matplotlib) (3.2.0)
Requirement already satisfied: aiofiles<25.0,>=22.0 in
c:\users\lenovo\anaconda3\lib\site-packages (from gradio) (24.1.0)
Requirement already satisfied: anyio<5.0,>=3.0 in
c:\users\lenovo\anaconda3\lib\site-packages (from gradio) (4.7.0)
Requirement already satisfied: audioop-lts<1.0 in
c:\users\lenovo\anaconda3\lib\site-packages (from gradio) (0.2.1)
Requirement already satisfied: fastapi<1.0,>=0.115.2 in
c:\users\lenovo\anaconda3\lib\site-packages (from gradio) (0.115.13)
Requirement already satisfied: ffmpy in c:\users\lenovo\anaconda3\lib\site-
packages (from gradio) (0.6.0)
Requirement already satisfied: gradio-client==1.10.3 in
c:\users\lenovo\anaconda3\lib\site-packages (from gradio) (1.10.3)
Requirement already satisfied: groovy~=0.1 in
c:\users\lenovo\anaconda3\lib\site-packages (from gradio) (0.1.2)
Requirement already satisfied: httpx>=0.24.1 in
c:\users\lenovo\anaconda3\lib\site-packages (from gradio) (0.28.1)
Requirement already satisfied: huggingface-hub>=0.28.1 in
c:\users\lenovo\anaconda3\lib\site-packages (from gradio) (0.33.1)
Requirement already satisfied: jinja2<4.0 in c:\users\lenovo\anaconda3\lib\site-
packages (from gradio) (3.1.6)
Requirement already satisfied: markupsafe<4.0,>=2.0 in
c:\users\lenovo\anaconda3\lib\site-packages (from gradio) (3.0.2)
Requirement already satisfied: orjson~=3.0 in
c:\users\lenovo\anaconda3\lib\site-packages (from gradio) (3.10.18)
Requirement already satisfied: pydantic<2.12,>=2.0 in
c:\users\lenovo\anaconda3\lib\site-packages (from gradio) (2.10.3)
Requirement already satisfied: pydub in c:\users\lenovo\anaconda3\lib\site-
packages (from gradio) (0.25.1)
Requirement already satisfied: python-multipart>=0.0.18 in
c:\users\lenovo\anaconda3\lib\site-packages (from gradio) (0.0.20)
Requirement already satisfied: pyyaml<7.0,>=5.0 in
c:\users\lenovo\anaconda3\lib\site-packages (from gradio) (6.0.2)
Requirement already satisfied: ruff>=0.9.3 in
c:\users\lenovo\anaconda3\lib\site-packages (from gradio) (0.12.0)
Requirement already satisfied: safehttpx<0.2.0,>=0.1.6 in
c:\users\lenovo\anaconda3\lib\site-packages (from gradio) (0.1.6)
```

```
Requirement already satisfied: semantic-version~=2.0 in
c:\users\lenovo\anaconda3\lib\site-packages (from gradio) (2.10.0)
Requirement already satisfied: starlette<1.0,>=0.40.0 in
c:\users\lenovo\anaconda3\lib\site-packages (from gradio) (0.46.2)
Requirement already satisfied: tomlkit<0.14.0,>=0.12.0 in
c:\users\lenovo\anaconda3\lib\site-packages (from gradio) (0.13.2)
Requirement already satisfied: typer<1.0,>=0.12 in
c:\users\lenovo\anaconda3\lib\site-packages (from gradio) (0.16.0)
Requirement already satisfied: typing-extensions~=4.0 in
c:\users\lenovo\anaconda3\lib\site-packages (from gradio) (4.12.2)
Requirement already satisfied: uvicorn>=0.14.0 in
c:\users\lenovo\anaconda3\lib\site-packages (from gradio) (0.34.3)
Requirement already satisfied: fsspec in c:\users\lenovo\anaconda3\lib\site-
packages (from gradio-client==1.10.3->gradio) (2025.3.2)
Requirement already satisfied: websockets<16.0,>=10.0 in
c:\users\lenovo\anaconda3\lib\site-packages (from gradio-client==1.10.3->gradio)
(15.0.1)
Requirement already satisfied: idna>=2.8 in c:\users\lenovo\anaconda3\lib\site-
packages (from anyio<5.0,>=3.0->gradio) (3.7)
Requirement already satisfied: sniffio>=1.1 in
c:\users\lenovo\anaconda3\lib\site-packages (from anyio<5.0,>=3.0->gradio)
(1.3.0)
Requirement already satisfied: annotated-types>=0.6.0 in
c:\users\lenovo\anaconda3\lib\site-packages (from pydantic<2.12,>=2.0->gradio)
(0.6.0)
Requirement already satisfied: pydantic-core==2.27.1 in
c:\users\lenovo\anaconda3\lib\site-packages (from pydantic<2.12,>=2.0->gradio)
(2.27.1)
Requirement already satisfied: click>=8.0.0 in
c:\users\lenovo\anaconda3\lib\site-packages (from typer<1.0,>=0.12->gradio)
(8.1.8)
Requirement already satisfied: shellingham>=1.3.0 in
c:\users\lenovo\anaconda3\lib\site-packages (from typer<1.0,>=0.12->gradio)
(1.5.0)
Requirement already satisfied: rich>=10.11.0 in
c:\users\lenovo\anaconda3\lib\site-packages (from typer<1.0,>=0.12->gradio)
(13.9.4)
Requirement already satisfied: colorama in c:\users\lenovo\anaconda3\lib\site-
packages (from click>=8.0.0->typer<1.0,>=0.12->gradio) (0.4.6)
Requirement already satisfied: certifi in c:\users\lenovo\anaconda3\lib\site-
packages (from httpx>=0.24.1->gradio) (2025.4.26)
Requirement already satisfied: httpcore==1.* in
c:\users\lenovo\anaconda3\lib\site-packages (from httpx>=0.24.1->gradio) (1.0.9)
Requirement already satisfied: h11>=0.16 in c:\users\lenovo\anaconda3\lib\site-
packages (from httpcore==1.*->httpx>=0.24.1->gradio) (0.16.0)
Requirement already satisfied: filelock in c:\users\lenovo\anaconda3\lib\site-
packages (from huggingface-hub>=0.28.1->gradio) (3.17.0)
Requirement already satisfied: requests in c:\users\lenovo\anaconda3\lib\site-
```

```
packages (from huggingface-hub>=0.28.1->gradio) (2.32.3)
Requirement already satisfied: tqdm>=4.42.1 in
c:\users\lenovo\anaconda3\lib\site-packages (from huggingface-
hub>=0.28.1->gradio) (4.67.1)
Requirement already satisfied: six>=1.5 in c:\users\lenovo\anaconda3\lib\site-
packages (from python-dateutil>=2.8.2->pandas) (1.17.0)
Requirement already satisfied: markdown-it-py>=2.2.0 in
c:\users\lenovo\anaconda3\lib\site-packages (from
rich>=10.11.0->typer<1.0,>=0.12->gradio) (2.2.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in
c:\users\lenovo\anaconda3\lib\site-packages (from
rich>=10.11.0->typer<1.0,>=0.12->gradio) (2.19.1)
Requirement already satisfied: mdurl~=0.1 in c:\users\lenovo\anaconda3\lib\site-
packages (from markdown-it-py>=2.2.0->rich>=10.11.0->typer<1.0,>=0.12->gradio)
(0.1.0)
Requirement already satisfied: charset-normalizer<4,>=2 in
c:\users\lenovo\anaconda3\lib\site-packages (from requests->huggingface-
hub>=0.28.1->gradio) (3.3.2)
Requirement already satisfied: urllib3<3,>=1.21.1 in
c:\users\lenovo\anaconda3\lib\site-packages (from requests->huggingface-
hub>=0.28.1->gradio) (2.3.0)
```

1.3 Load the Titanic Dataset

We'll start by loading the dataset into a pandas DataFrame.

[3]:
```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Load Titanic Dataset
url = r"E:\Data backup 25.6.25\Downloads\titanic_ dataset_final.csv"
df = pd.read_csv(url)
df
```

[3]:
```
     PassengerId  Survived  Pclass  \
0              1         0       3
1              2         1       1
2              3         1       3
3              4         1       1
4              5         0       3
..           …         …       …
886          887         0       2
887          888         1       1
888          889         0       3
889          890         1       1
890          891         0       3
```

```
                                                  Name     Sex   Age  SibSp  \
0                              Braund, Mr. Owen Harris    male  22.0      1
1    Cumings, Mrs. John Bradley (Florence Briggs Th…  female  38.0      1
2                               Heikkinen, Miss. Laina  female  26.0      0
3         Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                             Allen, Mr. William Henry    male  35.0      0
..                                                 …     …   …    …
886                              Montvila, Rev. Juozas    male  27.0      0
887                       Graham, Miss. Margaret Edith  female  19.0      0
888           Johnston, Miss. Catherine Helen "Carrie"  female   NaN      1
889                              Behr, Mr. Karl Howell    male  26.0      0
890                                Dooley, Mr. Patrick    male  32.0      0

     Parch            Ticket     Fare Cabin Embarked
0        0         A/5 21171   7.2500   NaN        S
1        0          PC 17599  71.2833   C85        C
2        0  STON/O2. 3101282   7.9250   NaN        S
3        0            113803  53.1000  C123        S
4        0            373450   8.0500   NaN        S
..     …               …      …   …      …
886      0            211536  13.0000   NaN        S
887      0            112053  30.0000   B42        S
888      2        W./C. 6607  23.4500   NaN        S
889      0            111369  30.0000  C148        C
890      0            370376   7.7500   NaN        Q

[891 rows x 12 columns]
```

1.4 View Statistical Summary

This gives basic descriptive statistics like count, mean, std deviation, etc.

[4]:
```
print(df.describe())
```

```
       PassengerId    Survived      Pclass         Age       SibSp  \
count   891.000000  891.000000  891.000000  714.000000  891.000000
mean    446.000000    0.383838    2.308642   29.699118    0.523008
std     257.353842    0.486592    0.836071   14.526497    1.102743
min       1.000000    0.000000    1.000000    0.420000    0.000000
25%     223.500000    0.000000    2.000000   20.125000    0.000000
50%     446.000000    0.000000    3.000000   28.000000    0.000000
75%     668.500000    1.000000    3.000000   38.000000    1.000000
max     891.000000    1.000000    3.000000   80.000000    8.000000

            Parch        Fare
count  891.000000  891.000000
mean     0.381594   32.204208
std      0.806057   49.693429
```

```
min      0.000000    0.000000
25%      0.000000    7.910400
50%      0.000000   14.454200
75%      0.000000   31.000000
max      6.000000  512.329200
```

1.5 Check for Missing Values

This shows the number of missing entries for each column.

[5]:
```
print("\nMissing Values:\n", df.isnull().sum())
```

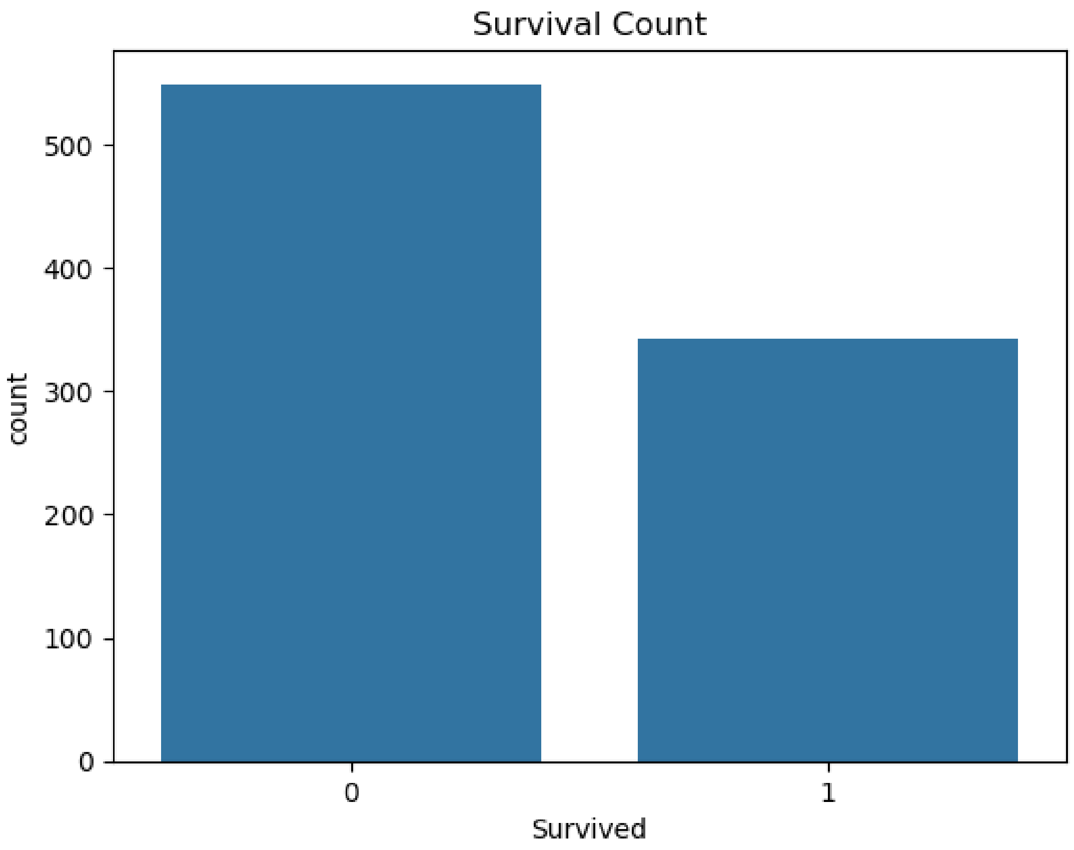
```

Missing Values:
 PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

1.6 Plot Survival Distribution

We visualize how many passengers survived vs. not survived.

[6]:
```
sns.countplot(x='Survived', data=df)
plt.title("Survival Count")
plt.show()
```



1.7 What if AI Could Do All This Automatically?

Let's now use a large language model (Mistral) through `ollama` to generate insights automatically.

1.8 AI-Generated Insights Using Mistral

We pass the summary to the LLM and get automatic interpretation.

[8]:
```
import ollama

def generate_insights(df_summary):
    prompt = f"Analyze the dataset summary and provide insights:
↪\n\n{df_summary}"
    response = ollama.chat(model="mistral", messages=[{"role": "user",␣
↪"content": prompt}])
    return response['message']['content']

# Generate AI Insights
summary = df.describe().to_string()
insights = generate_insights(summary)
```

```
print("\n -- AI-Generated Insights:\n", insights)
```

```

 -- AI-Generated Insights:
  This dataset appears to be the Titanic passenger manifest data, which includes
variables such as PassengerID, Survived (1 if survived, 0 if not), Pclass
(ticket class), Age, SibSp (number of siblings or spouses aboard the ship),
Parch (number of parents or children aboard), and Fare.

Here are some insights from the dataset summary:

1. The total number of passengers is 891. There were no missing values for any
variable in this dataset.

2. The mean age of passengers is approximately 29.7 years, with a standard
deviation of around 14.53 years. This suggests a wide distribution of ages among
the passengers.

3. Majority of the passengers survived (mean Survived = 0.38), but there were
still many who did not survive (mean Survived = 0.62 would mean all passengers
survived).

4. Most passengers traveled in 3rd class (mean Pclass = 2.31, median Pclass =
3.0), suggesting a high number of passengers traveling in the cheapest class.

5. On average, each passenger had about 0.5 family members or travel companions
aboard the ship (SibSp + Parch).

6. The fare varies greatly, with a mean of around 32.20 and a standard deviation
of approximately 49.69. This indicates that there was significant disparity in
ticket prices among passengers.

7. The youngest passenger was only 0.42 years old, while the oldest was 80 years
old. The median age is 28, with 25% and 75% of passengers being younger (20.125)
and older (38) than this value, respectively.

8. Interestingly, despite the large number of passengers in 3rd class, it seems
that a non-zero percentage of first-class passengers did not survive (mean
Survived = 0.68 for Pclass=1). This could potentially be an interesting factor
to explore further when analyzing survival rates based on different ticket
classes.
```

1.9 Create a Basic Web App for EDA Using Gradio

This app lets users upload a CSV file and get insights.

[9]:
```
import gradio as gr
import pandas as pd
```

```python
def eda_analysis(file):
    df = pd.read_csv(file.name)
    summary = df.describe().to_string()
    insights = generate_insights(summary)  # This should be defined separately
    return insights

# Create Web Interface
demo = gr.Interface(
    fn=eda_analysis,
    inputs=gr.File(label="Upload your CSV file", file_types=[".csv"]),
    outputs=gr.Textbox(label="AI-Generated Insights", lines=20),
    title="AI-Powered EDA with Mistral",
    description="""
    <div style="text-align: center;">
        <h3 style="color: #4CAF50; margin-bottom: 0;">Explore Your Dataset␣
 ↪Instantly</h3>
        <p style="color: #555;">Upload a CSV file and get an instant,␣
 ↪intelligent summary and analysis powered by AI.</p>
    </div>
    """,
    article="""
    <p style="text-align: center; font-size: 14px; color: gray;">
        Built using <strong>Pandas</strong> and <strong>Gradio</strong>. AI␣
 ↪insights are generated by a custom model powered by Mistral.
        <br>Developed for automated exploratory data analysis (EDA).
    </p>
    """,
    theme="huggingface"  # You can also try "default", "soft", or "grass"
)

# Launch App
demo.launch(share=True)
```

```
C:\Users\Lenovo\anaconda3\Lib\site-packages\gradio\blocks.py:1180: UserWarning:
Cannot load huggingface. Caught Exception: 404 Client Error: Not Found for url:
https://huggingface.co/api/spaces/huggingface (Request ID:
Root=1-685e9283-3fb0de2e00eacb090c3e543f;d93e573a-7699-4138-933b-dea9a97fbd20)

Sorry, we can't find the page you are looking for.
  warnings.warn(f"Cannot load {theme}. Caught Exception: {str(e)}")
```

```
* Running on local URL:  http://127.0.0.1:7860
* Running on public URL: https://34a9e0105dc7e46b81.gradio.live

This share link expires in 1 week. For free permanent hosting and GPU upgrades,
run `gradio deploy` from the terminal in the working directory to deploy to
Hugging Face Spaces (https://huggingface.co/spaces)
```

```
<IPython.core.display.HTML object>
```

[9]:

1.10 Final Web App with Summary, Missing Values, Insights, and Visuals

This version also adds histograms and a correlation heatmap automatically.

[12]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import gradio as gr
import os

# Generate Insights (placeholder)
def generate_insights(summary):
    return "Key Insights:\n- Dataset has been summarized.\n- Add␣
 ↪model-generated insights here."

# Create visualizations
def generate_visualizations(df):
    plot_paths = []
    os.makedirs("plots", exist_ok=True)

    for col in df.select_dtypes(include=['number']).columns:
        plt.figure(figsize=(6, 4))
        sns.histplot(df[col], bins=20, kde=True, color="skyblue")
        plt.title(f"Distribution of {col}")
        plot_path = f"plots/{col}_hist.png"
        plt.savefig(plot_path)
        plot_paths.append(plot_path)
        plt.close()

    return plot_paths

# EDA function
def eda_analysis(file):
    df = pd.read_csv(file.name)

    # Fill missing values
    for col in df.select_dtypes(include=['number']).columns:
        df[col].fillna(df[col].median(), inplace=True)

    for col in df.select_dtypes(include=['object']).columns:
        df[col].fillna(df[col].mode()[0], inplace=True)

    # Summary
    summary = df.describe(include='all').to_string()
    missing = df.isnull().sum().to_string()

    # Insights and visualizations
    insights = generate_insights(summary)
    plot_paths = generate_visualizations(df)

    report = f"Summary Statistics:\n\n{summary}\n\nMissing Values:
 ↪\n{missing}\n\nAI Insights:\n{insights}"

    return report, plot_paths

# Gradio Interface
demo = gr.Interface(
    fn=eda_analysis,
    inputs=gr.File(label="Upload your CSV file", file_types=[".csv"]),
    outputs=[
        gr.Textbox(label="EDA Report", lines=30),
        gr.Gallery(label="Visualizations")
    ],
    title="AI-Powered Exploratory Data Analysis",
    description="""
    Upload a CSV file to get automatic EDA insights. This includes summary␣
 ↪statistics, missing value handling, and visualizations.
    """,
    article="""
    Built using Pandas, Seaborn, and Gradio. You can enhance this with custom␣
 ↪models for deeper insights.
    """,
    theme="huggingface"
)

demo.launch(share=True)
```

```
C:\Users\Lenovo\anaconda3\Lib\site-packages\gradio\blocks.py:1180: UserWarning:
Cannot load huggingface. Caught Exception: 404 Client Error: Not Found for url:
https://huggingface.co/api/spaces/huggingface (Request ID:
Root=1-685e9741-58cddf7b65c7c13251909fb9;24b70883-cefd-4fbd-be16-5a8fc21deaec)

Sorry, we can't find the page you are looking for.
  warnings.warn(f"Cannot load {theme}. Caught Exception: {str(e)}")

* Running on local URL:  http://127.0.0.1:7862
* Running on public URL: https://0d85562b21700e438a.gradio.live

This share link expires in 1 week. For free permanent hosting and GPU upgrades,
run `gradio deploy` from the terminal in the working directory to deploy to
Hugging Face Spaces (https://huggingface.co/spaces)
```

```
<IPython.core.display.HTML object>
```

[12]:

```
C:\Users\Lenovo\AppData\Local\Temp\ipykernel_8108\1815311375.py:33:
FutureWarning: A value is trying to be set on a copy of a DataFrame or Series
through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  df[col].fillna(df[col].median(), inplace=True)
C:\Users\Lenovo\AppData\Local\Temp\ipykernel_8108\1815311375.py:36:
FutureWarning: A value is trying to be set on a copy of a DataFrame or Series
through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  df[col].fillna(df[col].mode()[0], inplace=True)
```

1.11 Conclusion

With just a few lines of code, we've: - Explored the Titanic dataset manually - Generated AI-powered summaries and visualizations - Built two different web apps using Gradio: one simple, one advanced - Integrated LLMs like Mistral using Ollama for smart insights

This is a scalable, interactive approach to rapid data exploration in real-world projects.

[ ]:

```
```