**📦 Imports and Dependencies**

python

import gradio as gr

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import ollama

* **gradio**: Used to build interactive web UIs.
* **pandas**: For reading and processing CSV files.
* **matplotlib.pyplot & seaborn**: For data visualizations.
* **ollama**: For chatting with an LLM model (like mistral) locally to generate text-based insights.

**🔍 Main EDA Function**

python

def eda\_analysis(file\_path):

df = pd.read\_csv(file\_path)

* Reads the uploaded CSV file into a **DataFrame df**.

**🧹 Handling Missing Value**

for col in df.select\_dtypes(include=['number']).columns:

df[col].fillna(df[col].median(), inplace=True)

* For **numeric columns**, fills missing values using the **median**.

for col in df.select\_dtypes(include=['object']).columns:

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

* For **categorical columns**, fills missing values using the **mode** (most frequent value).

**📈 Generating Summary and Missing Values Report**

summary = df.describe(include='all').to\_string()

missing\_values = df.isnull().sum().to\_string()

* df.describe(include='all') generates summary stats (like count, mean, unique values).
* .isnull().sum() counts nulls column-wise.
* .to\_string() converts these summaries into string format for display.

**🤖 LLM-Generated Insights**

insights = generate\_ai\_insights(summary)

* Calls an **AI function** (defined later) to analyze the data summary and return **natural language insights**.

**📊 Generating Visualizations**

plot\_paths = generate\_visualizations(df)

* Generates and saves plots (histograms and heatmap), returns a list of image file paths.

**📦 Final Return**

return f"\n Data Loaded Successfully!\n\n Summary:\n{summary}\n\n Missing Values:\n{missing\_values}\n\n AI Insights:\n{insights}", plot\_paths

* Returns a **multi-section textual report** and the **list of plot image paths**.

**🤖 LLM Insight Generator Function**

def generate\_ai\_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']

* Sends the data summary to a **local Mistral LLM** using ollama.chat(...) to generate textual insights.
* Expects ollama and mistral to be installed locally (ollama run mistral should work independently).

**📊 Plot Generator Function**

def generate\_visualizations(df):

plot\_paths = []

* Initializes an empty list to collect file paths of saved plots.

**1. Histograms for Numeric Columns**

python

CopyEdit

for col in df.select\_dtypes(include=['number']).columns:

...

path = f"{col}\_distribution.png"

...

* Loops over numeric columns.
* Plots histogram + KDE using **seaborn**, saves each plot as PNG.
* Appends each image path to plot\_paths.

**2. Correlation Heatmap**

numeric\_df = df.select\_dtypes(include=['number'])

if not numeric\_df.empty:

...

path = "correlation\_heatmap.png"

* If there are numeric columns, generates a **correlation matrix heatmap**.
* Saves it as "correlation\_heatmap.png" and adds it to the list.

return plot\_paths

* Returns all plot file paths.

**🧪 Gradio Interface**

demo = gr.Interface(

fn=eda\_analysis,

inputs=gr.File(type="filepath"),

outputs=[gr.Textbox(label="EDA Report"), gr.Gallery(label="Data Visualizations")],

...

)

* fn=eda\_analysis: Uses your EDA function as the main logic.
* inputs: Accepts a **file input** (CSV).
* outputs: Displays a **textbox report** and **gallery of visualizations**.
* title, description: Displayed at the top of the UI.

**🚀 Launch the Gradio App**

demo.launch(share=True)

* Starts the Gradio web app.
* share=True: Generates a public URL you can share and test on the web.

**🔁 Summary of Workflow:**

1. Upload CSV.
2. App cleans and summarizes data.
3. AI (via Ollama + Mistral) generates natural language insights.
4. Histograms and heatmaps are saved and displayed.
5. Outputs are returned as report + visual gallery.

Top of Form

Bottom of Form