Project 2 - Automated Analysis

Your task is to:

1. Write a Python script that uses an LLM to analyze, visualize, and narrate a story from a dataset.
2. Convince an LLM that your script and output are of high quality.

Write a Python script

Your submission must be a Python script, autolysis.py submitted via a Git repository.

The Python script must accept a single CSV filename like this: **uv run autolysis.py dataset.csv** *(Read more about uv at the end of this document).*

This should create, in your current directory, the following files:

- A single Markdown file called README.md with results of your automated analysis, written as a story.
- 1-3 charts as PNG providing supporting data visualizations. Name them as *.png. Add the images in your README.md.

You can try this on these sample datasets:

- goodreads.csv: 10,000 books from GoodReads with their genres, ratings, etc.
- happiness.csv: Data from the World Happiness Report
- media.csv: The course faculty's rating of movies, TV series, and books.

Notes:

- **Create a single Python script**: Don't load other local scripts (e.g. utils.py) or files (e.g. system-prompt.txt). Keep the entire code in autolysis.py. This eases LLM evaluation.
- Create and use your AI Proxy Token. The free credits of $5 are now available for new accounts. The project can be well completed under $2, if you are required to pay for the token. Please understand token pricing before using. KaroStartup will not be paying or reimbursing for these AI Proxy tokens.
- **Use the AIPROXY_TOKEN environment variable**. DON'T commit your AI Proxy token to your repository. Instead, set the AIPROXY_TOKEN environment variable before running your script. Use os.environ["AIPROXY_TOKEN"] as the token in your script.
- **Stick to GPT-4o-Mini**. This is the only generation model that AI Proxy currently supports. When this page says "LLM", it means GPT-4o-Mini.

Analyze the data

Your script should work with *any* valid CSV file.

Since you don't know in advance what the data looks like, don't make assumptions. Instead:

1. **Do generic analysis** that will apply to all datasets. For example, summary statistics, counting missing values, correlation matrices, outliers, clustering, hierarchy detection, etc.
2. **Ask the LLM** to analyze the data. Send the LLM your filename, column names & types, and additional context (like summary statistics, example values, etc.) to the LLM. Then
    i. **For code**. Have the LLM give you Python code and run it. This is risky because LLM code might fail and your program might terminate. Use with caution.
    ii. **For summaries**. Have the LLM summarize your generic analysis. Use liberally.
    iii. **For function calls**. Have the LLM suggest specific function calls or analyses that will give you more insights, and run them. Use liberally.

Here are some ideas for analysis that may be insightful. *Some* of these are covered in this course, but don't limit yourself.

- Outlier and Anomaly Detection: You might find errors, fraud, or high-impact opportunities.
- Correlation Analysis, Regression Analysis, and Feature Importance Analysis: You might find what to improve to impact an outcome.
- Time Series Analysis: You might find patterns that help predict the future.
- Cluster Analysis: You might find natural groupings for targeted marketing or resource allocation.
- Geographic Analysis: You might find where the biggest problems or opportunities are.
- Network Analysis: You might find what to cross-sell or collaborate with.

Notes:

- **Don't send the entire dataset to the LLM**: LLMs are bad at numbers - even arithmetic. Also, you'll run out of tokens. Instead, do your analysis in Python and send relevant summaries.
- **You can consult the LLM multiple times**. If one analysis doesn't work, ask for another. Or, share the results of one analysis to help the next one.

Visualize your results

When you have your analysis, visualize the results.

For analysis that you wrote the code for, you know the structure of the output. So you can write functions to create the chart(s) for the structure they generate.

For example, if you create a correlation matrix, you can write a function to visualize it as a heatmap.

You need a Python-based library to create charts. We suggest Seaborn but you're welcome to use Matplotlib. We wouldn't recommend Bokeh, Plotly or Altair unless you know how to get them to work without a browser.

You could get creative and ask the LLM to generate the code for your chart based on the data. This is risky because LLM code might fail and your program might terminate. Use with caution.

Notes:

- **Export as PNG**. Save all your charts as PNG files (with different file names) in the current directory.
- **Don't send the entire analysis to the LLM**. You might run out of tokens. Send only what might be significant analysis.

Narrate a story

Use the LLM to write a story about your analysis. You can pass it your data structure, analysis, and even your charts. Have it describe:

1. The data you received, briefly
2. The analysis you carried out
3. The insights you discovered
4. The implications of your findings (i.e. what to do with the insights)

Save this as README.md.

Notes:

- **Keep images small**. 512x512 px images are ideal. That's the size of 1 tile. Or, send detail: low to reduce cost.

Submit your script

- Create a new public repository in your GitHub account with an MIT license.
- Add your code: autolysis.py
- Create directories called goodreads/, happiness/, and media/.
- Run your script on the respective CSV files and commit the **output** in that directory. (Don't commit the input CSVs. Just README.md and *.png files.)
- Commit and push these.
- Submit your project through this link: https://docs.google.com/forms/d/e/1FAIpQLSdVlN8airffzimiTTDlVA0hak5pMeBBTg0P29fNGD_lAE_ZiQ/viewform?usp=sharing

Evaluation

Here is how your script will be evaluated.

Submission

- **REQUIRED**: If your repository is public and has an MIT License. **NOTE** If you don't make your repo public or add an MIT license, the code will not be evaluated.
- If the repository has the required files
    - autolysis.py
    - goodreads/README.md
    - goodreads/*.png
    - happiness/README.md
    - happiness/*.png
    - media/README.md
    - media/*.png
- If uv run autolysis.py dataset.csv runs without errors using the instructor's AIPROXY_TOKEN environment variable
    - If uv run autolysis.py goodreads.csv runs without errors and creates README.md and *.png
    - If uv run autolysis.py happiness.csv runs without errors and creates README.md and *.png
    - If uv run autolysis.py media.csv runs without errors and creates README.md and *.png
    - If all the above run without errors and create the correct files (bonus)

Your code is passed to an LLM. Your submission is accepted if:

- If code is well structured, logically organized, with appropriate use of functions, clear separation of concerns, consistent coding style, meaningful variable names, proper indentation, and sufficient commenting for understandability.
- If code uses robust analytical techniques. Statistical methods, comprehensiveness of analysis, innovative analytical techniques, and dynamic analysis based on data exploration.
- If code uses appropriate visualization types, enhances charts with titles, axis labels, legends, and annotations, and uses colors effectively.
- If code crafts clear, context-rich prompts to guide the LLM on the narrative, includes relevant results, ensures proper Markdown formatting, logically sequences the narratives (data description, analysis, insights, implications), integrates visualizations at the right places, and prompts the LLM to emphasie significant findings and implications.

- If code makes uses LLMs efficiently, minimizing token usage by avoiding sending large data and using concise prompts.
- If code uses dynamic prompts and function calling
- If code uses vision capabilities and multiple calls to LLMs (agentic workflows).

We don't know the exact prompts we will use for this. We won't share these either. But this gives you an idea of what we're looking for.

Output

We will run your code against the output from 3 datasets:

1. The first will be the output from *one* of the datasets you submitted. We will not share which one, and it may vary for each person.
2. Dataset 2: We won't be sharing this dataset. Your code should work with any CSV file.
3. Dataset 3: Same as above

The LLM will review your README.md and *.png files on these criteria:

- If README.md is well-structured, using headers, lists, and emphasis appropriately. The narrative clearly describes the data, analysis performed, insights gained, and implications.
- If the analysis demonstrates a deep understanding of the data, utilizing appropriate statistical methods and uncovering meaningful insights.
- If the PNG images are relevant, well-designed, and enhance the narrative by effectively illustrating key findings.

REMEMBER: LLMs give different results each time. Just because it worked for you doesn't mean it'll work when we evaluate it. So be robust in your prompts and code.

Criteria for Letter of Recommendation:

- **Code diversity**. You're welcome to copy code and learn from each other. But we encourage diversity too. We will use code embedding similarity (via text-embedding-3-small, dropping comments and docstrings) and give bonus marks for most unique responses. (That is, if your response is similar to a lot of others, you lose these marks.)
- **Engaging and interesting**. We'll read your output. If it tugs at our hearts or blows our minds, we'll give bonus marks to a few lucky students.

Prompt injection is fine. LLMs can be coerced via prompt injection. You're welcome to try it. We'll try hard with our prompts to reduce this possibility. But if you succeed, you deserve an LOR.

Deadline

The deadline for the project is 31st January, 2025.

FAQ

- **Can I use my own OpenAI API key?** Yes
- **Where should I store the AI Proxy token?** In the AI_PROXY environment variable. Don't commit it.
- **Won't passing images to OpenAI cost too much?** Use "detail": "low" to reduce it.
- **What files should I commit to my GitHub repo?**
    - autolysis.py
    - goodreads/README.md
    - goodreads/*.png
    - happiness/README.md
    - happiness/*.png
    - media/README.md
    - media/*.png
- **How should I run my autolysis.py?** uv run autolysis.py /path/to/input.csv
- **Where should I save the README.md and PNG files?** In the current working directory.
- **LLMs are error prone. Should my code retry on failure?** Yes. E.g. use a library like tenacity.
- **Which datasets will you evaluate?** The project mentions "*one* of the datasets you submitted". We'll run uv run autolysis.py on either goodreads.csv, happiness.csv, or media.csv, randomly, and evaluate it. We will also evaluate it on 2 secret datasets.
- **I get a CLONE FAILED ... returned non-zero exit status 128.?** Your GitHub repo URL is wrong or your repo is private.
- **I get a HTTP 429 error.** OpenAI is tired. Retry.

Reading Material

uv

uv is a fast Python package manager that's becoming the standard for running Python scripts.

Install uv. Set the AIPROXY_TOKEN environment variable. Then run uv run autolysis.py dataset.csv instead of python autolysis.py dataset.csv. This automatically installs Python and your dependencies and runs the script.

Dependencies are declared using inline script metadata. Make sure all dependencies are mentioned at the top of your script. For example:

```
# /// script
# requires-python = ">=3.11"
# dependencies = [
#   "httpx",
#   "pandas",
# ]
# ///
```

Video for reference: https://youtu.be/igWlYl3asKw?t=1240

OpenAI Function Calling

OpenAI supports [Function Calling]() -- a way for LLMs to suggest what functions to call and how.

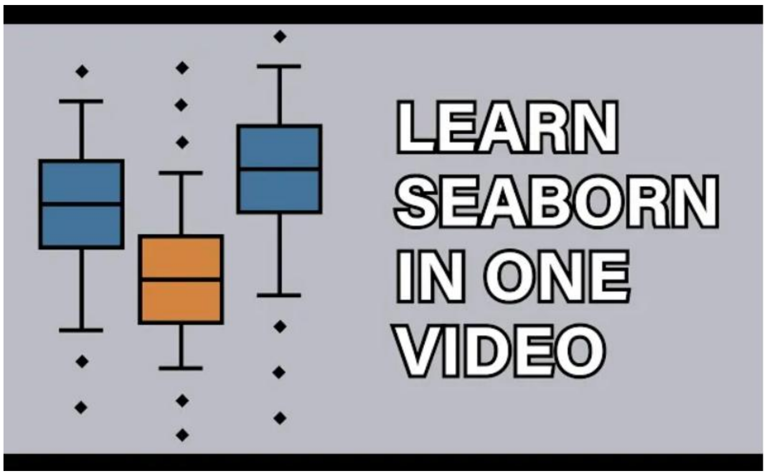
FUNCTIONS EXPLAINED
openai_function_calling.py — langchain-experiments (Workspace)
openai_function_calling.py
openai-functions > openai_function_calling.py
Spaces: 4 UTF-8 LF Python 3.10.11 ('ai-experiments': conda)

```
ion_descriptions_multiple = [
        "name": "get_flight_info",
        "description": "Get flight information between t          ions",
        "parameters": {
            "type": "object",
            "properties": {
                "loc_origin": {
                    "type": "string"
                    "description":                    airport, e.g. DUS",
                },
                "loc_destination": {
                    "type": "string",
                    "description": "The destination airport, e.g. HAM",
                },
            },
            "required": ["loc_origin", "loc_destination"],
        },
    },
    {
```

Video for reference: https://www.youtube.com/watch?v=aqdWSYWC_LI

Data visualization with Seaborn

[Seaborn]() is a data visualization library for Python. It's based on Matplotlib but a bit easier to use, and a bit prettier.



Video for reference: https://www.youtube.com/watch?v=6GUZXDef2U0