Supplementary Files for Preliminary Analysis of Datasets using GPT and Wolfram Language

This main folder contains all the necessary files for the preliminary analysis of datasets using GPT and Wolfram Language. Below are the details of each folder and file

Note:

- You need to have the appropriate API key from OpenAI (paid) and HuggingFace (free) to run any GPT models present.
- You also need to install the LLMs from Ollama and HuggingFace. Getting models from Ollama and installing them on your system is necessary to run all the files, including Python files or the Streamlit web app.

Folders and Files:

- **Research folder:** This folder contains all the research conducted throughout the project development process. It includes files with and without errors, representing various attempts and experiments made to achieve the project's goals. The research folder is essential for tracking progress, troubleshooting issues, and documenting the evolution of ideas.
  - **Dockerfile:** The Dockerfile included in the research folder is crucial for creating a Docker image that encapsulates all the required dependencies, libraries, tools, and repositories necessary for running the project. Using Docker ensures consistency and reproducibility across different environments and simplifies the setup process for deploying the application.
- **Streamlit application folder:** This folder contains all the files necessary to run the developed Streamlit Web Application. It includes Python scripts, HTML templates, CSS files, and any other resources required for the application's frontend and backend functionalities.
- **Dataset Folder:** This folder contains most of the datasets used in the project. It serves as a repository for the data required for conducting analysis, training models, and testing algorithms. The datasets cover a wide range of domains and provide valuable resources for experimentation and validation of the project's methodologies

- **Codegemma_x_LangChain.ipynb:** This Jupyter Notebook contains the appropriate code snippets for running the 'codegemma' model, which is based on Ollama. It provides instructions and examples for utilizing the 'codegemma' model within the LangChain framework for data analysis and processing tasks.
- **Llama2_x_LangChain.ipynb:** Similar to the previous notebook, this file contains code snippets specific to the 'llama2' model, another Ollama-based language model. It demonstrates how to utilize the 'llama2' model within the LangChain framework for various natural language processing tasks.
- **Llama3_x_LangChain.ipynb:** This notebook focuses on the 'llama3' model, showcasing its capabilities and usage within the LangChain framework. It provides examples and guidelines for leveraging the 'llama3' model for text analysis and generation tasks.
- **LLaVa_x_LangChain.ipynb:** This notebook contains code snippets for working with the 'llava' model, an Ollama-based language model. It demonstrates how to integrate the 'llava' model into the LangChain framework and utilize its features for text processing and analysis purposes.
- **Mistral_x_LangChain.ipynb:** Similar to the previous notebooks, this file provides instructions and examples for utilizing the 'mistral' model within the LangChain framework. It explores the capabilities of the 'mistral' model and demonstrates its usage for natural language processing tasks.
- **Mixtral_x_LangChain.ipynb:** This notebook focuses on the 'mixtral' model, showcasing its functionalities and usage within the LangChain framework. It provides code snippets and examples for leveraging the 'mixtral' model for various text-related tasks.
- **OpenAI_GPT_x_LangChain.ipynb:** This notebook contains code snippets for running the 'gpt-4' model, which is based on OpenAI's GPT architecture. It demonstrates how to use the 'gpt-4' model within the LangChain framework for natural language processing tasks.
- **Snip.ipynb:** This notebook contains the latest snippets based on the project, including examples of code snippets, functions, or techniques used in the development process. It may include optimizations, GPU utilization techniques, or other relevant information for enhancing the project's performance.

These folders and files collectively contribute to the development, experimentation, and documentation of the preliminary analysis of datasets using GPT. They serve as valuable resources for understanding, implementing, and extending the project's functionalities.

Installing Local Models:

To install and use the local models, follow these steps:

- Install Unix/Linux on Windows (if using). Alternatively, you can directly run ollama setup in Linux and Mac.
- Download Ollama setup for Linux.
- Open CMD prompt and run the command '**ollama pull**' to pull a model from the repository.
- Once the model is successfully installed, run '**ollama serve'** to start its server.
- To run, simply open any of the Ollama-based Jupyter Notebook files and execute the code. It will automatically start generating results.