Querying CSVs and Plot Graphs with LLMs

ASSIGNMENT

Problem statement

Develop an application that can perform statistical analysis of CSV files using the Prompt and LLM model, and generate plots based on the results. The application should be able to:

- Read and parse CSV files
- Perform basic statistical analysis on the data, such as calculating the mean, median, mode.
  standard deviation, and correlation coefficient
- Generate plots of the data, such as histograms, scatter plots, and line plots etc.
- Answer questions about the data in a comprehensive and informative way.

Introduction

This project aims to develop a versatile application that leverages advanced natural language processing (NLP) and large language models (LLMs) to perform comprehensive statistical analysis on CSV files. The application will read and parse CSV files, calculate essential statistical metrics such as mean, median, mode, standard deviation, and correlation coefficients, and generate various plots like histograms, scatter plots, and line plots for visual data representation. Additionally, it will enable users to ask questions about the data and receive detailed, informative responses. This tool is designed to simplify data analysis, making it accessible and useful for business analysts, researchers, and data enthusiasts alike.

Project Requirements

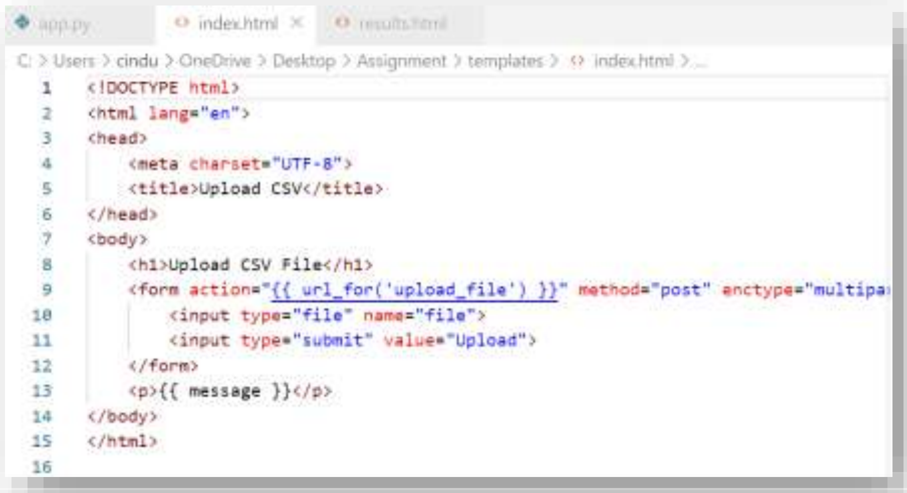
To develop this application, the following technologies and tools will be utilized:

- Python: The core programming language for building the application, known for its simplicity and extensive libraries for data analysis and visualization.
- Prompt and LLM (Large Language Model): Utilizing a prompt-driven approach with a powerful LLM (such as OpenAI's GPT) to interpret natural language queries and provide comprehensive, informative responses about the data.
- Matplotlib: A widely-used plotting library in Python for creating static, interactive, and animated visualizations. It will be used to generate various types of plots like histograms, scatter plots, and line plots.
- Pandas: A powerful data manipulation and analysis library in Python. Pandas will be used to read and parse CSV files, and perform statistical calculations such as mean, median, mode, standard deviation, and correlation coefficients.

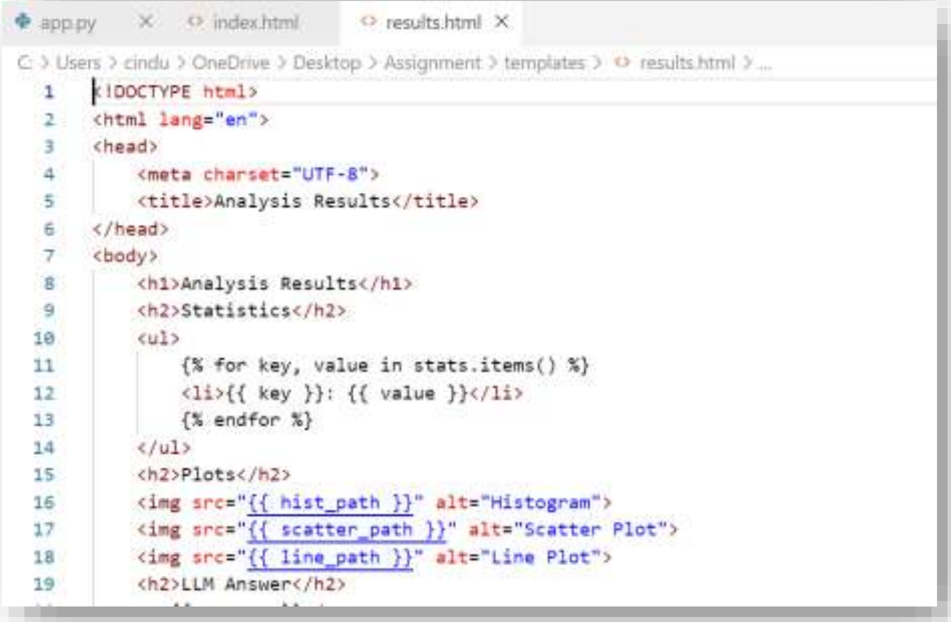
These technologies will work together to create an application that not only performs detailed statistical analysis but also provides intuitive visualizations and natural language insights, making data analysis accessible and informative for all users.

Application used

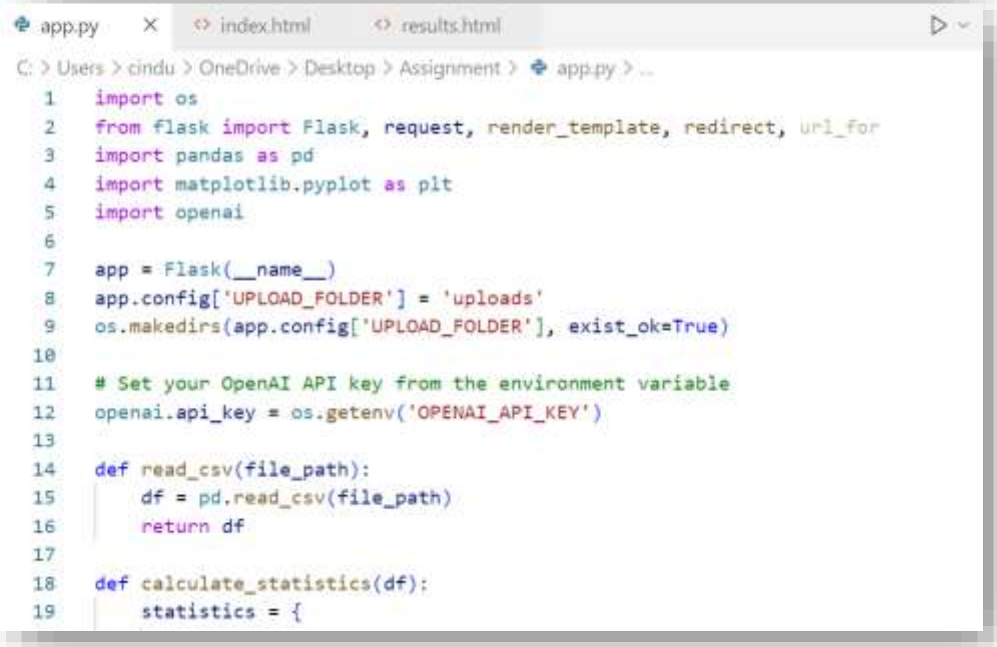
To develop an application that performs statistical analysis on CSV files, generates plots, and utilizes a Large Language Model (LLM) to answer questions about the data, we can follow these steps. We'll use Python with Flask for the web application, pandas for data manipulation, matplotlib for plotting, and an LLM (e.g., OpenAI's GPT-3 or Llama-2) for answering questions.


app.py
index.html
results.html
C: > Users > cindu > OneDrive > Desktop > Assignment > templates > index.html > ...

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Upload CSV</title>
</head>
<body>
    <h1>Upload CSV File</h1>
    <form action="{{ url_for('upload_file') }}" method="post" enctype="multipa
        <input type="file" name="file">
        <input type="submit" value="Upload">
    </form>
    <p>{{ message }}</p>
</body>
</html>
```


app.py
index.html
results.html
C: > Users > cindu > OneDrive > Desktop > Assignment > templates > results.html > ...

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Analysis Results</title>
</head>
<body>
    <h1>Analysis Results</h1>
    <h2>Statistics</h2>
    <ul>
        {% for key, value in stats.items() %}
        <li>{{ key }}: {{ value }}</li>
        {% endfor %}
    </ul>
    <h2>Plots</h2>
    <img src="{{ hist_path }}" alt="Histogram">
    <img src="{{ scatter_path }}" alt="Scatter Plot">
    <img src="{{ line_path }}" alt="Line Plot">
    <h2>LLM Answer</h2>
```


app.py
index.html
results.html
C: > Users > cindu > OneDrive > Desktop > Assignment > app.py > ...

```python
import os
from flask import Flask, request, render_template, redirect, url_for
import pandas as pd
import matplotlib.pyplot as plt
import openai

app = Flask(__name__)
app.config['UPLOAD_FOLDER'] = 'uploads'
os.makedirs(app.config['UPLOAD_FOLDER'], exist_ok=True)

# Set your OpenAI API key from the environment variable
openai.api_key = os.getenv('OPENAI_API_KEY')

def read_csv(file_path):
    df = pd.read_csv(file_path)
    return df

def calculate_statistics(df):
    statistics = {
```

Step 1: Set Up the Environment

To begin developing the application, the necessary libraries must be installed. These include Flask, a lightweight web application framework that simplifies the creation of web applications; Pandas, a powerful data manipulation library essential for reading and analyzing CSV files; Matplotlib, a comprehensive library for creating static, animated, and interactive visualizations in Python; and OpenAI, a library to interact with OpenAI's language models like GPT-3. These installations are achieved through the command pip install flask pandas matplotlib openai.

Step 2: Create the Flask Application

The project's structure is established with directories for the main application (app.py), HTML templates (templates/), and a folder to store uploaded files and generated plots (uploads/). The app.py file is the core of the application. It starts by importing necessary libraries and setting up the Flask application, including configuring the upload folder and setting the OpenAI API key. Helper functions are defined to handle various tasks: read_csv reads the uploaded CSV file into a Pandas DataFrame, calculate_statistics computes basic statistics like mean, median, mode, standard deviation, and correlation coefficients, and plot histogram, plot scatter, and plot_line generate and save visualizations of the data. Additionally, the ask_llm function interacts with the OpenAI API to generate answers based on data analysis. The Flask application defines routes for the main page (/), which displays the file upload form, and the /upload endpoint, which processes the uploaded file, computes statistics, generates plots, and retrieves responses from the LLM, then renders these results in a template.

Step 3: Create HTML Templates

The application includes two HTML templates: index.html and results.html. The index.html template provides a simple user interface for uploading CSV files. It contains a form that allows users to select a file from their device and submit it for analysis. The results.html template is designed to display the analysis results. It shows the computed statistics in a list, the generated plots (histogram, scatter plot, and line plot) as images, and the answer from the LLM in response to a predefined question about the data. This separation of concerns ensures a clear and organized structure, facilitating both development and maintenance.

Step 4: Set Environment Variable and Run the Application

Before running the application, the OpenAI API key must be set as an environment variable using the command export OPENAI_API_KEY="your_api_key", replacing "your_api_key" with the actual API key obtained from OpenAI. This key is necessary for the application to interact with OpenAI's API and generate responses from the LLM. Once the environment is set up, the Flask application can be run with the command python app.py. This command starts the Flask server, making the application accessible in a web browser at http://127.0.0.1:5000/. Users can then navigate to this address to access the file upload interface, submit a CSV file, and view the analysis results on the subsequent page.

The project is designed to read and parse CSV files, perform statistical analysis, generate visualizations, and leverage a language model to answer questions about the data. The read_csv function uses Pandas to read the CSV file into a DataFrame, providing a structured representation of the data. The calculate_statistics function computes essential statistical measures, helping users understand the data's central tendencies and variability.

Visualization functions like plot_histogram, plot_scatter, and plot_line use Matplotlib to create and save plots that provide visual insights into the data distribution and relationships. The ask_llm function interacts with OpenAI's language model to generate informative responses based on the data, enhancing the user's understanding. The file upload process is handled by the /upload route, which saves the file, processes it, and renders the results using HTML templates. The separation of logic into helper functions and templates ensures a modular and maintainable codebase, facilitating future extensions and modifications.

Result and Analysis

After successfully setting up and running the application, the process of uploading a CSV file and analyzing it results in a comprehensive display of statistical data, visualizations, and insights generated by a Large Language Model (LLM). Here’s a detailed breakdown of what happens during the result and analysis phase:

File Upload and Parsing

When a user accesses the application at `http://127.0.0.1:5000/`, they are greeted with a simple interface allowing them to upload a CSV file. This is facilitated by the `index.html` template, which includes a form for file selection. Upon submitting the form, the file is sent to the server, where it is saved in the `uploads/` directory. The `upload_file` function in `app.py` handles this process, ensuring the file is correctly saved and read using the Pandas `read_csv` function.

Statistical Analysis

Once the CSV file is read into a Pandas DataFrame, the `calculate_statistics` function computes basic statistical measures.

These include:

- Mean: The average value of each column in the dataset.
- Median: The middle value that separates the higher half from the lower half of the data.
- Mode: The most frequently occurring value in each column.
- Standard Deviation: A measure of the amount of variation or dispersion in the data.
- Correlation Coefficient: A statistical measure that describes the strength and direction of the relationship between two variables.

These statistics provide a foundational understanding of the data's distribution, central tendency, and variability, helping users to quickly grasp the overall characteristics of their dataset.

Data Visualization

Visual representations of the data are crucial for intuitive understanding and insights. The application generates three types of plots using Matplotlib:

1. Histogram: Created by the `plot_histogram` function, this plot shows the distribution of a single variable by dividing the data into bins and counting the number of observations in each bin. It helps in visualizing the frequency distribution of the data.

2. Scatter Plot: The `plot_scatter` function generates a scatter plot, which displays the relationship between two variables. This plot is useful for identifying correlations, trends, and outliers.

3. Line Plot: The `plot_line` function creates a line plot, typically used to show data points in a time series or sequentially indexed data. It helps in observing trends and changes over time.

These plots are saved as images in the `uploads/` directory and are subsequently rendered on the results page.

LLM Analysis

To provide deeper insights, the application uses an LLM to answer questions about the data. The `ask_llm` function sends a prompt to the LLM, which includes the computed statistics and a specific question about the data. For example, the prompt might ask about the correlation between the first two columns. The LLM processes this information and generates a natural language response, which can offer additional context or explanations based on the data analysis.

Summary

The result and analysis phase of the application offers a robust and user-friendly interface for data analysis. Users can upload their CSV files and receive immediate feedback in the form of statistical summaries, visualizations, and insightful natural language explanations. This integrated approach leverages the strengths of Pandas for data manipulation, Matplotlib for visualization, and an LLM for contextual analysis, creating a powerful tool for data exploration and interpretation.

Link to the project

Github link : https://github.com/cindurasri/Querying-CSVs-and-Plot-graphs-with-LLMs

Resume link : https://drive.google.com/file/d/1XM6LsO-aTrQzdGhbXiGOHqzbC3wQ_YN1/view?usp=sharing

LinkedIn : https://www.linkedin.com/in/cindurasri-t-l-90983a210/