Report based on the project pipeline

Different components used in the pipeline-

1. ASR - ASR stands for Automatic Speech Recognition. It's a technology that enables computers to understand and transcribe spoken language into text. ASR systems use algorithms to analyze audio input and convert it into written words. These systems are widely used in various applications such as virtual assistants, voice-activated devices, dictation software, and customer service automation.

2. NMT - NMT stands for Neural Machine Translation. It's a type of machine translation that uses artificial neural networks to translate text from one language to another. NMT systems work by processing entire sentences or paragraphs as a whole, rather than breaking them down into smaller parts like older statistical machine translation systems. This allows NMT systems to produce more fluent and contextually accurate translations, particularly for languages with complex syntax or semantics. NMT has significantly advanced the quality of machine

translation and is widely used in various translation applications and services.

3. LangChain is a framework for developing applications powered by large language models (LLMs).

LangChain simplifies every stage of the LLM application lifecycle:

Development: Build your applications using LangChain's open-source building blocks and components. Hit the ground running using third-party integrations and Templates.
Productionisation: Use LangSmith to inspect, monitor and evaluate your chains, so that you can continuously optimize and deploy with confidence.
Deployment: Turn any chain into an API with LangServe.

4. Autogen - AutoGen is a framework that enables the development of LLM applications using multiple agents that can converse with each other to solve tasks. AutoGen agents are customizable, conversable,

and seamlessly allow human participation. They can operate in various modes that employ combinations of LLMs, human inputs, and tools.

- ◆ AutoGen enables building next-gen LLM applications based on multi-agent conversations with minimal effort. It simplifies the orchestration, automation, and optimization of a complex LLM workflow. It maximizes the performance of LLM models and overcomes their weaknesses.
- ◆ It supports diverse conversation patterns for complex workflows. With customizable and conversable agents, developers can use AutoGen to build a wide range of conversation patterns concerning conversation autonomy, the number of agents, and agent conversation topology.
- ◆ It provides a collection of working systems with different complexities. These systems span a wide range of applications from various domains and complexities. This demonstrates how AutoGen can easily support diverse conversation patterns.

◆ AutoGen provides enhanced LLM inference. It offers utilities like API unification and caching, and advanced usage patterns, such as error handling, multi-config inference, context programming, etc.

Web scraping and beautiful soup -

Web scraping is the process of extracting data from websites. It involves fetching HTML content from web pages and then parsing and extracting relevant information. Beautiful Soup is a Python library commonly used for web scraping tasks. It provides convenient methods for navigating and manipulating HTML and XML documents.

Here's a breakdown of how web scraping and Beautiful Soup work:

1. Fetching HTML Content: The first step in web scraping is to fetch the HTML content of the web page you want to scrape. This can be done using Python libraries like

`requests`, which make HTTP requests to the web server and retrieve the HTML response.

```python
import requests

url = 'https://example.com'
response = requests.get(url)
html_content = response.text
```

2. Parsing HTML with Beautiful Soup: Once you have the HTML content, you can parse it using Beautiful Soup. This library creates a parse tree from the HTML and provides methods for navigating, searching, and modifying the parse tree.

```python
from bs4 import BeautifulSoup

soup = BeautifulSoup(html_content, 'html.parser')
```

3. Navigating the Parse Tree: Beautiful Soup allows you to navigate the HTML document's structure by accessing tags, attributes, and text content.

   ```python
   # Accessing tags
   title_tag = soup.title

   # Accessing attributes
   link_tag = soup.a
   href_attr = link_tag['href']

   # Accessing text content
   paragraph_text = soup.p.text
   ```

4. Searching for Elements: You can search for specific elements in the HTML document using methods like `find()` and `find_all()`.

   ```python
   # Find a specific tag
   first_heading = soup.find('h1')
   ```

```python
   # Find all occurrences of a tag
   all_paragraphs = soup.find_all('p')

   # Find elements with specific class or id
   element_with_class = soup.find(class_='classname')
   element_with_id = soup.find(id='element-id')
```

5. Extracting Data: Once you've located the desired elements, you can extract their text content, attributes, or other information.

```python
   # Extracting text content
   paragraph_text = first_heading.text

   # Extracting attribute values
   href_value = link_tag['href']
```

6. Iterating Over Elements: You can iterate over lists of elements obtained from search results to extract data from multiple occurrences.

```python
for paragraph in all_paragraphs:
    print(paragraph.text)
```

7. Handling Errors: Web scraping may encounter errors like missing elements or invalid HTML. Beautiful Soup provides mechanisms to handle such errors gracefully.

```python
if first_heading:
    print(first_heading.text)
else:
    print("Heading not found.")
```

8. Advanced Usage: Beautiful Soup supports advanced features like prettifying HTML, navigating through the parse tree using CSS selectors, and working with XML documents.

Beautiful Soup simplifies the process of web scraping by providing a high-level interface for working with HTML documents. However, it's essential to review the website's terms of service and ensure compliance with legal and ethical guidelines when scraping data from websites.

Data Visualization

Data visualization is the graphical representation of data to provide insights and facilitate understanding. It involves presenting data in a visual format such as charts, graphs, and maps, making it easier to identify patterns, trends, and relationships within the data.

-Importance in Data Analysis:

- Simplifies complex data: Visual representations make it easier for analysts to comprehend complex datasets by presenting information in a more intuitive format.
- Facilitates decision-making: Visualizations help stakeholders quickly grasp key insights, enabling informed decision-making based on data-driven evidence.
- Enhances communication: Visualizations are effective tools for conveying information to a broad

audience, including non-technical stakeholders, promoting better understanding and engagement.

- Types of Visualizations:

- Charts: Bar charts, line charts, pie charts, and scatter plots are commonly used to represent numerical data.
- Graphs: Network graphs, tree maps, and Sankey diagrams visualize relationships and hierarchies within data.
- Maps: Geographic information systems (GIS) and choropleth maps display spatial data, facilitating analysis of location-based trends.
- Dashboards: Interactive dashboards provide a comprehensive overview of multiple datasets, allowing users to explore data dynamically.

- Tools for Data Visualization:

- Programming libraries: Python libraries like Matplotlib, Seaborn, and Plotly offer extensive capabilities for creating custom visualizations.
- Business intelligence (BI) tools: Platforms like Tableau, Power BI, and Qlik provide user-friendly interfaces for building interactive dashboards and reports.

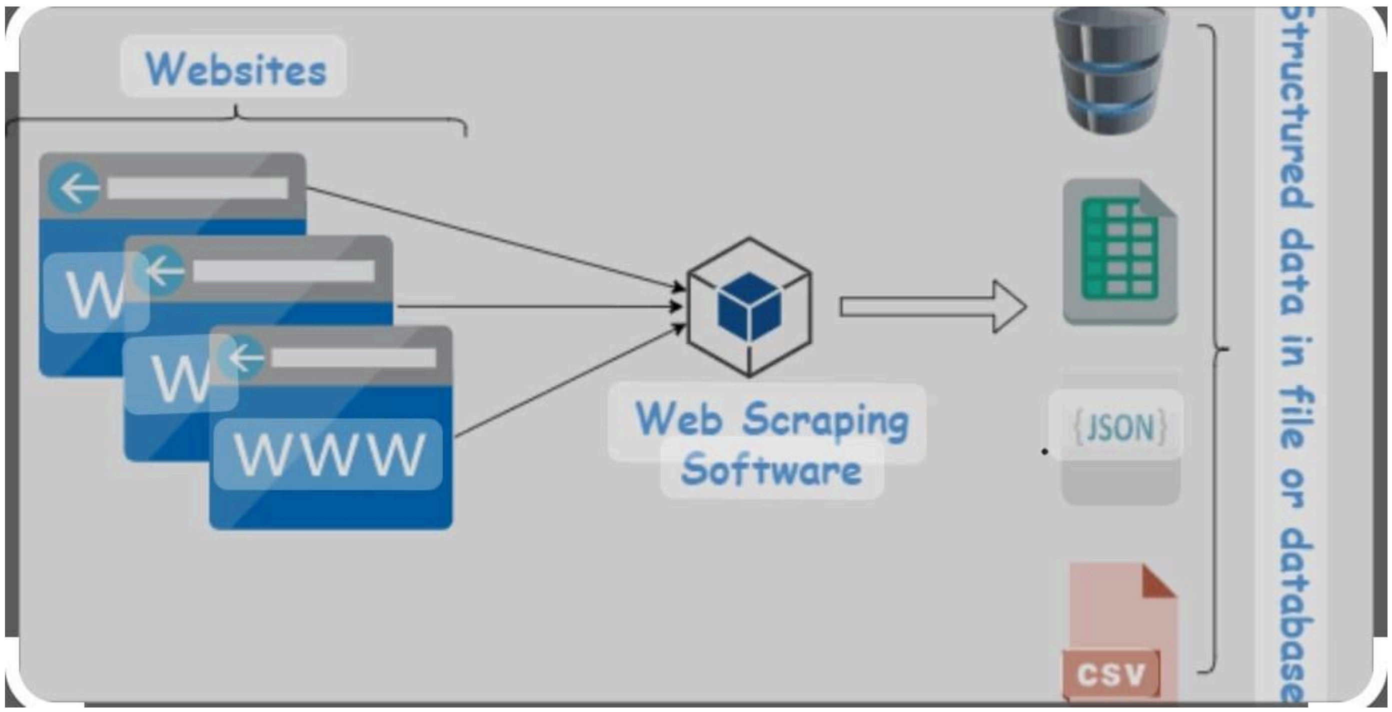
- Web-based tools: Online tools such as Google Data Studio and Chart.js enable users to create and share visualizations without programming skills.

- Importance in the Health Tech Industry:

- Decision support: Visualizations aid healthcare professionals in interpreting medical data, facilitating diagnosis and treatment decisions.
- Patient engagement: Interactive visualizations empower patients to understand their health metrics, encouraging proactive management of chronic conditions.
- Public health monitoring: Visualizations enable epidemiologists and policymakers to track disease outbreaks, monitor population health trends, and allocate resources effectively.
- Research insights: Visualizations assist researchers in analyzing clinical trial data, identifying correlations, and discovering new insights into disease mechanisms and treatments.
- Healthcare system optimization: Visualizations help healthcare administrators optimize resource allocation, improve operational efficiency, and

enhance patient outcomes through data-driven insights.

Data visualization plays a crucial role in transforming raw data into actionable insights, driving innovation and improvement across various sectors, including healthcare and technology. Its ability to communicate complex information effectively makes it an indispensable tool for decision-makers and analysts alike.



Tech-Stack:

- AI Agent Framework: Autogen by Microsoft
- Web Scraping: Beautiful Soup
- Data Visualization: Python scripting (e.g., Matplotlib, Seaborn)
- Language Model Integration: OpenAI API
- Automation: End-to-end workflow automation

Actual Workflow:
So taking all this into use,I have written the code which deals with the integration of Gen AI , LLM and web scraping. I have made an AI Agent using Autogen by Microsoft. The agent works to get data pertaining to the medical domain using web scraping (majorly beautiful soup) . This data after acquisition is fed into a python script to get the data visualisation and plots for analysis. LLMs have been utilized to their full potential in this code. I have used the OpenAI API key to integrate the LLM. The entire workflow can be automated using this code.

Notebook:

Tech_Assignment3.ipynb

References:

- https://github.com/AI4Bharat/IndicTrans2
- https://github.com/microsoft/autogen
- https://github.com/openai/openai-python