IRisk Lab Data Discovery and Consolidation
Week 8 Report
Oct. 17
Rohit Valmeekam

<u>What Was Accomplished</u>

- ❖ Advanced Web Scraping Concepts:
  - ➢ Dynamic Web Pages:
    - ■ Expanding beyond static web pages, I explored techniques for scraping dynamic content. Dynamic pages use JavaScript to load data dynamically, making traditional scraping methods ineffective. I delved into the use of tools like Selenium, which allows for automated interactions with web pages, enabling the retrieval of dynamically loaded content.
    - ■ Handling AJAX Requests:
      - ● AJAX requests play a crucial role in modern web applications. To scrape data from pages making asynchronous requests, I learned to intercept and handle these requests using libraries like Scrapy and Puppeteer. This ensures a comprehensive extraction of data, including content loaded after the initial page load.
  - ➢ Dealing with Login and Authentication:
    - ■ Many websites require user authentication to access certain data. I covered methods for handling login forms programmatically, enabling access of authenticated areas of a website. This is particularly relevant when scraping insurance data from portals or dashboards.
  - ➢ Anti-Scraping Measures:
    - ■ As web scraping becomes more prevalent, websites implement anti-scraping measures to protect their data. I explored techniques to bypass these measures, such as rotating IP addresses, using user-agents, and introducing delays in our scraping scripts.
  - ➢ Coding Techniques for Web Scraping
    - ■ XPath and CSS Selectors:
      - ● While Beautiful Soup provides a convenient way to navigate HTML documents, understanding XPath and CSS selectors allows for more fine-grained control over element selection. This is particularly useful when dealing with complex HTML structures.
- ❖ Integration with Flask and APIs
  - ➢ Flask Framework:
    - ■ I explored the integration of web scraping into Flask.

        - Flask provides an ideal environment for creating web applications and APIs, making it a suitable platform for incorporating scraped data into practical applications.
    - Creating API Endpoints:
        - Leveraging Flask, I learned to create API endpoints that serve as interfaces for accessing scraped data. This facilitates seamless communication between the web scraping scripts and external applications or services.
    - Real-time Data Updates:
        - By combining web scraping with Flask APIs, I can establish real-time data updates. This is particularly valuable for insurance-related applications, ensuring that users have access to the latest information on premiums, policies, and other relevant data.

Next Steps

- Start the creation of an API that can automate web-scraping for insurance purposes
- Continue building out the data input structure