Final write-up

Our team develops a web application that will serve as a centralized location for users to access and manage their healthcare data. This application will be designed with a user-friendly interface that simplifies the process of accessing and navigating through insurance claims and EHR records.

Through our web application, users will be able to easily view their insurance claims and the status of their claims. They can also access and view their EHR records, including medical histories and other health-related data. Our aim is to provide users with a comprehensive and organized view of their healthcare information that they can access from anywhere and at any time. Our application is built using the Python Flask framework. We chose it because it offered us the flexibility, we needed to develop a customized web application that met the specific requirements of our project. Besides, we developed our frontend using HTML, CSS, and JavaScript. To handle data storage, we are using SQLite as our database management system. To ensure seamless communication between the frontend and backend, we have developed several APIs that support data extraction, storage, and updating. These APIs serve as a connection point between the frontend and backend of our application, enabling users to access and manage their healthcare information easily.

We have some exciting new features to announce that will enhance the user experience and functionality of our application.

First, we have added a Dropdown Search Box that allows users to search for specific items quickly and easily within our application. With this feature, users can easily find what they are looking for without having to scroll through long lists or pages.

Second, we have added a new feature that allows users to mark new matches and unmarked existing matches within our application. This feature is particularly useful for insurance providers who identify the mismatched EHR and insurance claims. Additionally, this new feature also provides users with an interface to adjust the result of the matching algorithm. It allows users to customize the way that matches are identified within the application, making it easier to fine-tune and optimize the matching process to their specific needs.

Third, we have added a Save New Checks feature that enables users to save new checks. This feature is particularly useful for users who regularly perform checks on specific items, as it allows them to save those checks quickly and safely in our database.

Finally, we have added a Mouseover Text Description feature that provides users with additional information and context about specific items within our application. With this feature, users can simply hover their mouse over an insurance code to access a brief description, making it easier for them to understand and use the information within our application.

Overall, these new features are designed to enhance the user experience and functionality of our application, making it easier and more convenient for users to access and manage their data. We are excited to introduce these new features and are confident that they will make a positive impact on our users' experience.

We have made some significant changes to our app. Firstly; we have transitioned from R Shiny to Python Flask. The decision to switch to Python Flask was based on several factors, including better flexibility, scalability, and customization options. We believe that Flask offers a more robust and efficient framework for our application, allowing us to better meet the needs of our users. In addition to switching to Python Flask, we have also converted our static data files to a dynamic SQLite database. This has enabled us to build more complex and interactive features within the application, providing users with greater control and flexibility in managing their data. To support data extraction, update, and storage, we have also built several APIs within our application. These APIs provide a standardized interface for accessing and manipulating data within the application, making it easier for users to work with and manage their data.
In terms of user experience, we have designed fully functional web pages to support various user actions. These pages are intuitive and user-friendly, making it easy for users to navigate the application and perform the actions they need to manage their data effectively.
Finally, we have worked to resolve any unknown errors within the application to ensure that it is functioning at its best. Our team has put in a great deal of effort to ensure that the application is stable, reliable, and provides an optimal user experience.

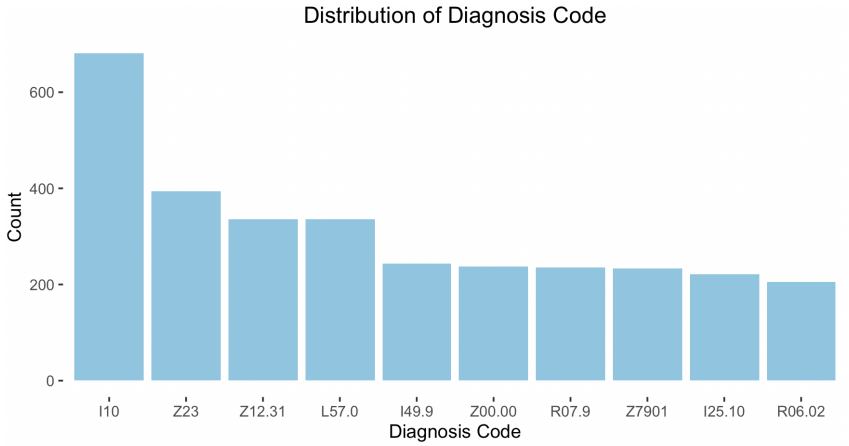
In summary, we have made significant changes to our application, including transitioning to Python Flask, converting static data files to a dynamic SQLite database, building several APIs, designing fully functional web pages, and resolving unknown errors. These changes are designed to enhance the user experience, improve functionality, and provide our users with greater control and flexibility in managing their data. Thank you for your attention, and we look forward to continuing to improve and enhance our application.

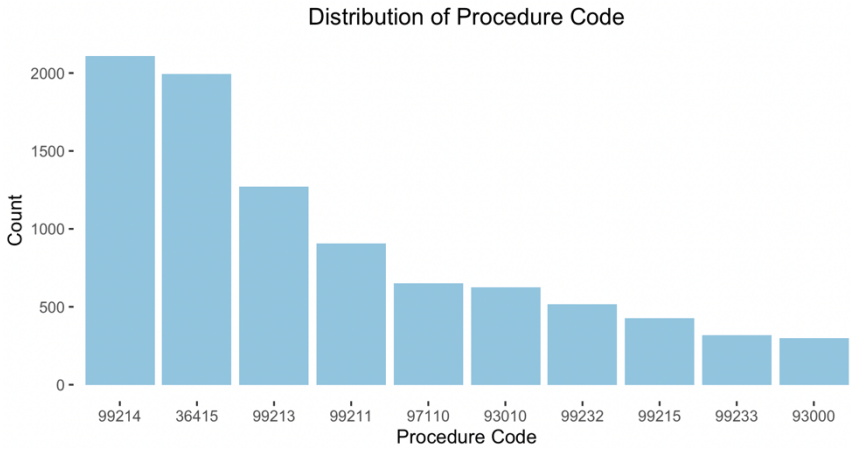
Data Analysis:

1. **EHR**

The EHR dataset contains information on various variables related to patients' health records. Stride ID consists of 202 unique IDs, used to uniquely identify each patient within the dataset. Service Date represents the date of the medical service, ranging from September 16, 2015, to March 31, 2019. Admit Date indicates the date when a patient was admitted to hospital, ranging from September 16, 2015, to March 30, 2019. Disch Date denotes the date when a patient was discharged from Stride's service, ranging from September 16, 2015, to March 31, 2019. Site ID represents site IDs, and in our case, all the patients are from Site 5. Diagnosis Code includes diagnosis codes such as I10 (Essential hypertension) and Z23 (Encounter for Immunization), among others, indicating patients' medical conditions. Procedure Code includes procedure codes such as 99214 and 36415, among others, which likely represent specific medical procedures or treatments provided by Stride.

The following plots show the top 10 diagnosis code and procedure code in the dataset.





2. Claims Data

Claims data contains information on various variables related to patients' fall injuries and medical visits. Stride ID also consists of 202 unique IDs. Follow-up Interview Month represents the month of follow-up interviews conducted, with values ranging from 4 to 40, indicating the timing of post-fall injury assessments. Index of Reported Fall Injury includes values 1, 2, and 3, which represent different levels or severity of reported fall injuries. Fall Injury Bone Break contains 21 different kinds of bone breaks, such as head/skull, face, neck, etc. Date of ER Visit represents the date when a fall-related injury resulted in a visit to the emergency room, ranging from December 2, 2015, to January 30, 2019. Date of doctor's office visit indicates the date when a fall-related injury resulted in a visit to a doctor's office, ranging from April 7, 2016, to February 26, 2019. Date of visit to other facility denotes the date when a fall-related injury resulted in a visit to another facility, such as a specialized clinic or hospital, ranging from December 3, 2015, to December 12, 2018. Admission Date for overnight hospitalization represents the date when a fall-related injury resulted in an overnight hospitalization, ranging from December 2, 2015, to January 9, 2019. The dataset also includes information about injury

types. The injuries listed include dislocated joint, head injury, cut with bleeding, sprain, or strain, bruising or swelling, and other injuries. The response options for each injury question are "Yes," "No," "Refused," or "DK" (don't know).

The following plot provided displays the distribution of the Index of Reported Fall Injury by interview month. It is evident from the plot that most patients experienced a relatively mild level of fall injury during the period.