Project Recap

Integrating EHR and claims data yields richer health data for administrators, researchers, and policymakers to assess. Clinical EHR data helps more accurately identify conditions. With comprehensive info about health before, during and after treatments, claims-linked EHR data boosts evidence-based insights. Our aims: 1) Improve UI for more efficient workflow & fix current errors; 2) Develop ML/DL algorithm for more accurate & faster matching of EHRs and insurance.

Milestones

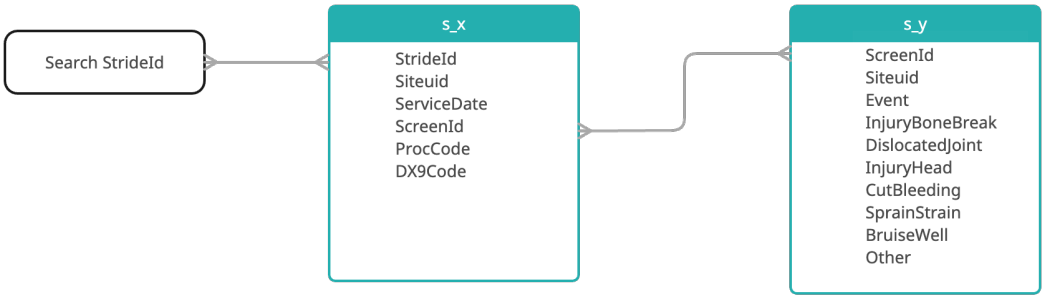
Current Status:

- Data has been collected and stored in a sqlite database.

The raw data was extracted from the Shiny app and used to create a new SQLite database. This database was designed to support information retrieval and update, enabling efficient management of the extracted data. Three tables were created to store the data: s_x, s_y, and ms. To better manage and analyze the insurance and EHR data, we created two separate tables: s_x for insurance data and s_y for EHR data.

The s_x table is designed to capture information related to insurance claims, such as the site where the claim was filed, the specific service provided, and the corresponding insurance codes. On the other hand, the s_y table is focused on recording EHR data related to specific screens and injuries, including the type of injury, the location where it occurred, and other relevant details.

To establish a relationship between the two tables, we had a common column, the screen ID, which is present in both s_x and s_y tables. Typically, when a user searches for a specific stride ID, we first extract the corresponding insurance data from the s_x table, using the stride ID to locate the relevant records. Then, we look up the s_y table to extract the corresponding EHR data for the insurance screen ID associated with the screen ID.



Besides, we have the third table, ms. The ms table is used to indicate whether each stride in s_x is matched with any records in s_y. This table contains columns such as s_x ID, s_y ID, and match status. The match status column is a binary variable indicating whether a match is found between s_x and s_y records.

Overall, we've created an SQLite database with 3 tables s_x, s_y, ms to store raw data from a Shiny app. The database enables info retrieval and update with an efficient solution for managing the data and retrieving useful info for analysis.

- Data analysis was performed to further understand the data
- Alpha version app has been implemented (html web development, APIs to search and update the database)
- A binary classification model has been built.

We reviewed literature and identified project improvements, then presented our research outline. After discussion and iterations, we migrated the app from R shiny to python flask. We're currently focusing on data cleaning, analysis, and web development. After separating the data into tables, we stored it in a database. We've analyzed the data to identify patterns and have developed an alpha version app with a search API that allows users to update the database. We're also working on a binary classification algorithm.

Adjustments:

- **Recreating the R Shiny App with Python Flask:**
  Due to time constraints and limitations with R Shiny, we have decided to recreate the app using Python Flask framework.

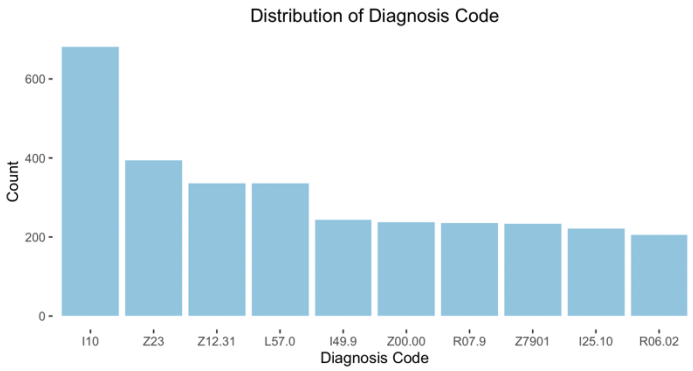
- **Matching algorithm optimization TBD:**
  We are currently working on migrating the app's functionality and manipulating the data, which has taken longer than anticipated. As a result, we have postponed the optimization of the matching algorithm with deep learning until a later date.

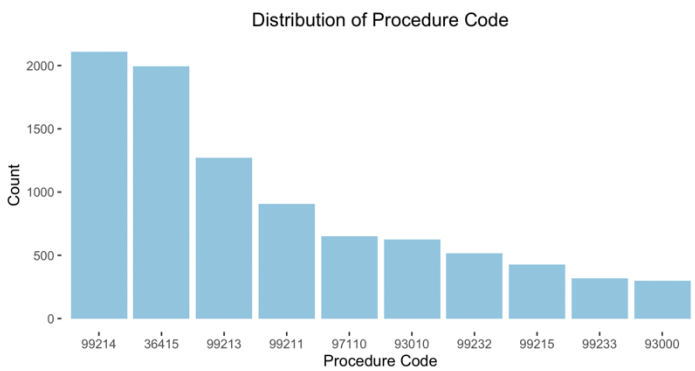
Current Work

1. Data Analysis:

- **EHR**

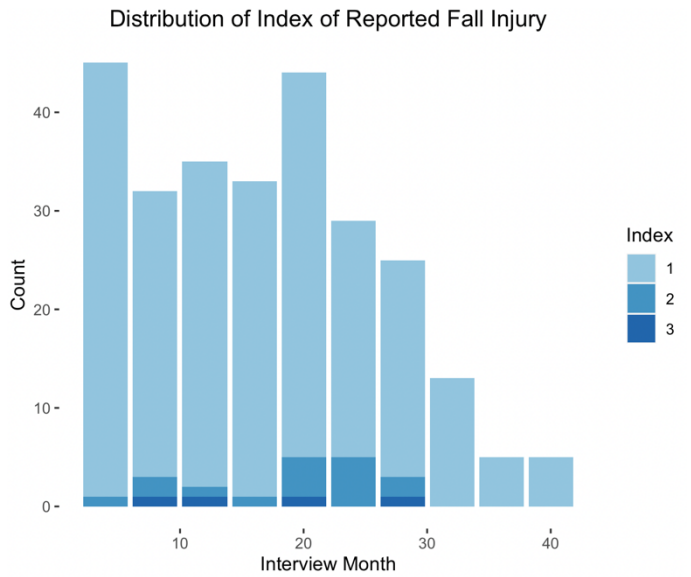
The EHR dataset contains information on various variables related to patients' health records. Stride ID consists of 202 unique IDs, used to uniquely identify each patient within the dataset. Service Date represents the date of the medical service, ranging from September 16, 2015, to March 31, 2019. 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.





- **Claims Data**
  Claims data includes information on patient fall injuries and medical visits, with unique Stride IDs and follow-up interview months ranging from 4 to 40. Reported fall injuries are indexed by severity level, and there are 21 different types of bone breaks recorded. Dates are also included for emergency room visits, doctor's office visits, visits to other facilities, and overnight hospitalizations due to fall-related injuries, with ranges from 2015 to 2019.

  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 time period.



2. Web app:

   The annotation interface in R Shiny was redeveloped into a web application using HTML, JavaScript, and Python Flask. The Bootstrap framework and Select2 plugin were used for the user interface. The web app comprises two tables, Table A and Table B, which are displayed on the webpage. The user can select a Match ID from a dropdown menu, and upon selection, an AJAX GET request is sent to the server to fetch data based on the chosen Match ID. The returned data is then used to populate Table A and Table B.

   Table A contains insurance claim data, while Table B contains information on the patient's medical history (EHRs). Before populating the tables with data, the column names of each table are displayed in the corresponding HTML table element. In Table B, the user can select entries as matches using checkboxes. Upon selecting entries, the selection results are updated in the database.