SPECFIC AIMS

The potential of real-world evidence from electronic health records (EHRs) and claims data to inform regulatory decision-making is currently being assessed (Sharma et al. [2020]). They are both important sources of electronic data that can be utilized for health services research and quality-of-care evaluations. Claims data have been utilized for research purposes for many years, but EHRs, due to their high level of detail, are gaining in popularity. While each data source has its own benefits, combining the two sources can create a synergy that is useful for research and policy applications. The ability to match EHR with insurance claims data has the potential to revolutionize the way patient care is analyzed and evaluated. EHR and claims data each provide only partial information about a patient's health journey, and integrating these sources of information provides a more comprehensive view of a patient's health status. The main purpose of claims data is to bill and pay for health services, while EHRs are designed for managing and recording patient care. However, the current process of matching EHR and claims data is time-consuming and prone to errors. Automating this process using machine learning and deep learning methods can improve the accuracy and efficiency of matching, reducing the time and effort required and improving healthcare overall (West et al. [2014]).

A study sponsored by the Agency for Healthcare Research and Quality (AHRQ) examined the process and outcome of combining administrative claims data and EHRs from a state Medicaid population and an academic medical center. The study group found that despite some challenges in merging and analyzing claims and clinical data, the combination of both sources of healthcare data produced a stronger analytic resource than either source alone, and should be further evaluated and improved.

Therefore, our project will focus on matching EHRs and claims data with the following specific aims:

Specific Aim 1: To develop a user-friendly web application using R Shiny that enables users to capture new checks, add notes, and navigate between matches. The upcoming version of the web application is set to undergo major enhancements, particularly in its front-end, which will incorporate new and improved features to optimize the workflow and provide a better user experience. The primary goal of these updates is to identify and resolve any existing issues in the current system while introducing novel features, such as next and previous match buttons, a check capture function, a notes text box, and an error resolution function. The addition of next and previous match buttons is intended to facilitate quicker navigation between matches in the application, thus improving the user's ability to move through the data with ease. The check capture function will enable users to effortlessly capture and store checks within the web application, streamlining the record-keeping process. The notes text box feature will provide users with a handy way to better understand the data columns, enabling them to perform more comprehensive analyses. Furthermore, an error resolution function will be introduced to help users quickly resolve any issues that may arise while using the app. To develop these new features, an experimental approach will be employed, involving the incorporation of new elements into the existing R Shiny application. Overall, these enhancements aim to optimize the application's functionality, streamline the workflow, and provide a more user-friendly experience.

Specific Aim 2: To develop a matching algorithm using machine learning and deep learning methods that can match EHRs with insurance records. By automating the matching process, the accuracy and efficiency of matching will improve, reducing the time and effort required and improving healthcare overall.To accomplish this aim, the first step will involve cleaning and preprocessing the data to ensure its quality and suitability for the algorithm. The data will then be split into train, validate, and test sets to enable effective training and testing of the algorithm. Traditional machine learning models and deep learning models will be developed and trained to match EHRs with insurance records. Performance evaluation will be conducted using various accuracy and ROC/AUC metrics to determine the most effective model. An experimental approach will be employed to improve the performance of the algorithm. This will involve fine-tuning the models to enhance their accuracy and efficiency, as well as testing them on larger datasets to determine scalability. The development of a matching algorithm using machine learning and deep learning methods is expected to significantly enhance the accuracy and efficiency of matching EHRs with insurance records, ultimately reducing the time and effort required for the process. This will enable healthcare providers to concentrate on delivering high-quality care to their patients, thereby improving healthcare outcomes by minimizing errors and streamlining the healthcare delivery process.

RESEARCH STRATEGY

A. Significance

The ability to match electronic health records (EHR) with insurance claims data is an innovative approach that has the potential to revolutionize the way patient care is analyzed and evaluated. This is because EHR and claims data each provide only partial information about a patient's health journey, and integrating these sources of information provides a more comprehensive view of a patient's health status.

EHR data provides robust information on demographics, diagnoses, encounters, laboratory results, procedures, and medication orders. However, it does not reflect activity outside of the care setting, such as prescription fills, and is incomplete for patients who receive care at multiple hospitals. On the other hand, medical and prescription claims provide a continuous clinical story that represents

care across providers and inclusive of pharmacy transactions. Claims data also captures costs and assesses medication compliance, but is short on clinical detail and lacks lab values, tumor staging information, and physicians' notes.

The integration of EHR and claims data offers health administrators, researchers, and policy makers an opportunity to draw insights from the richest versions of patient health data. By using clinical data from the EHR, condition identification can be significantly improved. For example, lab result data elements can support the identification of people with chronic kidney disease even without a coded diagnosis. Similarly, vital sign data can help identify people with hypertension despite the lack of a claim-based diagnosis. There are various data elements available in the EHR that, when analyzed, can allow the identification of a condition that was either not recognized or not coded for by the physician. EHR data enables better condition identification by providing access to data elements that allow one to impute a diagnosis, even if that diagnosis was never made.

Claims-linked EHR data offers integrated evidence with a fuller representation of reality and helps researchers follow the full story of a subject's health before, during, and after their disease treatment. Studies have shown that most patients' claims data is available in a continuous 2-3 year window before their cancer diagnosis appears in the EHR, and the average overlapping time period between the EHR and insurance claims can be as high as 93% depending on the type of cancer (Unknown [2022]). Therefore, researchers who use claims-linked EHR data can track the long-term outcomes of different treatment paradigms and conduct studies of treatment sequencing, switching, and drug adherence.

The development of matching applications that integrate EHR and claims data, such as the one proposed here, will have a significant impact on the next-generation patient care system. The ability to analyze patient health status using both EHR and claims data at the patient level provides a more complete picture of patient health, enabling health care professionals to make better-informed decisions about patient care.

B. Innovation

User Interface We are going to design the following key features for our shiny app.

- **Capture New Check**: The administrator can manually check the match box, and the change will be stored in our database.
- **Note Text Box**: Users will have the ability to add notes to each check, which will be stored in the system and can be easily accessed later.
- **Add Next and Previous Match Buttons**: Users will have the ability to navigate between matches using next and previous buttons, which will provide a more intuitive and efficient experience.
- **Resolve Errors**: We will investigate the error with specific subjects (id=712, etc) and implement a solution to ensure that the issue is resolved and does not occur again in the future.

Matching Algorithm Accurate matching between EHR and claims data is essential in this study. Traditionally, deterministic matching has been used, which employs a predefined set of matching rules to match patients across datasets. Deterministic matching can be fast and accurate when there is a high degree of overlap in the patient identifiers, but it may mismatch or generate false positives if the data is incomplete or in high-dimensions. To address this challenge, we propose the use of deep learning (DL) and natural language processing (NLP) techniques to enhance the accuracy and completeness of EHR and claims data matching.

- DL techniques are able to handle large volumes of complex data, such as EHR and claims data, which are often diverse and include various data types and formats. Moreover, deep learning models can be retrained and improve their accuracy and effectiveness over time as more data becomes available, making them a valuable tool for matching and integration of EHR and claims data in the future.
- NLP can be used to identify patterns from unstructured data such as physician notes which can then be used to link the data across different sources. For example, an NLP algorithm could be trained to recognize instances where a patient's medication is mentioned in both the EHR and claims data, even if the name or dosage of the medication is slightly different between the two sources. This can help ensure that the patient's medication history is accurately captured in both data sets.

The proposed DL and NLP techniques have the potential to improve patient care and research outcomes by enabling the analysis of richer versions of patient health status using both EHR and claims data.

C. Research Plan

To develop a user-friendly web application, we define the following research questions:

- What is the accuracy of matching EHR and insurance claims data?
- What is the impact of matching EHR and insurance claims data on patient outcomes?
- How can we improve the accuracy of matching EHR and insurance claims data?

Then we will conduct the research in the following steps:

- **Identify the data sources**: The two main data sources for this study are EHR data and insurance claims data. We need to identify the specific sources of these data and obtain the necessary permissions to use them.
- **Determine the matching criteria**: Matching EHR and insurance claims data requires the identification of common data elements that can be used to link the two datasets. This includes patient name, date of birth, gender, and medical record number.
- **Develop a matching algorithm**: Once the matching criteria are identified, we aim to develop a matching algorithm that can efficiently and accurately link the two datasets. This may involve using deterministic or probabilistic matching methods. Deterministic matching involves exact matching of data elements, while probabilistic matching uses statistical methods to match data elements that may be similar but not exact.
- **Train the algorithm**: If a probabilistic matching algorithm is used, the algorithm needs to be trained using a subset of the data that has been manually matched. This will help the algorithm learn the patterns and relationships between the data elements.
- **Test the algorithm**: The algorithm needs to be tested on a separate subset of the data that has not been used for training. This will help identify any issues or errors in the algorithm. Validate the algorithm: The algorithm needs to be validated on a larger dataset to ensure that it is accurate and effective when applied to real-world data.
- **Evaluate the accuracy of the matching algorithm**: The accuracy of the matching algorithm needs to be evaluated to determine its effectiveness. This can be done by comparing the matched data to a gold standard dataset that has been manually verified.
- **Analyze the matched data**: Once the two datasets are successfully matched, the researcher can analyze the data to answer the research questions. This may involve exploring the relationship between the EHR and insurance claims data, identifying patterns of care, or examining the impact of matching on patient outcomes.
- **Draw conclusions and make recommendations**: The final step is to draw conclusions from the data analysis and make recommendations for improving the accuracy and effectiveness of matching EHR and insurance claims data.

Specific Aim 1:

Hypothesis: Our project aims to develop a web application using R Shiny that allows users to capture new checks, add notes, and easily navigate between matches. The app will have an enhanced front-end with additional features that improve the user experience and provide a smoother workflow. Additionally, we will address any existing errors in the current system.

Rationale: This is what our user interface looks like now.

Figure 1: Current User Interface

To get information about a particular subject, users can enter the subject ID in the left box and click the “Record” button. Our shiny app will then display basic information about the subject in the top right, and all stride records associated with the subject in the bottom right. For instance, if the user searches for subject 698, our app will display six stride records related to that subject. Each record has a “match” option, which our model uses to predict whether the record matches our needs. If a record is ticked, it means our model has predicted it to be a match. In the example shown in the figure, the subject has two stride records that match our requirements. Users can also manually tick the “match” column to mark a record as a match, and the results will be stored in our database automatically to improve the model’s performance.

However, there are some errors that occur when searching for specific subjects. For example, if we search for id=712, our shiny app cannot fetch the stride record table for this person, and thus errors are displayed directly on our front-end as follows. This may confuse our users as they may not be aware of its meaning. Therefore, we need to dive deep into our back-end and database to figure out the source for this kind of error. If the error is unavoidable, we will use an alarm or messages to make it more clear and concise for our users.

Experimental Approach: In order to enhance the functionality of our shiny application, we will develop several application features using the R programming language to perform a wider variety of tasks and improve the overall user experience.

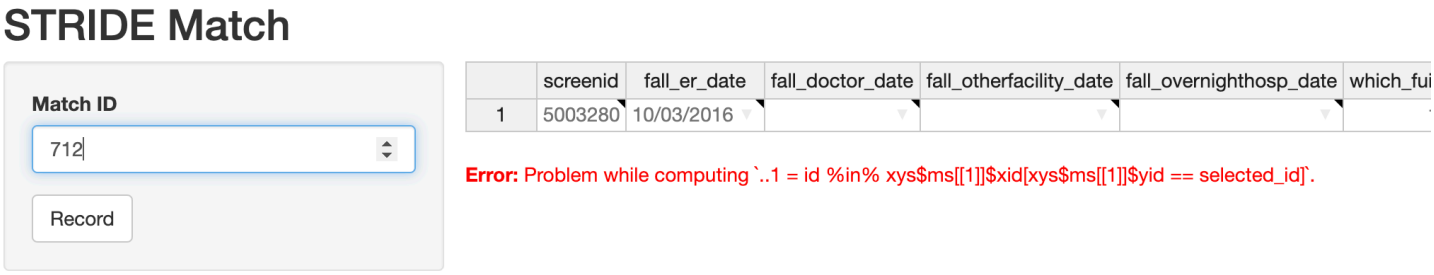


Figure 2: Current User Interface (Error)

- We will be introducing a “capture new check” feature that will enable users to create a new check within the application. This feature will provide users with greater flexibility in how they use the application and will make it easier for them to track and manage their progress.
- We will add a “note text box” feature that will allow users to add notes to their work as they progress through the application. This feature will help users to stay organized and will make it easier for them to remember important details as they work.
- In addition to these features, we will be introducing “next” and “previous” match buttons that will allow users to navigate through their work more easily. This feature will be particularly useful for users who are working on larger projects and need to move quickly between different parts of the application.
- We will dive deep into the current shiny application and develop a comprehensive testing plan that includes several unit tests. These tests will be designed to identify any bugs or errors that may be present in the application, and will help us to ensure that the app is delivered to users in a bug-free state.

By using these experimental techniques to develop new features in our shiny application, we hope to improve the overall user experience and make the application more versatile and useful to users.

Interpretation of Results: Our designed user interface is displayed below.

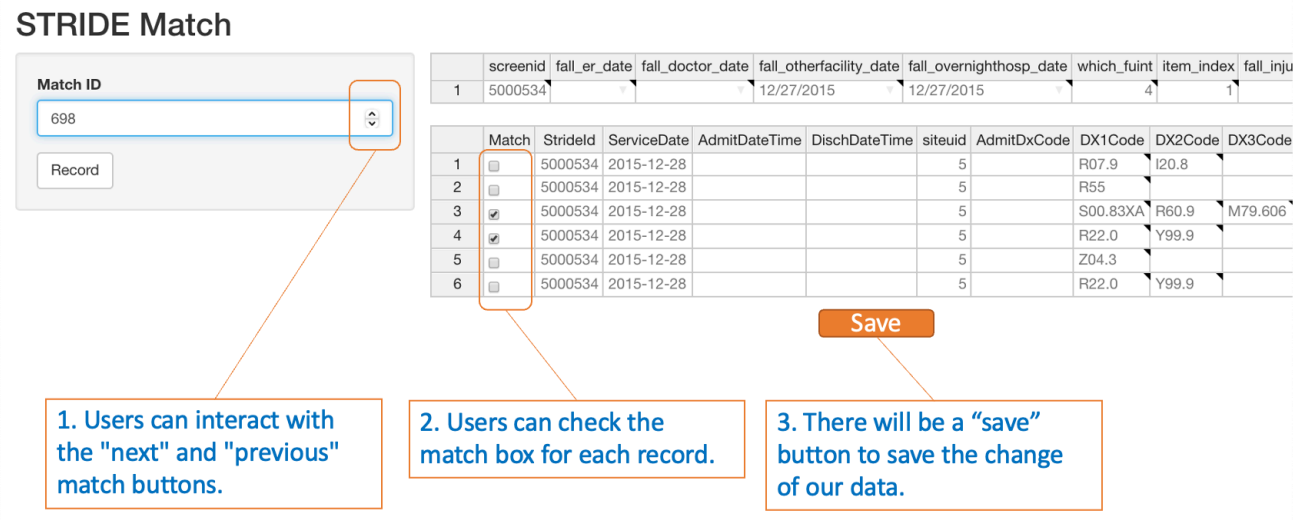


Figure 3: Designed User Interface

Potential Problems and Alternative Approaches: As we go through our development process, it’s possible that we may encounter some issues.

- The first feature we are introducing is the “capture new check” feature. While this feature provides users with greater flexibility and the ability to track and manage their progress more easily, it also presents potential issues with data privacy and security. To address these concerns, we will need to ensure that user data is securely stored and encrypted, and that only authorized users

have access to it. Additionally, we will need to carefully consider how this feature is integrated into the overall application design and user interface, to ensure that it is easy to use.

- The second feature we are introducing is the “note text box” feature. While this feature provides users with a useful tool for staying organized and remembering important details, it may also present challenges with information overload and cluttered user interface. To address these concerns, we will need to carefully consider how notes are stored and displayed within the application, and provide users with the ability to easily manage and organize their notes as needed.
- The third feature we are introducing is the “next” and “previous” match buttons, which will allow users to navigate through their work more easily. While this feature provides users with greater convenience and efficiency, it may also present challenges with information overload and navigation complexity. To address these concerns, we will need to carefully consider how the buttons are integrated into the overall application design and user interface, and provide users with the ability to customize their navigation preferences as needed.
- Finally, our comprehensive testing plan presents its own set of potential problems and alternative approaches. While unit tests are an effective tool for identifying bugs and errors in the application, they may not capture all potential issues that users may encounter in real-world scenarios. To address this concern, we will need to supplement our unit tests and gather feedback from a diverse range of users to ensure that the application meets the needs and expectations of its target audience.

In conclusion, the four features we are introducing, along with our testing plan, present potential challenges that need to be carefully evaluated and addressed to ensure the success of our shiny application. By taking a thoughtful and strategic approach to the development and testing process, we can overcome these challenges and deliver an application that is intuitive, user-friendly, and bug-free.

Specific Aim 2: Develop matching algorithm.

Hypothesis: Our hypothesis is that by using machine learning and deep learning methods, we can develop a match algorithm that can accurately and efficiently match EHRs with insurance

Rationale: Manual matching between the EHRs and insurance records is a complex and labor-intensive task that requires experienced staff. By automating the matching process, we can reduce the time and effort required to match records, while also improving the accuracy of the matching. This will help to improve the overall efficiency of the healthcare system and provide better care to patients.

We plan to randomly split our preprocessed data into three parts: train, validate and test. We will use our train data to train the model, in other words, to derive the appropriate parameters/model to fit the data. And we will tune our hyper-parameters based on observation on the performance of the validation data. Last, we test our model on the test data, which is part of our full data, but has no relation with our choice of the model/parameters. So it is the optimal choice to test our model on the test data.

Experimental Approach: Suppose the insurance record and EHR pairs are our observations Xs, and matching or not is the response which has only two values. So we transformed our task to a binary classification problem.

To test our hypothesis, we plan to explore two directions: traditional machine learning methods and deep learning methods. The experimental approach will involve the following steps:

- **Data cleaning and preprocessing**: We will clean and preprocess the data to get rid of outliers, contaminated records, and do feature selection if necessary.
- **Data splitting**: We will randomly split our preprocessed data into three parts: train, validate and test.
- **Data statistics**: We will collect the specific statistics of our preprocessed data, such as the distribution of the Xs and Ys to check if the label is balanced. If not, we will further do up/down-sampling to prepare for our model.
- **Model selection**: We will do experiments on both traditional machine learning models such as logistic regression, decision tree, and random forest, and deep learning models such as neural networks. We will try various combinations of layers and models depending on the data statistics and network performance.
- **Performance evaluation**: We will use the accuracy and ROC/AUC metric to evaluate our model performance. We will also compare the performance of the traditional machine learning models and deep learning models.

Interpretation of Results: The accuracy and ROC/AUC metric will be used to evaluate the performance of our model. Accuracy is simply the percentage of the correct predictions, which is limited when the data is imbalanced. So we use ROC/AUC score. It tells us how efficient the model is. The higher the AUC, the better the model’s performance at distinguishing between the positive and negative classes. An AUC score of 1 means the classifier can perfectly distinguish between the classes, which is, if they match or not.

Potential Problems and Alternative Approaches: One potential problem we foresee is how the raw data can be encoded to machine learnable features properly. Based on our preliminary research on the data, it is quite noisy mixed with null values, natural language, characters and numbers. It can be tough if the features cannot be embedded very well and it will definitely affect the model performance. To solve this challenge, we aim to find the patterns beneath the data and will try various encoding methods on it, including pre-trained embeddings although we know transfer learning is hard to do.

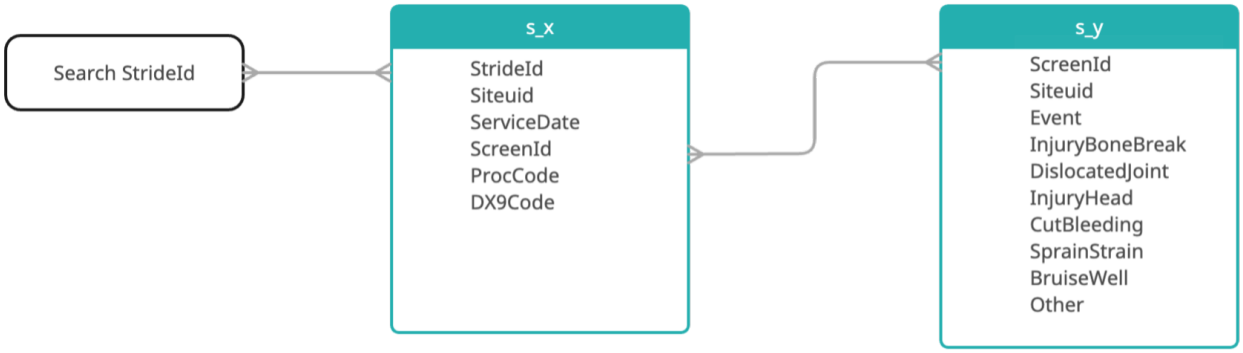
MILESTONES

A. 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.

B. 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.

C. Alpha version app has been implemented (html web development, APIs to search and update the database).

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.

DETAILED WORK

A. 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.

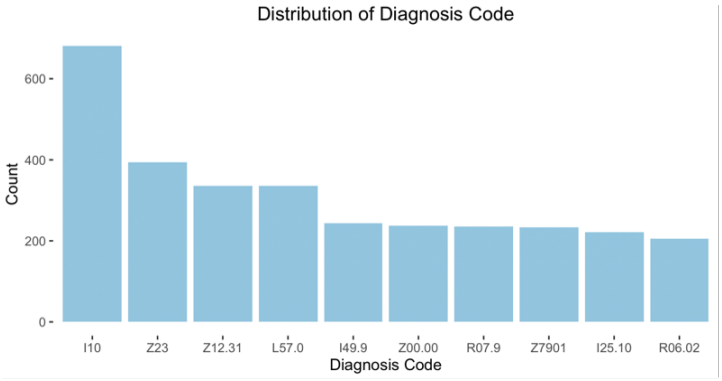


Figure 4: Distribution of Diagnosis Code

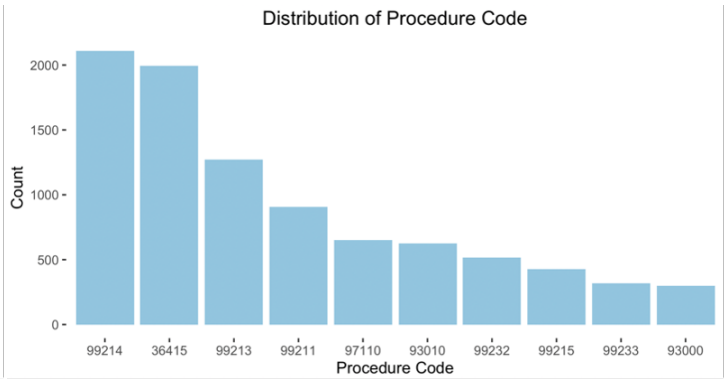


Figure 5: Distribution of Procedure Code

The following plot provides a summarized view of fall conditions in the EHR data

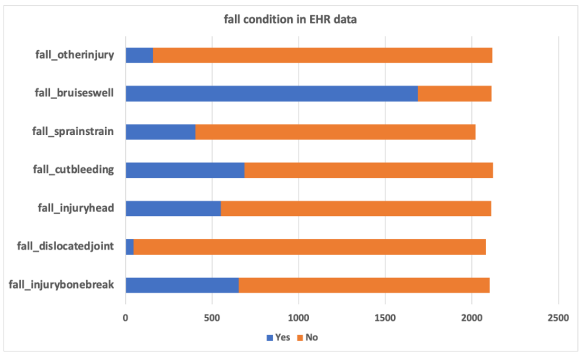


Figure 6: Descriptive Analysis of Fall Conditions

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 the majority of patients experienced a relatively mild level of fall injury during the time period.

B. Feature Development

Our team has developed a web application that serves as a centralized location for users to access and manage their healthcare data. The application is designed with a user-friendly interface that simplifies the process of accessing and navigating through insurance claims

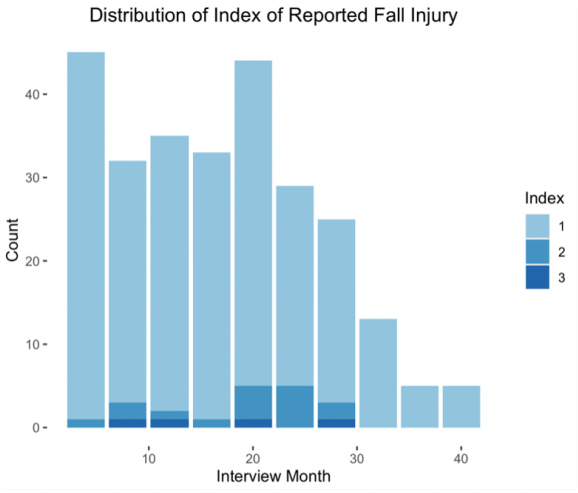


Figure 7: Distribution of Index of Reported Fall Injury

and EHR records.

The application enables users 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. The 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.

The application is built using the Python Flask framework due to its flexibility in developing a customized web application that met the specific requirements of the project. The frontend was developed using HTML, CSS, and JavaScript, while SQLite was used as the database management system for data storage.

Several APIs were developed to support data extraction, storage, and updating, ensuring seamless communication between the frontend and backend of the application.

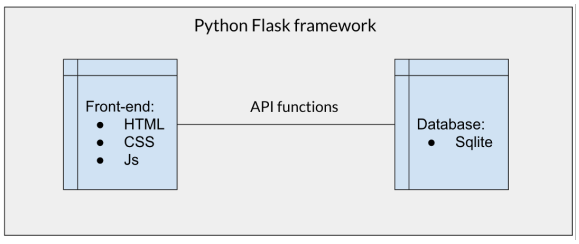


Figure 8: Tech Stacks Overview

Exciting new features have been added to the application to enhance the user experience and functionality.

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

- **Mark and Unmark:** 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.

- **Save New Checks:** 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 quickly and safely save those checks in our database.

- **Mouseover Text Description:** 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.

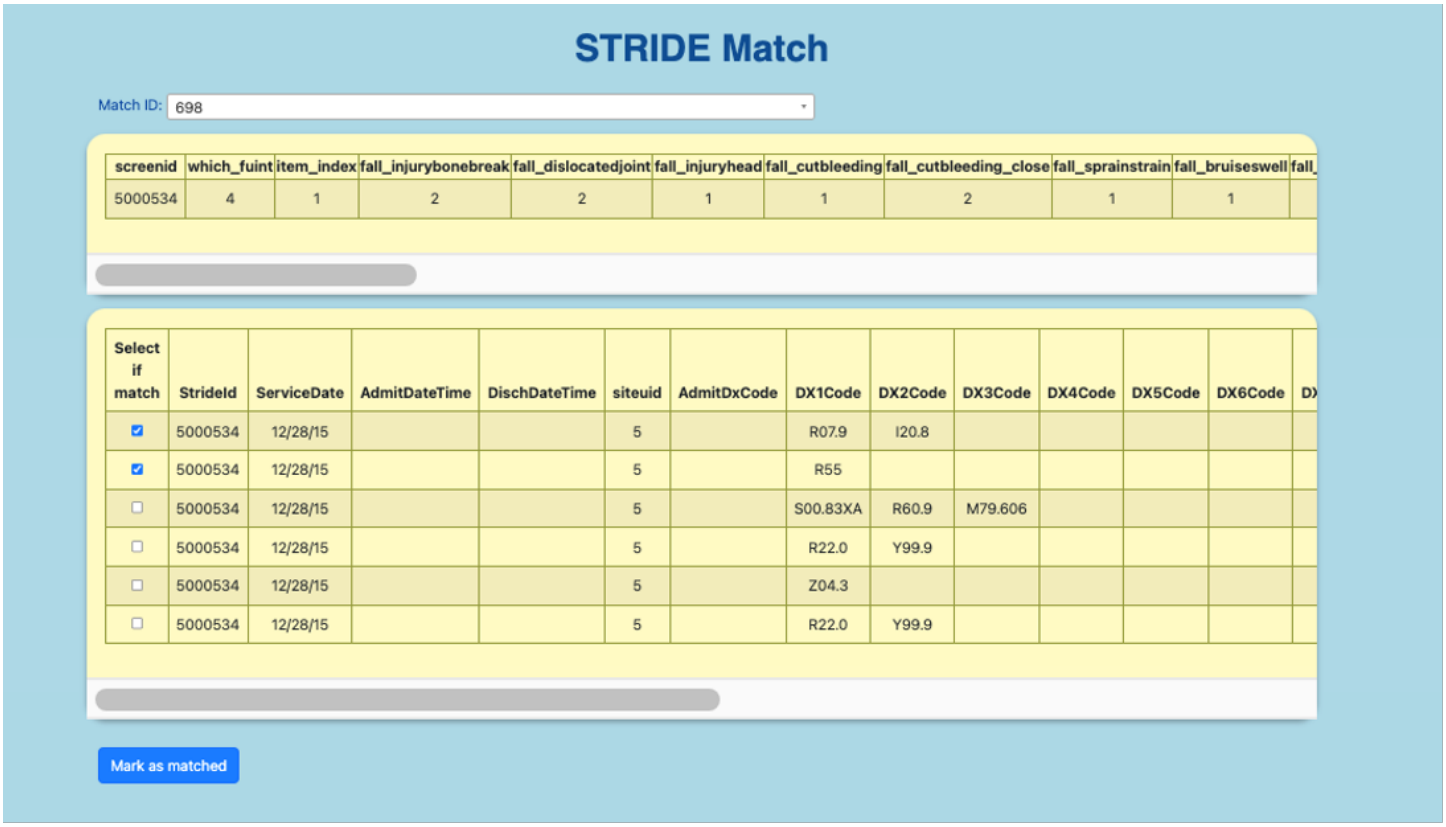


Figure 9: Current UI

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.

C. Application Enhancements

We have made some significant changes to our app.

- **From R Shiny to Python Flask:** 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.
- **From Static Data Files to Dynatic SQLite database:** 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.
- **APIs Implementaion:** 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.
- **Web Interaction Page Development:** 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.
- **Resolved Unkown Errors:** 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.

PROEJECT SUMMARY

The project aims to develop a user-friendly web application using Python Flask that will allow users to capture new checks, add notes, and navigate between matches. The team proposed to develop a matching algorithm using machine learning and deep learning methods that can match EHRs with insurance records but postponed this aim due to time constraints. The project's significance lies in its potential to revolutionize the way patient care is analyzed and evaluated by combining the benefits of EHRs and claims data. Furthermore, the developed application is designed to facilitate the integration of EHR and claims data, which is a critical step towards improving patient care and outcomes. By combining these two types of data, healthcare professionals can gain a more comprehensive and accurate understanding of patients' health status, history, and needs. This can help to improve diagnosis, treatment, and care coordination, leading to better health outcomes for patients.

The team has analyzed two datasets - the EHR dataset and the claims data set. We have developed a web application that provides users with a comprehensive and organized view of their healthcare information. The application features a Dropdown Search Box, Mark and Unmark feature, Save New Checks, and Mouseover Text Description. These features aim to enhance the user experience and functionality of the application, making it easier and more convenient for users to access and manage their data. Compared to the older version of stride matching application, we made significant changes to our application, including transitioning to Python Flask from R Shiny, converting static data files to a dynamic SQLite database, building several APIs, designing fully functional web pages, and resolving unknown errors.

In terms of future work, developing a matching algorithm using machine learning and deep learning methods is a crucial step towards fully realizing the potential of the developed ecosystem. By automating the process of matching EHRs with insurance records, the algorithm can save time and effort for healthcare professionals and provide more accurate and efficient results. Additionally, further refining the UI page to make it even more user-friendly can enhance the overall user experience and encourage wider adoption of the application.

In conclusion, the developed web application successfully meets the project aims and provides a powerful ecosystem for matching EHR and claims data. With further development and refinement, it has the potential to revolutionize the way patient care is analyzed and evaluated, leading to better health outcomes for patients.

References

Manvi Sharma, Michael L Johnson, Hui Zhao, Sharon H Giordano, and Holly M Holmes. Concordance between electronic health record data and medicare part d claims data for oral anticancer drug use. *JAMA Network Open*, 3(4):e203821–e203821, 2020.

Unknown. Abstracts of the 38th international conference on pharmacoepidemiology: Advancing pharmacoepidemiology and real-world evidence for the global community, august 26-28, 2022, copenhagen, denmark. *Pharmacoepidemiol Drug Saf*, 31 Suppl 2:3–678, 2022. ISSN 1099-1557. doi: 10.1002/pds.5518. URL https://doi.org/10.1002/pds.5518.

Suzanne L West, William Johnson, Wendy Visscher, Marianne Kluckman, Yue Qin, and Ann Larsen. The challenges of linking health insurer claims with electronic medical records. *Health informatics journal*, 20(1):22–34, 2014.