Lowering Wait Times in Emergency Rooms using Process Mining and Machine Learning

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE COURSE

CHBE 492: Chemical and Biological Engineering Thesis,
Topic: Data Analytics and Intelligent Systems

Prepared by:

Atharva Laud – 35127042

Supervisor:

Dr. Bhushan Gopaluni

THE UNIVERSITY OF BRITISH COLUMBIA

April 16th, 2024

Contents

List of Abbreviations

Abbreviation	Definition
ED/ER	Emergency Department/Emergency Room
KPI	Key Performance Indicators
ML	Machine Learning
AI	Artificial Intelligence
NLP	Natural Language Processing
RPA	Robotic Process Automation
RF	Random Forest
ICD	International Classification of Diseases
UTI	Urinary Tract Infections
BTI	Brain Trauma Injuries
SVM	Support Vector Machines

Acknowledgements

I extend my deepest gratitude to my thesis supervisor, Dr. Bhushan Gopaluni, and Ph.D. student Gregory Li, for their invaluable support and guidance throughout my thesis. This project has been an enriching learning experience, offering me profound insights into research methodologies and the practical application of machine learning tools to optimize societal processes.

Abstract

This thesis examines the use of process mining and machine learning to improve the operational efficiency of emergency departments, specifically targeting patients with urinary tract infections and brain trauma injuries. Key to the study is the transformation of complex patient data into structured event logs and the creation of a predictive machine learning model that utilizes patient vitals—such as heart rate, temperature, and blood pressure—to predict medication administration. Despite the model's dependence solely on patient vitals for predictions, it produces detailed process maps that enhance our understanding of emergency department workflows. However, the model's accuracy is constrained by the lack of detailed patient data and the encrypted nature of sensitive information, leading to challenges in accurately predicting medication types. With improvements in data quality and access to more specific patient details, the model could substantially advance the efficacy of emergency department processes, aligning with stringent healthcare standards for machine learning applications. The thesis underscores the potential for more sophisticated machine learning models to refine emergency care, pending enhancements in data collection and model training methodologies to address current shortcomings.

Introduction

Emergency rooms (ERs) are fundamental to modern healthcare, offering immediate and 24/7 medical care to those in urgent need. Their role is crucial in handling acute medical emergencies, serving as the primary response in public health crises, and ensuring access to life-saving treatments, especially for vulnerable populations. Beyond patient care, ERs offer a rich environment for healthcare research due to the diversity of patient cases and the need for quick, effective decision-making. This makes them ideal for studying workflow efficiencies and patient outcomes. In Canada, ER wait times present a significant challenge with an average wait time during late 2023 measured to be 22 hours (Canadian Medical Association, no date). Hence, leading to risks for patient health and exposing systemic inefficiencies in healthcare. This problem offers a unique opportunity for data-driven research, which can provide insights into patient flow, resource allocation, and care delivery bottlenecks. By analyzing real-time data and event logs, modern-day software technologies can allow for the mapping of patient journeys and the identification of patterns to optimize ER processes. Addressing long ER wait times in Canada through process mining represents a crucial step in improving the quality and timeliness of emergency healthcare services, transcending academic research to impact real-world healthcare delivery.

Background and Literature Review

Emergency Room Workflow

Emergency rooms are dynamic and fast-paced, often characterized by a continuous influx of patients and demanding work hours for the staff. To address the challenges inherent in these

settings, it's crucial first to understand their operational workflow, which typically involves the following steps (Jarvis, 2016):

1. Patient Entry: The patient arrives and is admitted to the ER.
2. Triage: Medical staff assess the patient's vital signs, medication history, and the severity of their condition.
3. Diagnosis: Doctors or physicians determine the nature of the illness and contemplate potential treatment options.
4. Treatment: The patient receives the appropriate medical care, which may range from surgery to the administration of medications like antibiotics.
5. Patient Discharge: The patient is discharged and exits the ER after receiving the necessary treatment.

Understanding this workflow is essential for identifying areas where process mining and data visualization can be effectively applied to enhance efficiency and patient care in emergency departments.

Process Mining

Process mining, bridging data science and process management, is crucial in healthcare due to the complexity of medical procedures. This technique extracts insights from event logs, detailing activities' sequences and timings, resources used, and patient information (De Roock & Martin, 2022). It employs methods like process discovery, conformance checking, and enhancement to create, assess, and improve process models from these logs (Mans et al., 2022). The figure below outlines how process mining can be applied to healthcare facilities for acquiring process insights.

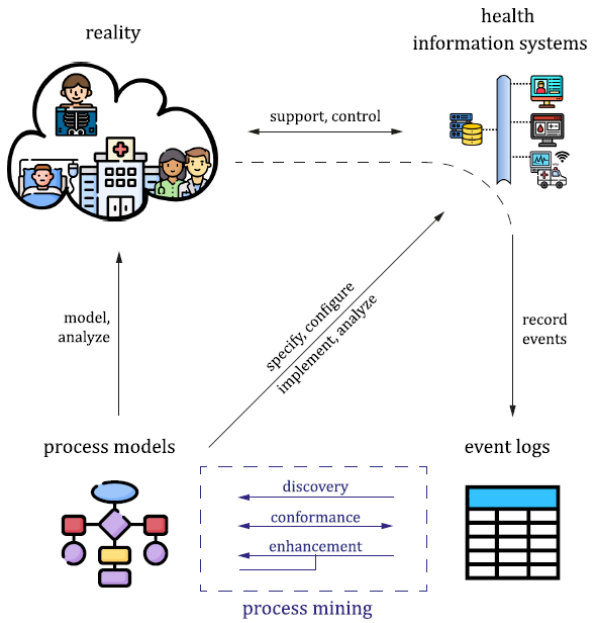

Figure 1: Diagram presenting how process mining models function in the healthcare sector (De Roock & Martin, 2022).

In healthcare, process mining optimizes operational efficiency, patient care, and adherence to clinical guidelines. It has been significant in areas such as oncology, surgery, cardiology, and primary care, utilizing various tools and algorithms to manage healthcare's inherent complexity. These tools and techniques used can vary, with some of the most common being ProM, a tool for process mining, and algorithms like Heuristics Miner and Fuzzy Miner, which are particularly adept at handling the variability inherent in healthcare processes (Rojas et al., 2016). Despite its progress, challenges in creating complex petri net diagrams indicate the need for further research, particularly in simplifying and optimizing less intricate medical cases (Sundari et al., 2020; Martin et al, 2020).

The evolution and impact of process mining in healthcare can be seen through several studies. Recent literature reviews have shed light on the stages of process mining projects, the crucial role of domain experts, and the application of Key Performance Indicators (KPIs) in these

studies (Munoz-Gama, 2022). The involvement of domain experts, particularly in validation and interactive analysis stages, highlights the importance of interdisciplinary collaboration in understanding and improving healthcare processes (Gatta et al, 2020). These experts play a key role in different stages of process mining projects, including problem identification, data extraction, and data preparation, thereby ensuring that the insights derived are relevant and actionable (De Roock & Martin, 2022). This comprehensive approach in the application of process mining in healthcare underscores its potential in improving healthcare processes and outcomes. Figure 2 below showcases a sample process mining workflow generated using the library pm4py, a common library for mapping processes in python.

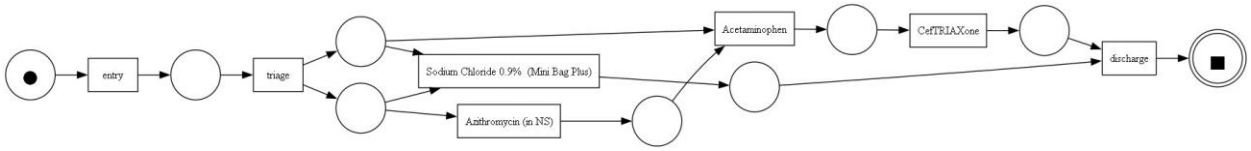

Figure 2: Sample treatment process flow generated using process mining algorithm: Alpha miner (image produced using pm4py library on python)

Machine Learning

Machine learning (ML), a cornerstone of artificial intelligence (AI), has significantly influenced the healthcare sector by employing statistical methods to enable models to learn from data and make predictions or decisions. Machine learning operates in two primary modes: supervised and unsupervised learning. Figure 3 below depicts the most common steps used for creating and applying machine learning models.

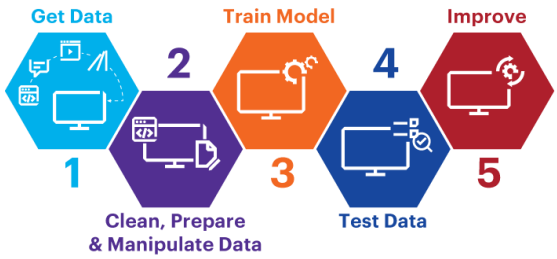

Figure 3: Machine learning model development workflow (Medium ML, 2020)

In the context of supervised machine learning, as delineated in the provided workflow figure above, models are carefully trained on datasets where each instance is associated with a labeled outcome. The procedure initiates with the acquisition of data, where the quality and quantity of the input-output pairs are paramount. This is followed by rigorous data preprocessing, a critical step that involves cleaning, normalization, and transformation to ensure optimal data quality for model training. The model is then trained to discern patterns and associations within the labeled data, effectively minimizing the disparity between its predictions and the actual outcomes, thereby enhancing its predictive prowess.

After the model's training phase, its performance is evaluated using a testing set—a collection of data reserved exclusively for assessment purposes to gauge the model's accuracy on previously unseen data. This phase is pivotal in validating the model's generalizability and reliability. Post evaluation, the model undergoes iterative refinements, with adjustments to its configuration and training data, in a bid to amplify accuracy and diminish error rates. Although this structured approach typically yields high precision in predictions, it is contingent on the availability of substantial labeled datasets—a requirement that may impose constraints due to the resource-intensive nature of data labeling. On the contrary, unsupervised learning, which is not illustrated in the above workflow, leverages unlabeled data to identify inherent patterns, excelling in

exploratory analysis but potentially faltering in delivering the level of precision in complex predictive tasks that supervised learning achieves.

In the healthcare domain, these machine learning techniques have been crucial, particularly in precision medicine. Here, they assist in predicting treatment outcomes based on various patient attributes (Davenport & Kalakota, 2019). Advanced forms of unsupervised machine learning, such as neural networks and deep learning, have been deployed for complex tasks like identifying potential cancerous lesions in radiological images. Neural networks, simulating the way neurons in the human brain process signals, and deep learning, characterized by multi-layered neural networks, handle highly intricate tasks, making them instrumental in healthcare innovations (Davenport & Kalakota, 2019).

Complementing machine learning in healthcare AI is Natural Language Processing (NLP), which interprets human language, crucial for analyzing clinical documentation and enhancing patient communication. Another example is Robotic Process Automation (RPA), which streamlines administrative tasks in healthcare by automating routine digital tasks (Davenport & Kalakota, 2019).

Random Forests and Decision Trees are foundational algorithms in machine learning, each with distinct yet complementary mechanics. A Decision Tree is a flowchart-like structure in which each internal node represents a "test" on an attribute, each branch represents the outcome of the test, and each leaf node represents a class label. The paths from root to leaf represent classification rules (Breiman et al., 1984). In contrast, a Random Forest is an ensemble approach that creates a 'forest' of multiple Decision Trees, which are trained on random subsets of the data, and then aggregates their predictions to improve accuracy and control over-fitting (Matsuki et al,

2016) (Dittman et al, 2011). The ensemble nature of Random Forests makes them more robust and accurate than individual Decision Trees.

In emergency rooms (ERs), these algorithms have been utilized for various predictive tasks such as triage prioritization, predicting patient outcomes, and hospital readmission rates (Lee et al., 2019). Decision Trees help in identifying the most critical factors affecting patient wait times, while Random Forests, with their higher accuracy and ability to handle imbalanced datasets, are used for more complex predictions such as the likelihood of a patient developing a certain condition based on their ER admission data (Kwon et al., 2018). These tools aid medical practitioners in making data-driven decisions that can lead to more efficient patient management and potentially save lives by prioritizing those in most urgent need of care.

Event Log Data Source and Content

Emergency event logs consist of several different variables and factors that need to be analyzed. Therefore, it is crucial be familiar with this content prior to modelling these complicated processes. The database being studied is widely known as “MIMIC IV” which contains event log data from the Harvard Medical Student Training Facility in Boston, MA. This database contains 6 different datasets specifically including: patient diagnosis, medication history, medicine administration, entry/exit times, vitals, and triage data. All sheets contain a “subject ID” and a “stay ID” to allow for cross-referencing between the datasets. Table 1 below briefly explains what each of these datasets contains, additional data samples and supplementary information can be found below in Appendix A.

Table 1: Explanation of medical database (MIMIC-IV) content and datasets (Johnson et al, 2023)

Dataset Title	Content	Notes and Comments
Triage	• Patient vitals, includes heart rate, blood pressure, etc. • Chief complaint, indicating the primary patient complaint • Acuity, indicating how close the patient is to death (out of 5, 0 being severe)	Triage vitals are measured prior to treatment and diagnosis to identify patient severity and priority.
Vitals	• Patient vitals, includes heart rate, blood pressure, etc. • Vitals measurement timestamp • Patient's self-reported pain level (out of 10, 10 being severe)	Patient vitals are re-recorded every 1-4 hours to decide on severity and treatment procedure.
Medication History	• Patient Medication History, with medication name and ID • Chart-time indicating when the staff member recorded past medication	Patient medication history is recorded to ensure treatment doesn't interfere with previous injuries/diseases.
Diagnosis	• Disease/injury name • Other symptoms • International Classification of Diseases (ICD) code to categorize types of injuries	ICD codes follow a specific naming convention, to help staff classify various types of diseases and the treatments associated with them.
Medicine Administration	• Chart-time indicating when the staff member added medicine to be administered • What and how the medication was administered • Medication name and ID	Chart-times for this are the best approximates as to when the medicine was administered, however it is fair to assume a +/- 5min interval.
Entry/Exit	• Entry Time • Exit Time • Method of transport (walk-in, ambulance, etc.)	Important to track overall patient wait times with different diseases.

Thesis Objectives and Research Questions

This thesis is dedicated to exploring how process mining techniques and data visualization can be utilized to reduce waiting times in emergency departments. Given the vast array of medical conditions treated in hospitals, the focus is on targeting less critical yet diagnostically challenging injuries and diseases. These specifically include urinary tract infections (UTIs), and

brain trauma injuries (BTIs) which often present the greatest opportunities for optimization in terms of diagnosis and treatment efficiency. The ICD codes are as follows (Please note the "X" indicates a placeholder for additional characters to specify the severity and location of the injury):

1. Brain Trauma Injuries
    i. Traumatic Subdural Hemorrhage: ICD-10 code: S06.5X, ICD-9 code: 804X, 850X
    ii. Traumatic Subarachnoid Hemorrhage: ICD-10 code: S06.6X, ICD-9 code: 854X, 800X
2. Urinary Tract Infections
    i. Pyelonephritis: ICD-10 code: N10.X, N11.X, ICD-9 code: 59010X, 59011X
    ii. Unspecified UTIs, ICD-10 code: N39.0, ICD-9 code: 5990X

Leveraging hospital data, this study aims to gain insights that can effectively streamline patient flow and bolster operational efficiency in emergency care. The objective of this research requires accomplishing the following goals:

1. Data Organization and Event Log Formatting: Transform and filter the dataset into a structured event log format suitable for process mining.
2. Machine Learning Model Development: Create and train a machine learning model that will predict medication administration for UTIs and BTIs.
3. Development of Specific Process Flows: Construct injury/disease-specific process flows using process mining.
4. Identification of Study Limitations and Constraints: Acknowledge and delineate the limitations and constraints of the study.

Therefore, the research questions for this study are as follows:

1. How can process mining techniques identify various medicinal administration pathways for BTIs, and UTIs?
2. How can machine learning derived process workflows enhance decision-making processes in emergency departments?
3. What are some challenges in integrating process mining and machine learning technologies in current ER protocols and how can they be overcome?

Methodology: Data Preprocessing and Reformatting

In the field of healthcare, data preprocessing and reformatting are pivotal steps in using machine learning algorithms to improve patient outcomes and streamline operations. This thesis section delves into the detailed preprocessing of diverse datasets, specifically focusing on UTIs, BTIs, and medication administration records. The primary objective is to refine and prepare these datasets for effective analysis, ensuring that the ML models developed can accurately identify patterns and propose solutions for reducing emergency room wait times. By applying targeted preprocessing techniques tailored to each dataset's unique characteristics and challenges, it aims to eliminate irrelevant data, reduce noise, and highlight the most significant features for analysis. The following paragraphs detail the specific preprocessing steps undertaken for UTI cases, BTI cases, and medication administration records, highlighting the rationale and methodologies employed to optimize data quality for machine learning applications.

UTI Cases Preprocessing

In the UTI data preprocessing phase for this study, the initial step involves filtering the diagnosis dataset for UTIs based on specific ICD codes. The selection criteria encompass all codes beginning with N390, 5990, N10, N11, 59010, and 59011. This deliberate filtering process targets the most prevalent UTI instances while also isolating a more refined set of cases for analysis. Furthermore, an additional screening layer is applied to exclude cases presenting multiple diagnoses alongside UTIs. The rationale behind this exclusion is to mitigate potential complications during model training phases, where medications administered for conditions other than UTIs could introduce biases or inaccuracies.

BTI Cases Preprocessing

In parallel with the approach for UTIs, the data preprocessing for BTIs also begins with a selective filtration based on ICD codes starting with S06, 800, 804, 850, and 854. Unlike the UTI dataset, for BTIs, all concurrent injuries and diseases diagnosed alongside concussions are retained within the dataset. This decision stems from the observation that comorbid conditions are commonly associated with brain injuries (Tator, 2013). Retaining these additional diagnoses allows for a more comprehensive analysis of treatment patterns, reflecting the real-world complexity of managing brain injuries and their frequent comorbidities. This inclusive approach is designed to enhance the understanding of how brain injuries, along with their associated conditions, are treated in practice, acknowledging their prevalent co-occurrence.

Medicine Administration Preprocessing

For the medicine administration dataset preprocessing, the dataset exhibits challenges such as duplicates and variations in medication dosages and sizes. It is essential to normalize these variations because medications sharing the same active ingredient should be considered

equivalent for the purposes of this analysis. However, this normalization must be approached with caution, as differences in administration methods (e.g., intravenous vs. oral) and doses can have significant implications on treatment effectiveness.

To address these complexities, the dataset will be processed to identify and consolidate medications based on their active ingredients while distinguishing between different administration routes and concentrations. This task will be accomplished using the regex (regular expression) library in Python. The objective of applying regular expressions is to thoroughly filter out extraneous details such as volume and other descriptors enclosed in parentheses, ensuring that the analysis accurately reflects the primary ingredient of the medications. By filtering out specific syntax patterns, the preprocessing aims to maintain the integrity of the medication data, including the diversity in medication types and concentrations, thus providing a solid foundation for subsequent analyses. The script attached in Appendix B details the specific string patterns that were filtered out.

Moreover, an important aspect to consider in the dataset analysis is the occurrence of numerous one-off cases, where certain medications appear infrequently, with their occurrences limited to just a few instances within the dataset. This pattern raises questions about how to optimally leverage this varied distribution of medication data for analytical purposes. To better comprehend the distribution of medication frequencies and to devise a strategy for incorporating this information into the analysis, the frequencies of different medications have been charted. This distribution is detailed in Figure 4, which visualizes the frequency of each medication across the dataset.

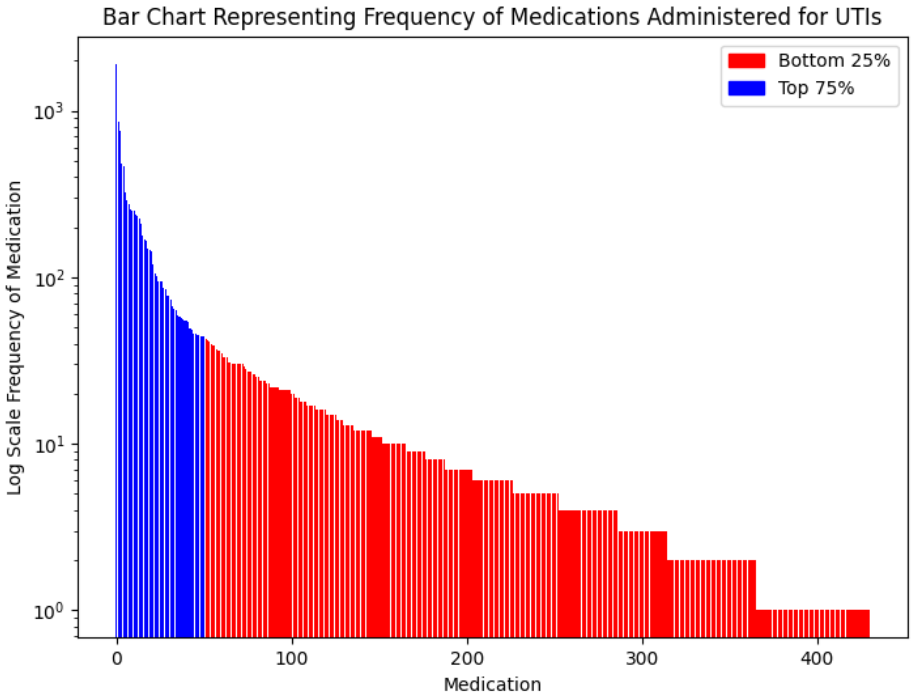

Figure 4: Log y-scale bar chart representing medications administered and their frequency for patients diagnosed with UTIs

When analyzing the medication frequency data on a log scale, a distinct pattern emerges, highlighting that a small subset of medications is used with high frequency across the dataset. It's particularly striking to note that the most common 49 medications account for the top 75% of all medication instances, while the remaining 380 medications collectively make up only 25% of the total instances of medication administration. This skewed distribution suggests a concentration of preference or efficacy among a limited number of medications for treating the conditions in question, particularly UTIs in this context.

To delve deeper into the reasons behind this distribution and to understand the specific roles of the less frequently used medications—the bottom 25%—an investigation into the medications is needed. For the sake of clarity and ease of analysis, a subset of the common findings from this investigation have been systematically organized into a table. This table aims to shed light on the

utility and application of these infrequently used medications, providing insights into their inclusion and usage patterns within the dataset, thereby offering a more nuanced understanding of medication strategies for UTI treatment. Additional medications that were uncommonly administered are detailed in Appendix A.

Table 2: Subset of infrequently administered medications with the frequency, purpose and it usage for UTIs

Bottom 25% Medication Name	Frequency	Purpose	How it is used to treat UTIs
Lithium Carbonate	2	Lithium is used to treat mania that is part of bipolar disorder (manic-depressive illness). It is also used daily to reduce the frequency and severity of manic episodes. ("Mayo Clinic", no date)	Mostly used only if patient is exhibiting manic behaviour during treatment or care.
Olanzapine	43	Olanzapine is used to treat schizophrenia. It may also be used alone or with other medicines (eg, lithium or valproate) to treat mania or mixed episodes that is part of bipolar disorder. ("Mayo Clinic", no date)	Mostly used only if patient is exhibiting schizophrenic behaviour during treatment or care.
Finasteride	5	Finasteride is used to treat men who have symptoms of benign prostatic hyperplasia (BPH) and male pattern hair loss, also called androgenetic alopecia. ("Mayo Clinic", no date)	Mostly administered to men who are at risk of BPH as well with UTIs, which is relatively uncommon.
Amoxycillin Capsule	1	Amoxicillin is used to treat bacterial infections in many different parts of the body. It is also used with other medicines (e.g., clarithromycin, lansoprazole) to treat H. pylori infection and duodenal ulcers. ("Mayo Clinic", no date)	Despite being a common antibiotic, it is rarely administered for UTIs

From the analysis, it becomes apparent that a significant portion of the medications cataloged in the dataset does not directly contribute to the treatment of UTIs. These medications are predominantly administered to patients who experience extended stays in the ED due to behavioral issues often associated with manic or depressive states, rather than the UTI itself. A parallel trend is observed in the dataset concerning BTIs, where a core set of 54 medications covers the top 75% of cases, leaving the remaining 380 medications to account for the bottom 25% of instances, as illustrated in Figure 5 below.

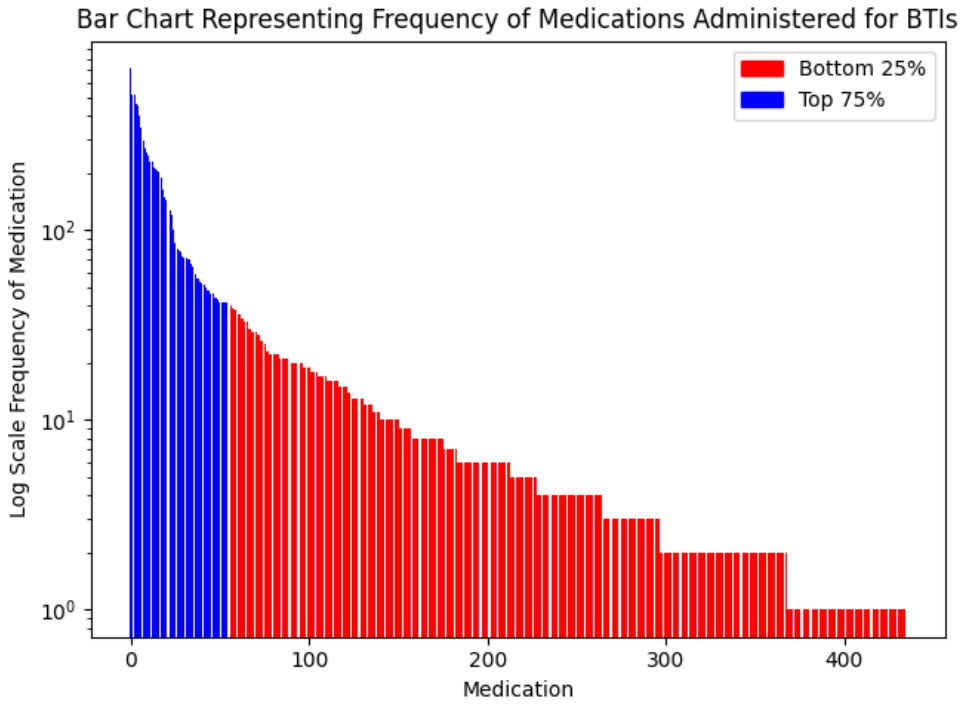

Figure 5: Log y-scale bar chart representing medications administered and their frequency for patients diagnosed with BTIs

Given this distribution, the decision to exclude the lower 25% of rarely used medications from both UTI and BTI analyses is strategic, aiming to minimize dataset noise and enhance the predictive performance of the model. This exclusion is predicated on the understanding that these

infrequently administered medications introduce a potential for increased error and could disproportionately skew the model's outcomes due to their limited relevance to the conditions being studied. By focusing on the medications most relevant and frequently used in the treatment of UTIs and BTIs, the analysis ensures a more accurate and focused examination of treatment patterns, thereby improving the overall reliability and efficacy of the model in predicting outcomes and optimizing emergency department wait times.

Overall Data Preprocessing

Following the initial filtering and specific preprocessing steps for each dataset, a comprehensive reformatting process is applied across all datasets. This process entails the meticulous removal of any blank entries, incorrect values, and data mismatches to ensure the integrity and consistency of the datasets. Concurrently, all values are systematically converted to their appropriate data types, including datetimes for timestamps, strings for textual data, and integers for numerical values. This conversion is critical for maintaining data uniformity and facilitating accurate analysis.

After these cleanup efforts, the datasets undergo a crucial transformation phase to prepare them for machine learning analysis. At this stage, any categorical or textual data that cannot be directly interpreted by the model—such as ICD codes and medication names—is encoded. The LabelEncoder function from the preprocessing module in Python's machine learning library, scikit-learn, is employed for this purpose. This encoding process converts these string values into a model-readable numeric format, ensuring that all data can be effectively utilized in the machine learning workflow.

Model Training and Testing Datasets

Once the datasets have been properly filtered and cleansed of any discrepancies, the next step involves leveraging the stay IDs unique to each dataset to amalgamate them into a comprehensive dataframe. This unified dataset will encompass all pertinent metrics and variables necessary for the analysis. By integrating these datasets, we establish a foundation that encapsulates a spectrum of patient information, treatment details, and outcomes. This holistic approach ensures that the machine learning model has access to a rich and diverse dataset, enhancing its ability to uncover meaningful patterns and insights that are critical for achieving the study's objectives of improving emergency room efficiency. Table 3 below showcases the corresponding input and target variables read by the model.

Table 3: Shows feature and target variables for the medication administration model

<table>
<tr><th colspan="2">X Parameters (Model Feature Variables)</th></tr>
<tr><th>Metric</th><th>Justification</th></tr>
<tr><td>Gender</td><td>Particularly useful for UTI cases, since medications can largely differ for men and women.</td></tr>
<tr><td>Heartrate</td><td rowspan="5">The following metrics are taken upon the patient's arrival and provide information on how serious the patient's condition is, which can significantly affect the medications administered.</td></tr>
<tr><td>Temperature</td></tr>
<tr><td>Oxygen saturation</td></tr>
<tr><td>Blood pressure (systolic/diastolic)</td></tr>
<tr><td>Respiratory rate</td></tr>
<tr><td>Acuity</td><td>Is useful because it provides the staff's perspective on the patient's severity and condition.</td></tr>
<tr><td>Encoded ICD Code</td><td>Provides a specific classification, which is useful since different cases are treated differently.</td></tr>
<tr><th colspan="2">Y Parameters (Model Target Variable)</th></tr>
<tr><td colspan="2">Array of Encoded Medications Administered</td></tr>
</table>

Figure 6 depicted below illustrates a segment of the input dataframe used in the machine learning model. In this representation, columns labeled A through J are designated as feature variables, which include various patient data and other relevant factors that the model utilizes to understand and predict outcomes. Conversely, columns K through T are assigned as target columns; these contain the encoded values of the medications administered to the patients. Each row in these target columns represents a list of medications given to a patient, where numerical values correspond to specific medications, and 0s are used to indicate the absence of additional medications. This setup ensures a structured approach to model training, where the relationship between patient characteristics (features) and their treatment regimens (targets) can be effectively learned and analyzed.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	index	acuity	temp	heartrate	resprate	sbp	dbp	o2sat	gender	condition_X	med1	med2	med3	med4	med5	med6	med7	med8	med9	med10
2	0	3	101.6	108	16	124	73	99	0	5	12	0	0	0	0	0	0	0	0	0
3	1	3	98.6	72	18	125	62	100	1	5	9	17	46	0	0	0	0	0	0	0
4	2	2	97.9	106	20	143	91	100	0	5	9	30	35	0	0	0	0	0	0	0
5	3	3	99	94	15	111	63	97	0	5	2	22	33	44	0	0	0	0	0	0
6	4	3	100.2	94	18	138	80	98	0	3	1	5	9	46	0	0	0	0	0	0
7	5	2	97.1	84	18	178	96	97	0	5	9	44	0	0	0	0	0	0	0	0
8	6	3	102	100	16	161	91	96	0	5	9	44	49	0	0	0	0	0	0	0
9	7	2	99.8	97	32	176	86	88	0	2	9	0	0	0	0	0	0	0	0	0
10	8	3	98.1	88	18	113	57	100	0	5	2	9	0	0	0	0	0	0	0	0
11	9	3	97.7	58	16	119	59	100	1	5	2	9	0	0	0	0	0	0	0	0

Figure 6: input dataframe format into the machine learning model

Before looking at the data splitting for training and testing, the dataset undergoes a crucial step where the rows are randomly shuffled. This randomization ensures that the division into training and testing sets is unbiased and representative of the overall dataset, preventing any inadvertent patterns or ordering in the data from influencing the model's learning process.

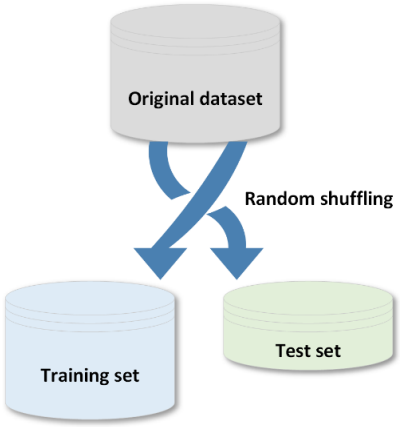

Figure 7: Image representing how training/testing sets are created from the original dataset (Machine Learning Algorithms, 2015)

In the subsequent model training phase, a strategic approach is adopted to optimize the model's learning and predictive accuracy. An 80:20 split between training and testing datasets is employed, allocating 80% of the data for training the model. This larger portion for training allows the model to immerse in a more extensive range of data, enabling it to uncover and learn from complex patterns and trends within the dataset. The remaining 20% serves as the testing set, which is used to rigorously evaluate the model's predictive performance and accuracy. Opting for this split over the traditional 70:30 ratio is particularly justified for this study due to the model's complexity and the intricate nature of the problem being addressed. This allocation of data not only facilitates a more detailed learning set for the model but also ensures a robust testing framework to validate the model's efficacy in real-world scenarios.

Dataset Value Counts and Statistics

Before delving into the model creation and performance evaluation of the model, it's essential to conduct a thorough examination of the dataset's characteristics, including statistics and value counts. This preliminary analysis provides crucial insights into the dataset's composition and the

distribution of cases across training and testing sets. Specifically, the split between training and testing datasets results in value counts as follows: 1,209 cases of BTIs and 2,242 cases of UTIs are allocated for training. Conversely, for testing purposes, there are 302 BTI cases and 560 UTI cases. This distribution ensures a comprehensive representation of both conditions, facilitating a balanced learning environment for the model.

Additionally, a crucial part of this preliminary analysis is the examination of primary patient vitals, namely temperature and heart rate, which are pivotal in assessing the severity and nature of the conditions under study. The distributions of these vitals for each case category (BTIs and UTIs) are visualized below in Figure 8. These visual representations are instrumental in identifying any underlying patterns or anomalies in the data, such as outliers or skewed distributions, that could impact model training and performance. Additional data distributions are included in Appendix A.

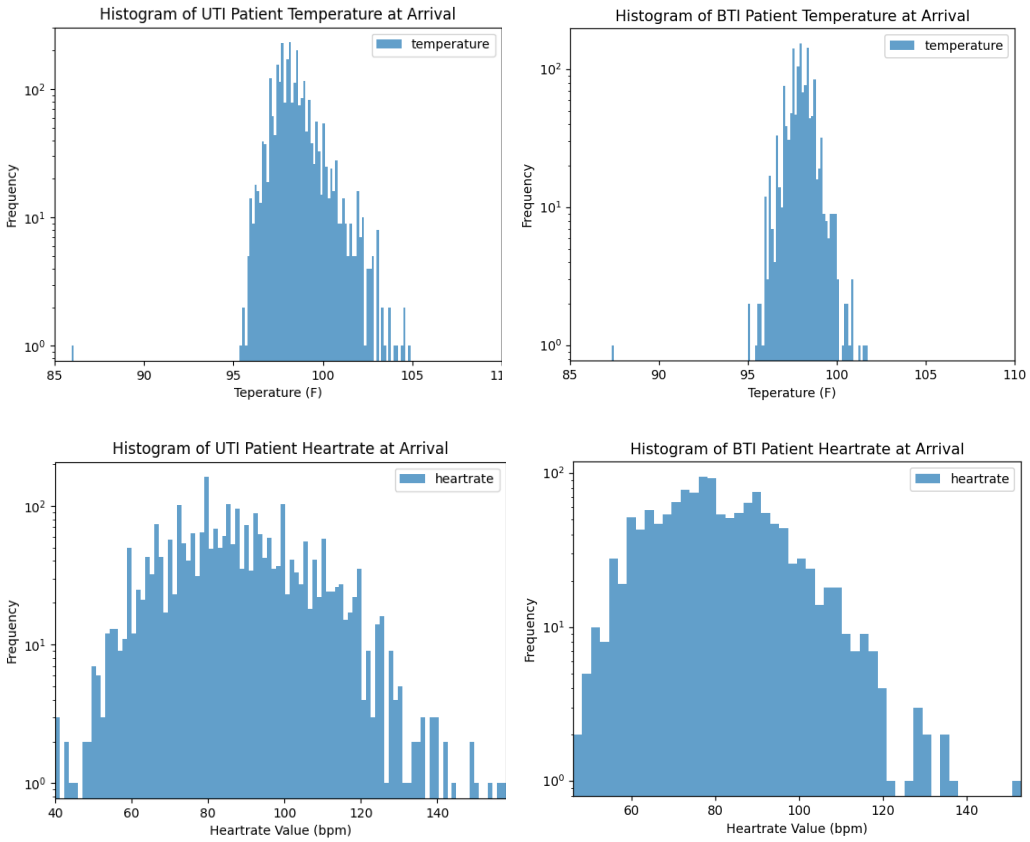

Figure 8: Shows the patient temperature and heart rate distributions as log-scale histograms for UTIs and BTIs

Model Testing and Performance

The testing dataset plays a crucial role in evaluating the model's performance and accuracy. Considering the size and diversity of the dataset, two machine learning algorithms were specifically chosen for this study: Random Forests and Decision Trees. These algorithms are particularly well-suited for the task due to their robust performance in handling large datasets. Furthermore, both Random Forests and Decision Trees excel in scenarios requiring multi-class classification, making them ideal for this application where the target variables involve multiple classes of medications. Their ability to manage the complexity and variability of the data ensures

a thorough assessment of the model's efficacy in predicting the correct medication administration based on patient features (Dittman et al, 2011).

The models were studied by a total of four different metrics: precision, recall, f1 score, and accuracy. The respective formulas for each are as follows (Kundu, 2022):

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives} \quad (1)$$

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives} \quad (2)$$

$$F1\ score = \frac{2 * Recall * Precision}{Recall + Precision} * 100 \quad (3)$$

$$Accuracy = \frac{True\ Positives + True\ Negatives}{Total\ Values} \quad (4)$$

By applying formulas 1 to 4 mentioned above, the results obtained are shown in the table below.

Table 4: Model prediction performance metrics for UTIs and BTIs

Condition	Random Forests (%) Precision/Recall/F1/Accuracy	Decision Trees (%) Precision/Recall/F1/Accuracy
UTI	92.59% / 89.68% / 90.57% / 83.69%	77.50% / 81.45% / 86.72% / 77.50%
BTI	92.34% / 89.88% / 90.62% / 83.73%	80.38% / 93.70% / 85.69% / 75.86%

As seen in the table, the increased number of estimators in the random forests model provides a better prediction accuracy of the medications administered. In multi-classification contexts, the diversity of estimators helps to cover a broader array of decision paths, capturing subtleties and variances across different classes more effectively than a single estimator could. Consequently, as the number of estimators increases, the ensemble's ability to generalize and accurately predict across a diverse set of classes improves, leading to enhanced performance on multi-class targets

such as medication types. This statistical strength makes Random Forests particularly valuable for datasets where the target variables involve multiple categories, as is common in medical or pharmacological applications (Kwon et al., 2018).

Nonetheless, the accuracy metrics derived from the table also account for the correctly predicted zeros within the dataset, corresponding to the absence of medication administration. While accurately predicting these zeros is important, as it reflects the model's capacity to recognize cases where no medication is warranted, this alone does not include an exhaustive measure of the model's precision across the various medication classes. To acquire a more granular view of the model's performance in predicting specific medications, further analysis was conducted. The model's accuracy for each medication class was plotted against the relative frequency of these medications within the testing dataset, deliberately excluding the zeros. This approach provides a more nuanced understanding of the model's predictive success rate for each individual medication, allowing for a precise evaluation of its efficacy in a clinical setting where such differentiation is critical. Figures 9 and 10 below show the discovered correlation for the random forests model.

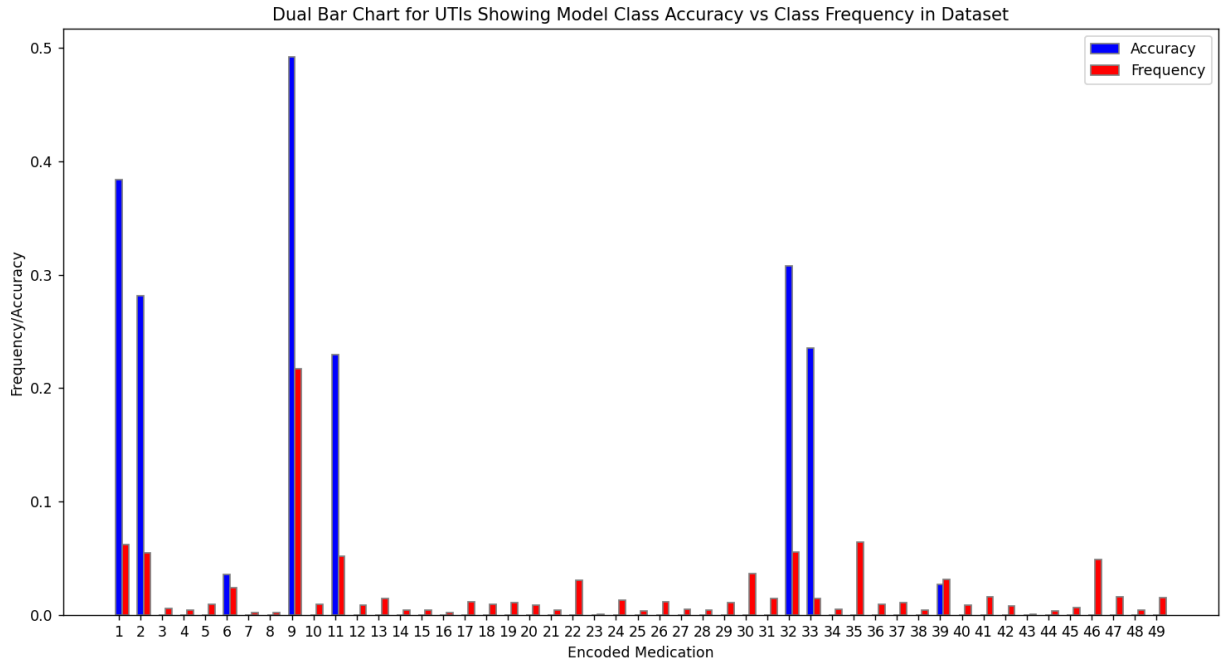

Figure 9: Dual bar chart representing model class prediction accuracy against the class relative frequency in the dataset for UTIs

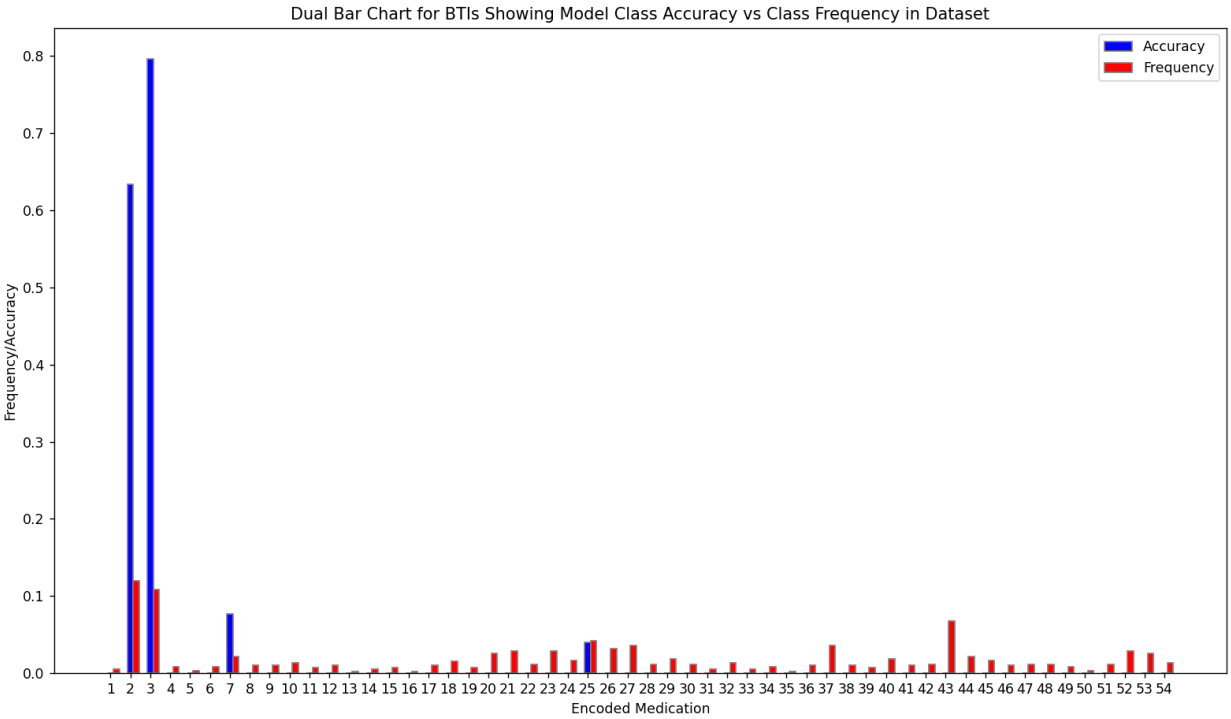

Figure 10: Dual bar chart representing model class prediction accuracy against the class relative frequency in the dataset for BTIs

Despite attempts to weight less frequently classes higher, it is imperative to acknowledge the influence of the persisting class imbalance on the model's performance. The analysis indicates a notable disparity in class frequencies, a common issue where majority classes dominate over minority ones, potentially leading to the underrepresentation and thus poor prediction of less frequent classes. However, upon examining the accuracy relative to frequency for each class, a non-linear relationship is evident. Therefore, the model's performance does not simply improve with the prevalence of a class; it also demonstrates the capability to accurately predict some of the less common classes, indicating a certain level of confidence.

Furthermore, the emergence of outliers in the accuracy versus frequency charts is particularly telling. Some classes with lower occurrence rates exhibit high predictive accuracy, suggesting that the model has effectively learned their distinctive features as specifically seen on medications numbered 6, 11, and 33 on Figure 9. On the flip side, certain more prevalent classes show lower predictive accuracy, signaling areas where the model may benefit from further training or feature engineering to enhance its understanding. The accuracy distribution across classes is also varied, with no clear pattern of accuracy increasing or decreasing with frequency. This variability can be attributed to several factors, including the inherent difficulty of the class, noise in the data, or model's ability to learn from the input features provided. Additionally, given the higher variability with how BTIs are usually diagnosed, it could prove to be more challenging for the model to accurately predict the correct medications.

The possibility of overfitting must also be considered, especially in classes that demonstrate exceptionally high accuracy and frequency, such as medications numbered 2 and 3 in Figure 10.

Since, the model has seen these medications occur significantly more than the rest, naturally it gravitates towards the majority class. Furthermore, it's essential to validate whether the model retains its performance on an independent test set to confirm that the high accuracy is not simply a reflection of the model's memorization of the training data. Lastly, the practice of stratified sampling or manually selecting training data could be a necessary methodological consideration. Without it, certain classes may be misrepresented in the training and validation process, leading to a skewed depiction of model accuracy. Stratification is crucial to ensure each class's proportional representation, fostering a more equitable and accurate model evaluation.

Moreover, the differences in predictive performance between the models for brain trauma injuries BTIs and UTIs could have medical reasons grounded in the nature of the conditions and the variables at play.

Traumatic brain injuries are complex and can involve various factors that may not be as prevalent or as quantifiable as those associated with UTIs. Predictive models for BTIs may struggle with minor classes due to the highly individualized nature of brain injuries, where factors like the exact location and severity of the injury, patient's age, and presence of other conditions can vary widely and influence outcomes significantly (Cleveland Clinic, no date). The predictive models might not capture all these nuances, especially when the training data does not represent the minor classes well enough.

Moreover, severe TBIs often involve a myriad of complications and interventions, which can be challenging to predict accurately due to their complexity and the smaller sample size of severe cases compared to more common, minor TBIs. Factors such as the need for intubation, blood transfusions during resuscitation, and the presence of in-hospital complications play crucial roles in patient outcomes and can significantly affect the model's performance (Khatri et al, 2021).

On the other hand, UTIs are generally less complex and more uniform in their presentation and treatment. This can make them easier to predict with machine learning models since the variables at play may be fewer and more consistent across different cases.

Additionally, after seeing how many classes have very low accuracies, the F1 score was once again calculated while excluding 0's to see the accuracy with respect to the model's precision and recall. Where we get the results in the table below.

Table 5: Model prediction accuracy for UTIs and BTIs after excluding 0's within the dataset

Condition	Random Forests F1 score	Decision Trees F1 score
UTI	34.59%	29.42%
BTI	41.44%	38.12%

Thus, it becomes evident that while the model exhibits a notable decline in performance scores, it retains the ability to determine the correct number of medications to be administered since the presence of 0's results in significantly higher performance metrics. However, the model encounters challenges in precisely identifying which specific medications need to be dispensed. This distinction underscores a critical area for improvement in the model's predictive capabilities, emphasizing the need to enhance its accuracy in discerning the appropriate medication types for effective clinical application.

Process Mining Workflows

Leveraging the machine learning model previously developed and the event log data, we are equipped to simulate patient flow within the ED, incorporating the dynamics of medication

administration. By employing the pm4py library, we can construct a sophisticated workflow that represents the patient journey from admission to discharge, visualized through the lenses of the medications administered. The choice of the alpha miner algorithm within pm4py is deliberate, due to its proficiency in navigating data with considerable noise—an inherent characteristic of the fast-paced and unpredictable ED environment.

When we isolate and model the cases where the medicine administration model achieves an accuracy of greater of equal to 90%, we unveil insightful patterns. The resulting two figures provide a visual narration of these patterns. They offer a detailed view into the decision-making process of the model, revealing how patients with different symptoms, and other variables follow specific medication regimes. This level of accuracy ensures that the workflow models generated are not only reliable but also actionable, potentially serving as a framework for optimizing medication flow and resource allocation in the ED.

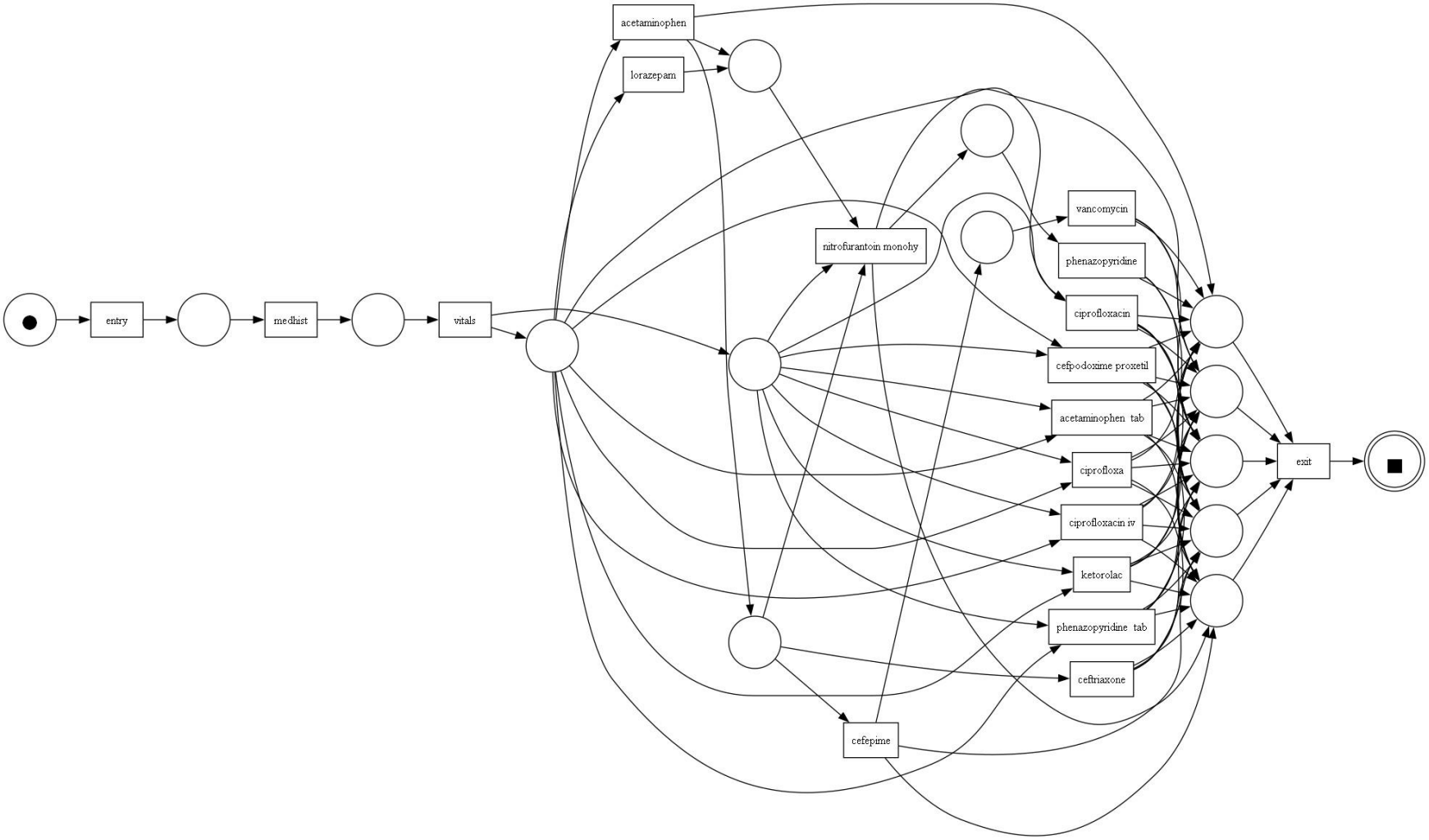

Figure 11: Process mining workflow for UTI cases where the ML model predicted medications with greater than or equal to 90% accuracy. (image produced using pm4py library on python)

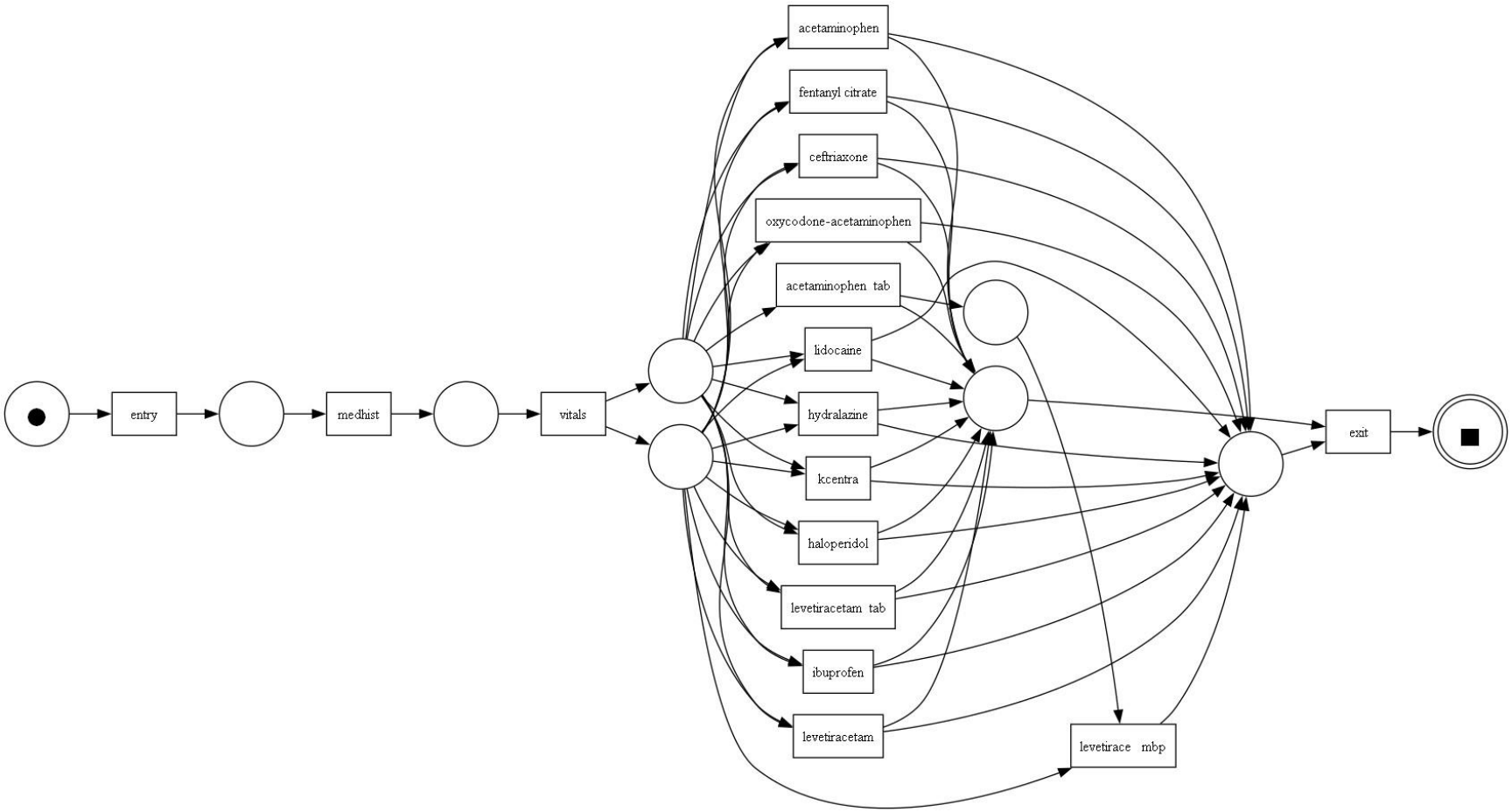

Figure 12: Process mining workflow for BTI cases where the ML model predicted medications with greater than or equal to 90% accuracy. (image produced using pm4py library on python)

The complexity of the process flows post-vitals measurement, as visualized in the provided diagrams, suggests that the machine learning model possesses the capability to predict a diverse array of medication pathways. This diversity is particularly pronounced in the case of UTIs, where the model reflects the clinical reality that many UTI treatments involve a sequence of follow-up medications. The intricacy of these treatment paths is captured and predicted by the model, highlighting its understanding of UTI management protocols.

In contrast to the UTI cases, the process flows for BTIs depicted by the model are markedly more simplified, suggesting a tendency towards predicting a limited variety of medication outcomes. This observation may stem from the multifaceted nature of BTI treatment protocols, which demand complex clinical judgments informed by a spectrum of patient-specific factors. Alternatively, it could indicate a prevailing clinical practice in which a more select range of medications is conventionally administered during the initial stages of ED response to BTIs (Cleveland Clinic, no date). Additionally, the possibility of model overfitting should be considered, particularly considering previous analyses that highlighted the possibility. The presence of overfitting could artificially narrow the scope of predicted medications, thereby simplifying the resultant process flows in a way that may not accurately reflect real-world scenarios.

In practical terms, the model's predictions, as seen in Figure 11, reveal potential efficiencies that could be harnessed in emergency care. For instance, the concurrent administration of medications like lorazepam and nitrofurantoin monohydrate suggests a pattern that could be anticipated in the ED. Identifying these predictive patterns facilitates pre-emptive strategies, which include pre-stocking medications for patients projected to require them upon assessment or implementing standing orders prior to patient arrival. Such measures refine the efficiency of

patient care upon their entry into the emergency department and ensure a proactive approach to medicinal supply management. This foresight not only expedites treatment delivery but also enhances the readiness of the medical inventory, optimizing the overall patient experience from triage to treatment.

Potential Errors and Study Limitations

The integrity and comprehensiveness of the data presented significant challenges for this study, primarily due to the necessary encryption imposed for patient privacy. Critical information such as patient age, exact visit dates, and patient count was either absent or encrypted, impeding the ability to make precise predictions. The omission of patient age is particularly impactful, as medication administration protocols can vary widely across different age groups.

Furthermore, a natural extension upon this research would involve predicting patient wait times within each stage of the ED and track how enhanced medicine administration could lower overall stay times. Yet, such predictions would require direct insights into ED congestion levels, which in turn depend on the number of patients present, which can be calculated using the exact dates and times of patient visits. Without access to precise datetime data, the ability to accurately gauge ED occupancy and thus predict wait times is significantly compromised. Furthermore, the unavailability of data on staff and equipment allocation further limits our study. Any delays in patient care due to insufficient resources cannot be captured by our model, yet they are often crucial factors influencing ED wait times (Marshall et al, 2023).

For conditions such as UTIs, the predictive accuracy of the medication administration model could be markedly improved with access to patient urine sample data, which is a critical diagnostic tool. The absence of such data in our dataset restricts the model's capacity to identify

underlying causes and appropriate medication responses (Mayo Clinic, no date). Moreover, translating these findings into real-world applications within EDs is constrained by rigorous legal frameworks and the high standards set for machine learning models in healthcare settings. The current model would require substantial enhancement and a richer data pool to fulfill the accuracy and regulatory requirements (Farhud & Zokaei, 2021).

Lastly, our analysis may contain inherent errors stemming from discrepancies in the dataset. Medical practitioners prioritize patient care and safety, which can sometimes lead to lapses in data collection. Such gaps or inconsistencies in the data, though unintentional, have the potential to introduce inaccuracies in the model's predictions, a factor that must be carefully considered when interpreting the results (Johnson et al, 2023).

Conclusion

In conclusion, this study successfully aggregated and formatted relevant data into coherent event logs for patients with UTIs and BTIs. Additionally, a ML model was developed and trained to predict medication administration. This model was further utilized to generate valuable insights through process mining and flow mapping techniques, highlighting its utility in understanding patient care dynamics.

However, the model's effectiveness is curtailed by suboptimal accuracy and performance scores, which casts doubt on its reliability for precise medication prediction in clinical settings. This limitation underscores a critical need for enhanced data quality and additional patient-specific information. Specifically, incorporating more detailed patient demographics such as age, or clinical data such as urine samples, could significantly improve the model's predictive power. With these improvements, there is a strong potential to not only boost the model's accuracy but

also to increase its applicability and trustworthiness in real-world medical settings, thereby enhancing patient care through more tailored and timely medication administration.

References

Conca1, T., Saint-Pierre1, C., Herskovic1, V., Sepúlveda1, M., Capurro2, D., Prieto3, F., Fernandez-Llatas4, C., Department, 1Computer Science, & Sepúlveda, C. A. (no date). *Multidisciplinary collaboration in the treatment of patients with type 2 diabetes in primary care: Analysis using process mining*. Journal of Medical Internet Research. Available at: https://www.jmir.org/2018/4/e127/ (Accessed: 15 October 2023)

Davenport, T. H., & Spanyi, A. (2019). *What process mining is, and why companies should do it*. Harvard Business Review. Available at: https://hbr.org/2019/04/what-process-mining-is-and-why-companies-should-do-it (Accessed: 07 November 2023)

De Roock, E., & Martin, N. (2022). *Process mining in healthcare – an updated perspective on the state of the art*. Journal of Biomedical Informatics. Available at: https://www.sciencedirect.com/science/article/pii/S1532046422000119 (Accessed: 07 November 2023)

Gatta, R., Vallati, M., Fernandez-Llatas, C., Martinez-Millana, A., Orini, S., Sacchi, L., Lenkowicz, J., Marcos, M., Munoz-Gama, J., Cuendet, M. A., de Bari, B., Marco-Ruiz, L., Stefanini, A., Valero-Ramon, Z., Michielin, O., Lapinskas, T., Montvila, A., Martin, N., Tavazzi, E., & Castellano, M. (2020). *What role can process mining play in recurrent clinical guidelines issues? A position paper*. MDPI. Available at: https://www.mdpi.com/1660-4601/17/18/6616 (Accessed: 27 November 2023)

Jarvis, P. R. E. (2016). *Improving emergency department patient flow*. Clinical and experimental emergency medicine. Available at: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5051606/ (Accessed: 03 November 2023)

Martin, N., Weerdt, J., Gatta, R., Marco-Ruiz, L., Gal, A., & Mertens, S. (2020). *Recommendations for enhancing the usability and understandability of process mining in Healthcare*. Artificial Intelligence in Medicine. Available at: https://www.sciencedirect.com/science/article/pii/S0933365720312276 (Accessed: 02 December 2023)

Martinez, A., Palao, C., & Fernandez, C. (no date). Integrated IOT intelligent system for the automatic ... - IEEE xplore. Available at: https://ieeexplore.ieee.org/document/8513638 (Accessed: 22 October 2023)

Martinez, A., Romero, A., Henriques, J., Bianchi, A., Fernandez, C., & Carvalho, P. (no date). *SCITEPRESS*. SciTePress. Available at:

https://www.scitepress.org/PublicationsDetail.aspx?ID=8u1xR%2BepQBY&t=1 (Accessed: 07 November 2023)

Martinez-Millana, A., Hulst, J. M., Boon, M., Witters, P., Fernandez-Llatas, C., Asseiceira, I., Calvo-Lerma, J., Basagoiti, I., Traver, V., Boeck, K. D., & Ribes-Koninckx, C. (no date). *Optimisation of children Z-score calculation based on New Statistical Techniques*. PLOS ONE. Available at: https://journals.plos.org/plosone/article?id=10.1371%2Fjournal.pone.0208362 (Accessed: 22 November 2023)

Munoz-Gama, J., Martin, N., Fernandez-Llatas, C., Johnson, O., Sepulveda, M., Galves-Yanjari, V., Rojas, E., Helm, E., Martinez-Milana, A., Aloini, D., Amantea, I., Andrews, R., Arias, M., & Beerepoot, I. (2022). *Process mining for healthcare: Characteristics and challenges*. Journal of Biomedical Informatics. Available at: https://www.sciencedirect.com/science/article/pii/S1532046422000107 (Accessed: 02 December 2023)

Rojas, E., Munoz-Gama, J., Sepulveda, M., & Capurro, D. (2016). *Process mining in Healthcare: A literature review*. Journal of Biomedical Informatics. Available at: https://www.sciencedirect.com/science/article/pii/S1532046416300296 (Accessed: 29 November 2023)

What is process mining?: Celonis: Process mining software. Celonis. (no date). Available at: https://www.celonis.com/process-mining/what-is-process-mining/ (Accessed: 02 December 2023)

Tator, C.H. (2013) *Concussions and their consequences: Current diagnosis, management and prevention*, *CMAJ : Canadian Medical Association journal = journal de l'Association medicale canadienne*. Available at: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3735746/ (Accessed: 07 April 2024).

Lithium (oral route) side effects (2020) *Mayo Clinic*. Available at: https://www.mayoclinic.org/drugs-supplements/lithium-oral-route/side-effects/drg-20064603?p=1#:~:text=Descriptions,and%20severity%20of%20manic%20episodes. (Accessed: 09 April 2024).

professional, C.C. medical (no date) *TBI*, *Cleveland Clinic*. Available at: https://my.clevelandclinic.org/health/diseases/8874-traumatic-brain-injury (Accessed: 13 April 2024).

Khatri N;Sumadhura B;Kumar S;Kaundal RK;Sharma S;Datusalia AK; (2021) *The complexity of secondary cascade consequent to traumatic brain injury: Pathobiology and potential treatments*, *Current neuropharmacology*. Available at: https://pubmed.ncbi.nlm.nih.gov/33588734/#:~:text=Patients%20who%20suffer%20TBI%20face,injuries%20have%20their%20own%20consequences. (Accessed: 13 April 2024).

Insight: Why are patients spending 22 hours in the emergency room waiting for a hospital bed? (no date) *Canadian Medical Association*. Available at: https://www.cma.ca/latest-stories/insight-why-are-patients-spending-22-hours-emergency-room-waiting-hospital-bed (Accessed: 24 October 2023).

Breiman, L. (1984) *Classification and Regression Trees*. 1st edn. Routledge.

Matsuki, K., Kuperman, V. and Van Dyke, J.A. (2016) *The Random Forests Statistical Technique: An examination of its value for the study of reading, Scientific studies of reading : the official journal of the Society for the Scientific Study of Reading*. Available at: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4710485/ (Accessed: 10 February 2024).

Lee SY;Chinnam RB;Dalkiran E;Krupp S;Nauss M; (2019) *Prediction of emergency department patient disposition decision for proactive resource allocation for admission, Health care management science*. Available at: https://pubmed.ncbi.nlm.nih.gov/31444660/ (Accessed: 29 October 2023).

Kwon JM;Lee Y;Lee Y;Lee S;Park J; (2018) *An algorithm based on Deep Learning for predicting in-hospital cardiac arrest, Journal of the American Heart Association*. Available at: https://pubmed.ncbi.nlm.nih.gov/29945914/ (Accessed: 05 April 2024).

Machine Learning Algorithms (2015) *packt*. Available at: https://subscription.packtpub.com/book/data/9781785889622/3/ch03lvl1sec21/creating-training-and-test-sets (Accessed: 05 February 2024).

Marshall, E.G., Miller, L. and Moritz, L.R. (2023) *Challenges and impacts from wait times for specialist care identified by Primary Care Providers: Results from the MAAP study cross-sectional survey, Healthcare management forum*. Available at: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10448708/ (Accessed: 01 March 2024).

Mayo Clinic Top-ranked hospital in the nation (no date) *Mayo Clinic*. Available at: https://www.mayoclinic.org/ (Accessed: 29 September 2023).

Farhud, D.D. and Zokaei, S. (2021) *Ethical issues of Artificial Intelligence in medicine and Healthcare, Iranian journal of public health*. Available at: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8826344/ (Accessed: 20 March 2024).

Johnson, A., Bulgarelli, L., Pollard, T., Celi, L. A., Mark, R., & Horng, S. (2023). *Mimic-IV-ED*. MIMIC-IV-ED v2.2. Available at: https://physionet.org/content/mimic-iv-ed/2.2/ (Accessed: 25 October 2023)

Dittman, D., Khoshgoftaar, T., Wald, R., Napolitano A., (2011) *Random forest: A reliable tool for patient response prediction, IEEE Xplore*. Available at: https://ieeexplore.ieee.org/abstract/document/6112389?casa_token=NufIquQZHLsAAAAA:eZjhutxfSl__xOmdH7IDEXpdT39dgkYmLy7bn3hahPbp-xIHrPcnxLk5pXoaxFzA12G_MxOV4g (Accessed: 16 February 2024).

Kundu, R. (2022) *F1 score in Machine Learning: Intro & Calculation*, *V7*. Available at: https://www.v7labs.com/blog/f1-score-guide (Accessed: 23 February 2024).

dev, Mr. (2020) *ML(machine learning) and it's models*, *Medium*. Available at: https://medium.com/@devendra.bhumca2014/ml-machine-learning-and-its-models-f44dcc5ae306 (Accessed: 10 April 2024).

Appendix A

	A	B	C	D	E	
1	subject_id	stay_id	seq_num	icd_code	icd_version	icd_title
2	10000032	32952584	1	4589	9	HYPOTENSION NOS
3	10000032	32952584	2	7070	9	UNSPECIFIED VIRAL HEPATITIS C WITHOUT HEPATIC COMA
4	10000032	32952584	3	V08	9	ASYMPTOMATIC HIV INFECTION
5	10000032	33258284	1	5728	9	OTH SEQUELA, CHR LIV DIS
6	10000032	33258284	2	78959	9	OTHER ASCITES
7	10000032	33258284	3	7070	9	UNSPECIFIED VIRAL HEPATITIS C WITHOUT HEPATIC COMA
8	10000032	33258284	4	V08	9	ASYMPTOMATIC HIV INFECTION
9	10000032	35968195	1	5715	9	CIRRHOSIS OF LIVER NOS
10	10000032	35968195	2	78900	9	ABDOMINAL PAIN UNSPEC SITE
11	10000032	35968195	3	V08	9	ASYMPTOMATIC HIV INFECTION
12	10000032	38112554	1	78959	9	OTHER ASCITES
13	10000032	38112554	2	7070	9	UNSPECIFIED VIRAL HEPATITIS C WITHOUT HEPATIC COMA
14	10000032	38112554	3	5715	9	CIRRHOSIS OF LIVER NOS
15	10000032	38112554	4	V08	9	ASYMPTOMATIC HIV INFECTION
16	10000032	39399961	1	78097	9	ALTERED MENTAL STATUS
17	10000032	39399961	2	34830	9	ENCEPHALOPATHY, UNSPECIFIED
18	10000084	35203156	1	R531	10	Weakness
19	10000084	35203156	2	G20	10	Parkinson's disease
20	10000084	36954971	1	R4182	10	Altered mental status, unspecified
21	10000084	36954971	2	F0390	10	Unspecified dementia without behavioral disturbance
22	10000108	32522732	1	7822	9	LOCAL SUPRFICIAL SWELLNG
23	10000108	36533795	1	5283	9	CELLULITIS/ABSCESS MOUTH
24	10000108	39513268	1	5283	9	CELLULITIS/ABSCESS MOUTH
25	10000115	30295111	1	S0181XD	10	Laceration w/o foreign body of oth part of head, subs encntr
26	10000115	30295111	2	V00131D	10	Fall from skateboard, subsequent encounter
27	10000115	38081480	1	S025XXA	10	Fracture of tooth (traumatic), init for clos fx
28	10000115	38081480	2	S0181XA	10	Laceration w/o foreign body of oth part of head, init encntr
29	10000115	38081480	3	W1789XA	10	Other fall from one level to another, initial encounter
30	10000117	30632130	1	S098XXA	10	Other specified injuries of head, initial encounter
31	10000117	30632130	2	S01111A	10	Laceration w/o fb of right eyelid and periocular area, init
32	10000117	30632130	3	W228XXA	10	Striking against or struck by other objects, init encntr
33	10000117	32642808	1	R1310	10	Dysphagia, unspecified
34	10000117	33176849	1	S72092A	10	Oth fracture of head and neck of left femur, init
35	10000117	33176849	2	W1830XA	10	Fall on same level, unspecified, initial encounter
36	10000178	31721172	1	M109	10	Gout, unspecified

Dataset A-1: Mimic IV Sample Data - Diagnosis Dataset

	A	B	C	D	E	F	G	H	I	J	
1	subject_id	stay_id	temperature	heartrate	resprate	o2sat	sbp	dbp	pain	acuity	chiefcomplaint
2	10000032	32952584	97.8	87	14	97	71	43	7	2	Hypotension
3	10000032	33258284	98.4	70	16	97	106	63	0	3	Abd pain, Abdominal distention
4	10000032	35968195	99.4	105	18	96	106	57	10	3	n/v/d, Abd pain
5	10000032	38112554	98.9	88	18	97	116	88	10	3	Abdominal distention
6	10000032	39399961	98.7	77	16	98	96	50	13	2	Abdominal distention, Abd pain, LETHAGIC
7	10000084	35203156	97.5	78	16	100	114	71	0	2	Confusion, Hallucinations
8	10000084	36954971	98.7	80	16	95	111	72	0	2	Altered mental status, B Pedal edema
9	10000108	32522732	98.21	83	20	100	112	81	5	3	L CHEEK ABSCESS
10	10000108	36533795	98.8	98	16	100	135	85	5	3	LEFT CHEEK SWELLING, Abscess
11	10000108	39513268	98.8	101	18	100	131	62	5	3	L FACIAL SWELLING
12	10000115	30295111	98.4	60	16	98	143	74	0	4	Suture removal
13	10000115	38081480	97.6	81	18	99	120	71	5	3	Laceration, s/p Fall
14	10000117	30632130	97.1	119	16	99	100	55	7	3	Head injury
15	10000117	32642808	97.6	81	16	100	148	83	0	3	Throat foreign body sensation
16	10000117	33176849	97.5	104	16	99	130	71	7	4	L Hip pain
17	10000178	31721172	98.5	82	16	100	167	86	8	3	R Foot pain
18	10000248	35106839	99.2	87	18	100	124	75	1	3	ANEMIA S/P FALL
19	10000285	36555703	99	100	16	96	142	83	5	3	Abd pain
20	10000285	37909301	98	80	19	100	141	99	5	3	LUQ abd pain
21	10000473	33267868	101.1	87	20	95	133	63	3	3	ILI
22	10000492	34094600	97.2	92	18	98	132	79	6	3	Vomiting
23	10000507	31021946	97.4	81	18	100	159	98	0	3	ETOH
24	10000526	31939255	97	83	18	100	127	80	0	4	ILI
25	10000635	31767754	97.1	56	16	100	139	67	0	2	Dizziness
26	10000635	38135254	98.2	56	16	100	145	76	6	3	L Shoulder pain, Visual changes
27	10000764	35420907	98.9	92	18	100	140	77	3	2	s/p Fall
28	10000891	37006387	98	81	18	98	155	89	2	3	Abnormal labs, Calf pain
29	10000898	30017875	97.7	88	16	94	140	77	2	3	Chest pain
30	10000935	32845079	97.6	117	18	95	128	74	10	3	WEAKNESS
31	10000935	35197384	97.8	120	20	98	121	77	0	3	VOMITING AND/OR NAUSEA
32	10000935	37059738	98.4	104	16	96	113	59	0	3	L LEG SWELLING
33	10000935	37290428	98.1	90	18	100	128	82	0	3	"HEAD IS NUMB"
34	10000935	37499077	98	116	24	99	100	49	0	2	DYSPNEA
35	10000951	35942449	97.5	89	16	100	101	57	5	3	R Ankle pain
36	10000980	30449161	94.8	57	18	98	151	84	0	2	Hypoglycemia

Dataset A-2: Mimic IV Sample Data - Triage Dataset

	A	B	C	D	E	F	G	H	I	J	K
1	subject_id	stay_id	charttime	temperature	heartrate	resprate	o2sat	sbp	dbp	rhythm	pain
2	10000032	32952584	2180-07-22 16:36		83	24	97	90	51		0
3	10000032	32952584	2180-07-22 16:43		85	22	98	76	39		0
4	10000032	32952584	2180-07-22 16:45		84	22	97	75	39		0
5	10000032	32952584	2180-07-22 17:56		84	20	99	86	51		
6	10000032	32952584	2180-07-22 18:37	98.4	86	20	98	65	37		
7	10000032	32952584	2180-07-22 18:38		85	16	99	83	45		0
8	10000032	32952584	2180-07-22 19:47	98.2	85	18	98	81	38		0
9	10000032	33258284	2180-05-06 23:04	97.7	79	16	98	107	60		0
10	10000032	35968195	2180-08-05 23:50	98.5	96	17	100	102	58		
11	10000032	35968195	2180-08-06 1:07	98.1	91	18	99	98	60		
12	10000032	38112554	2180-06-26 18:42	97.9	76	18	95	95	64		5
13	10000032	38112554	2180-06-26 20:54	97.9	86	17	93	96	57		
14	10000032	39399961	2180-07-23 7:19		79	18	93	86	57		asleep
15	10000032	39399961	2180-07-23 7:20		78	18	95	88	57		unable
16	10000032	39399961	2180-07-23 8:30		78	18	94	82	55		uta
17	10000032	39399961	2180-07-23 9:02		94	16	95	73	51		unable
18	10000032	39399961	2180-07-23 9:28	98.1	86	18	94	93	54		uta
19	10000032	39399961	2180-07-23 10:01		81	18	92	82	54		uta
20	10000032	39399961	2180-07-23 10:16		83	16	92	76	48		uta
21	10000032	39399961	2180-07-23 10:46		83	14	96	93	46		u
22	10000032	39399961	2180-07-23 11:35		92	16	95	83	49		asleep
23	10000032	39399961	2180-07-23 11:51	99	93	16	95	84	40		asleep
24	10000032	39399961	2180-07-23 12:54		96	18	97	86	45		asleep
25	10000084	35203156	2160-11-20 20:39	97.5	78	16	100	114	71		0
26	10000084	35203156	2160-11-21 0:46	98	67	18	95	137	76		0
27	10000084	35203156	2160-11-21 2:35	98	68	16	96	103	74		0
28	10000084	36954971	2160-12-27 18:34	98.7	80	16	95	111	72		0
29	10000084	36954971	2160-12-27 21:04	98.5	70	18	98	109	69		
30	10000084	36954971	2160-12-27 23:45	97.7	77	18	100	120	70		
31	10000084	36954971	2160-12-28 6:03	97.7	73	16	100	119	72		
32	10000084	36954971	2160-12-28 6:47		78	16	100	121	76		
33	10000084	36954971	2160-12-28 6:48		77	18	100	129	76		
34	10000084	36954971	2160-12-28 11:34	98.5	74	16	96	108	67		
35	10000084	36954971	2160-12-28 14:30	98.7	80	16	100	119	84		0
36	10000108	32522732	2163-09-16 17:56	98.2	66	15	100	111	62		3

Dataset A-3: Mimic IV Sample Data - Vitals Dataset

	A	B	C	D	E	F	G	H	I
1	subject_id	hadm_id	stay_id	intime	outtime	gender	race	arrival_transport	disposition
2	10000032	22595853	33258284	2180-05-06 19:17	2180-05-06 23:30	F	WHITE	AMBULANCE	ADMITTED
3	10000032	22841357	38112554	2180-06-26 15:54	2180-06-26 21:31	F	WHITE	AMBULANCE	ADMITTED
4	10000032	25742920	35968195	2180-08-05 20:58	2180-08-06 1:44	F	WHITE	AMBULANCE	ADMITTED
5	10000032	29079034	32952584	2180-07-22 16:24	2180-07-23 5:54	F	WHITE	AMBULANCE	HOME
6	10000032	29079034	39399961	2180-07-23 5:54	2180-07-23 14:00	F	WHITE	AMBULANCE	ADMITTED
7	10000084	23052089	35203156	2160-11-20 20:36	2160-11-21 3:20	M	WHITE	WALK IN	ADMITTED
8	10000084	29888819	36954971	2160-12-27 18:32	2160-12-28 16:07	M	WHITE	AMBULANCE	HOME
9	10000108	27250926	36533795	2163-09-27 16:18	2163-09-28 9:04	M	WHITE	WALK IN	HOME
10	10000108		32522732	2163-09-16 16:34	2163-09-16 18:13	M	WHITE	WALK IN	HOME
11	10000108		39513268	2163-09-24 16:14	2163-09-24 21:02	M	WHITE	WALK IN	HOME
12	10000115		30295111	2154-12-17 16:37	2154-12-17 16:59	M	WHITE	WALK IN	HOME
13	10000115		38081480	2154-12-10 2:04	2154-12-10 5:59	M	WHITE	WALK IN	HOME
14	10000117	22927623	32642808	2181-11-14 21:51	2181-11-15 2:06	F	WHITE	WALK IN	ADMITTED
15	10000117	27988844	33176849	2183-09-18 8:41	2183-09-18 20:20	F	WHITE	WALK IN	ADMITTED
16	10000117		30632130	2183-07-17 10:30	2183-07-17 11:31	F	WHITE	WALK IN	HOME
17	10000178		31721172	2157-04-08 9:58	2157-04-08 15:30	F	ASIAN	WALK IN	HOME
18	10000248	20600184	35106839	2192-11-29 18:44	2192-11-30 3:32	M	WHITE	WALK IN	ADMITTED
19	10000285		36555703	2161-11-08 14:19	2161-11-08 21:06	M	OTHER	WALK IN	HOME
20	10000285		37909301	2159-11-26 14:22	2159-11-26 19:17	M	OTHER	WALK IN	HOME
21	10000473		33267868	2138-03-15 20:07	2138-03-16 4:04	M	ASIAN - SOUTH EAST ASIAN	WALK IN	HOME
22	10000492		34094600	2129-08-04 12:35	2129-08-04 16:26	M	WHITE	WALK IN	HOME
23	10000507		31021946	2151-07-02 3:31	2151-07-02 6:45	M	WHITE	AMBULANCE	HOME
24	10000526		31939255	2160-06-12 19:02	2160-06-12 20:33	F	UNKNOWN	UNKNOWN	LEFT WITHOUT BEING SEEN
25	10000635		31767754	2138-09-29 10:54	2138-09-29 16:53	F	BLACK/AFRICAN AMERICAN	WALK IN	HOME
26	10000635		38135254	2141-08-15 11:32	2141-08-15 17:06	F	BLACK/AFRICAN AMERICAN	WALK IN	HOME
27	10000764	27897940	35420907	2132-10-14 19:31	2132-10-14 23:32	M	WHITE	AMBULANCE	ADMITTED
28	10000891		37006387	2143-05-23 17:58	2143-05-23 23:30	F	WHITE	WALK IN	HOME
29	10000898		30017875	2188-03-13 13:32	2188-03-13 20:44	F	WHITE	WALK IN	HOME
30	10000935	21738619	35197384	2187-07-11 9:26	2187-07-11 13:07	F	BLACK/AFRICAN AMERICAN	AMBULANCE	ADMITTED
31	10000935	25849114	37499077	2187-10-10 11:56	2187-10-10 19:09	F	BLACK/AFRICAN AMERICAN	AMBULANCE	ADMITTED
32	10000935	26381316	32845079	2187-08-23 14:37	2187-08-23 22:46	F	BLACK/AFRICAN AMERICAN	AMBULANCE	ADMITTED
33	10000935		37059738	2187-09-26 14:48	2187-09-26 17:50	F	BLACK/AFRICAN AMERICAN	AMBULANCE	HOME
34	10000935		37290428	2186-05-27 7:26	2186-05-27 13:13	F	BLACK/AFRICAN AMERICAN	WALK IN	HOME
35	10000951		35942449	2155-09-30 20:15	2155-09-30 23:16	F	WHITE	WALK IN	HOME
36	10000980	20897796	31236252	2193-08-14 21:25	2193-08-15 2:22	F	BLACK/AFRICAN AMERICAN	WALK IN	ADMITTED

Dataset A-4: Mimic IV Sample Data – Patient Entry/Exit Dataset

	A	B	C	D	E	F	G	H	I
1	subject_id	stay_id	charttime	name	gsn	ndc	etc_rn	etccode	etcdescription
2	10000032	32952584	2180-07-22 17:26	albuterol sulfate	28090	21695042308	1	5970	Asthma/COPD Therapy - Beta 2-Adrenergic Agents, Inhaled, Short Acting
3	10000032	32952584	2180-07-22 17:26	calcium carbonate	1340	10135021101	1	733	Minerals and Electrolytes - Calcium Replacement
4	10000032	32952584	2180-07-22 17:26	cholecalciferol (vitamin D3)	65241	37205024678	1	670	Vitamins - D Derivatives
5	10000032	32952584	2180-07-22 17:26	emtricitabine-tenofovir [Truvada]	57883	35356007003	1	5849	Antiretroviral - Nucleoside and Nucleotide Analog RTIs Combinations
6	10000032	32952584	2180-07-22 17:26	fluticasone [Flovent HFA]	21251	49999061401	1	371	Asthma Therapy - Inhaled Corticosteroids (Glucocorticoids)
7	10000032	32952584	2180-07-22 17:26	furosemide	8209	10544058430	1	250	Diuretic - Loop
8	10000032	32952584	2180-07-22 17:26	lactulose	3143	11845042013	1	478	Colonic Acidifier (Ammonia Inhibitor)
9	10000032	32952584	2180-07-22 17:26	nicotine	16426	10939070733	1	5604	Smoking Deterrents - Nicotine-Type
10	10000032	32952584	2180-07-22 17:26	peg 3350-electrolytes	62533	10572010001	1	2889	Laxative - Saline/Osmotic Mixtures
11	10000032	32952584	2180-07-22 17:26	raltegravir [Isentress]	63231	35356011006	1	6072	Antiretroviral - HIV-1 Integrase Strand Transfer Inhibitors
12	10000032	32952584	2180-07-22 17:26	rifaximin [Xifaxan]	66295	54868615400	1	5844	Rifamycins and Related Derivative Antibiotics
13	10000032	32952584	2180-07-22 17:26	sulfamethoxazole-trimethoprim	9396	10544094020	1	2656	Antibacterial Folate Antagonist - Other Combinations
14	10000032	32952584	2180-07-22 17:26	tiotropium bromide [Spiriva with HandiHaler]	50714	35356021530	1	6461	Asthma/COPD - Anticholinergic Agents, Inhaled Long Acting
15	10000032	32952584	2180-07-22 17:26	tramadol	23139	10135051901	1	583	Analgesic Opioid Agonists
16	10000032	33258284	2180-05-06 21:39	albuterol sulfate	28090	21695042308	1	5970	Asthma/COPD Therapy - Beta 2-Adrenergic Agents, Inhaled, Short Acting
17	10000032	33258284	2180-05-06 21:39	emtricitabine-tenofovir [Truvada]	57883	35356007003	1	5849	Antiretroviral - Nucleoside and Nucleotide Analog RTIs Combinations
18	10000032	33258284	2180-05-06 21:39	ergocalciferol (vitamin D2)	2169	17236064401	1	670	Vitamins - D Derivatives
19	10000032	33258284	2180-05-06 21:39	furosemide	8208	10544013130	1	250	Diuretic - Loop
20	10000032	33258284	2180-05-06 21:39	ipratropium bromide [Atrovent HFA]	59081	12280037213	1	6460	Asthma/COPD - Anticholinergic Agents, Inhaled Short Acting
21	10000032	33258284	2180-05-06 21:39	nicotine	16426	10939070733	1	5604	Smoking Deterrents - Nicotine-Type
22	10000032	33258284	2180-05-06 21:39	peg 3350-electrolytes	62533	10572010001	1	2889	Laxative - Saline/Osmotic Mixtures
23	10000032	33258284	2180-05-06 21:39	raltegravir [Isentress]	63231	35356011006	1	6072	Antiretroviral - HIV-1 Integrase Strand Transfer Inhibitors
24	10000032	33258284	2180-05-06 21:39	spironolactone [Aldactone]	6818	16729022601	1	5658	Diuretic - Aldosterone Receptor Antagonist, Non-selective
25	10000032	33258284	2180-05-06 21:39	spironolactone [Aldactone]	6818	16729022601	2	6043	Aldosterone Receptor Antagonists
26	10000032	35968195	2180-08-05 22:11	emtricitabine-tenofovir [Truvada]	57883	35356007003	1	5849	Antiretroviral - Nucleoside and Nucleotide Analog RTIs Combinations
27	10000032	35968195	2180-08-05 22:11	lactulose	3143	11845042013	1	478	Colonic Acidifier (Ammonia Inhibitor)
28	10000032	35968195	2180-08-05 22:11	nicotine	16426	10939070733	1	5604	Smoking Deterrents - Nicotine-Type
29	10000032	35968195	2180-08-05 22:11	raltegravir [Isentress]	63231	35356011006	1	6072	Antiretroviral - HIV-1 Integrase Strand Transfer Inhibitors
30	10000032	35968195	2180-08-05 22:11	rifaximin [Xifaxan]	66295	54868615400	1	5844	Rifamycins and Related Derivative Antibiotics
31	10000032	35968195	2180-08-05 22:11	tiotropium bromide [Spiriva with HandiHaler]	50714	35356021530	1	6461	Asthma/COPD - Anticholinergic Agents, Inhaled Long Acting
32	10000032	35968195	2180-08-05 22:11	tramadol	23139	10135051901	1	583	Analgesic Opioid Agonists
33	10000032	38112554	2180-06-26 19:14	albuterol sulfate	28090	21695042308	1	5970	Asthma/COPD Therapy - Beta 2-Adrenergic Agents, Inhaled, Short Acting
34	10000032	38112554	2180-06-26 19:14	calcium carbonate	1340	10135021101	1	733	Minerals and Electrolytes - Calcium Replacement
35	10000032	38112554	2180-06-26 19:14	emtricitabine-tenofovir [Truvada]	57883	35356007003	1	5849	Antiretroviral - Nucleoside and Nucleotide Analog RTIs Combinations
36	10000032	38112554	2180-06-26 19:14	furosemide	8209	10544058430	1	250	Diuretic - Loop

Dataset A-5: Mimic IV Sample Data – Medication History Dataset

	A	B	C	D	E	F	G
1	subject_id	stay_id	charttime	med_rn	name	gsn_rn	gsn
2	10000032	32952584	2180-07-22 17:59	1	Albuterol Inhaler	1	5037
3	10000032	32952584	2180-07-22 17:59	1	Albuterol Inhaler	2	28090
4	10000032	35968195	2180-08-05 22:29	1	Morphine	1	4080
5	10000032	35968195	2180-08-05 22:55	2	Donnatol (Elixir)	1	4773
6	10000032	35968195	2180-08-05 22:55	3	Aluminum-Magnesium Hydrox.-Simet	1	2701
7	10000032	35968195	2180-08-05 22:55	3	Aluminum-Magnesium Hydrox.-Simet	2	2716
8	10000032	35968195	2180-08-05 22:55	4	Ondansetron	1	15869
9	10000032	35968195	2180-08-05 22:55	4	Ondansetron	2	61716
10	10000032	38112554	2180-06-26 16:58	1	Morphine	1	4080
11	10000032	38112554	2180-06-26 16:59	2	Ondansetron	1	15869
12	10000032	38112554	2180-06-26 16:59	2	Ondansetron	2	61716
13	10000032	39399961	2180-07-23 8:11	1	Lactulose	1	3067
14	10000032	39399961	2180-07-23 8:11	1	Lactulose	2	29054
15	10000032	39399961	2180-07-23 8:11	1	Lactulose	3	68217
16	10000032	39399961	2180-07-23 8:12	2	CeftriaXONE (Mini Bag Plus)	1	
17	10000032	39399961	2180-07-23 8:32	3	Calcium Gluconate	1	1356
18	10000032	39399961	2180-07-23 8:32	3	Calcium Gluconate	2	63951
19	10000032	39399961	2180-07-23 8:32	3	Calcium Gluconate	3	66576
20	10000032	39399961	2180-07-23 8:32	4	Dextrose 50%	1	1989
21	10000032	39399961	2180-07-23 10:19	5	Lactulose	1	3067
22	10000032	39399961	2180-07-23 10:19	5	Lactulose	2	29054
23	10000032	39399961	2180-07-23 10:19	5	Lactulose	3	68217
24	10000032	39399961	2180-07-23 13:12	6	Lactulose	1	3067
25	10000032	39399961	2180-07-23 13:12	6	Lactulose	2	29054
26	10000032	39399961	2180-07-23 13:12	6	Lactulose	3	68217
27	10000084	35203156	2160-11-21 1:10	1	Pravastatin	1	20741
28	10000084	36954971	2160-12-27 23:42	1	Lidocaine Jelly 2% (Glydo)	1	38861
29	10000108	36533795	2163-09-27 20:37	1	Oxycodone-Acetaminophen	1	4222
30	10000108	36533795	2163-09-27 20:37	2	Ampicillin-Sulbactam	1	
31	10000108	36533795	2163-09-28 2:46	3	Ampicillin-Sulbactam	1	
32	10000108	39513268	2163-09-24 20:45	1	Penicillin V Potassium	1	8879
33	10000108	39513268	2163-09-24 20:46	2	Penicillin V Potassium	1	8879
34	10000178	31721172	2157-04-08 14:02	1	Colchicine 0.6mg TAB	1	8334
35	10000178	31721172	2157-04-08 14:03	2	Ibuprofen 400mg TAB	1	8348
36	10000178	31721172	2157-04-08 14:03	3	Colchicine 0.6mg TAB	1	8334

Dataset A-6: Mimic IV Sample Data – Medicine Administration Dataset

Table A-1: Infrequently administer medications in lower 25% of dataset

Medication	Used for UTI	Primary Use If Not for UTI
Gastroview	No	Diagnostic agent in radiographic exams
Divalproex	No	Mood stabilizer, used in bipolar disorder
Adenosine	No	Treats certain types of cardiac arrhythmias

Atenolol Tablet	No	Beta-blocker for hypertension, heart issues
Cetirizine	No	Antihistamine for allergies
Clobazam	No	Anticonvulsant for epilepsy
Calcium Gluconate	No	Calcium supplement
Fluconazole	Sometimes	Antifungal, used for yeast infections including some urinary tract fungal infections
Simethicone	No	Antiflatulent, used to relieve gas
Vitamin D Tablet	No	Vitamin D supplement
Clindamycin	Sometimes	Antibiotic, used for bacterial infections, including some cases of bacterial UTIs

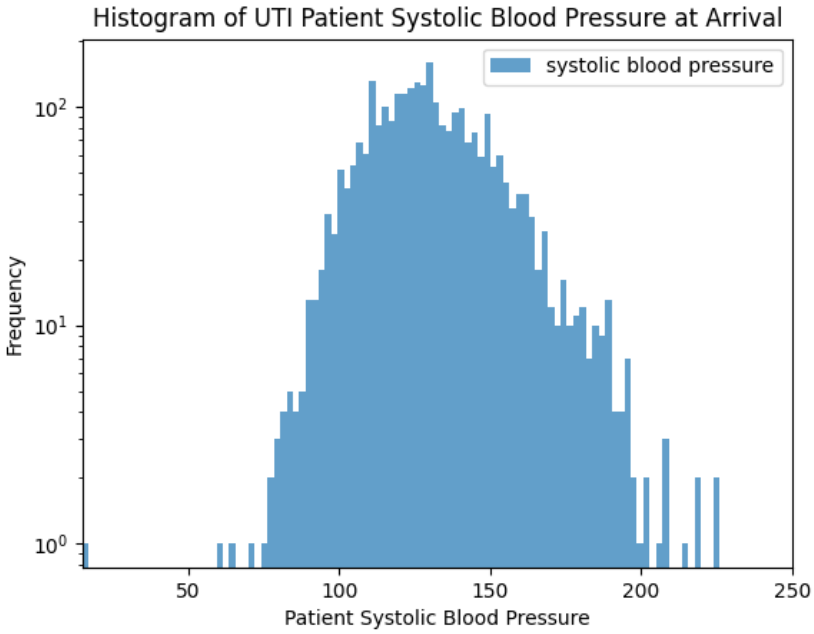

Figure A-1: Histogram for UTI Patient Systolic Blood Pressure upon arrival

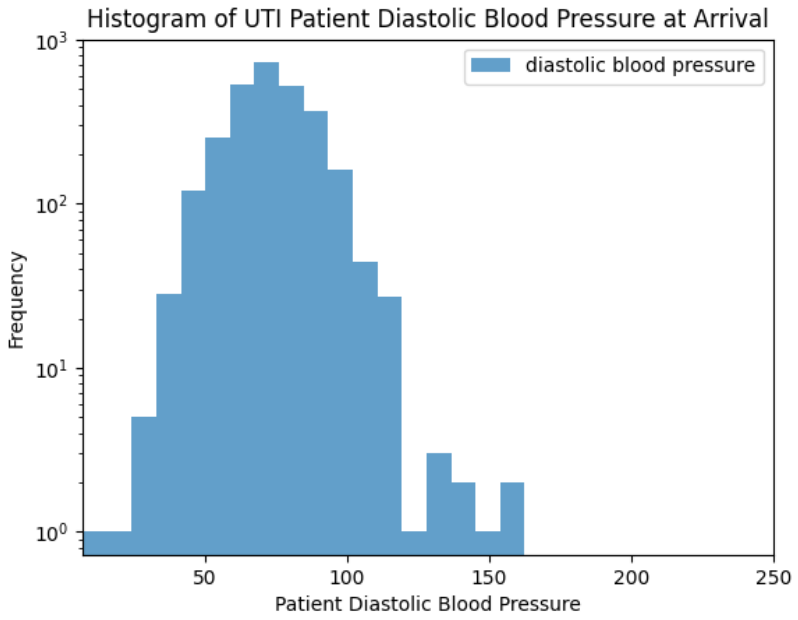

Figure A-2: Histogram for UTI Patient Diastolic Blood Pressure upon arrival

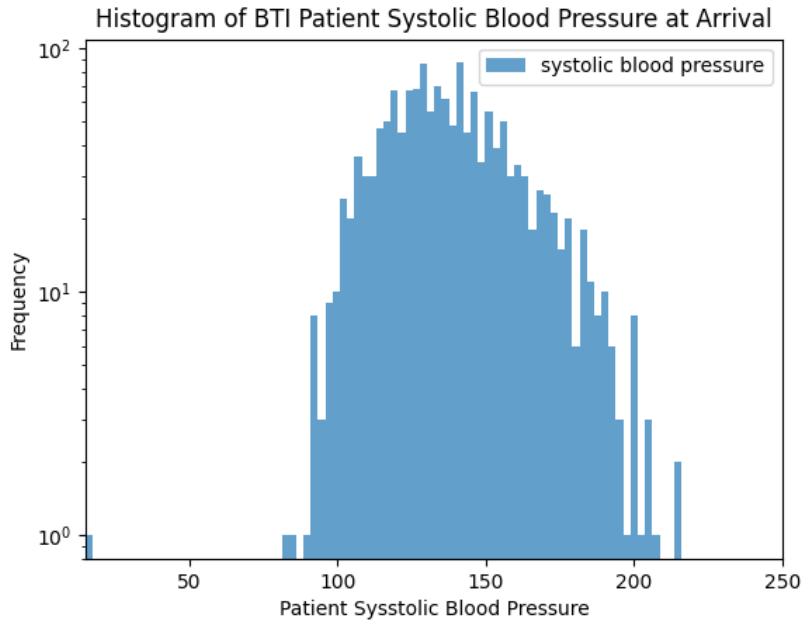

Figure A-3: Histogram for BTI Patient Systolic Blood Pressure upon arrival

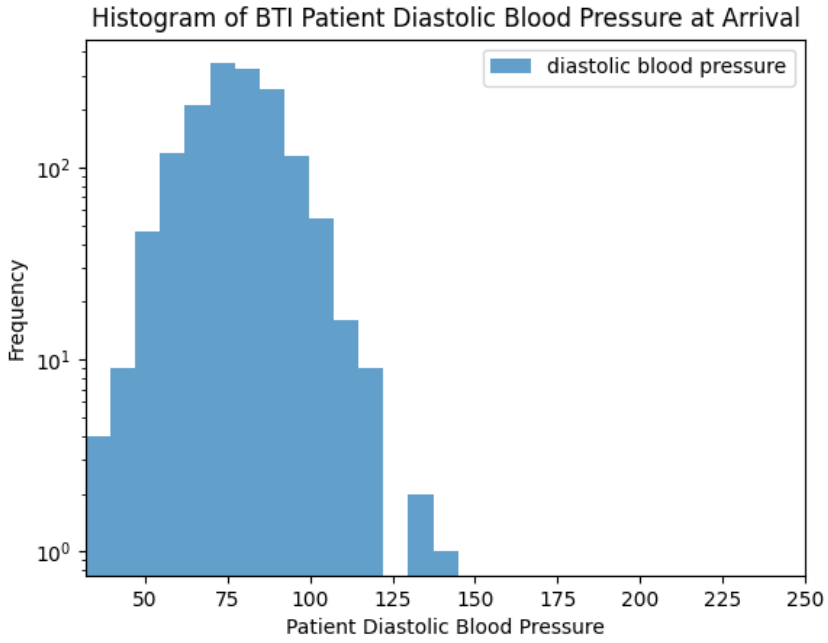

Figure A-4: Histogram for BTI Patient Diastolic Blood Pressure upon arrival

Appendix B

This appendix presents the script used for the study. Please note that a few parameters in the script were modified to obtain different data visuals or values.

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
import re
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.multioutput import MultiOutputClassifier
from datetime import datetime, timedelta
import pm4py
from pm4py.algo.discovery.alpha import algorithm as alpha_miner
from pm4py.visualization.petri_net import visualizer as pn_visualizer
from datetime import datetime as dataframe
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier
import matplotlib.patches as mpatches
import matplotlib.pyplot as plt
from sklearn.metrics import precision_score, recall_score, f1_score

# Read .csv's
diagnosisdf = pd.read_csv(r"C:\Users\athar\OneDrive\Documents\UBC 4th
Year\Thesis\mimic-iv-ed-2.2\ed\diagnosis.csv\diagnosis.csv")
edstaysdf = pd.read_csv(r"C:\Users\athar\OneDrive\Documents\UBC 4th
Year\Thesis\mimic-iv-ed-2.2\ed\edstays.csv\edstays.csv")
triagedf = pd.read_csv(r"C:\Users\athar\OneDrive\Documents\UBC 4th
Year\Thesis\mimic-iv-ed-2.2\ed\triage.csv\triage.csv")
medrecondf = pd.read_csv(r"C:\Users\athar\OneDrive\Documents\UBC 4th
Year\Thesis\mimic-iv-ed-2.2\ed\medrecon.csv\medrecon.csv")
pyxisdf = pd.read_csv(r"C:\Users\athar\OneDrive\Documents\UBC 4th
Year\Thesis\mimic-iv-ed-2.2\ed\pyxis.csv\pyxis.csv")
vitalsdf = pd.read_csv(r"C:\Users\athar\OneDrive\Documents\UBC 4th
Year\Thesis\mimic-iv-ed-2.2\ed\vitalsign.csv\vitalsign.csv")

# CLEANUP DATAFRAMES
# rename columns to avoid duplicate column titles
medrecondf = medrecondf.rename(columns={'charttime': 'histcharttime'})
vitalsdf = vitalsdf.rename(columns={'charttime': 'vitcharttime'})
```

```
# convert columns with timestamps into datetime values and store columns as int
medrecondf['histcharttime'] = pd.to_datetime(medrecondf['histcharttime'],
format='mixed')
edstaysdf['intime'] = pd.to_datetime(edstaysdf['intime'], format='mixed')
edstaysdf['outtime'] = pd.to_datetime(edstaysdf['outtime'], format='mixed')
pyxisdf['charttime'] = pd.to_datetime(pyxisdf['charttime'], format='mixed')
vitalsdf['vitcharttime'] = pd.to_datetime(vitalsdf['vitcharttime'],
format='mixed')
triagedf['stay_id'] = triagedf['stay_id'].astype('int')
edstaysdf['gender'] = edstaysdf['gender'].replace({'M': 1, 'F': 0})

# Filter for UTIs: Pylonephritis & Unspecified UTIs
diagnosisdf = diagnosisdf.drop_duplicates(subset=['stay_id'], keep=False)  # keep
line remove duplicates to filter only for UTIs
icd_uti_codes = ('N390', '5990', 'N10', 'N11', '59010', '59011')  # icd 9 and 10
included
icd_bti_codes = ('S06', '800', '804', '850', '854')
bti_conditions = '|'.join(icd_bti_codes)
uti_conditions = '|'.join(icd_uti_codes)
uti_df = diagnosisdf[diagnosisdf['icd_code'].str.startswith((icd_uti_codes))]  #
Filter according to uti codes or bti codes, adjust as needed

# Create a set of stay_IDs that meet the "UTI" conditions and apply to
medications administered
uti_set = set(uti_df['stay_id'])
pyxisdf = pyxisdf.loc[pyxisdf['stay_id'].isin(uti_set)]

# Function to simplify medication names using regex patterns
def simplify_med_name(name):
    # Normalize the medication name: remove parts that start with numbers
    simplified_name = re.sub(r'\b\d+[^ ]*', '', name)
    # Remove volume, brands and other details while keeping primary medication
name and method of administration (tablet, vaccine, etc.)
    simplified_name = re.sub(r'\s*\d*\.?\d*\s*(mg|gm|ml)[^a-zA-Z]*', '',
simplified_name, flags=re.IGNORECASE)
    simplified_name = re.sub(r'\s*\([^)]*\)', '', simplified_name)  # Remove
anything in parentheses
    simplified_name = re.sub(r'\s*vial|bag', '', simplified_name,
flags=re.IGNORECASE)  # Remove specific package types
    simplified_name = simplified_name.strip().lower()  # Convert to lower case
and strip whitespace
    return simplified_name

# Apply the simplification function to the medication column and create a df with
new med names and frequency
```

```
pyxisdf['simplified_medication'] = pyxisdf['name'].apply(simplify_med_name)
countsdf = pyxisdf['simplified_medication'].value_counts()
countsdf = countsdf.reset_index()
countsdf.columns = ['simplified_medication', 'count']
oneoffs = list(countsdf.loc[countsdf['count'] <= 3]['simplified_medication'])
oneoff_df = pyxisdf.loc[pyxisdf['simplified_medication'].isin(oneoffs)]
er_oneoff = edstaysdf.loc[edstaysdf['stay_id'].isin(oneoff_df['stay_id'])]
er_oneoff['staytime'] = er_oneoff['outtime'] - er_oneoff['intime']
edstayuti = edstaysdf.loc[edstaysdf['stay_id'].isin(uti_set)]
edstayuti['stay'] = edstayuti['outtime'] - edstayuti['intime']
meanstay = np.mean(edstayuti['stay'])
meanoneoff = np.mean(er_oneoff['staytime'])

# Set limit to only keep top 75% of medications
medcount_lim = sum(countsdf['count']) * 0.75
tally = 0
indexcounter = 0

# Filter out lwoer 25% of medications
for index, medrow in countsdf.iterrows():
    add = medrow['count']
    tally += add
    indexcounter += 1
    if tally >= medcount_lim:
        count_filter = countsdf.iloc[:indexcounter, :]
        break

simp_med_100 = count_filter['simplified_medication']

# Initialize a color array with 'blue' for all items
colors = ['blue'] * len(countsdf)

# Update the color array to 'red' for the bottom 25%
for idx in countsdf.index:
    if idx > indexcounter:
        colors[idx] = 'red'

plt.bar(countsdf.index, countsdf['count'], color=colors)
plt.title('Bar Chart Representing Frequency of Medications Administered for
BTIs')
red_patch = mpatches.Patch(color='red', label='Bottom 25%')
blue_patch = mpatches.Patch(color='blue', label='Top 75%')
plt.legend(handles=[red_patch, blue_patch])
plt.xlabel('Medication')
plt.ylabel('Log Scale Frequency of Medication')
```

```
plt.yscale('log')
plt.show()

pyxisdf_singular = pyxisdf.drop_duplicates(subset=['stay_id',
'simplified_medication']).reset_index()

pyxisdf_singular =
pyxisdf_singular.loc[pyxisdf_singular['simplified_medication'].isin(simp_med_100)
]

uti_set = pyxisdf_singular['stay_id']
medrecondf = medrecondf.loc[medrecondf['stay_id'].isin(uti_set)]
uti_df = uti_df.loc[uti_df['stay_id'].isin(uti_set)]

dataframe = pyxisdf_singular[['stay_id', 'simplified_medication']]

def create_transposed_df(df):
    # Pivot the table to get unique ids as rows and unique medications as columns
    # Fill missing values with 0, as those combinations do not exist in the input
table
    transposed_df = pd.crosstab(df['stay_id'], df['simplified_medication'])

    # Convert the table to use 1s and 0s instead of counts (which are 1s and 0s
in this case already)
    transposed_df = transposed_df.applymap(lambda x: 1 if x > 0 else 0)

    # Reset the index to make 'id' a column again and rename the columns
    transposed_df.reset_index(inplace=True)
    transposed_df.columns = ['stay_id'] + ['med' + col.upper() for col in
transposed_df.columns[1:]]

    return transposed_df

def encode_pad_transpose(df, stay_id_col, medication_col):
    """
    Encodes and pads the medication data of a DataFrame based on stay_id, and
provides a dictionary of encoded values to medication names.

    Parameters:
    - df: DataFrame containing the data.
    - stay_id_col: Name of the column containing stay IDs.
    - medication_col: Name of the column containing medication names.

    Returns:
    - DataFrame with stay_id, and padded, encoded medication columns.
```

```
    - Dictionary of encoded values to medication names.
    """

    # Step 1: Encode 'medication_col'
    label_encoder = LabelEncoder()
    df = df.sort_values(by=[stay_id_col, medication_col])
    df['med_encoded'] = label_encoder.fit_transform(df[medication_col]) + 1  # +1
so 0 can be used for padding

    # Create a dictionary of encoded values to medication names
    value_to_med_name = {index + 1: label for index, label in
enumerate(label_encoder.classes_)}

    # Step 2: Group by 'stay_id' and collect encoded values into lists
    grouped =
df.groupby(stay_id_col)['med_encoded'].apply(list).reset_index(name='med_list')

    # Step 3: Find the maximum list length for padding
    max_len = grouped['med_list'].str.len().max()

    # Step 4: Pad lists and transpose into separate columns
    for i in range(max_len):
        grouped[f'med{i+1}'] = grouped['med_list'].apply(lambda x: x[i] if i <
len(x) else 0)

    # Drop the 'med_list' column and ensure the DataFrame is in the desired
format
    grouped.drop(columns=['med_list'], inplace=True)

    return grouped, value_to_med_name

admstrd_grouped = encode_pad_transpose(dataframe, 'stay_id',
'simplified_medication')[0]
admstrd_dict = encode_pad_transpose(dataframe, 'stay_id',
'simplified_medication')[1]
recon_grouped = encode_pad_transpose(medrecondf, 'stay_id', 'etccode')[0]
uti_grouped = encode_pad_transpose(uti_df, 'stay_id', 'icd_code')[0]
triage = triagedf[['stay_id','acuity', 'temperature', 'heartrate']]

binarymed = create_transposed_df(dataframe)
triage = triagedf[['stay_id','acuity', 'temperature', 'heartrate', 'resprate',
'sbp', 'dbp', 'o2sat']]
merge_df2 = pd.merge(uti_grouped, binarymed, on='stay_id', how='inner')
merge_df1 = pd.merge(edstaysdf[['stay_id','gender']], merge_df2, on='stay_id',
how='inner')
```

```
merge_df = pd.merge(triage, merge_df1, on='stay_id', how='inner')
binary_df = merge_df.iloc[:, 1:]

merge_df2 = pd.merge(uti_grouped, admstrd_grouped, on='stay_id', how='inner')
merge_df1 = pd.merge(edstaysdf[['stay_id','gender']], merge_df2, on='stay_id',
how='inner')
merge_df = pd.merge(triage, merge_df1, on='stay_id', how='inner')
merge_df = merge_df.iloc[:, 1:]
merge_df = merge_df.dropna()

plt.hist(merge_df['sbp'], bins=100, alpha=0.7, label='systolic blood
pressure')  # Adjust the number of bins as necessary
plt.xlabel('Patient Sysstolic Blood Pressure')
plt.ylabel('Frequency')
plt.title('Histogram of BTI Patient Systolic Blood Pressure at Arrival')
plt.xlim(min(merge_df['sbp']), 250)
plt.legend()
plt.yscale('log')  # Set y-axis to logarithmic scale

plt.show()

# MODEL TRAINING AND TESTING
# Separate the features and the target variables
X = merge_df.iloc[:, :-10]
y = merge_df.iloc[:, -10:]

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=40)

# Initialize the RandomForestClassifier
rf = RandomForestClassifier(n_estimators=500, random_state=42)

# Create the MultiOutputClassifier
multi_target_rf = MultiOutputClassifier(rf, n_jobs=-1)

# Train the model
multi_target_rf.fit(X_train, y_train)

# Predict the target variable for the test set
y_pred = multi_target_rf.predict(X_test)

# Collecting scores for each output
precisions = []
recalls = []
```

```
f1_scores = []

for i in range(y_test.shape[1]):
    precisions.append(precision_score(y_test.iloc[:, i], y_pred[:, i],
average='macro'))
    recalls.append(recall_score(y_test.iloc[:, i], y_pred[:, i],
average='macro'))
    f1_scores.append(f1_score(y_test.iloc[:, i], y_pred[:, i], average='macro'))

print(precisions)
print(recalls)
print(f1_scores)

# Averaging the scores across all outputs
overall_precision = np.mean(precisions)
overall_recall = np.mean(recalls)
overall_f1 = np.mean(f1_scores)

print("Overall Precision: {:.4f}".format(overall_precision))
print("Overall Recall: {:.4f}".format(overall_recall))
print("Overall F1 Score: {:.4f}".format(overall_f1))

# Since we have multiple targets, we need to calculate accuracy for each one
accuracies = y_test.columns.map(lambda col: accuracy_score(y_test[col],
y_pred[:,list(y_test.columns).index(col)]))

y_test_array = y_test.to_numpy()

def calculate_accuracy_and_frequency(pred_array, test_array):
    # Ensure both arrays have the same shape
    assert pred_array.shape == test_array.shape, "Arrays must have the same
shape."

    # Overall accuracy
    overall_accuracy = np.mean(pred_array == test_array)
    print(f"Overall Accuracy: {overall_accuracy:.2%}")

    # Total number of elements in test_array
    total_elements = test_array.size - np.count_nonzero(test_array == 0)

    acc_list = []
    freq_list = []

    # Accuracy and frequency for each number from 0 to 24
    for number in range(1, np.max(test_array)):
```

```
        mask = test_array == number
        num_occurrences = np.sum(mask)
        if num_occurrences == 0:
            print(f"number {number} is not present")
        else:
            accuracy = np.mean(pred_array[mask] == test_array[mask])
            frequency = num_occurrences / total_elements
        acc_list.append(accuracy)
        freq_list.append(frequency)

    return acc_list, freq_list

acc_list1 = calculate_accuracy_and_frequency(y_pred, y_test_array)[0]
freq_list1 = calculate_accuracy_and_frequency(y_pred, y_test_array)[1]

# Positions of the left bar-groups
barWidth = 0.3
r1 = np.arange(1, len(acc_list1)+1)
r2 = [x + barWidth for x in r1]

# Create blue bars
plt.bar(r1, acc_list1, color='blue', width=barWidth, edgecolor='grey',
label='Accuracy')

# Create red bars (middle of group)
plt.bar(r2, freq_list1, color='red', width=barWidth, edgecolor='grey',
label='Frequency')

# General layout
plt.xticks([r for r in range(1, 1 + len(acc_list1))])
plt.ylabel('Frequency/Accuracy')
plt.xlabel('Encoded Medication')
plt.title('Dual Bar Chart for BTIs Showing Model Class Accuracy vs Class
Frequency in Dataset')
plt.legend()
plt.show()

# Separate method to calculate accuracy, since if a medicaiton is predicted in
column 1 however is present in column 2, the prediction should still be treated
as correct
ypred = pd.DataFrame(y_pred)
count = 0
count1 = 0
TP_count = 0
FP_count = 0
```

```
FN_count = 0
accuracy = []
Prec = []
Recall = []
F_1 = []
unique_case_pred = set()
unique_case_act = set()

for i, row in ypred.iterrows():
    # Accuracy Calculation
    for x in range(0, 10):
        if row[x] != 0:
            if row[x] in list(y_test.iloc[i, :]):
                count += 1
                TP_count += 1
            else:
                FP_count += 1
        elif row[x] == y_test.iloc[i, x]:
            count += 1
            TP_count += 1
        else:
            FN_count += 1
    accuracy.append(count/10)
    prec = (TP_count/(TP_count + FP_count))
    if TP_count + FN_count != 0:
        recall = (TP_count/(TP_count + FN_count))
    else:
        recall = 0
    if prec + recall != 0:
        F1 = (2 * prec * recall)/(prec + recall)
    else:
        F1 = 0
    Prec.append(prec)
    Recall.append(recall)
    F_1.append(F1)
    if (count/10) >= .9:
        count1 += 1
        processed_row = sorted([y for y in row if y != 0])
        unique_case_pred.add(tuple(processed_row))

    processed_row_act = sorted([z for z in row if z != 0])
    unique_case_act.add(tuple(processed_row_act))
    count = 0
    TP_count = 0
    FP_count = 0
```

```
    FN_count = 0

unique_pred_list = [list(row) for row in unique_case_pred]
unique_act_list = [list(row) for row in unique_case_act]

def replace_with_med_name(num_list, num_to_med_dict):
    replaced_list = []
    for sublist in num_list:
        med_names = [num_to_med_dict[num] for num in sublist if num in
num_to_med_dict]
        replaced_list.append(med_names)
    return replaced_list

# Replace numbers with medication names
med_names_list = replace_with_med_name(unique_pred_list, admstrd_dict)

print(med_names_list)
print(count1/len(y_test))

print('random forests classifier model accuracy:',
np.round(np.mean(accuracy)*100, 3), '%')
print('random forests classifier model prec:', np.round(np.mean(Prec)*100, 3),
'%')
print('random forests classifier model recall:', np.round(np.mean(Recall)*100,
3), '%')
print('random forests classifier model f1:', np.round(np.mean(F_1)*100, 3), '%')

# Separate the features and the target variables
X = merge_df.iloc[:, :-10]  # All columns except the last seven
y = merge_df.iloc[:, -10:]  # Just the last seven columns

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Initialize the DecisionTreeClassifier
dt = DecisionTreeClassifier(random_state=42)

# Create the MultiOutputClassifier wrapping the DecisionTreeClassifier
multi_target_dt = MultiOutputClassifier(dt, n_jobs=-1)

# Train the model
multi_target_dt.fit(X_train, y_train)

# Predict the target variable for the test set
```

```
y_pred = multi_target_dt.predict(X_test)

# Since we have multiple targets, calculate accuracy for each one
accuracies = {col: accuracy_score(y_test[col], y_pred[:,
list(y_test.columns).index(col)]) for col in y_test.columns}

ypred = pd.DataFrame(y_pred)
count = 0
count1 = 0
TP_count = 0
FP_count = 0
FN_count = 0
accuracy = []
Prec = []
Recall = []
F_1 = []
unique_case_pred = set()
unique_case_act = set()

for i, row in ypred.iterrows():
    # Accuracy Calculation
    for x in range(0, 10):
        if row[x] != 0:
            if row[x] in list(y_test.iloc[i, :]):
                count += 1
                TP_count += 1
            else:
                FP_count += 1
        elif row[x] == y_test.iloc[i, x]:
            count += 1
            TP_count += 1
        else:
            FN_count += 1
    accuracy.append(count/(10))
    prec = (TP_count/(TP_count + FP_count))
    recall = (TP_count/(TP_count + FN_count))
    F1 = (2 * prec * recall)/(prec + recall)
    Prec.append(prec)
    Recall.append(recall)
    F_1.append(F1)
    # print('acc:', count/7, '\n')
    if (count/10) >= .9:
        count1 += 1
        processed_row = sorted([y for y in row if y != 0])
        unique_case_pred.add(tuple(processed_row))
```

```
    # if F1 >= 0.9:
    #     print(row, y_test.iloc[i, :])

    processed_row_act = sorted([z for z in row if z != 0])
    unique_case_act.add(tuple(processed_row_act))
    count = 0
    TP_count = 0
    FP_count = 0
    FN_count = 0

print('decision trees classifier model accuracy:',
np.round(np.mean(accuracy)*100, 3), '%')
print('decision trees classifier model prec:', np.round(np.mean(Prec)*100, 3),
'%')
print('decision trees classifier model recall:', np.round(np.mean(Recall)*100,
3), '%')
print('decision trees classifier model f1:', np.round(np.mean(F_1)*100, 3), '%')

# Optionally, print the accuracies for each target
for target, acc in accuracies.items():
    print(f"Accuracy for {target}: {acc}")

# EVENT LOG APPLICATION TO CREATE PROCESS FLOWS
# Create event log and decode the medications used
def create_eventlog(pred_meds_list):
    holder = 1
    df_list = []
    for meds in pred_meds_list:
        meds_len = len(meds)
        event_list = ['entry', 'medhist', 'vitals'] + meds + ['exit']

        time_list = []
        for num in range(0, meds_len + 4):
            time = datetime(2020, 1, 1, 0, 0, 0)
            time_int = time + timedelta(minutes=10*num)
            time_list.append(time_int)

        df = pd.DataFrame({
            'case:concept:name': [str(holder)] * len(event_list),
            'concept:name': event_list,
            'time:timestamp': time_list
        })

        holder += 1
```

```
        df_list.append(df)
    return df_list

# Store event log as a dataframe to be read later
df1 = create_eventlog(pred_meds_list=med_names_list)
master_df = pd.DataFrame()

for frames in df1:
    master_df = pd.concat([master_df, frames], ignore_index=True)

string_range = [str(i) for i in range(1, 15+1)]

df_1 = master_df.loc[master_df['case:concept:name'].isin(string_range)]

# Create process flows using pm4py library
log = pm4py.convert_to_event_log(df_1)
net, initial_marking, final_marking = alpha_miner.apply(log)
gviz = pn_visualizer.apply(net, initial_marking, final_marking,
parameters={"node_font_size": 20, "edge_font_size": 20})
pn_visualizer.view(gviz)
```