Advancing Few-Shot Multi-Label Medication Prediction in Intensive Care Units: The FIDDLE-Rx Approach

Xinghe Chen

Submitted in partial fulfillment of the
requirements for the degree of
Master of Science in Computer Science
Fu Foundation School of Engineering and Applied Science

COLUMBIA UNIVERSITY IN THE CITY OF NEW YORK

2023

Abstract

Contemporary intensive care units (ICUs) are navigating the challenge of enhancing medical service quality amidst financial and resource constraints. Machine learning models have surfaced as valuable tools in this context, showcasing notable effectiveness in supporting healthcare delivery. Despite advancements, a gap remains in real-time medical interventions. To bridge this gap, we introduce `FIDDLE-Rx`, a novel, data-driven machine learning approach designed specifically for real-time medication recommendations in ICUs. This method leverages the eICU Collaborative Research Database (eICU-CRD) for its analysis, which encompasses diverse electronic health records from ICUs (ICU-EHRs) sourced from multiple critical care centers across the US. `FIDDLE-Rx` employs the Flexible Data-Driven Pipeline (`FIDDLE`) for transforming tabular data into binary matrix representations and standardizes medication labels using the RxNorm (`Rx`) API . With the processed dataset, `FIDDLE-Rx` applies various machine learning models to forecast the requirements for 238 medications. Compared with previous studies, `FIDDLE-Rx` stands out by extending the scope of the research of ICU-EHRs beyond mortality prediction, offering a more comprehensive approach to enhancing critical care. The experimental results of our models demonstrate high efficacy, evidenced by their impressive performance across two key metrics: the area under the receiver operating characteristic curve (AUROC) and the area under the precision-recall curve (AUPRC). Remarkably, these results were achieved even when the model was trained with just 20% of the database, underlining its strong generalizability. By broadening the scope of ICU-EHRs research to encompass real-time medication recommendations, `FIDDLE-Rx` presents a scalable and effective solution for improving patient care in intensive care environments.

Table of Contents

Acknowledgements

The author extends heartfelt acknowledgements to:

Professor Shalmali Joshi, his thesis supervisor and master's research advisor, for her invaluable insights and guidance throughout this journey; alongside his master's research advisor, Professor Noémie Elhadad, for her tremendous support during his master's study. *They have imparted to the author the knowledge to accomplish what he can change.*

His beloved fiancée, Miss Murphy Hu, for her enduring companionship in New York and New Jersey, and her life-long commitment through every challenge. *She has granted the author the courage to embrace the unchangeable.*

His beloved parents, for gifting him the life-altering chance to explore academia at Columbia University, across 6,828 miles from home. *They have blessed the author with the wisdom to discern the differences between one from the other.*

His one and only best friend, Miss Mico Zhao, for a decade of steadfast support. From China to England, and then America, her constant presence has inspired the author's unwavering dedication to doing the right thing.

His new friend in New York, Miss Jessi Yang, a serendipitous meeting blossoming into what the author hopes will be a lasting friendship. He insists, under all circumstances, that she is the unrivaled Mario Party pro player.

Without each one of you, this work could not have been completed.

Chapter 1: Introduction

Electronic health records (EHRs) are a comprehensive digital repository of patient data compiled during healthcare operations. These records contain a broad spectrum of information, including medication histories, diagnoses, laboratory orders and results, procedures, insurance claims, and other healthcare services [1]. As such, the contexts and objectives of EHRs are highly variable. Some EHR systems, initially designed for billing purposes [2, 3], fall short in facilitating clinical workflows. Others, however, are developed for patient monitoring [4, 5], aiming to improve diagnosis and treatment and thus provide substantial benefits for clinical research.

Among all EHRs, those collected in Intensive Care Units (ICU), called ICU-EHRs, are particularly noteworthy as they provide direct insights into real-time patient care in ICUs. Nevertheless, ICUs face specific challenges, including (a) financial limitations, (b) resource-allocation difficulties, and (c) staffing issues, which can affect the quality of care [6]. Therefore, there has been a focus on investigating data-driven machine learning models to address clinical challenges [7, 8]. Recent studies have demonstrated the effectiveness of machine learning techniques applied to such ICU-EHRs for predicting patient mortality within specific cohorts, including those with cerebral infarction [9], hip fracture [10], lung cancer [11], and chronic kidney disease [12]. The accuracy of these predictive tasks is contingent upon clinicians' diligent data collection. Therefore, such machine learning tasks rely on the behaviors and insights of clinicians rather than on physiological signals that may aid clinical decision-making [13]. Among these studies, the extensive pre-filtering of ICU-EHRs plays a crucial role in reducing data size, which aids machine learning models in effectively recognizing patterns within the simplified sample distribution. However, this approach limits the models' applicability to a broader patient population, while mortality prediction alone yields limited

insights into patient care in critical settings. To conclude, there is an essential requirement to design machine learning models that go beyond mortality forecasts and can suggest real-time medical interventions based on ICU-EHRs [14].

Building upon the challenges in ICU operations and the necessity of realizing real-time medical interventions at ICUs, this study introduces a data-driven deep learning (DL) approach, `FIDDLE-Rx`, for treatment recommendation in ICUs. Developed in collaboration with H. Wang's work [15], `FIDDLE-Rx` is designed for the simultaneous prediction of multiple medications. It leverages the eICU Collaborative Research Database (eICU-CRD), a substantial database comprising ICU-EHRs from various centers across the US. For dataset creation, `FIDDLE-Rx` compiles an extensive dataset with over two million records, integrating the Flexible Data-Driven Pipeline (`FIDDLE`) [16] and RxNorm (`Rx`) API. This dataset features detailed patient care information and 238 standardized medication labels that stand for 238 different kinds of medications. Based on the dataset, `FIDDLE-Rx` builds a series of different few-shot multi-label medication prediction models to forecast the requirement for these 238 medications.

In summary, `FIDDLE-Rx` achieves several key objectives: (a) It forecasts the required medications for the next four hours, based on an analysis of eight hours of ICU-EHRs from a patient. (b) It implements a standardized pipeline to associate patients with medication labels, utilizing the RxNorm API. (c) It demonstrates the effectiveness of data engineering in handling large-scale datasets for multi-label prediction tasks. (d) It showcases its ability to generalize, performing few-shot predictions effectively even with limited training sample sizes. The effectiveness of `FIDDLE-Rx` is demonstrated by high scores in both the area under the receiver operating characteristic (AUROC) and the area under the precision-recall curve (AUPRC), which highlights both its generalizability and its practicality. Thus, `FIDDLE-Rx` presents a scalable and effective approach for real-time medication intervention, showing promise for future use in enhancing patient care in intensive care settings.

Chapter 2: Technical Approach

This chapter introduces the methods of `FIDDLE-Rx`, including `FIDDLE`, data engineering, model design, model training, and evaluation strategies.

2.1 `FIDDLE`

In this study, we utilize data from eICU-CRD, encompassing ICU-EHRs of 200,859 patient admissions across 208 ICU centers in the US, spanning from 2013 to 2015 [4]. This extensive dataset offers a broad sample distribution, enabling a thorough investigation into the generalizability of our pipeline.

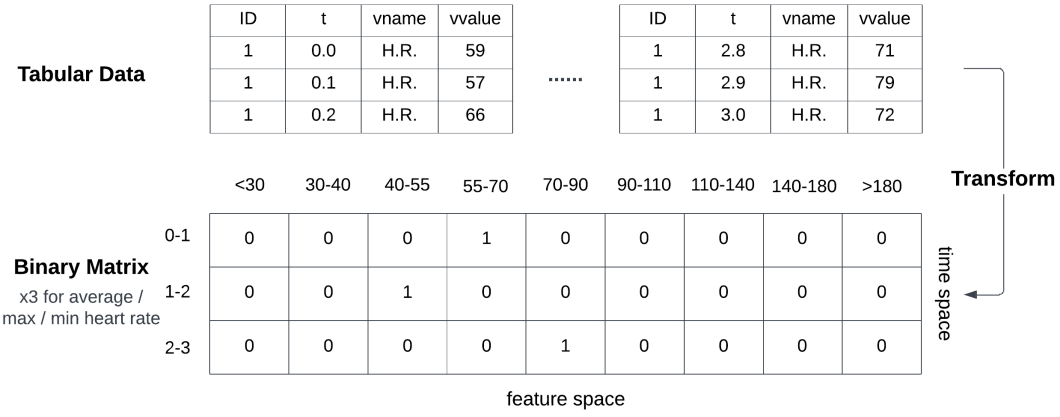

Figure 2.1: An example demonstrates the transformation of a patient's heart rate (H.R.) data into a binary matrix representation. The heart rate measurements within each time window are processed to calculate the matrix representation of their average, maximum, and minimum values. Subsequently, these three matrices are concatenated along the feature space axis, forming the final binary matrix representation.

Before applying machine learning techniques, substantial effort is needed to preprocess the ICU-EHRs. eICU-CRD comprises 18 structured SQL tables. For example, *patient* table contains demographic information; *vitalPeriodic* table lists time-series vital signs; *lab*

table details laboratory tests; and *note* table includes physician and nurse assessment documentation. However, the eICU-CRD presents challenges with missing values and significant variation in the harmonization of its tables. This inconsistency, particularly with some values being non-numerical, complicates the application of mathematical analysis. To address this, `FIDDLE` [16] is applied to eICU-CRD to combine and transform these tables into a binary matrix representation. The transformation process includes the following stages:

- Combine tables: To combine all tables, the tabular data in eICU-CRD are converted to rows with four columns `ID`, `t`, `vname`, and `vvalue`. The `ID` is unique for each patient. `t` is the timestamp, with `t=0` meaning admission time. If `t` is null, the variable is time-invariant, such as demographical information. The `vname` is consistent across all patients and encodes the variable's name. The `vvalue` is either numerical or a string and must not be null. Variables are classified as numerical or categorical based on `vvalue`, where categorical data is one-hot encoded, and numerical data may be kept as continuous or binned. For handling missing values in this process, imputation techniques with carry-forward methods [17] are utilized. xi

- Pre-filter: Rows with timestamps t that are outside the observation period $[0,T]$ are removed, and variables that are rarely shown are also excluded. A threshold, $\theta_1 \times 100\%$, is set to filter out such less frequent variables.

- Transform: Time-invariant variables across all patients are concatenated to form a 2D matrix of dimensions $N \times d$, where N represents the number of patients and d the number of time-invariant features. For time-dependent variables, a 3D matrix with dimensions $N \times D \times L$ is constructed, where D is the number of time-dependent features and L the number of timestamps. All variables are converted to a binary representation during this transformation to improve their suitability for machine learning applications. For example, a patient's heart rate can be encoded into a binary matrix, as shown in Fig. 2.1. An extra time dimension is added to the time-invariant matrix to facilitate the concatenation.

- Post-filter: After the data transformation, a feature is determined to be less informative if the frequency of its zero/one values falls below $\theta_2 \times 100\%$. Features meeting this criterion are excluded to reduce the data size.

In this study, the thresholds θ_1 and θ_2 are both set to 0.01. The selection criteria include patients who are at least 18 years old and have an ICU stay of at least 48 hours. This results in a total of 77,066 subjects being included in the analysis. Detailed mathematical procedures, such as the bin size and number of bins used for converting continuous variables into a binary matrix representation, can be found in [16]. Additionally, to facilitate multi-label medication prediction, medication-related variables are excluded from the analysis. After applying `FIDDLE`, the resulting matrix, M, has dimensions of 77,066 (number of patients) by 1,733 (number of features) by 48 (T in hours). Furthermore, in [15], each patient is associated with 238 medication labels (y) for each hour. Thus, y is of shape 77,066 (number of patients) by 238 (binary medication labels) by 48 (T in hours).

2.2 Data Engineering

Matrix	Dimension	Method	Corresponding Model
M	(77066, 1733, 48)	FIDDLE	/
M'	(77066 $\times$ 36, 1733, 8)	Sliding Window	Transformer, Attention
M''	(77066 $\times$ 36, 1733, 1)	Temporal Aggregation	DNN
M'''	(77066 $\times$ 36, 200, 1)	PCA	DNN, LR, RF, XGBoost
y	(77066, 238, 48)	Label Generator	/
y'	(77066 $\times$ 36, 238, 1)	Temporal Aggregation	All Models

Table 2.1: Summary of matrix notations along with their dimensions, methods to derive them, and the corresponding models in which they are utilized.

Following the FIDDLE approach, more comprehensive data engineering techniques are applied to M and y to construct the final database. As shown in Fig. 2.2, we use an eight-hour sliding window operation to matrix M along its temporal axis. This results in an aggregated matrix M', which encapsulates the complete time-series information. Concurrently, dataset y

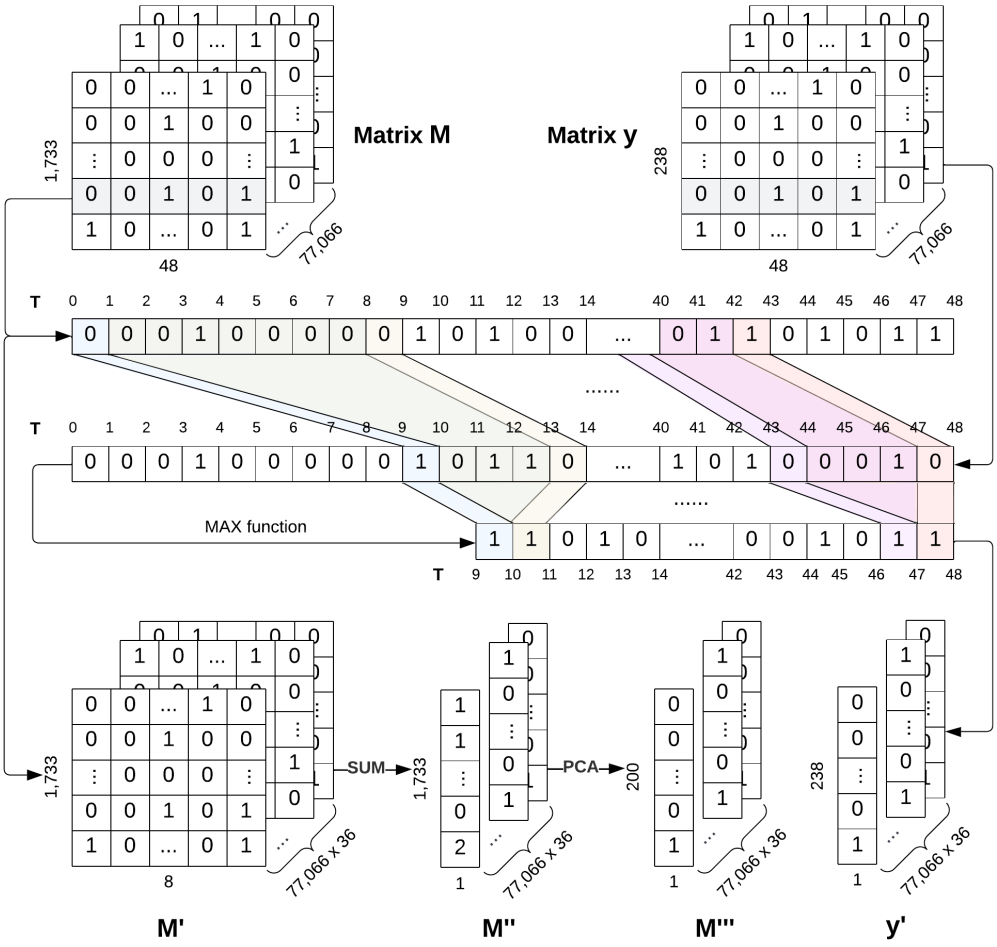

Figure 2.2: Schematic representation of the data transformation pipeline. The process begins with matrix M and y, which undergo sequential transformations to yield M', M'', M''', and y'.

is processed using a four-hour maximum filter, denoted as $\mathrm{MAX}_t^{t+3}\mathrm{value}(\mathrm{ID}, \mathrm{label}, t)$, commencing from time $t = 9$. This generates the corresponding label matrix y'. Afterward, we perform a summation operation across the temporal dimension of matrix M'. By effectively aggregating the time dimension, it facilitates the use of non-time-series models. In the final step, we reduce the dimensionality of the feature space to 200 components using principal component analysis (PCA). This results in a more compact matrix M''', which accelerates model training. For detailed information on this process, refer to section 2.3. Finally, dataset y' is aligned with either M', M'', or M''' to construct a dataset that encapsulates 8-hour summary features of patients and their corresponding 4-hour medication labels, separated by a 1-hour interval. For details on the application of these matrices in this study, see Table 2.1.

2.3 Principal Component Analysis

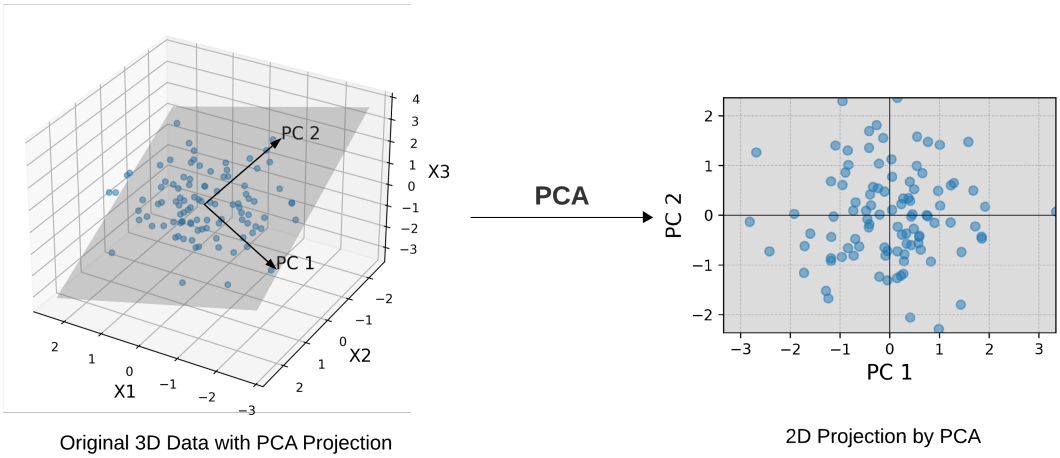

Figure 2.3: Transformation from three-dimensional database to two-dimensional space via PCA, with PC 1 and PC 2 representing the two principal component axes.

Although M'' has already eliminated the time dimension, the feature space dimension of 1,733 is still relatively large, particularly for non-DL models. This scenario motivates the application of PCA. PCA is a widely employed technique in data analysis for machine learning, primarily functioning in dimensionality reduction. Specifically, it distills a multi-dimensional

dataset into principal components comprising fewer dimensions while capturing the maximum variance within the data [18]. This procedure is realized through a linear transformation that identifies the most informative directions of the original data and projects the data onto new dimensions. In doing so, PCA effectively compresses the dataset and maintains its structural integrity with minimal information loss. In machine learning applications, the concentrated variance captured by PCA not only improves computational efficiency but also mitigates overfitting, contributing to more generalizable models [19].

Fig. 2.3 illustrates the transformation of a sample three-dimensional database into a two-dimensional representation using PCA. This research implements a dimensionality reduction from the initial feature space of 1,733 dimensions down to 200. This reduction not only allows for more efficient data compression but also improves the ability to recognize patterns within the data. Such streamlining substantially increases the training speed for non-DL methods, which employ a separate binary classifier for each medication. Throughout the PCA process, the model is fitted using the training data, and this fitting is then applied to transform the validation and test datasets. Details on the train/validation/test splitting strategy can be found in Section 2.6.1.

2.4 DL Models

In this study, we include three DL models: a transformer, a deep neural network (DNN), and an attention model. The purpose of incorporating the attention and transformer models is to evaluate the effectiveness of the self-attention mechanism in pattern recognition of time-series data. The transformer model builds upon this by employing a sequence of Transformer encoder blocks containing more comprehensive layers, thus maximizing its capacity to handle complex data sequences. Additionally, we include a vanilla DNN model to assess the efficacy of our data engineering approach.

This section aims to explain the transformer model, which was first proposed by A Vaswani et al. [20]. For comprehensive details on the DNN and attention models, refer to [15].

Figure 2.4 provides an overview of the transformer model. Initially, the input data is

embedded in a fully connected dense layer with an embedding size of 64. This is followed by a 2-layer encoder block designed to learn the structure and correlation of the time-series inputs. Subsequently, the hidden states of the inputs are processed through a multi-label head, which generates the 238 medication predictions. The following subsections delve into the mechanisms of each layer.

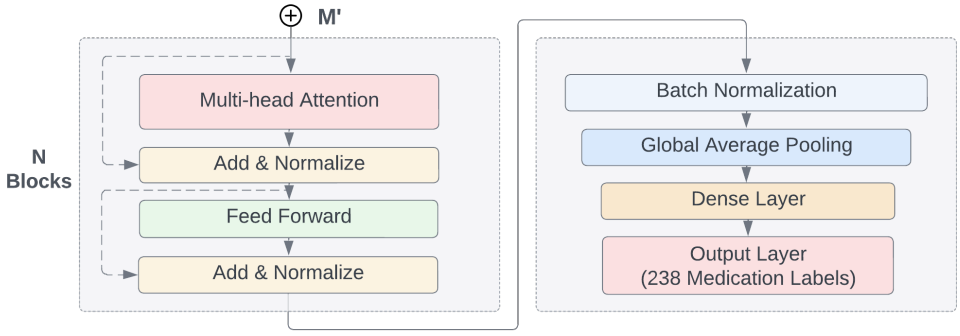

Figure 2.4: An overview of the architecture of the modified transformer model. The dotted lines represent residual connections. The term "N Blocks" indicates that the local architecture is replicated sequentially N times. The symbol $\oplus$ represents a dense layer used for input embedding.

2.4.1 Multi-head Attention

In the transformer encoder architecture, the attention mechanism is a critical component. It operates on three primary vectors: Query (Q), Key (K), and Value (V). These vectors are derived from the input embeddings. The attention function serves to map a query and a set of key-value pairs to an output. The output is computed as a weighted sum of the values, where each value's weight is computed by a compatibility function of the query with the corresponding key.

The attention weights are computed using the dot product of the Query with the Key:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.1}$$

Here, d_k ($d_k = 64$ in this work) is the dimension of the key vectors. The scaling factor,

$\sqrt{d_k}$, stabilizes the variance. The softmax function is

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}, \tag{2.2}$$

where z_i represents the i-th element of the input vector $\mathbf{z}$, and K is the total number of elements in the vector. The softmax function computes the exponential of z_i, divided by the sum of the exponents of all elements in the vector, effectively converting the input values into a probability distribution.

Additionally, the multi-head architecture enables the model to learn distinct representation subspaces at various positions. This is achieved by applying the attention mechanism h times, where h is the number of heads.

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O \tag{2.3}$$

$$\text{where head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{2.4}$$

In the above equations, W_i^Q, W_i^K, and W_i^V are parameter matrices for the i-th attention head, and W^O is the output linear projection matrix. In this study, we set the number of heads in the model to four for best performance.

2.4.2 Add & Normalize

The Add & Normalize layer, positioned after the multi-head attention and feed-forward layers, plays a significant role in stabilizing and accelerating the training process.

It consists of residual connections (the "Add" part) and layer normalization (the "Normalize" part). Residual connections, inspired by ResNet [21], allow the output of one layer to bypass some layers and directly add to a later layer's output, formulated as $\mathcal{F}(\mathbf{x}) + \mathbf{x}$ where $\mathbf{x}$ is the input and $\mathcal{F}$ is the function implemented by the layers. This approach mitigates the vanishing gradient problem, enabling deeper network architectures. Layer normalization, applied after the residual connection, normalizes the inputs across features, defined as

$\text{LN}(\mathbf{x}) = \gamma \left(\frac{\mathbf{x}-\mu}{\sqrt{\sigma^2+\epsilon}} \right) + \beta$ where μ and σ^2 are the mean and variance of the features, and γ and β are learnable parameters. Here, ϵ is a small constant added to avoid division by zero. This normalization stabilizes learning and allows for faster training with higher learning rates.

2.4.3 Feed Forward

The feed-forward layer in the encoder block is a critical component that follows the multi-head attention layer. This layer is fully connected and consists of two linear transformations with a ReLU activation in between. Mathematically, it can be represented as $\text{FFN}(\mathbf{x}) = \max(0, \mathbf{x}W_1 + b_1)W_2 + b_2$, where $\mathbf{x}$ is the input to the feed-forward layer, W_1 and W_2 are weight matrices, and b_1 and b_2 are bias vectors. The ReLU (Rectified Linear Unit) activation function, denoted by $\max(0, \cdot)$, introduces non-linearity into the model and enables it to learn more complex patterns in the data. Unlike the multi-head attention mechanism that allows the model to focus on different positions of the input sequence, the feed-forward layer operates on each position separately and identically. This design choice allows the encoder to integrate information from different representation subspaces at each position. In this study, the dimensions of the linear layers are fine-tuned to 64 to achieve optimal performance.

2.4.4 Multi-label Head

The multi-label head of the model is composed of several key components. Initially, a batch normalization (BN) layer is employed to stabilize the model by normalizing over each batch. Following this, a one-dimensional global average pooling layer is utilized to effectively eliminate the last dimension by taking the average of all units. This reduction aids in decreasing the model's complexity and computational cost. Finally, a dense layer, followed by a sigmoid classification head, is implemented. This dense layer serves to interpret the features extracted by previous layers, and the sigmoid classification head is used to generate the predictions for all medication labels, outputting values between 0 and 1 that represent the probability of each medication.

2.5 Non-DL Models

This study uses three non-DL methods – Logistic Regression (LR), Random Forest (RF), and XGBoost – as baseline models. LR is selected for its simplicity, particularly in linear contexts. RF, known for handling non-linear data, offers robustness in complex classification scenarios. XGBoost is included for its high performance and efficiency with diverse datasets. These models are suitable for binary prediction tasks, allowing for training distinct classifiers for each medication.

2.5.1 Logistic Regression

Logistic Regression (LR) is a method for predictive analysis that aims at explaining data and elucidating the association between a single binary dependent variable and one or more independent variables of varied types, including nominal, ordinal, interval, or ratio-level [22]. The logistic regression model is expressed through the logistic function, which is:

$$P(Y = 1) = \frac{1}{1 + e^{-(\beta_0+\beta_1 X_1+\beta_2 X_2+\ldots+\beta_k X_k)}} \tag{2.5}$$

In this formula, $P(Y = 1)$ represents the likelihood of the dependent variable being equivalent to a specific case (coded as "1"), given the independent variables. β_0 is the intercept, while $\beta_1, \beta_2, \ldots, \beta_k$ are the coefficients of the independent variables $X_1, X_2, \ldots, X_k$. The e is the base of natural logarithms. The coefficients are estimated from the training data using maximum likelihood estimation, delineating both the magnitude and direction of the correlation between the predictor variables and the response variable. The LR model is effective in situations where the predicted outcome is binary [22].

2.5.2 Random Forest

Random Forest (RF) is an ensemble learning method that constructs multiple decision trees during training and combines the trees' output for the final result. This approach lever-

ages the strength of multiple decision-making models, enhancing overall predictive accuracy and reducing the risk of overfitting [23]. Each tree in the RF contributes a vote toward the final prediction. The outcome is determined by the majority vote across all trees in classification tasks. This method is particularly effective in handling large datasets with high dimensionality of features, as it inherently performs feature selection and provides robustness against noise [23].

2.5.3 XGBoost

XGBoost, standing for eXtreme Gradient Boosting, is a state-of-the-art implementation of the gradient boosting algorithm, differing significantly from RF [24]. Unlike RF's parallel approach, XGBoost builds one tree at a time, where each tree is designed to correct the errors made by its predecessors. This is a key aspect of the boosting technique.

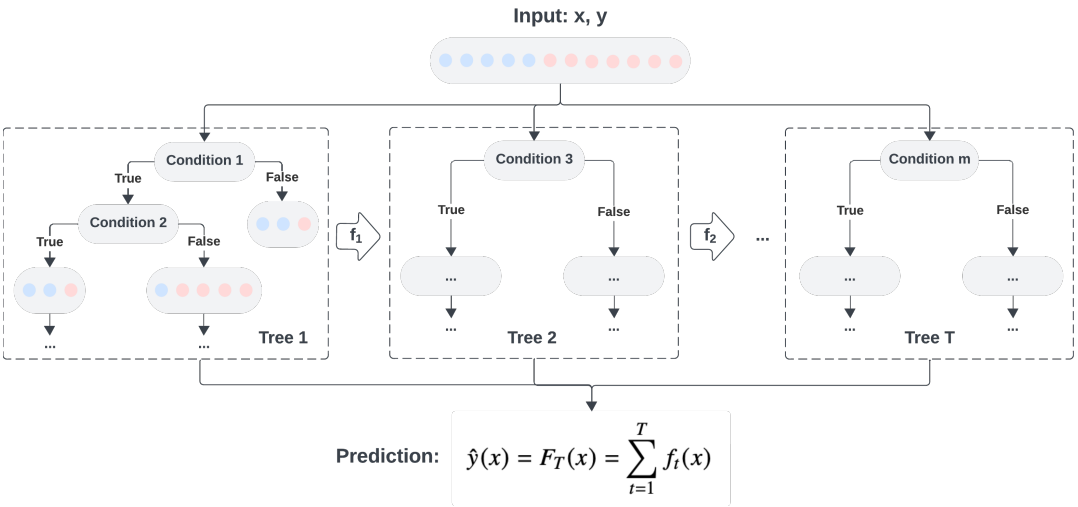

Figure 2.5: An illustration of the XGBoost algorithm that demonstrates how it sequentially builds trees, showcasing the series-like progression of this process.

As shown in Fig. 2.5, the model's prediction at step t, denoted as $F_t(x)$, is updated by the prediction, $f_t(x)$, from the new decision tree t. The update process is described as:

$$F_t(x) = F_{t-1}(x) + f_t(x), \; t = 2, 3, 4... \tag{2.6}$$

where $F_{t-1}(x)$ is the cumulative prediction up to the previous step. Thus, the final prediction $\hat{y}$ for a given instance x after T rounds of boosting is the sum of the predictions from all the trees:

$$\hat{y}(x) = F_T(x) = \sum_{t=1}^{T} f_t(x), \tag{2.7}$$

where T represents the total number of trees constructed or the number of boosting rounds completed, and $f_t(x)$ is the output of the t^{th} tree. XGBoost is designed to focus on learning residuals, that is, the discrepancies between actual and predicted values in the current model. It uses these residuals as targets for subsequent trees, thereby iteratively minimizing errors and enhancing prediction accuracy. Through a gradient-based optimization approach, XGBoost methodically reduces the loss function, enabling each new tree to adapt to complex data patterns more effectively. Additionally, the algorithm incorporates regularization, penalizing over-complex trees, thereby preventing overfitting.

Through its sequential tree building, combined with gradient descent and regularization, XGBoost achieves high efficiency and accuracy, particularly in handling large and complex datasets. Its robustness to overfitting and its computational efficiency make it a powerful tool for a wide range of problems, including regression, classification, and ranking tasks [24].

2.6 Experiments

This section introduces the methodology adopted for training and evaluating the models in this study. The computational infrastructure utilized for conducting all experiments comprises a high-performance server equipped with dual NVIDIA A6000 GPUs, dual AMD EPYC 64-core CPUs, and 256 GB of RAM, providing robust computational power essential for handling complex calculations and large datasets.

2.6.1 Model Training

In this study, all DL models are trained for multi-label prediction. In contrast, for the three non-DL models, 238 binary classifiers are trained, one for each medication. This approach

is necessitated as the non-DL models do not inherently support multi-label prediction and are incapable of capturing time-series information effectively. Table 2.1 presents the databases utilized for each respective model.

Throughout the training process, a fixed 20% of the data is allocated as the test set for evaluating model performance, and a fixed 16% is designated as the validation set for fine-tuning. The remaining 64% is used for training. To examine scaling behavior, eleven percentages on a logarithmic scale, from as little as 0.12% to the entire 64%, are selected as training sets. With very small training sets, each medication's positive examples are seen infrequently, creating a test set over 150 times larger than the training set. This scenario tests the models' few-shot learning capabilities. Five-fold cross-validation is conducted for all models to ensure their reliability. Please note that the XGBoost model demands significantly more system RAM. Therefore, for a training size of 64%, this requirement exceeds a reasonable limit, leading to the exclusion of this experiment. For more details on the training schema, please refer to [15].

2.6.2 Model Evaluation

In this work, all models are evaluated with AUROC and AUPRC scores. These two metrics are essential in assessing multi-label prediction models, particularly in the context of imbalanced datasets [25].

- **AUROC**: This curve is a graphical representation that plots the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings. The TPR, also known as sensitivity or recall, is juxtaposed with the FPR, which is calculated as one minus the specificity. The AUROC serves as a measure of a model's capability to differentiate between classes. An AUROC value of 1 signifies perfect classification, whereas a value of 0.5 suggests performance equivalent to random guessing.

- **AUPRC**: The Precision-Recall Curve, on the other hand, plots precision against recall. Precision is defined as the ratio of true positives to all positive predictions, while recall is TPR. The AUPRC becomes particularly crucial in scenarios with class imbalances.

It concentrates on the classifier's performance in identifying the positive class, thereby being more sensitive to the detection of rare events. A higher AUPRC value indicates a more effective model in managing positive class predictions, which is of significant importance in fields such as medical diagnosis, where the cost of false negatives is high [26].

For a comprehensive comparison, both AUROC and AUPRC are calculated for each medication across all models.

Chapter 3: Results

3.1 PCA Dimension Reduction

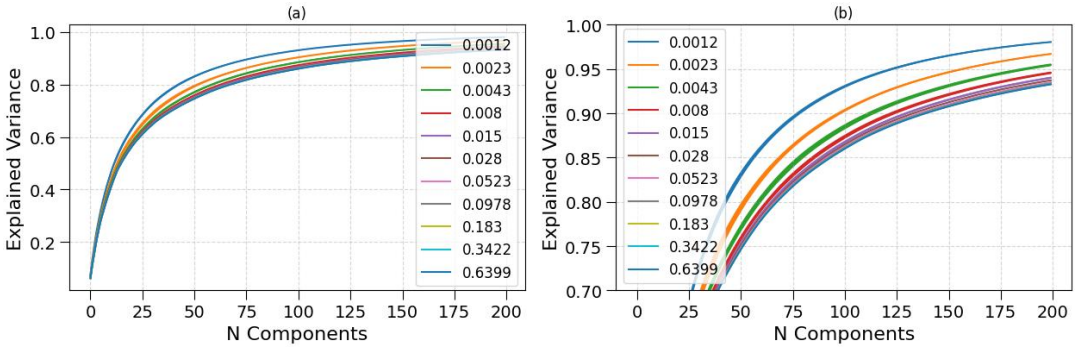

Figure 3.1: PCA explained variance at various training proportions: (a) illustrates the full variance range while (b) zooms in on the [0.7, 1.0] segment.

To preserve data integrity and optimize the database size for non-DL models, the number of principal components is determined to be 200. Fig 3.1 demonstrates that with 200 components, the explained variance exceeds 0.925 for all training proportions, and the curve appears to converge at this point. This indicates that our chosen configuration avoids significant information loss, and increasing the component count would only lead to an unnecessary expansion of the database size.

3.2 Performance over All Medications

Fig. 3.2 presents the performance of all evaluated models in this study, illustrating the average AUROC and AUPRC scores across different medications and varying training proportions. Notably, all models demonstrate a consistent scaling behavior: both AUROC and AUPRC scores increase with larger training proportions. The transformer model demonstrat-

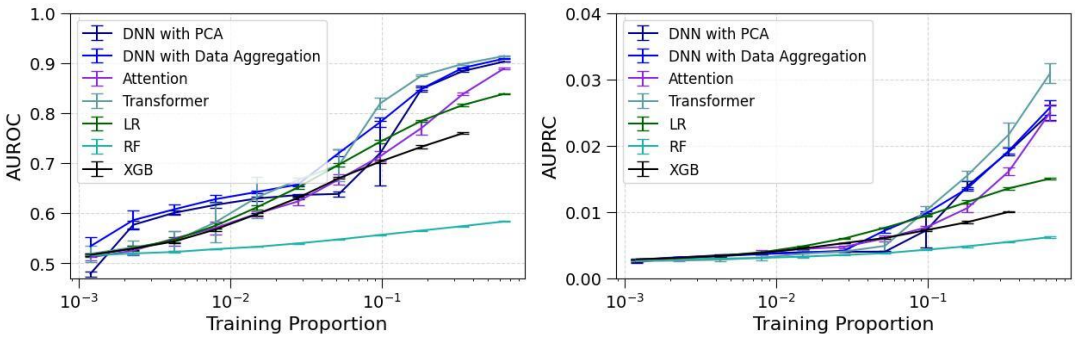

Figure 3.2: Average AUROC and AUPRC scores for all medications across various training proportions, calculated for each model. For the corresponding database used in each model, see Table 2.1.

ing the best performance requires a minimum of 10% of the data for training to achieve an average AUROC of over 0.8. Remarkably, when trained with less than 20% of the dataset, this model attains an AUROC of 0.88 and an AUPRC of 0.015. These metrics significantly exceed those of the best baseline model, LR, which achieves an AUROC of 0.77 and an AUPRC of 0.011. In general, non-DL baselines, though trained for each medication, are outperformed by all DL models. While XGBoost shows enhanced results through gradient boosting and effective sparse data management, LR consistently delivers superior performance. This is attributed to the prediction problem's binary nature. Among the four DL models, the attention model, despite leveraging the M' database, exhibits limited proficiency in capturing patterns within this time-series dataset. Furthermore, its performance is notably sensitive to training size, surpassing non-DL baselines only when utilizing the second-largest training dataset. The performance of the two DNN models is commendable, even when compared to the top-performing transformer model. This suggests that the temporal data aggregation and PCA for dimension reduction effectively reduce data size, enabling faster training while minimizing information loss.

In this imbalanced dataset, while AUROC scores are decent, AUPRC scores are less than ideal. This is mainly because the positive frequency does not exceed 0.2% for most medications beyond the top four frequent. (refer to [15] for details). Hence, for a random classifier,

the expected AUPRC is just 0.002 [26], indicating that extremely specific patterns are necessary to predict the use of a particular medication. Thus, an AUPRC of 0.01 is effectively five times better. This scenario is typical in complex critical care settings, where individual patient assessments are crucial [27, 28]. The difference between the AUROC and the AUPRC suggests that these models possess high sensitivity but acceptable precision. In practical terms, this means that when the model predicts a medication is needed, it requires further verification by a clinician. Conversely, if the model indicates no need for a medication, the prediction is highly reliable. To further examine which medications yield high AUPRC scores, an analysis of the AUROC and AUPRC scaling curves for the top-five medications is conducted.

3.3 Top-Five Medications

Fig. 3.3 and Fig. 3.4 demonstrate the scaling behavior of AUROC and AUPRC curves, offering a detailed evaluation of the performance of DL and non-DL models for the top five medications, as ranked by AUPRC scores, respectively. The performance of the top five medications—sodium chloride, potassium chloride, furosemide, magnesium sulfate, and aspirin—remains consistent across all models, though their ranking order may vary.

Across all DL models, top-five medication predictions show that the transformer model (see Fig. 3.3a) leads in performance. However, this lead is less pronounced compared to its performance averaged across 238 medications. In comparison with attention models, the transformer's AUPRC scores at higher training proportions are even marginally lower (Fig. 3.3b). Notably, the performance of the DNN models, both with temporal data aggregation and PCA (Figs. 3.3c and 3.3d), only decreases slightly. This further suggests that data engineering processes are particularly effective for medications that are easier to predict. These findings imply that for more frequently prescribed medications, complex models may not be necessary. And this result aligns well with intuitive data analysis principles.

Regarding non-DL models (Fig. 3.4), their performance, even for the top-five medications, remains unsatisfactory. The AUROC scores do not reach 0.8 at any training proportion. Given that the DNN with PCA confirms minimal information loss during data engineering, the lower

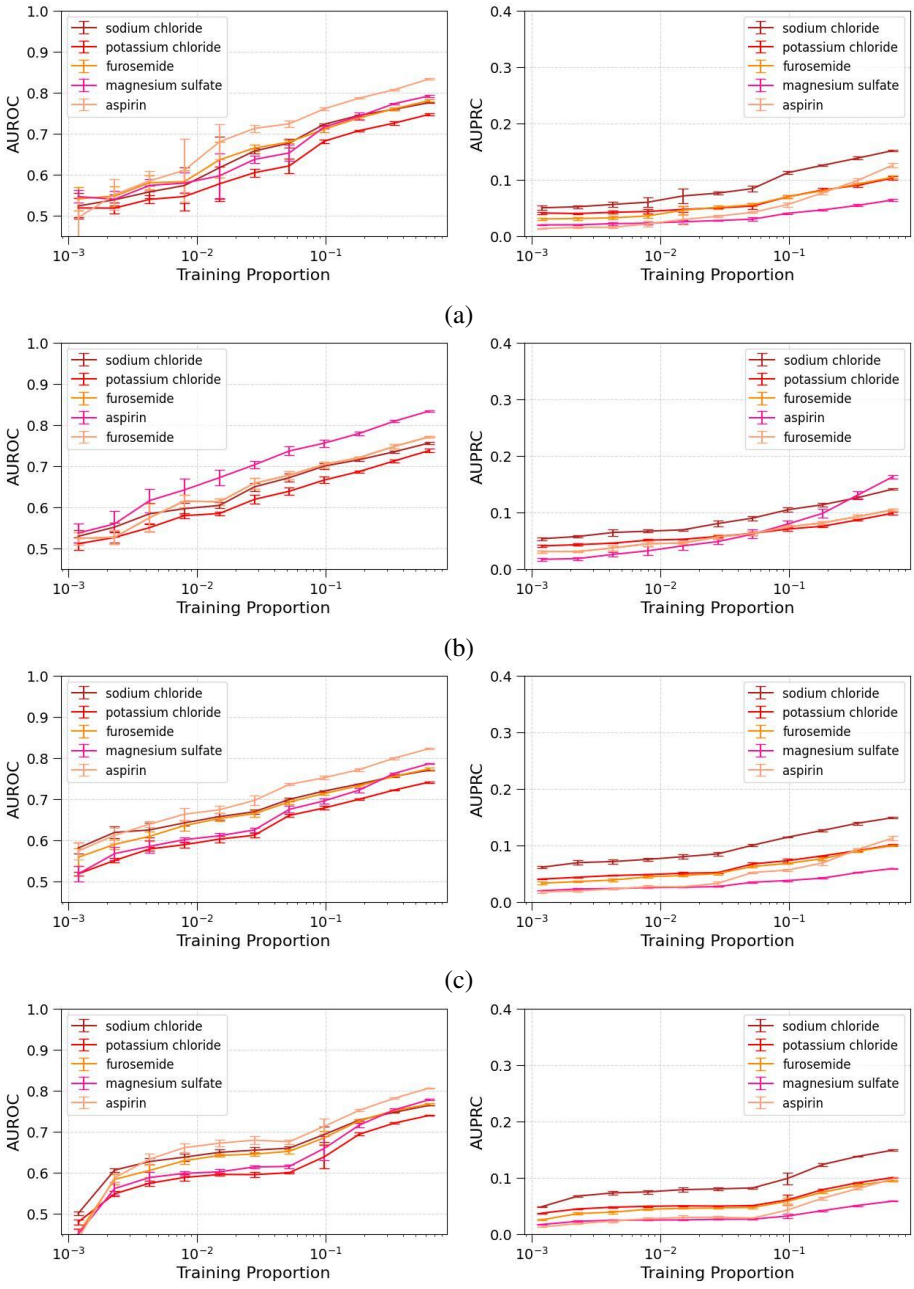

Figure 3.3: The AUROC and AUPRC scores at various training proportions for the top five medications of all DL models: (a) the transformer model utilizing matrix M', (b) the attention model with matrix M', (c) the DNN model incorporating matrix M'' which applies temporal aggregation, and (d) the DNN model with matrix M''', employing PCA for dimension reduction.

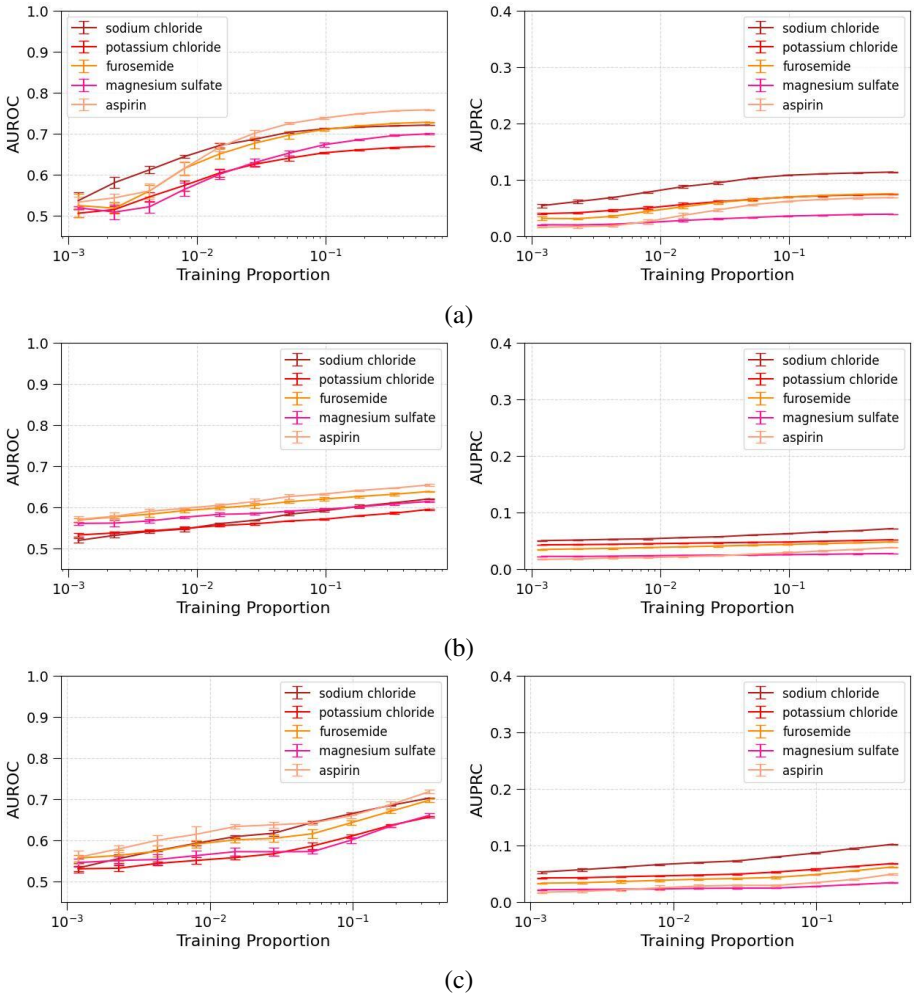

Figure 3.4: The AUROC and AUPRC scores at various training proportions for the top five medications of all non-DL models: (a) LR, (b) RF, and (c) XGBoost, all employing matrix M'''.

performance of non-DL models is mainly due to their inability to learn the underlying patterns in the database. This establishes that while sophisticated models are not essential for simpler medication prediction tasks, the use of DL models remains necessary.

Chapter 4: Conclusion

"What is the expected outcome of the patient?" is an ongoing research topic with notable progress. However, at the core of our research is the question: "Which medication(s) should be prescribed?". This question, complex but crucial, sheds light on the potential of DL techniques in supporting real-time decision-making in critical care. To answer this question, we introduced the `FIDDLE-Rx` framework, which realizes real-time medication intervention in resource-limited critical care environments. The `FIDDLE-Rx` encompasses several key components: (a) the `FIDDLE` model, adept at managing missing data and transforming the tabular ICU-EHRs into an efficient binary matrix representation; (b) a data engineering phase that enriches and compresses the dataset at the same time, links medical features with medication labels, thereby optimizing the database for faster model training and improved performance; (c) a final phase of training DL and non-DL models to provide actionable insights for clinicians; and (d) a scaling behavior analysis to determine the minimal sample size necessary for large-scale analysis. Utilizing just less than 20% of the database, the transformer model attains an AUROC of 0.88 and an AUPRC of 0.015. This result significantly outperforms the best non-DL baseline (LR), which achieves an AUROC of 0.77 and an AUPRC of 0.011. The gap between AUROC and AUPRC in our models indicates high sensitivity. Practically, this means model predictions of medication needs should be clinician-reviewed for accuracy, while their predictions against medication use are highly reliable. This highlights the models' potential in aiding clinical decisions, especially in preventing unnecessary prescriptions. The combination of standardized procedures, interpretable data engineering, and promising results in medication prediction positions `FIDDLE-Rx` towards few-shot multi-label medication prediction from ICU-EHRs. Further efforts should focus on enhancing the models' precision to advance the development of a more sophisticated medication recommendation system.

References

[1] M. R. Cowie *et al.*, "Electronic health records to facilitate clinical research," *Clinical Research in Cardiology*, vol. 106, pp. 1–9, 2017.

[2] M. A. Hendrickson, G. B. Melton, and M. B. Pitt, "The review of systems, the electronic health record, and billing," *Jama*, vol. 322, no. 2, pp. 115–116, 2019.

[3] W.-Q. Wei, P. L. Teixeira, H. Mo, R. M. Cronin, J. L. Warner, and J. C. Denny, "Combining billing codes, clinical notes, and medications from electronic health records provides superior phenotyping performance," *Journal of the American Medical Informatics Association*, vol. 23, no. e1, e20–e27, 2016.

[4] T. J. Pollard, A. E. Johnson, J. D. Raffa, L. A. Celi, R. G. Mark, and O. Badawi, "The eicu collaborative research database, a freely available multi-center database for critical care research," *Scientific data*, vol. 5, no. 1, pp. 1–13, 2018.

[5] A. E. Johnson *et al.*, "Mimic-iii, a freely accessible critical care database," *Scientific data*, vol. 3, no. 1, pp. 1–9, 2016.

[6] D. Shillan, J. A. Sterne, A. Champneys, and B. Gibbison, "Use of machine learning to analyse routinely collected intensive care unit data: A systematic review," *Critical care*, vol. 23, pp. 1–11, 2019.

[7] M. Greco, P. F. Caruso, and M. Cecconi, "Artificial intelligence in the intensive care unit," in *Seminars in Respiratory and Critical Care Medicine*, vol. 42, 2020, pp. 002–009.

[8] M. Ghassemi, T. Naumann, P. Schulam, A. L. Beam, I. Y. Chen, and R. Ranganath, "A review of challenges and opportunities in machine learning for health," *AMIA Summits on Translational Science Proceedings*, vol. 2020, p. 191, 2020.

[9] Y. Ouyang *et al.*, "Interpretable machine learning models for predicting in-hospital death in patients in the intensive care unit with cerebral infarction," *Computer Methods and Programs in Biomedicine*, vol. 231, p. 107 431, 2023.

[10] M. Lei *et al.*, "A machine learning-based prediction model for in-hospital mortality among critically ill patients with hip fracture: An internal and external validated study," *Injury*, vol. 54, no. 2, pp. 636–644, 2023.

[11] T. Huang, D. Le, L. Yuan, S. Xu, and X. Peng, "Machine learning for prediction of in-hospital mortality in lung cancer patients admitted to intensive care unit," *Plos one*, vol. 18, no. 1, e0280606, 2023.

[12] Z. Ye *et al.*, "The prediction of in-hospital mortality in chronic kidney disease patients with coronary artery disease using machine learning models," *European Journal of Medical Research*, vol. 28, no. 1, pp. 1–13, 2023.

[13] B. K. Beaulieu-Jones *et al.*, "Machine learning for patient risk stratification: Standing on, or looking over, the shoulders of clinicians?" *NPJ digital medicine*, vol. 4, no. 1, p. 62, 2021.

[14] Y. Si *et al.*, "Deep representation learning of patient data from electronic health records (ehr): A systematic review," *Journal of biomedical informatics*, vol. 115, p. 103 671, 2021.

[15] H. Wang, "Towards few-shot multi-label medication prediction in icu: A data-driven approach to enhanced icu-ehrs," Unpublished master's thesis, Columbia University, 2023.

[16] S. Tang, P. Davarmanesh, Y. Song, D. Koutra, M. W. Sjoding, and J. Wiens, "Democratizing ehr analyses with fiddle: A flexible data-driven preprocessing pipeline for structured clinical data," *Journal of the American Medical Informatics Association*, vol. 27, no. 12, pp. 1921–1934, 2020.

[17] M. M. Churpek, R. Adhikari, and D. P. Edelson, "The value of vital sign trends for detecting clinical deterioration on the wards," *Resuscitation*, vol. 102, pp. 1–5, 2016.

[18] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and intelligent laboratory systems*, vol. 2, no. 1-3, pp. 37–52, 1987.

[19] R. Bro and A. K. Smilde, "Principal component analysis," *Analytical methods*, vol. 6, no. 9, pp. 2812–2831, 2014.

[20] A. Vaswani *et al.*, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[21] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[22] S. Menard, *Applied logistic regression analysis*. Sage, 2002.

[23] L. Breiman, "Random forests," *Machine learning*, vol. 45, pp. 5–32, 2001.

[24] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.

[25] S. Raschka, "Model evaluation, model selection, and algorithm selection in machine learning," *arXiv preprint arXiv:1811.12808*, 2018.

[26] M. Yu, Y.-C. Tham, T. H. Rim, D. S. Ting, T. Y. Wong, and C.-Y. Cheng, "Reporting on deep learning algorithms in health care," *The Lancet Digital Health*, vol. 1, no. 7, e328–e329, 2019.

[27] M. Alameddine, K. N. Dainty, R. Deber, and W. J. B. Sibbald, "The intensive care unit work environment: Current challenges and recommendations for the future," *Journal of critical care*, vol. 24, no. 2, pp. 243–248, 2009.

[28] B. T. Thompson *et al.*, "Challenges in end-of-life care in the icu: Statement of the 5th international consensus conference in critical care: Brussels, belgium, april 2003: Executive summary," *Critical care medicine*, vol. 32, no. 8, pp. 1781–1784, 2004.