Interpretability, Fairness, and Data Scarcity in Machine Learning

by

Muhang (Tony) Tian

A Dissertation Presented to the

FACULTY OF COMPUTER SCIENCE DEPARTMENT

DUKE UNIVERSITY

In Partial Fulfillment of the Requirements for

GRADUATION WITH DISTINCTION IN COMPUTER SCIENCE

Dissertation Committee:

Cynthia Rudin, Ph.D.

Earl D. McLean, Jr. Professor of Computer Science

Anru Zhang, Ph.D.

Eugene Anson Stead, Jr. M.D. Associate Professor of Biostatistics & Bioinformatics

Brandon Fain, Ph.D.

Assistant Professor of the Practice in Computer Science

April 2024

Acknowledgments

I want to thank Professor Brandon Fain for introducing me to the beauty of Computer Science research during my sophomore summer year in the CS+ program and for being a supportive mentor for my academic life and choices outside of research. I would like to thank Professor Cynthia Rudin for providing the opportunity to work on interpretable machine learning with other excellent Duke undergraduate students in the Data Science Competition class, as well as the experience of working within her lab during the summer. I want to thank Professor Anru Zhang for providing me with guidance and an opportunity to conduct research on diffusion models and work with other interesting Duke undergraduate students.

I would also like to thank Duke University for its supportive, nurturing faculty and student population during my undergraduate research experiences and studies. This environment is crucial for me to grow as a thinker and learner and to gain my own research skills.

Lastly, I would like to thank my parents for moving from Kunming to Shanghai, a choice that allowed me to gain a better education and provided a chance for my admission to Duke University.

Table of Contents

List of Tables

List of Figures

Abstract

Digitization and the internet have made data extensively available in the current century. Machine learning, a computational and statistical technique capable of drawing insights from data, has demonstrated its capability to provide support for human-relevant tasks in pattern recognition, generative modeling, and agent control. Progress in machine learning has brought tremendous improvement in performance across various tasks aforementioned. Nevertheless, despite these advancements, challenges persist in implementing additional constraints on machine learning techniques in order to more appropriately support humankind. When applied to human-relevant tasks, it is vital for machine learning techniques to take into consideration their effect on social inequality and trustworthiness. Additionally, although data availability has increased, certain fields like healthcare contain barriers to data access that could thwart the development of machine learning. Motivated by these challenges, my research has concentrated on developing methods that address interpretability, fairness, and data scarcity within machine learning, aiming to refine their application and efficacy in human-relevant domains.

Chapter 1

Introduction

Due to digitization and the internet, data has become widely available in the current century. Machine learning, due to its ability to recognize patterns in data, has become a popular field of study due to its capability to make predictions, model data distributions, and control agents.

Nevertheless, despite the advances in machine learning, it remains a challenge to allow machine learning techniques to consider additional constraints in order to better support human-relevant tasks. For instance, while machine learning models could score outstanding predictive performance metrics, they could possibly harm society due to their inherent unfairness and inequity beliefs drawn from the training data, strengthening the inequality already present in the current society. Furthermore, while current complex machine learning models could provide outstanding performance measured by evaluation metrics, they are often unfathomable for humans to comprehend the reasons behind their predictions. Their "black-box" nature implicitly could inhibit their usage in high-stake settings, such as those in healthcare and medicine, where the cost of fully trusting an opaque model that could make false predictions is too high to bear. Moreover, while data availability has increased in today's age, data access is strictly

controlled in some domains, such as medicine and healthcare, due to privacy concerns. In these settings, it is difficult to develop machine learning models or conduct research due to a lack of data.

Therefore, to empower machine learning techniques, it remains a crucial task to consider interpretability, fairness, and tackling data scarcity challenges. Ideally, we would want our techniques to create equality between its users, provide a means of transparency that informs people about the reason for predictions, and have a level of tolerance for data scarcity. Motivated by these aspirations, my past research during my undergraduate studies has focused on designing techniques that address each of these factors in machine learning and exploring their potential solutions.

In this paper, I will discuss techniques for incorporating fairness considerations within the context of reinforcement learning, interpretability concepts within risk prediction for critical care patients, and synthetic data generation techniques to tackle data scarcity issues in electronic health records.

Chapter 2

Fairness in Reinforcement Learning

2.1 Introduction

Reinforcement learning (RL) is a sub-field of machine learning that focuses on training an effective policy of an agent. To understand why fairness could be important in RL, let us see the following hypothetical example:

Assuming a delivery company would like to deploy an RL agent to optimize for completing deliveries, and the agent successfully optimizes this objective and contributes to increased revenues of the company. After some time, the company heard complaints that services to some delivery locations had become worse than before the agent deployment. To seek a solution, the company's engineers fine-tuned the reward weights associated with deliveries to those locations but only found that other locations are now being neglected.

This is a case where data-driven algorithms may be generally performant but fail on structured subsets of input. The engineers did take a correct approach well-known in RL — "reward shaping" on individual objectives to achieve the desired behavior of achieving high delivery rates across *all* customer locations. However, standard RL,

where the reward signal is a scalar value, and the goal is to maximize cumulative discounted return, might naturally learn a policy that prioritizes "easy" to optimize regions (such as clusters of many tightly packed locations with many deliveries) at the expense of more difficult ways to achieve reward. Furthermore, because standard techniques rely on learning a *stationary* policy, the agent would continue to prioritize the same customers day after day. Solving this problem could involve problem-specific fine-tuning of rewards.

Compared with previous works, I take a different approach by studying nonlinear welfare optimization in the context of Multi-objective Reinforcement Learning (MORL). A Multi-Objective Markov Decision Process (MOMDP) is a Markov Decision Process where rewards are vectors instead of scalars. The elements in this vector can be different criteria we would like to consider, like cost and time, or as individual utilities of *"users"* to whom the learning agent should be *fair*. In the example above, the customers are the users, and the vector reward tracks how well the learning agent performs for each user in terms of delivery rates.

Solving a MOMDP involves maximizing some function of the cumulative reward vector. Due to the linearity of expectation, linear functions such as weighted arithmetic mean are the most straightforward to use. However, this approach has its limitations since any particular weight may result in policies undesirable from a fairness perspective since any linear function may ignore the utility of some users. For instance, for equal weights, a policy that gives user 1 a utility of 10 and user 2 a utility of 0 is preferred over another policy that yields a utility of 4 for both. Thus, I aim to study a more general class of *welfare* functions with a particular emphasis on *nonlinear* welfare functions that optimize for fairness and efficiency.

Optimizing a nonlinear welfare function in a MOMDP is a challenging task. Firstly,

the Bellman optimality principles [1][2] no longer hold, and stationary policies (where actions depend on the current state and not on past history) are no longer necessarily optimal. Nevertheless, this result is quite intuitive for fairness, as the notion of equality depends on the amount of resources already allocated to the users.

A fair MORL agent could be useful in real-world settings as well. In telecommunications, one may wish to allocate bandwidth that balances the quality of service across various locations. Moreover, in autonomous driving, one may want to balance vehicle speed and passenger comfort [3].

In this chapter, I would like to present my work, namely *Welfare Q-learning*, that uses a non-stationary action selection policy along with non-linear updates to approximate the welfare maximization objective effectively. I also present some additional discussions on the specific reason for optimizing for expected welfare rather than the welfare of expectation.

2.2 Related Work

Multi-objective reinforcement learning (MORL) algorithms include *single-policy* and *multi-policy* methods [4]. Single-policy methods use a scalarization function to reduce the problem to scalar optimization for a single policy. The simplest form is linear scalarization, applying a weighted sum on the Q vector [5].

Multi-policy methods search for a set of policies that approximate the Pareto frontier of the problem. For instance, the *convex hull value-iteration algorithm* [6] computes the deterministic stationary policies on the convex hull of the Pareto front. *Pareto Q-learning* [5] integrates temporal difference algorithms with Pareto dominance relations to learn a set of Pareto dominating policies. *Stochastic mixture policy* [7] combines

multiple deterministic base policies with a convex combination, choosing a base policy with a given probability at the start of each episode. Given that the size of the Pareto frontier may grow exponentially with the dimensionality of the problem, I mainly focus on single-policy methods with nonlinear scalarizing functions (in this case, the welfare functions).

Fairness in Reinforcement Learning has been recently considered, beginning with [8] in a scalar setting. More directly related to our work, [9] investigated the (Deep) MORL problem of learning a fair policy to optimize the *Generalized Gini Social Welfare function* using nonlinear scalarization. [10] studied a similar problem and considered maximizing concave welfare functions generally and Nash welfare specifically, showing an optimal approach to optimizing the welfare of expected rewards in the tabular setting and an extension to the function approximation setting.

My work differs from these in two major ways: (1) I seek to optimize the expected welfare, rather than the welfare of expected rewards, which is fundamentally more challenging computationally, and the welfare of expected rewards could also tolerate unfair outcomes within individual trajectories. (2) The agent learns a non-stationary policy, as stationary policies may be far from optimal for optimizing expected (nonlinear) welfare.

I formulate the welfare functions based on the resource allocation literature [11]. One canonical example of a fair welfare function is the Nash Social Welfare (NSW) function. It derives from Nash's solution to the bargaining game [12] and its n-player extension [13]. Its log transform is known as the proportional fairness objective. More recent studies have shown NSW maximization provides outstanding fairness guarantees when allocating both divisible and indivisible goods [14].

2.3 Preliminaries

A Multi-objective Markov Decision Process (MOMDP) consists of a finite set $\mathcal{S}$ of states, a starting state $s_1 \in \mathcal{S}$, a finite set $\mathcal{A}$ of actions (let $\mathcal{A}(s)$ denote the subset of actions available in state s), and probabilities $\mathcal{P}_{a,s,s'} \in [0,1]$ that determine the probability of transitioning to state s' from state s after taking action a. Probabilities are normalized so that $\sum_{a \in \mathcal{A}(s),s'} \mathcal{P}_{a,s,s'} = 1$ for all s. We also have a reward function $\boldsymbol{R}(s,a) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^n$ for taking action a in state s.[1] Some states may be *terminal*, meaning they transition only to themselves and yield $\mathbf{0}$ reward.

Each of the n dimensions of the reward vector correspond to one of the multiple objectives that are to be maximized. At each time step t, the agent observes state $s_t \in \mathcal{S}$, takes action $a_t \in \mathcal{A}(s_t)$, and receives a reward vector $\boldsymbol{r}_t = \boldsymbol{R}(s_t, a_t) \in \mathbb{R}^n$. The environment, in turn, transitions into s_{t+1} with probability $\mathcal{P}_{a_t,s_t,s_{t+1}}$. Where clear from context, we will often omit the subscript and simply write the immediate reward vector as $\boldsymbol{r}$.

A *trajectory* is a sequence of state, action, reward tuples $\tau = (s_1, a_1, \boldsymbol{r}_1), ..., (s_T, a_T, \boldsymbol{r}_T)$. A trajectory that begins in the starting state s_1 and ends in a terminal state defines an *episode*. For a discounting factor $\gamma \in [0,1)$, the *discounted cumulative return* of a trajectory is the vector

$$\boldsymbol{G}(\tau) = \sum_{t=1}^{\infty} \gamma^{t-1} \boldsymbol{r}_t.$$

A *stationary policy* is function $\pi(a \mid s) : \mathcal{S} \times \mathcal{A} \to [0,1]$ that forms a probability distribution such that $\sum_{a \in \mathcal{A}(s)} \pi(a \mid s) = 1$ for all s. Such a policy is *stationary* since the probability with which an action is selected depends only on the current state. More generally, a policy (not necessarily stationary) is a function $\pi(a \mid \tau, s)$ that may

[1] For simplicity of exposition, we assume rewards are deterministic.

additionally depend on a given trajectory (intuitively, the history prior to reaching state s).

An *action value function* is defined as the expected total reward starting from s, taking action a, and following policy π thereafter:

$$\boldsymbol{q}_\pi(s,a) := \mathbb{E}_{\tau\sim\pi}\left[\sum_{k=0}^{\infty}\gamma^k \boldsymbol{r}_{t+k} \;\middle|\; s_t = s, a_t = a\right].$$

Most *value-based* approaches aim to solve the learning problem by finding an estimate of $\boldsymbol{q}_\pi(s,a)$. Such an estimate is denoted as $\boldsymbol{Q}_\pi(s,a)$.

2.4 Methodology

2.4.1 Expected Welfare Maximization

In contrast to some prior work mentioned in Section 2.2 [9, 10], I aim to optimize the expected welfare objective $\mathbb{E}_{\tau\sim\pi}\big[W\big(\boldsymbol{G}(\tau)\big)\big]$ rather than the welfare of the expectation $W\big(\mathbb{E}_{\tau\sim\pi}[\boldsymbol{G}(\tau)]\big)$. Using *Jensen's inequality*, the following can be obtained [15]:

$$\mathbb{E}_{\tau\sim\pi}\big[W\big(\boldsymbol{G}(\tau)\big)\big] \leq W\big(\mathbb{E}_{\tau\sim\pi}[\boldsymbol{G}(\tau)]\big). \tag{2.1}$$

Thus, I aim to optimize for the lower bound (which is also a more computationally challenging objective) in order to avoid treating policies as "fair" that are unfair in every particular trajectory and satisfy fairness only across trajectories on average. This idea can be illustrated with the following toy example:

Consider MOMDP in Figure 2.1 with $n = 2$ users. Assume we aim to learn a policy that maximizes NSW. There is a stochastic policy π_1 that yields discounted cumulative

reward of $(1, 0)$ with probability 0.5 and $(0, 1)$ with probability 0.5. There is another deterministic policy π_2 that yields $(0.5 - \epsilon, 0.5 - \epsilon)$ (where $\epsilon > 0$ is small). The NSW of expected reward under π_1 is 0.5, even though with probability 1, the NSW of every trajectory generated by π_1 is 0. In contrast, the NSW of π_2 is always $0.5-\epsilon$. Therefore, by choosing to maximize the expected welfare, our agent would prefer π_2.

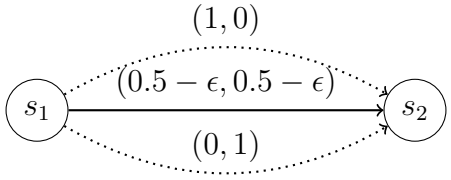


Figure 2.1: Example MOMDP. Dotted lines represent trajectories generated by π_1, solid line for π_2

This example shows the intuition for the algorithm is designed to maximize $\mathbb{E}_{\tau\sim\pi}\big[W\big(\boldsymbol{G}(\tau)\big)\big]$. The goal is to find a policy that generates trajectories with high expected welfare, a stronger property than generating high welfare of expected rewards. However, this objective is also more computationally challenging. In fact, it is *APX-hard* to optimize a policy for NSW of cumulative returns, even under a deterministic environment [16]. This result is obtained via a reduction from the problem of allocating indivisible goods.

2.4.2 Algorithm

I now present the algorithm show in Algorithm 1, *Welfare Q-Learning*, which implements a variant on *Q-learning* [17], a model-free temporal-difference learning algorithm [18]. The high-level intuition for temporal-difference learning is to update the Q-values on the next timestamp estimate in order to converge to that value and gradually close the temporal difference. The algorithm differs in two major ways from standard Q-learning.

1. Q table updates are chosen to maximize the (potentially) nonlinear welfare func-

tion W, and each value $\boldsymbol{Q}(s, a)$ is a vector in $\mathbb{R}^n$ corresponding to an estimate of the future reward vector possible that maximizes welfare.

2. Behavior policy for action selection is non-stationary. We keep track of the discounted cumulative return vector $\boldsymbol{r_{acc}}$ within a trajectory until the current time stamp and select the action that maximizes total estimated welfare, including that already accumulated and future estimates.

Algorithm 1 *Welfare Q-Learning*

Parameters: Learning rate $\alpha \in (0, 1]$, Discount factor $\gamma \in [0, 1)$, exploration rate $\epsilon > 0$, welfare function W
Require: Initialize $\boldsymbol{Q}(s, a)$ for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$ arbitrarily except $\boldsymbol{Q}(\bar{s}, \cdot) \leftarrow 0$ for terminal states $\bar{s}$
for each episode **do**
    Initialize $s \leftarrow s_1$, $\boldsymbol{r_{acc}} \leftarrow \boldsymbol{0}$, $c \leftarrow 0$
    repeat ▷ each step in an episode

$$a \leftarrow \begin{cases} \text{a uniform random action} & \text{with } \Pr(\epsilon) \\ \arg\max_{a'} W\left(\boldsymbol{r_{acc}} + \gamma^c \boldsymbol{Q}(s, a')\right) & \text{otherwise} \end{cases}$$

        Take action a, observe $\boldsymbol{r}, s'$
        $a^* \leftarrow \arg\max_a W[\gamma \boldsymbol{Q}(s', a)]$
        $\boldsymbol{Q}(s, a) \leftarrow \boldsymbol{Q}(s, a) + \alpha[\boldsymbol{r} + \gamma \boldsymbol{Q}(s', a^*) - \boldsymbol{Q}(s, a)]$
        $s \leftarrow s'$
        $\boldsymbol{r_{acc}} \leftarrow \boldsymbol{r_{acc}} + \gamma^c \boldsymbol{r}$
        $c \leftarrow c + 1$
    until s is terminal
end for

2.5 Results

To benchmark *Welfare Q-Learning*, I designed a simulation environment shown in Figure 2.2[2]. In this grid world, the agent is a taxi driver whose goal is to deliver passengers from their origins to their destinations. There are n origin-destination pairs,

[2]The implementation is available at https://github.com/MuhangTian/Fair-MORL-AAMAS

one for each dimension of reward, and the agent earns reward in that dimension when dropping off a passenger from that origin-destination pair. There are an unlimited number of passengers for each origin-destination pair, but the taxi can only take one passenger at a time. This constraint enforces objectives to be conflicting, thus the agent's fairness performance becomes more important—it should provide its delivery service to each origin successfully and fairly over time, without ignoring origins that are more difficult to deliver (such as number 3 origin/destination pair in Figure 2.2).

The results demonstrate that (a) *Welfare Q-Learning* is effective in finding policies with high expected welfare compared with other baselines, (b) the rate of convergence depends on n, the dimensionality of the reward space, and (c) linear scalarization and mixture policies are generally inadequate for optimizing fair welfare functions. All results for all algorithms are obtained using an average of NSW and utilitarian welfare of $\boldsymbol{r}_{acc}$ for each episode over 50 runs. For all the experiments, each unit on the x-axis corresponds to an episode, which equals to 10000 timesteps or action selections in the environment. The duration of a timestep is the same across all methods.

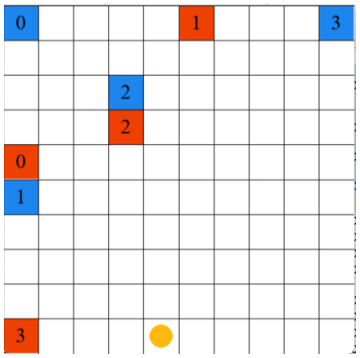
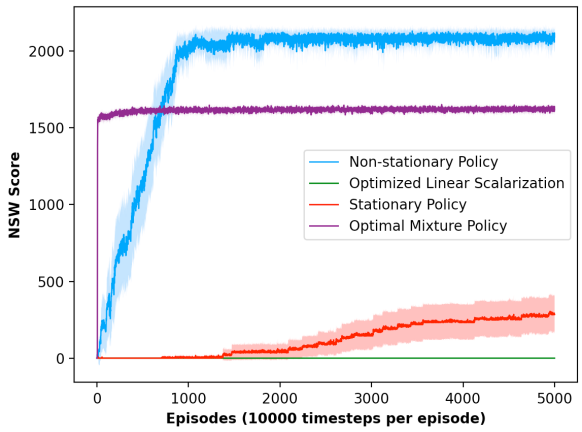
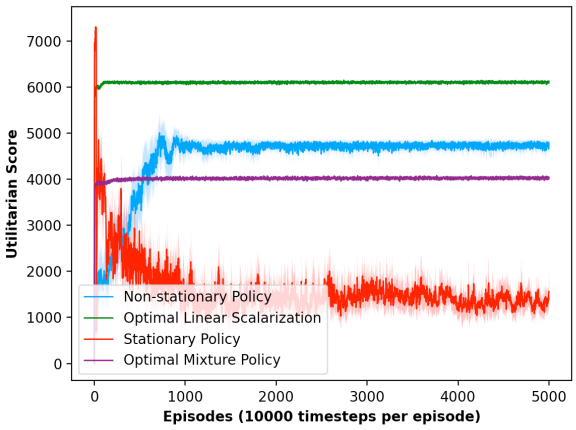


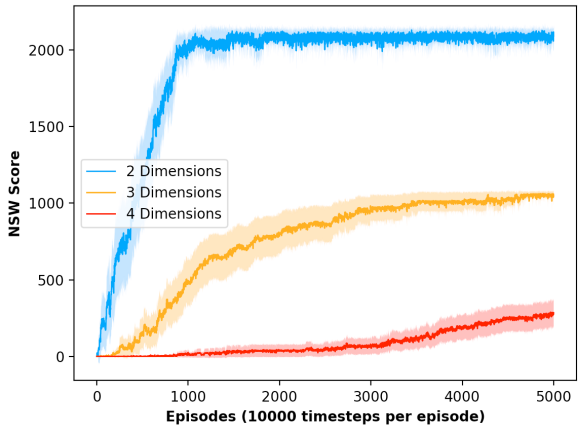
Figure 2.2: Taxi Simulation Environment



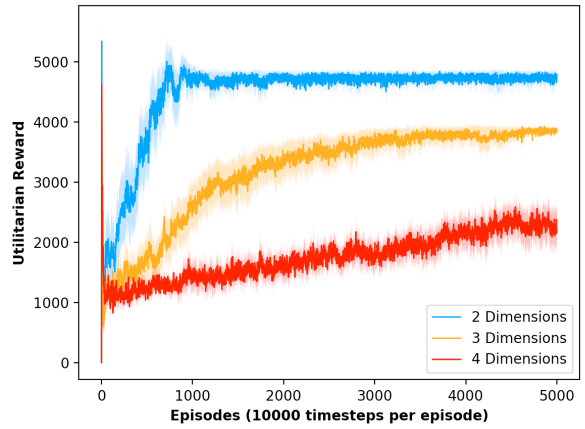
(a) Online Performance (NSW)



(b) Online Performance (Utilitarian)



(c) Taxi with Different Dimensions (NSW)



(d) Taxi with Different Dimensions (Utilitarian)

Figure 2.3: Experiment Results for Taxi Environment. Non-stationary Policy is Welfare Q-Learning

2.5.1 Metrics, Methods, and Baseline Algorithms

All of the experiments attempt to optimize NSW (results for other welfare functions are provided in the Appendix). I measure the NSW function on $\boldsymbol{r}_{acc}$ earned thus far. NSW satisfies all necessary desiderata for fairness plus scale invariance, and is an intermediate welfare function between the extremes of egalitarian and utilitarian social welfare [16]. For comparison, I also show utilitarian social welfare (the arithmetic mean of reward vectors) alongside NSW.

Since the geometric mean can be numerically unstable, I implement the log transform of NSW as the objective in practice. That is, instead of maximizing $\text{NSW}(\boldsymbol{R}) = (\prod_i^n R_i)^{1/n}$, I equivalently maximize $\sum_i^n \ln(R_i + \lambda)$, where $\lambda > 0$ is included as a smoothing factor in case $R_i = 0$. Due to the nature of the NSW function, the NSW of rewards with negative elements is undefined or alternatively can be defined as $-\infty$. Thus, the scope of this section is restricted to policies that yield all non-negative accumulated returns.

2.5.1.1 Baseline Algorithms

Welfare Q-Learning is compared against three baselines.

1. *Optimal Linear Scalarization.* A simple MORL technique is to apply linear scalarization on the Q-table [19]. Given weights $\boldsymbol{w} \in \mathbb{R}^n$, where $\sum_{i=1}^n w_i = 1$ for n objectives, let $SQ(s,a) = \sum_{i=1}^n w_i Q(s,a)_i$, where $Q(s,a)_i$ is the Q-value for i^{th} objective. For each time step, $SQ(s,a)$ is treated by the algorithm as the objective to perform both action selection with ϵ-*greedy* and learning updates of $\boldsymbol{Q}(s,a)$. The weights $\boldsymbol{w}$ are chosen using that performed best on the NSW objective as determined by a grid search through combinations of $\boldsymbol{w}$. This baseline is included to demonstrate the limitations of linear scalarization on nonlinear objectives and show the importance of our nonlinear learning updates in Algorithm 1.

2. *Stationary Policy.* Algorithm 1, *Welfare Q-Learning*, learns a particular Q-table corresponding to a (potentially) nonlinear welfare function, then performs non-stationary action selection. By contrast, I also show the results if one applies a stationary ϵ-greedy action selection on the same learned Q-table. That is, the stationary policy algorithm does not consider $\boldsymbol{r}_{acc}$ in its action selection. This baseline is included to demonstrate the importance of non-stationary action selection.

3. *Optimal Mixture Policy.* [7] proposed the idea of combining multiple Pareto Optimal base policies into a single mixture policy. I chose our base policies as those that optimize each dimension of the reward vector independently. The algorithm then uses one of these policies for I time steps, then switches to the next. To determine the optimal value of I for optimizing NSW, grid search is used. I use the resulting optimal value I^*. This baseline examines the effectiveness of intuitive approaches (combining optimal policies for each user) for optimizing fairness.

2.5.2 Results and Discussions

Results are shown in Figure 2.3. *Welfare Q-Learning* achieves the maximum average NSW score among all the algorithms and still manages to achieve the second-highest utilitarian score. Non-stationary policy outperforms the stationary policy on the same Q-table for both NSW and utilitarian score. No Note that a stationary policy that optimizes NSW on this environment must essentially make a large loop, always taking each origin-destination pair in turn, whereas a non-stationary policy can selectively optimize a single origin-destination pair for several time steps before switching to another pair.

Linear scalarization has the lowest average NSW since there simply does not exist a set of weights that would produce accumulated returns in all dimensions. It achieves highest utilitarian score since it favors to complete delivery for closest origin/destination pairs (such as index 2 pair in Figure 2.2). The mixture policy performs generally well but slightly lower than that of *Welfare Q-Learning*, this is because an optimal fair policy in this environment does have the structure of alternating between optimizing on different dimensions at a time. Although the mixture policy converges quickly (each dimension independently is very easy to optimize), such performance is also subject to

finding the optimal interval for the taxi environment I^* (227 timesteps) via a search through the parameter space, which involves a computational cost not reflected by the figures.

For *Welfare Q-Learning*, there is an inverse correlation between the dimensionality of the reward space n and the rate of convergence, as shown in Figure 2.3d. A possible explanation is that the increase in dimensionality increases the size of the Q-table, which is of size $|\mathcal{S}| \times |\mathcal{A}| \times n$, thus more updates are needed to converge.

2.6 Appendix

2.6.1 Taxi Environment Descriptions

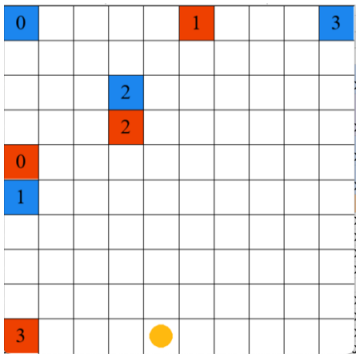


Figure 2.4: Visualization of Taxi grid world, orange circle is the taxi, origins are blue squares, destinations are red squares, with numbers indicating the corresponding origin and destination pairs

1. State space: contains information about location of taxi on the grid, whether there is passenger in taxi, destination of the passenger in taxi
2. Action space: move north, south, east, west, pick passenger, drop passenger
3. Reward function:

$$\boldsymbol{R_t} = \begin{cases} \mathbf{0} & \text{if } a_t \in \{\text{north, south, east, west }\} \text{ or } a_t \text{ is a valid pick or drop} \\ \mathbf{-10} & \text{if invalid pick or drop is performed} \\ R_i = 30, R_j = 0, \forall j \neq i & \text{if passenger from origin } i \text{ in taxi is delivered correctly} \end{cases}$$

2.6.2 Experimental Results for Other Welfare Functions

Experiments are also performed for *Welfare Q-Learning* based on P-welfare and *egalitarian* welfare functions (Figure 2.5). We chose a range of values of P between $[-1, 1]$ and recorded each of its performances with NSW and utilitarian score.

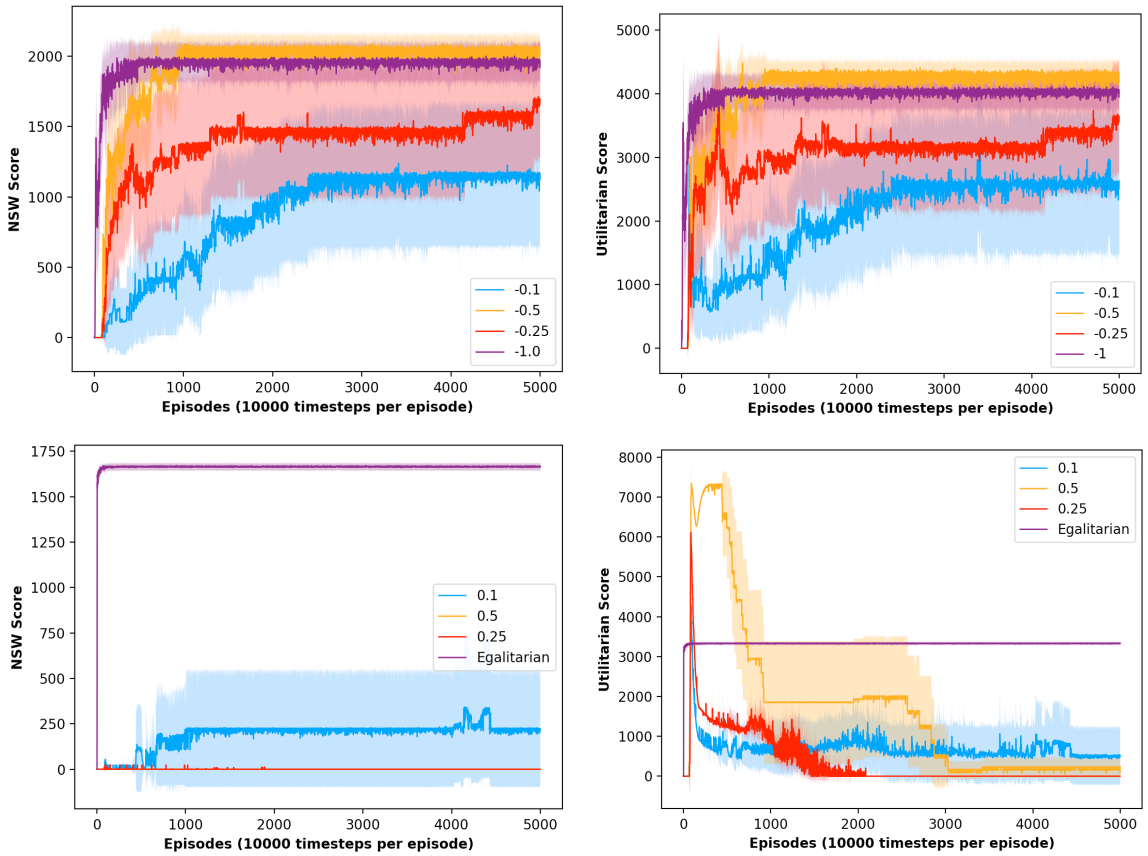


Figure 2.5: Experimental Results for Other Welfare Functions

Chapter 3

Interpretable Machine Learning for Critical Care

3.1 Introduction and Related Work

In-hospital mortality risk prediction is an essential task in medical decision-making [20, 21, 20, 22], supporting medical practitioners to better estimate patients' states and allocate resources appropriately for treatment and triaging [23, 24, 25].

Mortality is often estimated with *severity of illness risk scores*, where, first, each feature is transformed into an integer-valued component function based on its degree of deviation from normal values, and then a nonlinear function transforms the sum of component functions into an estimate of risk. Risk scores are designed to be easy to comprehend, troubleshoot, and use in practice. A method for constructing more accurate (but still interpretable) severity of illness risk scores could save lives and assist with better allocation of resources.

The severity of illness risk scores have been constructed in various ways since the early 1980's, starting with the APACHE [26], SOFA [27, 28], APACHE II [29], and SAPS

[30] scores, as well as the more recent APACHE IV [31] score. All of these scores were built using a combination of basic statistical techniques and domain expertise. Statistical hypothesis testing was generally used for variable selection, and techniques like logistic regression and locally weighted least squares [32] were often used for combining variables. This process left many manual choices for analysts: at what significance level should we stop including variables? Of the many features selected by hypothesis testing, how should we choose which ones would be included? How should the cutoffs for risk increases for each variable be determined? How do the risk scores from logistic regression become integer point values that doctors can easily sum, troubleshoot, and understand? While a variety of heuristics have been used to answer these questions, ideally, the answers would be determined automatically by an algorithm that optimizes predictive performance. Since humans, even equipped with heuristics, are not naturally adept at high-dimensional constrained optimization, it is particularly important that these models are *sparse* in the number of features so they are easy to calculate in practice.

Recently, more modern statistical and machine learning (ML) approaches have been used to create interpretable models for predicting mortality without the need for manual intervention. Specifically, the OASIS score [33] was built using a genetic algorithm [34] to select predictive variables, particle swarm optimization [35] to determine integer scores for variables' deciles, and logistic regression to transform integer scores into probabilities. However, genetic algorithm and particle swarm optimization approaches can be insufficient, leading to the possibility of improved performance using other techniques.

State-of-the-art "black box" ML approaches have been applied to the mortality risk prediction, aiming to improve predictive performance [36, 37, 38, 39, 40, 41]. For

instance, OASIS+ researchers [37] used a variety of black box ML algorithms (such as random forest [42] and XGBoost [43]) on OASIS features to develop ML models that mostly outperform other severity of illness scores such as OASIS and SAPS II. The black box models combine variables in highly nonlinear ways and are difficult to troubleshoot and understand in practice, which is why, to the best of my knowledge, OASIS+ models have not been adopted for mortality risk prediction in ICUs.

In this chapter, I introduce GroupFasterRisk— an interpretable machine learning algorithm capable of producing a set of diverse, high-quality risk scores — to generate severity of illness scores. GroupFasterRisk automates the entire process of feature selection, cutoffs for risk increases, and integer weight assignments. This approach optimizes more carefully than the approach of OASIS and another risk-score generation method called AutoScore [44], is much more scalable than its predecessor RiskSLIM [45, 46], and is more customizable than its predecessor FasterRisk [47]. GroupFasterRisk optimization process yields higher-quality interpretable models than competitors; in fact, I also show in the results section that its models are as performant as black box ones.

3.2 Methodology

3.2.1 Experimental Setup

3.2.1.1 Datasets

I consider the Medical Information Mart for Intensive Care III (MIMIC III) [48] and the eICU Collaborative Research Database (eICU) [49] datasets. Training of the models is performed on MIMIC III. I selected a subset of 49 features (including physiological measurements, lab measurements, and patient comorbidities) from the union of features

in existing severity of illness scores based on a ranking by AUROC. A transformation of continuous and categorical features into binary ones is applied, and indicator variables are used to provide information on whether the missing values are known. To test for generalization, the eICU dataset is used for out-of-distribution (OOD) testing.

3.2.1.2 Predictive Metrics

In this chapter, the area under the receiver-operating characteristic curve (AUROC) and the area under the precision-recall curve (AUPRC) are adopted as the metrics for predictive accuracy. Since the datasets are highly imbalanced, AUROC alone may not accurately capture the performance of models on the minority class (expired patients) [50]; AUPRC is used as an additional evaluation metric to provide a more complete view of the predictive accuracy.

3.2.1.3 Sparsity Metrics

A model's sparsity is informally defined as a way of measuring the model's size. For linear models such as logistic regression, explainable boosting machine (EBM) [51], AutoScore [44], and GroupFasterRisk, sparsity is the total number of coefficients, intercepts, and multipliers. For tree-based models like XGBoost [52], AdaBoost [53], and Random Forest [42], sparsity is the number of splits in all trees.

3.2.1.4 Calibration Metrics

High AUROC and AUPRC do not ensure that the model precisely estimates the risk probability. This is because they are rank statistics [54]. To evaluate the reliability of the predictions, I use the Brier score, Hosmer-Lemeshow χ^2 statistics (HL χ^2), and the standardized mortality ratio (SMR) [55, 56]. C-statistics is used for HL χ^2, calculated from deciles of predicted probabilities. Paired t-test is used for statistical testing.

3.2.2 Algorithm

Consider a dataset $\mathcal{D} = \{1, \tilde{\boldsymbol{x}}_i, y_i\}_{i=1}^n$, where $y_i \in \{0, 1\}$ is a label, $\tilde{\boldsymbol{x}}_i \in \mathbb{R}^p$ is a binarized feature vector, $\boldsymbol{x}_i \in \mathbb{R}^q$ is the raw feature vector, and 1 is added for ease of notation and represents an intercept. The set of feature indices $\{1, ..., p\}$ is arbitrarily partitioned into Γ disjoint sets (groups), denoted as $\{G_k\}_{k=1}^{\Gamma}$. Let $\mathcal{D}/m = \{1/m, \tilde{\boldsymbol{x}}_i/m, y_i\}_{i=1}^n$ be a scaled dataset, scaled by a multiplier m, $m > 0$ (m will be learned by the algorithm). Consider hypothesis space of linear models $\boldsymbol{w}^\top \tilde{\boldsymbol{x}} + w_0$, where $\boldsymbol{w} \in \mathbb{R}^p$ and $w_0 \in \mathbb{R}$. I denote $\boldsymbol{w}_{G_k} \in \mathbb{Z}^{|G_k|}$ as entries in $\boldsymbol{w}$ that belong to a group G_k.

The problem of creating risk scores is formulated as an optimization problem in Equation (3.1). The goal is to obtain *integer-valued* solutions of a *sparse* (ℓ_0 regularized) logistic regression under sparsity, group sparsity, and box constraint. A solution is denoted as $(\boldsymbol{w}, w_0)$. The sparsity constraint λ (Equation (3.1b)) is the number of non-zero elements in the solution vector $\boldsymbol{w}$ and directly controls the model complexity. Group sparsity constraint γ (Equation (3.1e), where $\mathbb{I}\{\cdot\}$ denotes the indicator function) allows users to control the number of partitions on the features. Box constraint $(\boldsymbol{a}, \boldsymbol{b})$, where $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^p$, (Equation (3.1c)) allows users to limit the solution values to their desired range, i.e., $w_j \in [a_j, b_j]$.

Overall, the problem of computing an integer-valued linear model is NP-hard [47]. However, we can solve it by finding a good approximate solution and using m as a multiplier to do so. While $(\boldsymbol{w}, w_0)$ must be integer-valued, the product $\left(\boldsymbol{w}^\top \tilde{\boldsymbol{x}}/m + w_0/m\right)$ can be real-valued. Therefore, I optimize logistic loss in for real-valued solution $\left(\boldsymbol{w}^\top \tilde{\boldsymbol{x}}/m + w_0/m\right)$ (Equation (3.1a)).

$$\min_{\boldsymbol{w}, w_0, m} \mathcal{L}(\boldsymbol{w}, w_0, \mathcal{D}/m) = \sum_{i=1}^{n} \log\left(1 + \exp\left(-y_i \frac{\boldsymbol{w}^\top \tilde{\boldsymbol{x}}_i + w_0}{m}\right)\right) \tag{3.1a}$$

$$\text{s.t. } \|\boldsymbol{w}\|_0 \leq \lambda, \boldsymbol{w} \in \mathbb{Z}^p, w_0 \in \mathbb{Z} \quad \textit{\# at most } \lambda \textit{ integer coefficients} \tag{3.1b}$$

$$w_j \in [a_j, b_j] \quad \forall j \in \{1, ..., p\} \quad \textit{\# control range of coefficients} \tag{3.1c}$$

$$m > 0 \quad \textit{\# expand solution space using multiplier} \tag{3.1d}$$

$$\sum_{k=1}^{\Gamma} \mathbb{I}\left\{\boldsymbol{w}_{G_k} \neq \boldsymbol{0}\right\} \leq \gamma. \quad \textit{\# at most } \gamma \textit{ groups, where } G_k \textit{ are the indices of group } k \tag{3.1e}$$

The optimization problem in Equation (3.1) is solved similarly to [47] in three steps:

1. Relax the integer value constraints and find a real-valued solution $(\boldsymbol{w}^{(*)}, w_0^{(*)})$ to sparse logistic regression under box constraint $(\boldsymbol{a}, \boldsymbol{b})$.

2. Based on $(\boldsymbol{w}^{(*)}, w_0^{(*)})$, swap one feature at a time to obtain a set of M sparse diverse real-valued solutions. Use an iterative subroutine to select the features, each of which must strictly satisfy the group constraint γ. Call this set Diverse Pool and denote the set by $\{(\boldsymbol{w}^{(t)}, w_0^{(t)})\}_{t=1}^{M}$. Diverse Pool's solutions are nearly as accurate as our original solution $(\boldsymbol{w}^{(*)}, w_0^{(*)})$.

3. For every solution in the Diverse Pool $\{(\boldsymbol{w}^{(t)}, w_0^{(t)})\}_{t=1}^{M}$, round all the continuous solutions to integers, thus assigning weights to the selected variables and producing a set of risk scores. Unlike direct rounding that could worsen solution quality, the algorithm use a subroutine that adapts multiplier m to extend the solution space when rounding the coefficients. A theoretical upper bound on the rounding error is proven in [47]; this choice permits us to maintain a high level of accuracy while rounding.

Together, the three steps allow GroupFasterRisk to produce multiple diverse sparse risk scores with high accuracy. Finally, for a given solution $(\boldsymbol{w}, w_0)$, risk predictions are

$P\left(Y=1 \mid \tilde{\boldsymbol{x}}\right)=\sigma\left(\left(\boldsymbol{w}^{\top} \tilde{\boldsymbol{x}}+w_0\right) / m\right)$, where $\sigma(\xi)=1 / \left(1+e^{-\xi}\right)$ is a sigmoid function. The Sequential Rounding algorithm [45] is used to find an integer risk score.

GroupFasterRisk provides the option of a monotonic correction so that the risk score is forced to increase (or decrease) along a variable. This allows users to better align the modeling process with domain knowledge.

For conciseness, I denote GroupFasterRisk models with the prefix GFR and group sparsity as the suffix. For instance, a GroupFasterRisk model trained with a group sparsity of 10 is GFR-10.

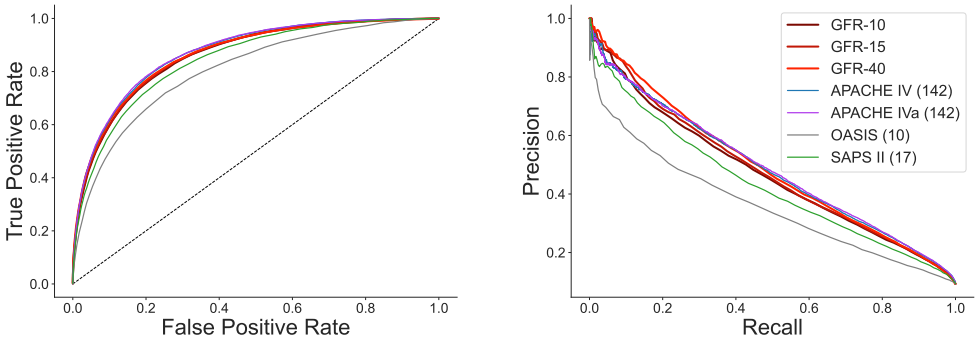
3.3 Results

3.3.1 All-cause Mortality Prediction

I first focus on evaluating how GroupFasterRisk performs when predicting all-cause in-hospital mortality, an important task in risk prognosis in critical care setting.

3.3.1.1 In-distribution Performance and Sparsity

GroupFasterRisk models predicted in-hospital mortality with the best AUROC and AUPRC across all internal evaluations on MIMIC III test folds (Figure 3.1b). Specifically, GFR-10 achieves an AUROC of 0.813 ($\pm$0.007) and AUPRC of 0.368 ($\pm$0.011), around 0.05 higher than OASIS. When using fifteen features, GFR-15 achieves an AUROC of 0.836 ($\pm$0.006) and AUPRC of 0.403 ($\pm$0.011), both around 0.05 higher than SAPS II (all the reported results are statistically significant with $p < 0.001$). GroupFasterRisk models are less complex than the competing scoring systems (Figure 3.1b) on MIMIC III. When using 10 features, GFR-10 has a model complexity of 42 ($\pm$0), whereas OASIS has 47. For fifteen features, GFR-15 is 48 ($\pm$4.9) while SAPS II is 58.



(a) ROC (left) and PR (right) curves for predicting all-cause in-hospital mortality on OOD evaluation. GroupFasterRisk models achieve better performance than all scoring system baselines except for APACHE IV/IVa on AUROC.

(b) GroupFasterRisk compared with the well-known severity of illness scores under different group sparsity constraints (equivalent to number of features). Evaluated on the internal MIMIC III dataset using 5-fold *nested* cross-validation, the best model from GroupFasterRisk is then evaluated in an OOD setting on the eICU cohort.
a. F is the number of features (equivalent to group sparsity) used by the model.
b. APACHE IV/IVa cannot be calculated on MIMIC III due to a lack of information for admission diagnoses.

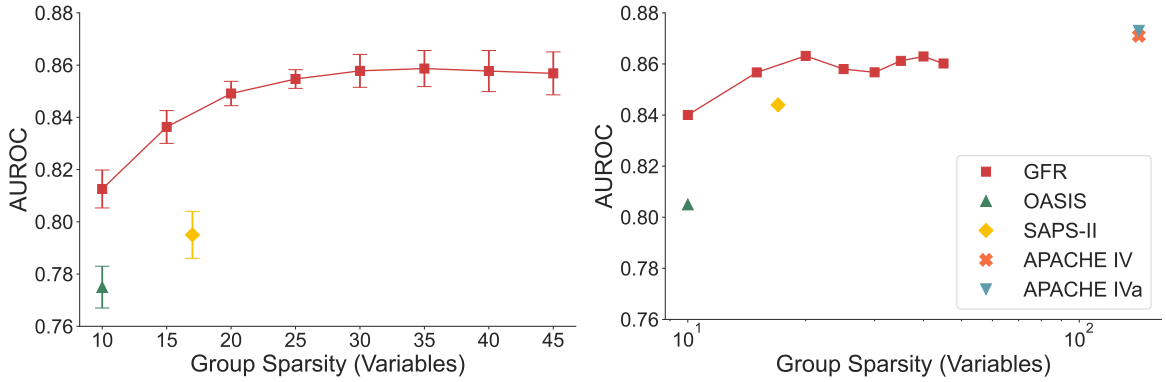
		Sparse				Not Sparse		
		GFR-10 $F=10$	OASIS $F=10$	GFR-15 $F=15$	SAPS II $F=17$	GFR-40 $F=40$	APACHE IV $F=142$	APACHE IVa $F=142$
MIMIC III Test Folds	AUROC	**0.813 ± 0.007**	0.775 ± 0.008	**0.836 ± 0.006**	0.795 ± 0.009	**0.858 ± 0.008**		
	AUPRC	**0.368 ± 0.011**	0.314 ± 0.014	**0.403 ± 0.011**	0.342 ± 0.012	**0.443 ± 0.013**		
	HL χ^2	**16.28 ± 2.51**	146.16 ± 10.27	**26.73 ± 6.38**	691.45 ± 18.64	**35.78 ± 11.01**		
	SMR	**0.992 ± 0.022**	0.686 ± 0.008	**0.996 ± 0.015**	0.485 ± 0.005	**1.002 ± 0.017**		
	Sparsity	**42 ± 0**	47	**48 ± 4.9**	58	**66 ± 8.0**		
eICU Test Set	AUROC	**0.844**	0.805	**0.859**	0.844	0.864	0.871	**0.873**
	AUPRC	**0.437**	0.361	**0.476**	0.433	**0.495**	0.487	0.489
	Sparsity	**34**	47	**50**	58	**80**	≥142	≥142

Figure 3.1: Comparison of GroupFasterRisk models with OASIS, SAPS II, APACHE IV, and APACHE IVa on all-cause in-hospital mortality prediction task.

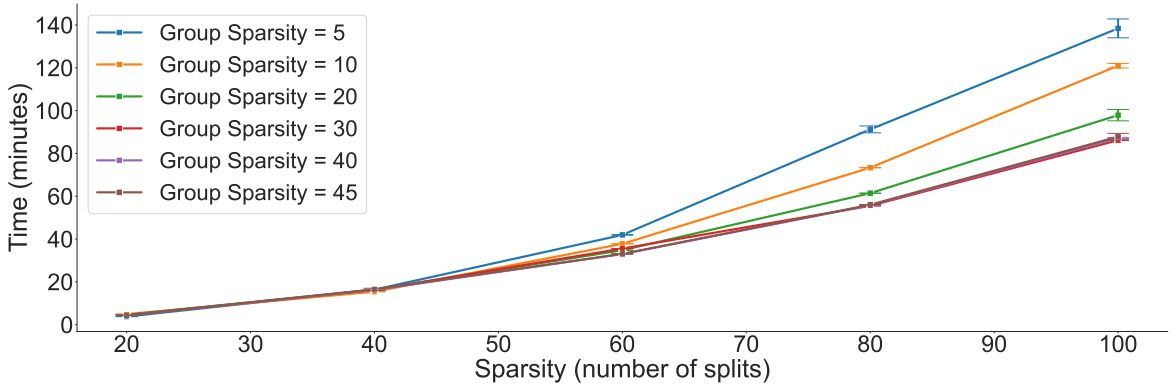
3.3.1.2 Out-of-distribution Performance and Sparsity

Further evaluation of GroupFasterRisk models is performed on the OOD eICU dataset (Figure 3.1b). I find that GFR-10 outperforms OASIS for both AUROC and AUPRC, with a noticeable margin of +0.039 and +0.075 for AUROC and AUPRC, respectively. Furthermore, GFR-15 achieves better predictive accuracy than SAPS II, with a margin of +0.015 for AUROC and +0.043 for AUPRC. The ROC and PR curves for the eICU dataset are shown in Figure 3.1a.

Although GroupFasterRisk is designed to optimize for sparse models, I included a more complex version, GFR-40, in our OOD evaluation for a thorough comparison with APACHE IV/IVa. GFR-40 outperforms APACHE IV/IVa in terms of AUPRC with a margin of +0.008 for IV and +0.006 for IVa. Although APACHE IV/IVa has higher AUROC scores (+0.007 for IV and +0.009 for IVa), GFR-40 uses significantly fewer features (40 compared to 142 for APACHE IV/IVa). In fact, APACHE cannot be calculated on the MIMIC III dataset, which highlights the disadvantage of complicated models in general.



(a) Performance of GroupFasterRisk under different levels of group sparsity. (Left) Internal evaluation on MIMIC III. (Right) OOD evaluation on eICU.



(b) Time consumption to train GroupFasterRisk under various sparsity and group sparsity constraints. Evaluated on Apple MacBook Pro, M2, using five repeated trials on the entire MIMIC III dataset, with sample size of 30,238 and 49 features.

Figure 3.2: Group sparsities and time consumption of GroupFasterRisk.

3.3.1.3 Group Sparsity and Runtime

Figure 3.2a shows the predictiveness of GroupFasterRisk models under different levels of group sparsity. I find that a higher group sparsity (equivalent to using more features) is positively correlated with an increase in AUROC. A similar observation is found for AUPRC. However, the increase in AUROC becomes relatively small after 30 variables. Note that under different group sparsity levels, GroupFasterRisk's models outperform OASIS and SAPS-II. Shown in Figure 3.2b, it takes at most two hours to train GroupFasterRisk on our MIMIC III cohort (recall that MIMIC III cohort has 30,238 patients). This is a relatively short amount of time considering that GroupFasterRisk is solving an NP-hard combinatorial optimization problem.

3.3.1.4 Fairness and Calibration

GroupFasterRisk produce reliable and fair risk predictions when we evaluated GroupFasterRisk models across various demographic subgroups, including ethnicity and gender (Table 3.1). GroupFasterRisk models are not particularly biased towards the majority race of Caucasians and are well-calibrated on specific subgroups in our eICU cohort. The models consistently achieve low Brier scores and HL χ^2 across subgroups. Except in a few cases, GroupFasterRisk models' Brier scores, HL χ^2, and SMR are better than those of OASIS, SAPS II, and APACHE IV/IVa. Further, among sparser models (no more than 17 variables), GroupFasterRisk achieve the highest AUROC and AUPRC.

3.3.2 Predictive Accuracy and Sparsity

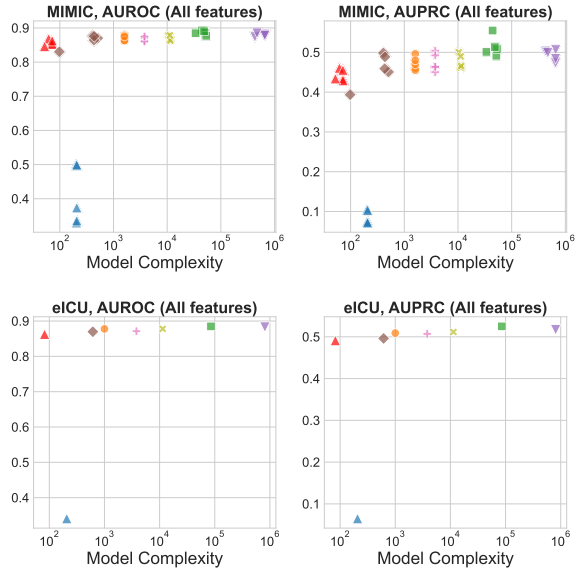
As observed in Figure 3.1, GroupFasterRisk models outperform existing risk scores in mortality prediction while being simpler. I further illustrate this point by comparing

Table 3.1: Fairness and calibration across population subgroups in eICU.

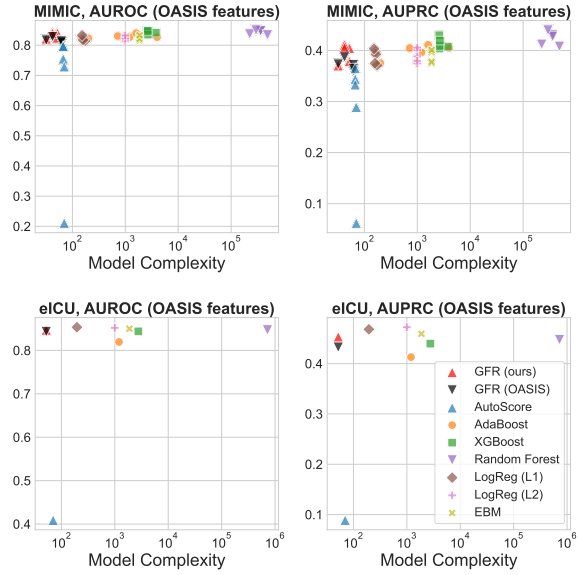
		Ethnicity (alphabetical order)						Gender	
		African American	Asian	Caucasian	Hispanic	Native American	Other/Unknown	Female	Male
Percentage (%)		11.17	1.49	76.91	3.86	0.68	4.68	45.08	54.90
AUROC (↑)	GFR-10	**0.829**	**0.833**	**0.837**	**0.856**	**0.881**	**0.849**	**0.835**	**0.840**
	OASIS	0.811	0.797	0.803	0.825	0.824	0.809	0.806	0.805
	GFR-15	**0.846**	**0.848**	**0.854**	**0.873**	**0.895**	**0.860**	**0.853**	**0.856**
	SAPS II	0.846	0.828	0.843	0.859	0.893	0.842	0.844	0.845
	GFR-40	0.859	0.861	0.859	0.881	0.902	0.873	0.857	0.865
	APACHE IV	0.873	0.858	0.869	0.890	**0.903**	0.884	0.867	0.875
	APACHE IVa	**0.875**	**0.866**	**0.870**	**0.893**	0.901	**0.886**	**0.869**	**0.876**
AUPRC (↑)	GFR-10	**0.415**	**0.390**	**0.422**	**0.480**	**0.558**	**0.418**	**0.418**	**0.429**
	OASIS	0.345	0.330	0.364	0.410	0.370	0.328	0.356	0.365
	GFR-15	**0.453**	**0.454**	**0.466**	**0.534**	**0.555**	**0.477**	**0.466**	**0.471**
	SAPS II	0.424	0.408	0.435	0.470	0.598	0.395	0.440	0.428
	GFR-40	**0.488**	**0.500**	**0.489**	**0.553**	**0.585**	**0.512**	**0.488**	**0.499**
	APACHE IV	0.488	0.467	0.484	0.536	0.536	0.479	0.478	0.493
	APACHE IVa	0.487	0.492	0.487	0.538	0.522	0.484	0.481	0.496
Brier Score (↓)	GFR-10	**0.064**	**0.070**	**0.068**	**0.065**	**0.059**	**0.065**	**0.068**	**0.067**
	OASIS	0.068	0.076	0.072	0.068	0.072	0.070	0.072	0.070
	GFR-15	**0.062**	**0.068**	**0.065**	**0.061**	**0.059**	**0.061**	**0.065**	**0.064**
	SAPS II	0.080	0.080	0.082	0.074	0.072	0.078	0.080	0.081
	GFR-40	**0.060**	**0.064**	**0.064**	**0.060**	**0.057**	**0.059**	**0.064**	**0.062**
	APACHE IV	0.063	0.069	0.066	0.062	0.066	0.064	0.066	0.064
	APACHE IVa	0.061	0.065	0.064	0.060	0.062	0.061	0.064	0.062
HL χ^2 (↓)	GFR-10	**27.90**	**11.00**	**113.70**	24.68	**5.48**	12.53	**58.65**	102.74
	OASIS	43.48	21.02	135.52	**5.23**	14.84	**11.75**	82.52	**79.11**
	GFR-15	**23.64**	**9.88**	**63.40**	**10.62**	**4.43**	**3.73**	**13.62**	**57.75**
	SAPS II	1070.09	94.34	6599.71	228.75	62.95	333.65	3575.48	4750.90
	GFR-40	**8.72**	**5.20**	**120.03**	**12.03**	**11.57**	**6.09**	**58.34**	**97.92**
	APACHE IV	308.51	34.51	1257.11	78.93	42.53	114.22	835.14	950.18
	APACHE IVa	167.60	13.04	502.27	42.78	23.21	62.48	372.68	384.89
SMR (∼ 1)	GFR-10	**0.946**	**0.915**	**1.028**	1.017	**0.949**	1.013	**0.993**	**1.031**
	OASIS	0.882	1.204	0.922	**0.994**	0.844	**1.002**	0.917	0.940
	GFR-15	**0.974**	**0.921**	**1.040**	**1.046**	**1.003**	**0.996**	**1.002**	**1.046**
	SAPS II	0.501	0.570	0.517	0.560	0.513	0.552	0.525	0.517
	GFR-40	**1.022**	**0.936**	**1.039**	**1.063**	**0.889**	**1.033**	**1.000**	**1.058**
	APACHE IV	0.663	0.732	0.731	0.710	0.606	0.697	0.716	0.725
	APACHE IVa	0.730	0.820	0.823	0.784	0.704	0.778	0.802	0.815

GroupFasterRisk with more complex ML approaches.

I conducted two experiments to assess the relationship between our methods' model complexity and AUROC or AUPRC. In the first experiment, I trained different ML models using the OASIS features, including Logistic Regression, Random Forest, AdaBoost, EBM, XGBoost, and AutoScore. I then compared their performance against our GFR-14 model (using our own features) and GroupFasterRisk trained on OASIS features, namely GFR-OASIS. In the second experiment, I trained the same ML models using all 49 features I obtained from the MIMIC III dataset. I compared these



(a) Performance vs. Complexity of GroupFasterRisk and baselines for all 49 features.



(b) Performance vs. Complexity of GroupFasterRisk and baselines for OASIS features.

Figure 3.3: Evaluation of GroupFasterRisk performance, sparsity and features

models with GFR-40.

I show results based on OASIS features in Figure 3.3b and results based on all features in Figure 3.3a. For both setups, I find that GroupFasterRisk models (GFR-14, GFR-40, GFR-OASIS) consistently achieve the best sparsity and high AUROC and AUPRC. AutoScore models are the least complex and rely on around 100 parameters, but their performance is substantially worse. Random Forest models achieve the highest AUROC and AUPRC scores, however, these models are very complex and rely on $\sim 10^6$ parameters, while GroupFasterRisk models use at most 82 parameters. Other methods such as ℓ_2-regularized Logistic Regression and EBM were as complex as boosted decision trees in terms of the total number of splits across all trees, $\sim 10^3$.

3.4 Discussion

Multiple aspects of this study are worth discussing. I firstly focus on the advantages of GroupFasterRisk and then on the limitations of this study.

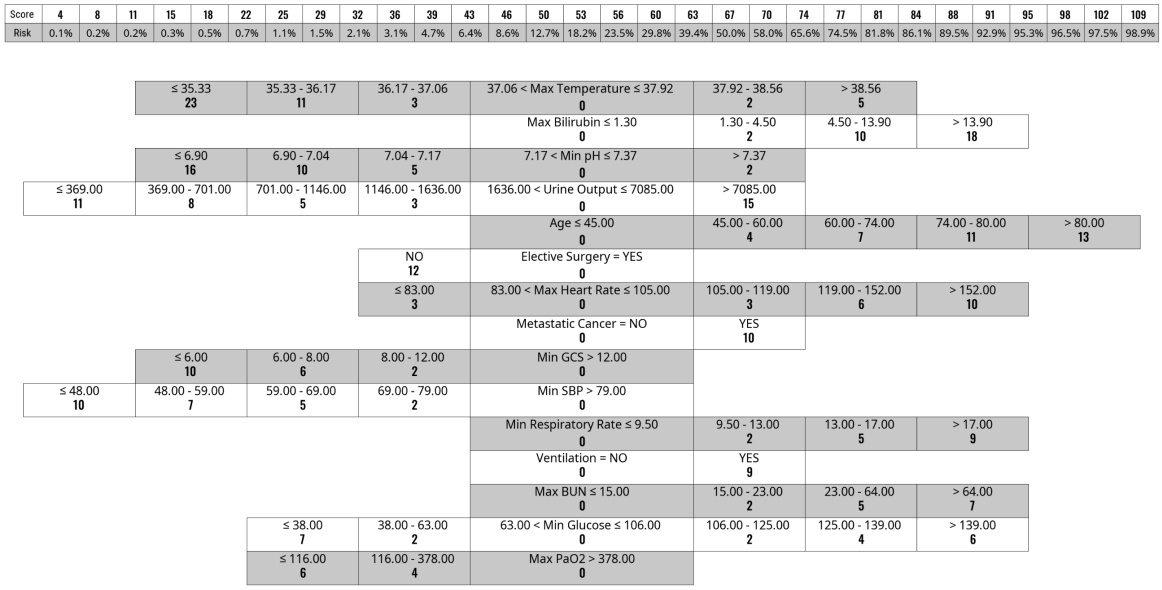


Figure 3.4: Risk score produced by GroupFasterRisk. This model has a group sparsity of 15 (GFR-15), which means that the model uses 15 features with multiple splits per feature.

3.4.1 Interpretability

GroupFasterRisk generates scorecard displays (such as the one in Figure 3.4). From those displays, people can quickly evaluate the correctness of the model and make adjustments if desired. For instance, the feature component scores, shown as the rows in Figure 3.4, allow medical practitioners to interpret the relationship between risk and the possible values of the features. Additionally, the group sparsity constraint enforces the selection of, at most, the top γ useful features, informing the user about the meaningful variables in the prediction-making process. Combined together, this score calculation

process from GroupFasterRisk models is transparent and interpretable to the user at any level of medical expertise, which could be beneficial for healthcare applications as it enables the discovery of new knowledge or potential bias in the model without the need for post-hoc explanations. Besides interpretability, GroupFasterRisk' models tend to be robust and fair across different ethnic and gender groups (Table 3.1).

3.4.2 Monotonic Correction

If the model does not suit medical knowledge due to empirical noise in data, practitioners could further adopt monotonicity corrections to adjust the model. I find that models corrected with monotonicity constraints demonstrate increased predictive accuracy in OOD evaluations compared to the original, uncorrected models (see Appendix for results without monotonic corrections). This suggests that incorporating domain-relevant knowledge into risk score design could further enhance performance.

3.4.3 Sparsity

The most accurate baselines we considered, APACHE IV/IVa, rely on 142 features. In comparison, GroupFasterRisk's most complex model, GFR-40, achieves similar performances as APACHE IV/IVa while requiring only 40 features (Figure 3.1). In practice, this 3.5 times difference in the number of features can be quite significant, especially when accounted for missing values or other collection errors that commonly occur in medical data [57]. While there are several ways to handle missing data [58, 59], these methods can negatively affect prediction accuracy, limit performance guarantees, and create an extra task for medical practitioners to complete in practice. Furthermore, compared to other ML approaches, GroupFasterRisk models are 1,000 times sparser in model complexity while achieving comparable performance (Figure 3.3a,

Figure 3.3b).

3.4.4 Limitations

There are some limitations in this study:

1. Although MIMIC III and eICU are the largest and most detailed publicly available datasets that have ever existed on ICU monitoring, they were collected from hospitals in the United States between 2001-2012 and 2014-2015, respectively, which limits this study from performing further evaluation on samples collected from other locations or time periods.

2. Changes in patterns over time are still not fully measured due to how MIMIC III and eICU collect data. For instance, vital signs in eICU are first recorded as one-minute averages and then stored as five-minute medians. Although this study primarily uses summary statistics, the data collection issues may still be affected by changes in measurement processing or aggregation.

3. To have more access to the measurements of the patients, the MIMIC III cohort in this study considers patients who stayed in the ICU for more than 24 hours (see Appendix for more details), which may cause bias in predicting mortality for patients admitted to the ICU for less than one day. Thus, to provide a more comprehensive evaluation, our eICU cohort includes all patients who have been admitted for more than 4 hours (this is consistent with cohorts used to create OASIS and APACHE IV). The results in the previous section fully support that GroupFasterRisk performs well under these shifted hours of ICU stay.

3.5 Appendix

3.5.1 Study Population

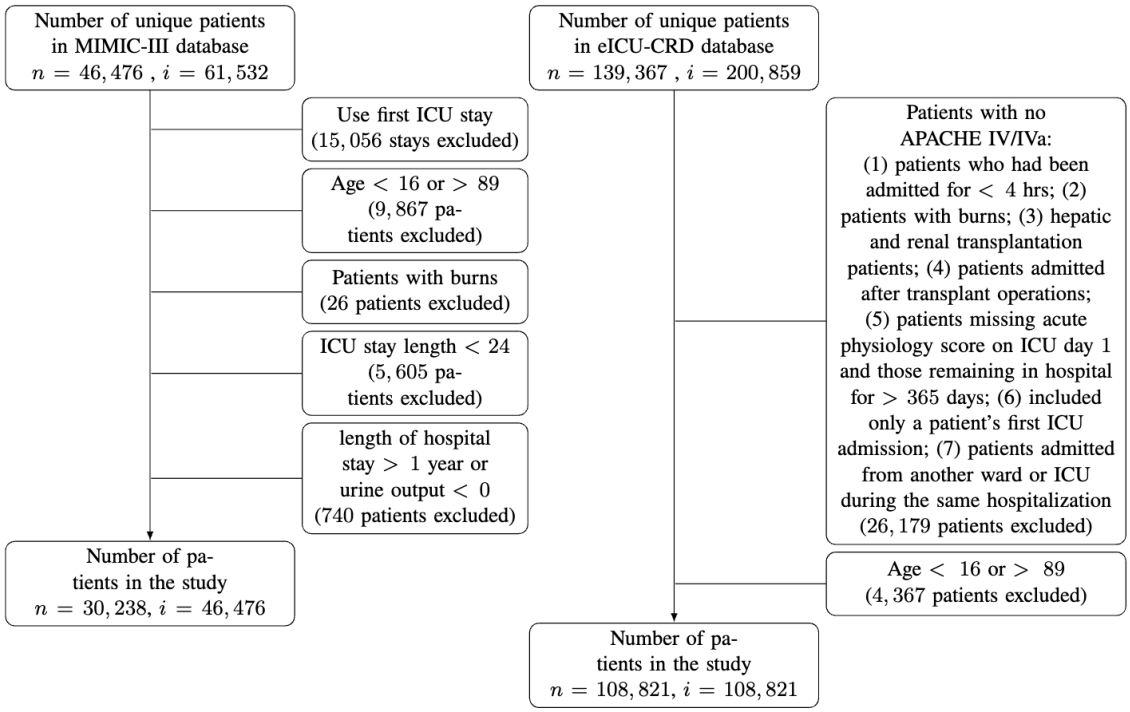


Figure 3.5: Flow chart of the study population selection on MIMIC III and eICU datasets. i and n are ICU stays and number of patients, respectively.

3.5.2 Data Processing

To include time series measurements such as vital signs, I extracted the minimum and maximum of these features during the first 24 hours of a patient's unit stay. This allows the algorithm to focus on the worst deviation from a normal range of values. Minimum and maximum values of time series data are often easier for medical practitioners to observe than other more sophisticated statistics (such as variance). Additionally, most existing severity of illness scores also rely on the worst values over a time period.

3.5.3 Feature Selection and Engineering

The feature selection process was divided into two stages.

1. I created a set of features by taking the union of features used by existing severity of illness scores. Specifically, I consider features from APS III [60], SAPS II [61], OASIS [33], SOFA [27], LODS [62], and SIRS [63]. I computed the area under the receiver-operating characteristic curve (AUROC) value for every feature individually. This provides us with a ranking of all features, and we selected the top 49 features for this study.

2. I transformed every continuous or categorical feature into a set of binary decision splits. This allows GroupFasterRisk to capture a non-linear step function for each feature when it learns coefficients for those decision splits.

   I used two methods to create the splits: 1) for binary or categorical features, a split was created between each pair of unique feature values for each feature. 2) I obtained the distribution of feature values based on the training data and computed quantiles of the distributions and used them as decision splits. Alternatively, the splits can be set as a hyperparameter. I create an indicator vector for missing values and do not perform imputation on them.

This preprocessing procedure makes GroupFasterRisk models generalized additive models (GAMs), which have been demonstrated to be as accurate as any black box ML model for most tabular data problems [64, 65, 51].

We perform an additional study on the effect of bin widths for continuous features in Table 3.2.

Table 3.2: GroupFasterRisk performance under various bin widths. To allow GroupFasterRisk to better utilize continuous variables, a binarization technique is applied, which transforms a continuous variable into B quantiles (bins).

Number of Bins (B)		100	50	20	10	5	4
Average Mean AUROC	Training	0.847 ± 0.030	0.849 ± 0.029	**0.856 ± 0.020**	0.850 ± 0.023	0.847 ± 0.016	0.841 ± 0.019
	Validation	0.828 ± 0.030	0.832 ± 0.028	**0.843 ± 0.018**	0.839 ± 0.022	0.840 ± 0.015	0.835 ± 0.018
Average Mean AUPRC	Training	0.444 ± 0.057	0.448 ± 0.056	**0.451 ± 0.043**	0.439 ± 0.046	0.430 ± 0.032	0.413 ± 0.036
	Validation	0.394 ± 0.047	0.401 ± 0.047	**0.414 ± 0.033**	0.413 ± 0.037	0.412 ± 0.028	0.401 ± 0.032

3.5.4 Optimization Procedure Outline

To solve the optimization problem in Equation (3.1), I solve three consecutive optimization sub-problems. In the first step, Equation (3.2), I approximately find a near-optimal solution for **sparse logistic regression** with sparsity and box constraints, denoted as λ and $(\boldsymbol{a}, \boldsymbol{b})$, respectively.

$$
\begin{aligned}
(\boldsymbol{w}^{(*)}, w_0^{(*)}) \in \underset{\boldsymbol{w}, w_0}{\arg\min}\, \mathcal{L}\left(\boldsymbol{w}, w_0, \mathcal{D}\right) = \sum_{i=1}^{n} \log\left(1 + \exp\left(-y_i\left(\tilde{\boldsymbol{x}}_i^\top \boldsymbol{w} + w_0\right)\right)\right) \\
\text{s.t. } \|\boldsymbol{w}\|_0 \leq \lambda, \boldsymbol{w} \in \mathbb{R}^p, w_0 \in \mathbb{R} \\
\forall j \in [p], w_j \in [a_j, b_j] \\
\sum_{k=1}^{\Gamma} \mathbb{I}\left\{\boldsymbol{w}_{G_k} \neq \boldsymbol{0}\right\} \leq \gamma.
\end{aligned}
\tag{3.2}
$$

Solving Equation (3.2) produces an accurate and sparse *real-valued* solution $(\boldsymbol{w}^{(*)}, w_0^{(*)})$ that satisfies both feature and group sparsity constraints.

In the second step, I aim to produce multiple *real-valued* **near-optimal sparse logistic regression solutions under group sparsity constraint**, which is formulated

as:

$$
\begin{aligned}
(\boldsymbol{w}^{(t)}, w_0^{(t)}) \text{ obeys } \mathcal{L}(\boldsymbol{w}^{(t)}, w_0^{(t)}, \mathcal{D}) &\leq \mathcal{L}(\boldsymbol{w}^{(*)}, w_0^{(*)}, \mathcal{D})(1+\epsilon_u) \\
\text{s.t. } \|\boldsymbol{w}^{(t)}\|_0 \leq \lambda, \boldsymbol{w}^{(t)} &\in \mathbb{R}^p, w_0^{(t)} \in \mathbb{R} \\
\forall j \in [p], w_j^{(t)} &\in [a_j, b_j] \\
\sum_{k=1}^{\Gamma} \mathbb{I}\left\{\boldsymbol{w}_{G_k}^{(t)} \neq \mathbf{0}\right\} &\leq \gamma.
\end{aligned}
\tag{3.3}
$$

In particular, in order to solve Equation (3.3), I delete a feature j_- with support in $supp(\boldsymbol{w}^{(*)})$ and add a new feature with index j_+. This procedure is repeated to turn the solution $(\boldsymbol{w}^{(*)}, w_0)$ into diverse sparse solutions with similar logistic loss. Note that during swapping, I only consider the alternative features that obey various constraints (including box constraint, groups-sparsity constraint, and monotonicity constraint) to ensure the new solutions are valid models.

$$
\text{Find all } j_+ \text{ s.t. } \min_{\delta \in [a_{j_+}, b_{j_+}]} \mathcal{L}(\boldsymbol{w}^{(*)} - w_{j_-}^{(*)} \boldsymbol{e}_{j_-} + \delta \boldsymbol{e}_{j_+}, w_0, \mathcal{D}) \leq \mathcal{L}(\boldsymbol{w}^{(*)}, w_0^{(*)}, \mathcal{D})\,(1+\epsilon_u)\,. \tag{3.4}
$$

I solve Equation (3.3) several times (set by the user as a hyper-parameter), after which we have a pool of distinct, almost-optimal sparse logistic regression models, and the top M models with the smallest logistic loss are selected, creating M solutions $\{(\boldsymbol{w}^{(t)}, w_0^{(t)})\}_{t=1}^{M}$. Note that the user can set ϵ_u and M arbitrarily, controlling the tolerance in logistic loss and the desired maximum quantity of diverse sparse solutions.

Lastly, for each solution in $\{(\boldsymbol{w}^{(t)}, w_0^{(t)})\}_{t=1}^{M}$, I compute an **integer risk score**, $(\boldsymbol{w}^{(+t)}, w_0^{(+t)})$,

by performing rounding to a *real-valued* solution:

$$\begin{aligned} \mathcal{L}\left(\boldsymbol{w}^{(+t)}, w_0^{(+t)}, \mathcal{D}/m^{(t)}\right) \leq \mathcal{L}\left(\boldsymbol{w}^{(t)}, w_0^{(t)}, \mathcal{D}\right) + \epsilon_t \\ \text{s.t. } \boldsymbol{w}^{(+t)} \in \mathbb{Z}^p, w_0^{(+t)} \in \mathbb{Z} \\ \forall j \in [p], w_j^{(+t)} \in [a_j, b_j] \\ \sum_{k=1}^{\Gamma} \mathbb{I}\left\{\boldsymbol{w}_{G_k}^{(+t)} \neq \mathbf{0}\right\} \leq \gamma. \end{aligned} \tag{3.5}$$

where ϵ_t is a gap in logistic loss with the near-optimal solution due to rounding. A theoretical upper bound on ϵ_t was proven in [47]. In order to round the coefficients, the following steps are performed: 1) we define the largest multiplier $m_{\max}$ as $\max\{\|\boldsymbol{a}\|_\infty, \|\boldsymbol{b}\|_\infty\}/\|\boldsymbol{w}^{(*)}\|_\infty$, and the smallest multiplier $m_{\min}$ to be 1. 2) select N_m equally spaced values within the range $[m_{\min}, m_{\max}]$, giving us a set of multipliers. 3) Using this set of multipliers, scale the dataset, obtaining $\{1/m, \tilde{\boldsymbol{x}}_i/m, y_i\}_{i=1}^n$. 4) Send the scaled dataset to the sequential rounding algorithm [47, 46], which rounds the coefficients one at a time to an integer that best maintains accuracy (not necessarily the nearest integer). Use the integer coefficients and multiplier with the smallest logistic loss as our final solution.

To perform monotonic correction, GroupFasterRisk allows users to set box constraints $(\boldsymbol{a}_{G_l}, \boldsymbol{b}_{G_l})$ for each feature l independently. These constraints can be imposed after any of the three GroupFasterRisk optimization steps. Since during the feature preprocessing, we transformed every continuous or categorical feature into a set of binary decision splits, each feature corresponds to a set of step functions. Imposing monotonicity is equivalent to forcing all coefficients for all step functions of one feature to be positive (for decreasing functions) or negative (for increasing functions). If $\boldsymbol{a}_{G_l}, \boldsymbol{b}_{G_l} \geq \mathbf{0}$, then $f_l(x_l)$ is monotonically decreasing; if $\boldsymbol{a}_{G_l}, \boldsymbol{b}_{G_l} \leq \mathbf{0}$, then component

function f_l is monotonically increasing.

Table 3.3: Ablation study on monotonicity correction. The evaluation is performed OOD on eICU. A performance boost is observed when monotonicity correction was applied, likely because correct domain information was included in the individual component scores of the features.

		Sparse				Not Sparse		
		GFR-10 $F = 10$	OASIS $F = 10$	GFR-15 $F = 15$	SAPS II $F = 17$	GFR-40 $F = 40$	APACHE IV $F = 142$	APACHE IVa $F = 142$
With monotonicity correction	AUROC	**0.844**	0.805	**0.859**	0.844	0.864	0.871	**0.873**
	AUPRC	**0.436**	0.361	**0.476**	0.433	**0.495**	0.487	0.489
No monotonicity correction	AUROC	0.840	0.805	0.857	0.844	0.863	0.871	**0.873**
	AUPRC	0.427	0.361	0.467	0.433	0.491	0.487	0.489

Chapter 4

Synthetic Electronic Health Records with Diffusion Models

4.1 Introduction

The Electronic Health Record (EHR) is a digital version of the patient's medical history maintained by healthcare providers. It includes information such as demographic attributes, vital signals, and lab measurements that are sensitive in nature and important for clinical research. Researchers have been utilizing statistical and machine learning (ML) methods to analyze EHR for a variety of downstream tasks such as disease diagnosis, in-hospital mortality prediction, and disease phenotyping [66, 67]. However, due to privacy concerns, EHR data is strictly regulated, and thus the availability of EHR data is often limited, creating barriers to the development of computational models in the field of healthcare. Widely used EHR de-identification methods to preserve patient information privacy are criticized for having high risks of re-identification of the individuals [68].

Instead of applying privacy-preserving methods that can adversely affect EHR data

utility [69], EHR synthetic data generation is one promising solution to protect patient privacy. Realistic synthetic data preserves crucial clinical information in real data while preventing patient information leakage [70, 71]. Synthetic data also has the added benefit of providing a larger sample size for downstream analysis than de-identifying real samples [72]. As a result, more research initiatives have begun to consider synthetic data sharing, such as the National COVID Cohort Collaborative supported by the U.S. National Institutes of Health and the Clinical Practice Research Datalink sponsored by the U.K. National Institute for Health and Care Research [73, 74]. With the advancement in ML techniques, applying generative models to synthesize high-fidelity EHR data is a popular research of interest [70]. Recent advances in generative models have shown significant success in generating realistic high-dimensional data like images, audio, and texts [75, 76], suggesting the potential for these models to handle EHR data with complex statistical characteristics.

Some representative work utilizing generative models for EHR data synthesis includes medGAN [77], medBGAN [78], and EHR-Safe [71]. However, most approaches to EHR data synthesis are GAN-based, and GANs are known for their difficulties in model training and deployments due to training instability and mode collapse [79]. Recently, diffusion probabilistic models have shown superb ability over GANs in generating high-fidelity image data [80, 81, 82]. A few studies thus propose to generate synthetic EHR data via diffusion models given their remarkable data generation performance [83, 84]. However, most EHR data synthesis methods, either GAN-based or diffusion-based, focus on binary or categorical variables such as the International Classification of Diseases (ICD) codes. Additionally, there is limited prior work on generating EHR data with temporal information, and most state-of-the-art time series generative models are GAN-based. The sole study that employs diffusion models for EHR time series overlooks discrete time series in its modeling process [85]. It resorts to Gaussian diffusion

for generating discrete sequences, treating them similarly to real-valued sequences but with further post-processing of the model output. These observations motivate me to bridge the gap by introducing a novel diffusion-based method to generate realistic EHR time series data with mixed variable types.

Specifically, this chapter aims to focus on the following:

- I present TimeDiff, a diffusion probabilistic model that uses a bidirectional recurrent neural network (BRNN) architecture for high-utility time series data generation.
- By combining multinomial and Gaussian diffusions, TimeDiff can simultaneously generate both real and discrete valued time series directly. To the best of my knowledge, TimeDiff is the first work in applying this mixed diffusion approach on EHR time series generation.
- I experimentally demonstrates that TimeDiff outperforms state-of-the-art methods for time series data generation by a big margin in terms of data utility. Additionally, our model requires less training effort than GANs.
- I further evaluate TimeDiff on potential applications in healthcare and show it can generate useful synthetic samples for ML model development while protecting patient privacy.

4.2 Related Work

4.2.1 Time Series Generation

Prior sequential generation methods using GANs rely primarily on binary adversarial feedback [86, 87], and supervised sequence models mainly focus on tasks such as pre-

diction [88], forecasting [89], and classification [90]. TimeGAN [91] was one of the first methods to preserve temporal dynamics in time series synthesis. The architecture comprises an embedding layer, recovery mechanism, generator, and discriminator, trained using both supervised and unsupervised losses. GT-GAN [92] considers the generation of both regular and irregular time series data using a neural controlled differential equation (NCDE) encoder [93] and GRU-ODE decoder [94]. This framework, combined with a continuous time flow processes (CTFPs) generator [95] and a GRU-ODE discriminator, outperformed existing methods in general-purpose time series generation. Recently, [96] proposed to generate time series data for forecasting and imputation using discrete or continuous stochastic process diffusion (DSPD/CSPD). Their proposed method views time series as discrete realizations of an underlying continuous function. Both DSPD and CSPD use either the Gaussian or Ornstein-Uhlenbec process to model noise and apply it to the entire time series. The learned distribution over continuous functions is then used to generate synthetic time series samples.

4.2.2 Diffusion Models

Diffusion models [97] have been proposed and achieved excellent performance in the field of computer vision and natural language processing. [80] proposed denoising diffusion probabilistic models (DDPM) that generate high-quality images by recovering from white latent noise. [98] proposed a vector-quantized diffusion model on text-to-image synthesis with significant improvement over GANs regarding scene complexity and diversity of the generated images. [99] suggested that the diffusion models with optimized architecture outperform GANs on image synthesis tasks. [100] proposed a diffusion model, Imagen, incorporated with a language model for text-to-image synthesis with state-of-the-art results. [101] introduced TabDDPM, an extension of DDPM for heterogeneous tabular data generation, outperforming GAN-based models. [102]

proposed ChiroDiff, a diffusion model that considers temporal information and generates chirographic data. Besides advancements in practical applications, some recent developments in theory for diffusion models demonstrate the effectiveness of this model class. Theoretical foundations explaining the empirical success of diffusion or score-based generative models have been established [103, 104, 105].

4.2.3 EHR Data Generation

A considerable amount of prior work has been done on generating EHR data. [77] proposed medGAN that generates EHR discrete variables. Built upon medGAN, [78] suggested two models, medBGAN and medWGAN, that synthesize EHR binary or discrete variables on International Classification of Diseases (ICD) codes. [106] developed a GAN that can generate high-utility EHR with both discrete and continuous data. [107] proposed the EHR Variational Autoencoder that synthesizes sequences of EHR discrete variables (i.e., diagnosis, medications, and procedures). [83] developed MedDiff, a diffusion model that generates user-conditioned EHR discrete variables. [84] created EHRDiff by utilizing the diffusion model to generate a collection of ICD diagnosis codes. [108] used continuous-time diffusion models to generate synthetic EHR tabular data. [109] applied TabDDPM to synthesize tabular healthcare data.

However, most existing work focuses on discrete or tabular data generation. There is limited literature on EHR time series data generation, and this area of research has not yet received much attention [110]. In 2017, RCGAN [87] was created for generating multivariate medical time series data by employing RNNs as the generator and discriminator. Until recently, [71] proposed EHR-Safe that consists of a GAN and an encoder-decoder module. EHR-Safe can generate realistic time series and static variables in EHR with mixed data types. [111] developed EHR-M-GAN that generates

mixed-type time series in EHR using separate encoders for each data type. Moreover, [85] suggested utilizing diffusion models to synthesize discrete and continuous EHR time series. However, their approach mainly relies on Gaussian diffusion and adopts a U-Net architecture [112]. The generation of discrete time series is achieved by taking argmax of softmax over real-valued one-hot representations. By contrast, our proposed method considers multinomial diffusion for discrete time series generation, allowing the generation of discrete variables directly.

4.3 Preliminaries

In this section, I explain diffusion models following the work of [97] and [80]. Diffusion models belong to a class of latent variable models formulated as $p_\theta(\boldsymbol{x}^{(0)}) = \int p_\theta(\boldsymbol{x}^{(0:T)})\, d\boldsymbol{x}^{(1:T)}$, where $\boldsymbol{x}^{(0)}$ is a sample following the data distribution $q(\boldsymbol{x}^{(0)})$ and $\{\boldsymbol{x}^{(t)}\}_{t=1}^{T}$ are latents with the same dimensionality as $\boldsymbol{x}^{(0)}$. The model consists of two stages: a forward process that gradually adds Gaussian noise to the original data sample and a reverse process that draws synthetic samples from learned distributions by gradually denoising from the pure Gaussian noise sample.

The *forward process* is defined as a Markov chain that gradually adds Gaussian noise to $\boldsymbol{x}^{(0)}$ via a sequence of variances $\left\{\beta^{(t)}\right\}_{t=1}^{T}$:

$$q\big(\boldsymbol{x}^{(1:T)}|\boldsymbol{x}^{(0)}\big) = \prod_{t=1}^{T} q\big(\boldsymbol{x}^{(t)}|\boldsymbol{x}^{(t-1)}\big), \quad q\big(\boldsymbol{x}^{(t)}|\boldsymbol{x}^{(t-1)}\big) := \mathcal{N}\big(\boldsymbol{x}^{(t)}; \sqrt{1-\beta^{(t)}}\boldsymbol{x}^{(t-1)}, \beta^{(t)}\boldsymbol{I}\big). \tag{4.1}$$

In practice, the variances $\beta^{(t)}$ in the forward process can be determined by reparameterization or via scheduling as a hyperparameter [113]. The process successively converts data $\boldsymbol{x}^{(0)}$ to white latent noise $\boldsymbol{x}^{(T)}$. The noisy sample $\boldsymbol{x}^{(t)}$ can be obtained directly from the original sample $\boldsymbol{x}^{(0)}$ by sampling from $q(\boldsymbol{x}^{(t)}|\boldsymbol{x}^{(0)}) = \mathcal{N}\big(\boldsymbol{x}^{(t)}; \sqrt{\bar{\alpha}^{(t)}}\boldsymbol{x}^{(0)}, (1-\bar{\alpha}^{(t)})\boldsymbol{I}\big)$,

where $\alpha^{(t)} = 1 - \beta^{(t)}$ and $\bar{\alpha}^{(t)} = \prod_{i=1}^{t} \alpha^{(i)}$.

The *reverse process* is the joint distribution $p_\theta(\boldsymbol{x}^{(0:T)}) = p(\boldsymbol{x}^{(T)}) \prod_{t=1}^{T} p_\theta(\boldsymbol{x}^{(t-1)}|\boldsymbol{x}^{(t)})$, which is a Markov chain that starts from white latent noise and gradually denoises noisy samples to generate synthetic samples:

$$p_\theta\big(\boldsymbol{x}^{(t-1)}|\boldsymbol{x}^{(t)}\big) := \mathcal{N}\big(\boldsymbol{x}^{(t-1)}; \boldsymbol{\mu}_\theta(\boldsymbol{x}^{(t)}, t), \boldsymbol{\Sigma}_\theta(\boldsymbol{x}^{(t)}, t)\big), \quad p\big(\boldsymbol{x}^{(T)}\big) := \mathcal{N}\big(\boldsymbol{x}^{(T)}; \boldsymbol{0}, \boldsymbol{I}\big). \quad (4.2)$$

Since computing $p_\theta(\boldsymbol{x}^{(0)}) = \int p_\theta(\boldsymbol{x}^{(0:T)})\ d\boldsymbol{x}^{(1:T)}$ directly is intractable, it is approximated by maximizing the variational lower bound on the log-likelihood:

$$\mathbb{E}\left[\log p_\theta(\boldsymbol{x}^{(0)})\right] \geq \mathbb{E}_q\left[\log \frac{p_\theta(\boldsymbol{x}^{(0:T)})}{q(\boldsymbol{x}^{(1:T)}|\boldsymbol{x}^{(0)})}\right] := \text{ELBO} \quad (4.3)$$

The evidence lower bound (ELBO) in Equation (4.3) can be rewritten in a form involving KL divergences between Gaussian distributions [80] shown in Equation (4.4). The objective is to maximize the variational lower bound on the log-likelihood which can be written in a form involving KL divergences between Gaussian distributions [80]:

$$\mathbb{E}\Big[\underbrace{\log p_\theta(\boldsymbol{x}^{(0)}|\boldsymbol{x}^{(1)})}_{A_1} - \underbrace{D_{\mathrm{KL}}\big(q(\boldsymbol{x}^{(T)}|\boldsymbol{x}^{(0)}) \parallel p(\boldsymbol{x}^{(T)})\big)}_{A_2} - \sum_{t=2}^{T} \underbrace{D_{\mathrm{KL}}\big(q(\boldsymbol{x}^{(t-1)}|\boldsymbol{x}^{(t)}, \boldsymbol{x}^{(0)}) \parallel p_\theta(\boldsymbol{x}^{(t-1)}|\boldsymbol{x}^{(t)})\big)}_{A_3}\Big]. \quad (4.4)$$

Specifically, A_1 can be interpreted as a reconstruction term that provides the log probability of the original data sample given $\boldsymbol{x}^{(1)}$. A_2 can be ignored in the optimization since both distributions are white noise and are not parameterized. A_3 is a term that ensures consistency between the ground truth, tractable forward process posterior, and our approximated and parameterized distribution. The forward process posterior can

be expressed as shown in Equation (4.5):

$$\tilde{\boldsymbol{\mu}}^{(t)}(\boldsymbol{x}^{(t)}, \boldsymbol{x}^{(0)}) = \frac{1}{\sqrt{\alpha^{(t)}}}\left(\boldsymbol{x}^{(t)} - \frac{\beta^{(t)}}{\sqrt{1-\bar{\alpha}^{(t)}}}\boldsymbol{\epsilon}^{(t)}\right), \quad \boldsymbol{\epsilon}^{(t)} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}),$$
$$q\big(\boldsymbol{x}^{(t-1)}|\boldsymbol{x}^{(t)}, \boldsymbol{x}^{(0)}\big) := \mathcal{N}\big(\boldsymbol{x}^{(t-1)}; \tilde{\boldsymbol{\mu}}^{(t)}(\boldsymbol{x}^{(t)}, \boldsymbol{x}^{(0)}), \tilde{\beta}^{(t)}\boldsymbol{I}\big), \quad \tilde{\beta}^{(t)} = \frac{1-\bar{\alpha}^{(t-1)}}{1-\bar{\alpha}^{(t)}}\beta^{(t)}. \tag{4.5}$$

Under a specific parameterization described in [80], the training objective can be expressed as follows:

$$\begin{aligned} &\mathbb{E}\left[\frac{1}{2(\sigma^{(t)})^2}\left\|\tilde{\boldsymbol{\mu}}^{(t)}\big(\boldsymbol{x}^{(t)}, \boldsymbol{x}^{(0)}\big) - \boldsymbol{\mu}_\theta\big(\boldsymbol{x}^{(t)}, t\big)\right\|^2\right] + C \\ &= \mathbb{E}\left[\frac{(\beta^{(t)})^2}{2(\sigma^{(t)})^2\alpha^{(t)}(1-\bar{\alpha}^{(t)})}\left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}^{(t)}}\boldsymbol{x}^{(0)} + \sqrt{1-\bar{\alpha}^{(t)}}\boldsymbol{\epsilon}, t)\right\|^2\right] + C, \end{aligned} \tag{4.6}$$

where C is a constant that is not trainable. Empirically, a neural network $\boldsymbol{s}_\theta$ is trained to approximate $\boldsymbol{\epsilon}$. This $\boldsymbol{\epsilon}$-prediction objective resembles denoising score matching, and the sampling procedure resembles Langevin dynamics using $\boldsymbol{s}_\theta$ as an estimator of the gradient of the data distribution [103, 104].

4.4 Methodology

In this section, I discuss the methodology for generating realistic synthetic EHR time series data. The scope of this work considers the generation of both numerical (real-valued) and discrete time series, as both are present in EHR. Specifically, let $\mathcal{D}$ denote our EHR time series dataset. Each patient in $\mathcal{D}$ has numerical and discrete multivariate time series $\boldsymbol{X} \in \mathbb{R}^{P_r \times L}$ and $\boldsymbol{C} \in \mathbb{Z}^{P_d \times L}$, respectively. L is the number of time steps, and P_r and P_d are the number of variables for numerical and discrete data types.

4.4.1 Diffusion Process on EHR Time Series

To generate both numerical and discrete time series, I consider a "mixed sequence diffusion" approach by adding Gaussian and multinomial noises. For numerical time series, I perform Gaussian diffusion by adding independent Gaussian noise similar to DDPM. The forward process is thus defined as:

$$q\big(\boldsymbol{X}^{(1:T)}|\boldsymbol{X}^{(0)}\big) = \prod_{t=1}^{T}\prod_{l=1}^{L} q\big(\boldsymbol{X}_{\cdot,l}^{(t)}|\boldsymbol{X}_{\cdot,l}^{(t-1)}\big), \tag{4.7}$$

where $q(\boldsymbol{X}_{\cdot,l}^{(t)}|\boldsymbol{X}_{\cdot,l}^{(t-1)}) = \mathcal{N}(\boldsymbol{X}_{\cdot,l}^{(t)}; \sqrt{1-\beta^{(t)}}\boldsymbol{X}_{\cdot,l}^{(t-1)}, \beta^{(t)}\boldsymbol{I})$ and $\boldsymbol{X}_{\cdot,l}$ is the l^{th} observation of the numerical time series. In a similar fashion as Equation (4.2), we define the reverse process for numerical features as $p_\theta(\boldsymbol{X}^{(0:T)}) = p(\boldsymbol{X}^{(T)})\prod_{t=1}^{T} p_\theta\big(\boldsymbol{X}^{(t-1)}|\boldsymbol{X}^{(t)}\big)$, where

$$\begin{aligned} p_\theta\big(\boldsymbol{X}^{(t-1)}|\boldsymbol{X}^{(t)}\big) &:= \mathcal{N}\big(\boldsymbol{X}^{(t-1)}; \boldsymbol{\mu}_\theta(\boldsymbol{X}^{(t)}, t), \tilde{\beta}^{(t)}\boldsymbol{I}\big), \\ \boldsymbol{\mu}_\theta(\boldsymbol{X}^{(t)}, t) = \frac{1}{\sqrt{\alpha^{(t)}}}&\left(\boldsymbol{X}^{(t)} - \frac{\beta^{(t)}}{\sqrt{1-\bar{\alpha}^{(t)}}}\boldsymbol{s}_\theta(\boldsymbol{X}^{(t)}, t)\right), \quad \tilde{\beta}^{(t)} = \frac{1-\bar{\alpha}^{(t-1)}}{1-\bar{\alpha}^{(t)}}\beta^{(t)}. \end{aligned} \tag{4.8}$$

In order to model discrete-valued time series, we use multinomial diffusion [114]. The forward process is defined as:

$$q\big(\tilde{\boldsymbol{C}}^{(1:T)}|\tilde{\boldsymbol{C}}^{(0)}\big) = \prod_{t=1}^{T}\prod_{p=1}^{P_d}\prod_{l=1}^{L} q\big(\tilde{\boldsymbol{C}}_{p,l}^{(t)}|\tilde{\boldsymbol{C}}_{p,l}^{(t-1)}\big), \tag{4.9}$$

$$q\big(\tilde{\boldsymbol{C}}_{p,l}^{(t)}|\tilde{\boldsymbol{C}}_{p,l}^{(t-1)}\big) := \mathcal{C}\big(\tilde{\boldsymbol{C}}_{p,l}^{(t)}; (1-\beta^{(t)})\tilde{\boldsymbol{C}}_{p,l}^{(t-1)} + \beta^{(t)}/K\big), \tag{4.10}$$

where $\mathcal{C}$ is a categorical distribution, $\tilde{\boldsymbol{C}}_{p,l}^{(0)} \in \{0,1\}^K$ is a one-hot encoded representation of $C_{p,l}$[1], and the addition and subtraction between scalars and vectors are performed

[1]We perform one-hot encoding on the discrete time series across the feature dimension. For example, if our time series is $\{0,1,2\}$, its one-hot representation becomes $\{[1,0,0]^\top, [0,1,0]^\top, [0,0,1]^\top\}$.

element-wise. The forward process posterior distribution is defined as follows:

$$q\big(\tilde{\boldsymbol{C}}_{p,l}^{(t-1)}|\tilde{\boldsymbol{C}}_{p,l}^{(t)},\tilde{\boldsymbol{C}}_{p,l}^{(0)}\big) := \mathcal{C}\left(\tilde{\boldsymbol{C}}_{p,l}^{(t-1)};\boldsymbol{\phi}/\sum_{k=1}^{K}\phi_k\right), \tag{4.11}$$

$$\boldsymbol{\phi} = \Big(\alpha^{(t)}\tilde{\boldsymbol{C}}_{p,l}^{(t)} + (1-\alpha^{(t)})/K\Big) \odot \Big(\bar{\alpha}^{(t-1)}\tilde{\boldsymbol{C}}_{p,l}^{(0)} + (1-\bar{\alpha}^{(t-1)})/K\Big). \tag{4.12}$$

The reverse process $p_\theta(\tilde{\boldsymbol{C}}_{p,l}^{(t-1)}|\tilde{\boldsymbol{C}}_{p,l}^{(t)})$ is parameterized as $q\big(\tilde{\boldsymbol{C}}_{p,l}^{(t-1)}|\tilde{\boldsymbol{C}}_{p,l}^{(t)},\boldsymbol{s}_\theta(\tilde{\boldsymbol{C}}_{p,l}^{(t)},t)\big)$. We train our neural network, $\boldsymbol{s}_\theta$, using both Gaussian and multinomial diffusion processes:

$$\mathcal{L}_{\mathcal{N}}(\theta) := \mathbb{E}_{\boldsymbol{X}^{(0)},\boldsymbol{\epsilon},t}\left[\left\|\boldsymbol{\epsilon} - \boldsymbol{s}_\theta\big(\sqrt{\bar{\alpha}^{(t)}}\boldsymbol{X}^{(0)} + \sqrt{1-\bar{\alpha}^{(t)}}\boldsymbol{\epsilon},t\big)\right\|^2\right], \tag{4.13}$$

$$\mathcal{L}_{\mathcal{C}}(\theta) := \mathbb{E}_{p,l}\left[\sum_{t=2}^{T} D_{\mathrm{KL}}\Big(q\big(\tilde{\boldsymbol{C}}_{p,l}^{(t-1)}|\tilde{\boldsymbol{C}}_{p,l}^{(t)},\tilde{\boldsymbol{C}}_{p,l}^{(0)}\big)\ \Big\|\ p_\theta\big(\tilde{\boldsymbol{C}}_{p,l}^{(t-1)}|\tilde{\boldsymbol{C}}_{p,l}^{(t)}\big)\Big)\right], \tag{4.14}$$

where $\mathcal{L}_{\mathcal{N}}$ and $\mathcal{L}_{\mathcal{C}}$ are the losses for numerical and discrete multivariate time series, respectively. The training of the neural network is performed by minimizing the following loss with stochastic gradient descent (such as Adam optimizer [115]):

$$\mathcal{L}_{\text{train}}(\theta) = \lambda\mathcal{L}_{\mathcal{C}}(\theta) + \mathcal{L}_{\mathcal{N}}(\theta), \tag{4.15}$$

where λ is a hyperparameter for creating a balance between the two losses. We investigate the effects of λ in the Appendix section.

4.4.2 Missing Value Representation

In medical applications, missing data and variable measurement times play a crucial role as they could provide additional information and indicate a patient's health status

[57]. I thus derive a missing indicator mask $\boldsymbol{M} \in \{0,1\}^{P_r \times L}$ [2] for each $\boldsymbol{X} \in \mathcal{D}$[3]:

$$M_{p,l} = \begin{cases} 0, & \text{if } X_{p,l} \text{ is present;} \\ 1, & \text{if } X_{p,l} \text{ is missing.} \end{cases} \tag{4.16}$$

Then $\boldsymbol{M}$ encodes the measurement time points of $\boldsymbol{X}$. If $\boldsymbol{X}$ contains missing values, I impute them in the initial value of the forward process, i.e., $\boldsymbol{X}^{(0)}$, using the corresponding sample mean. Nevertheless, $\boldsymbol{M}$ retains the information regarding the positions of missing values. This method generates both discrete and numerical time series, allowing us to represent and generate $\boldsymbol{M}$ as a discrete time series seamlessly.

4.4.3 TimeDiff Architecture

In this section, I describe the architecture for TimeDiff. A commonly used architecture in DDPM is U-Net [112]. However, most U-Net-based models are tailored to image generation tasks, requiring the neural network to process pixel-based data rather than sequential information [116, 80, 82]. Even its one-dimensional variant, 1D-U-Net, comes with limitations such as restriction on the input sequence length (which must be a multiple of U-Net multipliers) and a tendency to lose temporal dynamics information during down-sampling. On the other hand, TabDDPM [101] proposed a mixed diffusion approach for tabular data generation but relied on a multilayer perceptron architecture, making it improper for multivariate time series generation.

To address this challenge of handling EHR time series, we need an architecture capable of encoding sequential information while being flexible to the input sequence length. The time-conditional bidirectional RNN (BRNN) or neural controlled differential equa-

[2] Or alternatively, $\boldsymbol{M} \in \{0,1\}^{P_d \times L}$ if the time series is discrete.

[3] For simplicity in writing, we refer to $\boldsymbol{X}$ only, but this procedure can also be applied on $\boldsymbol{C}$.

tion (NCDE) [93] can be possible options. After careful evaluation, I found BRNN without attention mechanism offers superior computational efficiency and have chosen it as the neural backbone $\boldsymbol{s}_\theta$ for all of our experiments. A more detailed discussion of NCDE is provided in Appendix.

4.4.3.1 Diffusion Step Embedding

To inform the model about the current diffusion time step t, we use sinusoidal positional embedding [117]. The embedding vector output from the embedding layer then goes through two fully connected (FC) layers with GeLU activation in between [118]. The embedding vector is then fed to a SiLU activation [118] and another FC layer. The purpose of this additional FC layer is to adjust the dimensionality of the embedding vector to match the stacked hidden states from BRNN. Specifically, I set the dimensionality of the output to be two times the size of the hidden dimension from BRNN. I denote the transformed embedding vector as $\boldsymbol{t}_{\text{embed}}$. This vector is then split into two vectors, each with half of the current size, namely $\boldsymbol{t}_{\text{embed\_scale}}$ and $\boldsymbol{t}_{\text{embed\_shift}}$. Both vectors share the same dimensionality as BRNN's hidden states and serve to inform the network about the current diffusion time step.

4.4.3.2 Time-conditional BRNN

In practice, BRNN can be implemented with either LSTM or GRU units. To condition BRNN on time, I follow these steps. I first obtain noisy samples from Gaussian (for numerical data) and multinomial (for discrete data) diffusion. The two samples are concatenated and fed to our BRNN, which returns a sequence of hidden states $\{\boldsymbol{h}_l\}_{l=1}^{L}$ that stores the temporal dynamics information about the time series. To stabilize learning and enable proper utilization of $\boldsymbol{t}_{\text{embed}}$, I apply layernorm [119] on $\{\boldsymbol{h}_l\}_{l=1}^{L}$. The normalized sequence of hidden states, $\{\tilde{\boldsymbol{h}}_l\}_{l=1}^{L}$, is then scaled and shifted using

$\{\tilde{\boldsymbol{h}}_l \odot (\boldsymbol{t}_{\text{embed\_scale}} + \mathbf{1}) + \boldsymbol{t}_{\text{embed\_shift}}\}_{l=1}^{L}$. These scaled hidden states contain information about the current diffusion step t, which is then passed through an FC layer to produce the final output. The output contains predictions for both multinomial and Gaussian diffusions, which are extracted correspondingly and used to calculate $\mathcal{L}_{\text{train}}$ in Equation (4.15).

4.5 Results

4.5.1 Datasets

I use four publicly available EHR datasets to evaluate TimeDiff: Medical Information Mart for Intensive Care III and IV (MIMIC III/IV) [48, 120], the eICU Collaborative Research Database (eICU) [121], and high time resolution ICU dataset (HiRID) [122]. In order to evaluate TimeDiff with state-of-the-art methods for time series generation on non-EHR datasets, I include Stocks and Energy datasets used in studies that proposed TimeGAN [91] and GT-GAN [92].

4.5.2 Baselines

I compare TimeDiff with eight methods: EHR-M-GAN [111], GT-GAN [92], TimeGAN [91], RCGAN [87], C-RNN-GAN [86], RNNs trained with teacher forcing (T-Forcing) [123, 124] and professor forcing (P-Forcing) [125], and discrete or continuous stochastic process diffusion (DSPD/CSPD) with Gaussian (GP) or Ornstein-Uhlenbeck (OU) processes [96].

4.5.3 Metrics

(1-2) *Discriminative and Predictive Scores:* A GRU-based discriminator is trained to distinguish between real and synthetic samples. The discriminative score is $|0.5 - \text{Accuracy}|$. For the predictive score, a GRU-based predictor is trained on synthetic samples and evaluated on real samples for next-step vector prediction based on mean absolute error over each sequence.

(3) *Train on Synthetic, Test on Real (TSTR):* I train ML models entirely on synthetic data and evaluate them on real test data based on the area under the receiver operating characteristic curve (AUC) for in-hospital mortality prediction. I compare the TSTR score to the Train on Real, Test on Real (TRTR) score, which is the AUC obtained from the model trained on real training data and evaluated on real test data.

(4) *Train on Synthetic and Real, Test on Real (TSRTR):* Similar to TSTR, I train ML models and evaluate them on real test data using AUC. I fix the size of real training data to 2000 and add more synthetic samples to train ML models. This metric evaluates the impact on ML models when their training data includes an increasing amount of synthetic data. It also simulates the practical scenario where practitioners use synthetic samples to increase the sample size of the training data for model development.

(5) *t-SNE:* I flatten the feature dimension and use t-SNE dimension reduction visualization [126] on synthetic, real training, and real testing samples. This qualitative metric measures the similarity of synthetic and real samples in two-dimensional space.

(6) *Nearest Neighbor Adversarial Accuracy Risk (NNAA):* This score measures the degree to which a generative model overfits the real training data, a factor that could raise privacy-related concerns [127]. It is the difference between two discriminative accuracies, AA_{test} and AA_{train}.

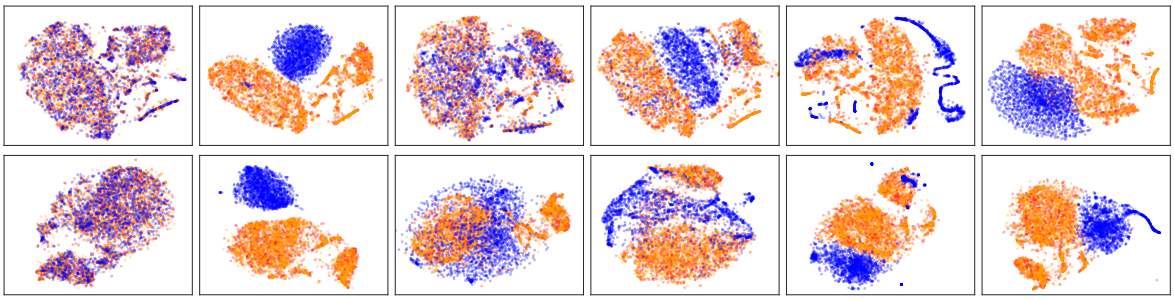
(7) *Membership Inference Risk (MIR):* An F1 score is calculated based on whether an adversary can correctly identify the membership of a synthetic data sample [128].

For all the experiments, I split each dataset into training and testing sets and used the training set to develop generative models. The synthetic samples obtained from trained generative models are then used for evaluation. We repeat each experiment over 10 times and report the mean and standard deviation of each quantitative metric. Further details for our experiments and evaluation metrics are discussed in Appendix.

4.5.4 Results

Table 4.1: Comparison of predictive and discriminative scores between TimeDiff and the baselines.

Metric	Method	Stocks	Energy	MIMIC-III	MIMIC-IV	HiRID	eICU
Discriminative Score (↓)	**TimeDiff**	**.048±.028**	**.088±.018**	**.028±.023**	**.030±.022**	**.333±.056**	**.015±.007**
	EHR-M-GAN	.483±.027	.497±.006	.499±.002	.499±.001	.496±.003	.488±.022
	DSPD-GP	.081±.034	.416±.016	.491±.002	.478±.020	.489±.004	.327±.020
	DSPD-OU	.098±.030	.290±.010	.456±.014	.444±.037	.481±.007	.367±.018
	CSPD-GP	.313±.061	.392±.007	.498±.001	.488±.010	.485±.007	.489±.010
	CSPD-OU	.283±.039	.384±.012	.494±.002	.479±.005	.489±.004	.479±.017
	GT-GAN	.077±.031	.221±.068	.488±.026	.472±.014	.455±.015	.448±.043
	TimeGAN	.102±.021	.236±.012	.473±.019	.452±.027	.498±.002	.434±.061
	RCGAN	.196±.027	.336±.017	.498±.001	.490±.003	.499±.001	.490±.023
	C-RNN-GAN	.399±.028	.499±.001	.500±.000	.499±.000	.499±.001	.493±.010
	T-Forcing	.226±.035	.483±.004	.499±.001	.497±.002	.480±.010	.479±.011
	P-Forcing	.257±.026	.412±.006	.494±.006	.498±.002	.494±.004	.367±.047
	Real Data	*.019±.016*	*.016±.006*	*.012±.006*	*.014±.011*	*.014±.015*	*.004±.003*
Predictive Score (↓)	**TimeDiff**	**.037±.000**	**.251±.000**	**.469±.003**	**.432±.002**	**.292±.018**	**.309±.019**
	EHR-M-GAN	.120±.047	.254±.001	.861±.072	.880±.079	.624±.028	.913±.179
	DSPD-GP	.038±.000	.260±.001	.509±.014	.586±.026	.404±.013	.320±.018
	DSPD-OU	.039±.000	.252±.000	.497±.006	.474±.023	.397±.024	.317±.023
	CSPD-GP	.041±.000	.257±.001	1.083±.002	.496±.034	.341±.029	.624±.066
	CSPD-OU	.044±.000	.253±.000	.566±.006	.516±.051	.439±.010	.382±.026
	GT-GAN	.040±.000	.312±.002	.584±.010	.517±.016	.386±.033	.487±.033
	TimeGAN	.038±.001	.273±.004	.727±.010	.548±.022	.729±.039	.367±.025
	RCGAN	.040±.001	.292±.005	.837±.040	.700±.014	.675±.074	.890±.017
	C-RNN-GAN	.038±.000	.483±.005	.933±.046	.811±.048	.727±.082	.769±.045
	T-Forcing	.038±.001	.315±.005	.840±.013	.641±.017	.364 ±.018	.547±.069
	P-Forcing	.043±.001	.303±.006	.683±.031	.557±.030	.445±.018	.345±.021
	Real Data	*.036±.001*	*.250±.003*	*.467±.005*	*.433±.001*	*.267±.012*	*.304±.017*



TimeDiff EHR-M-GAN DSPD-GP GT-GAN TimeGAN RCGAN

Figure 4.1: t-SNE for eICU (1st row) and MIMIC-IV (2rd row). Synthetic samples in **blue**, real training samples in **red**, and real testing samples in **orange**.

4.5.4.1 Predictive and Discriminative Scores

As presented in Table 4.1, I observe that TimeDiff consistently achieves the lowest discriminative and predictive scores across six datasets compared to all baselines. TimeDiff achieves significantly lower discriminative scores and close-to-real predictive scores on all four EHR datasets. For instance, TimeDiff yields a 95.4% lower mean discriminative score compared to DSPD-GP and obtains a 1.6% higher mean predictive score than real testing data on the eICU dataset. For non-EHR datasets, TimeDiff achieves a 37.7% lower and a 60.2% lower mean discriminative scores on the Stocks and Energy datasets than GT-GAN while having similar mean predictive scores as using real testing data.

4.5.4.2 t-SNE

As shown in Figure 4.1, the synthetic samples produced by TimeDiff demonstrate remarkable overlap with both real training and testing data, indicating their high fidelity.

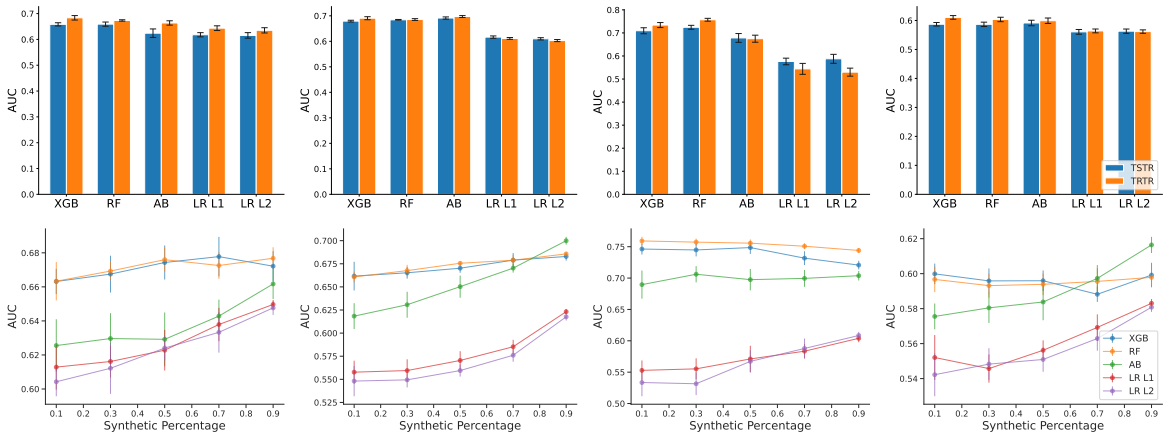
4.5.4.3 Runtime

I compare the number of hours to train TimeDiff with EHR-M-GAN, TimeGAN, and GT-GAN. I used Intel Xeon Gold 6226 Processor and Nvidia GeForce 2080 RTX Ti for runtime comparison of all models. As indicated by Table 4.2, TimeDiff can produce high-fidelity synthetic samples with less training time compared to GAN-based approaches.

4.5.4.4 In-hospital Mortality Prediction

In-hospital mortality prediction is one of the most important downstream tasks utilizing EHR data [129, 130]. To evaluate the utility of the generated EHR time series samples using TimeDiff, I perform in-hospital mortality prediction using six ML algorithms: XGBoost (XGB) [52], Random Forest (RF) [42], AdaBoost (AB) [53], and ℓ_1 and ℓ_2 regularized Logistic Regression (LR L1/L2) [131]. The prediction models are trained using synthetic samples from TimeDiff and assessed on real testing data.

As indicated in Figure 4.2, I observe that models trained using pure synthetic samples have similar AUCs compared to those trained on the real training data. Furthermore, to simulate the practical scenario where synthetic samples are used for data augmentation, I calculate the TSRTR scores for each ML model. I observe that most ML models achieve better performances as more synthetic samples are added. This observation is also consistent with our previous findings, demonstrating the high fidelity of our synthetic data.



MIMIC-III eICU HiRID MIMIC-IV

Figure 4.2: (Top) comparison of TSTR with TRTR scores; (Bottom) TSRTR score.

Table 4.2: Runtime comparisons (hours).

Dataset	**TimeDiff**	EHR-M-GAN	TimeGAN	GT-GAN
MIMIC-III	**2.7**	18.9	10.8	21.8
MIMIC-IV	**2.7**	28.8	29.5	47.3
HiRID	**2.5**	29.7	46.2	58.3
eICU	**8.7**	87.1	110	59.1

4.5.4.5 NNAA and MIR

As indicated in Table 4.4, I observe that TimeDiff consistently scores around 0.5 for both AA_{test} and AA_{train} while having low NNAA and MIR scores. This suggests that TimeDiff produces high-fidelity synthetic samples and does not overfit its training data. By contrast, although still mostly having low NNAA and MIR scores, all the baselines have higher AA_{test} and AA_{train}.

4.5.4.6 Ablation Study

I further investigate whether performing multinomial diffusion on missing indicators for discrete sequence generation is useful. I compare the proposed method with Gaussian diffusion on the missing indicators, and these post-processing methods are applied to transform real-valued model predictions into discrete sequences: (1) direct rounding;

(2) argmax on the softmax of real-valued, one-hot encoded representations[4]. I compare these methods on MIMIC-III/IV and eICU. HiRID is excluded from this ablation study since it is complete and does not contain missing values. Table 4.3 shows that the synthetic data has much higher utility when multinomial diffusion is adopted.

Table 4.3: Ablation study on generating missing indicators using multinomial diffusion.

Metric	Method	MIMIC-III	MIMIC-IV	eICU
Discriminative Score (↓)	with Gaussian and rounding	.355±.020	.121±.025	.030±.018
	with Gaussian and softmax	.088±.023	.155 ±.032	.042±.045
	with multinomial	**.028±.023**	**.030±.022**	**.015±.007**
Predictive Score (↓)	with Gaussian and rounding	.486±.005	.433±.003	.312±.031
	with Gaussian and softmax	.472±.004	.434±.002	.320±.035
	with multinomial	**.469±.003**	**.432±.002**	**.309±.019**

[4]The synthetic one-hot encoding is not discrete since Gaussian diffusion is used. This method is also adopted by [85] for the generation of discrete time series with diffusion models.

Table 4.4: Privacy score evaluations.

Metric	Method	MIMIC-III	MIMIC-IV	HiRID	eICU
AA_{test} ($\sim$0.5)	**TimeDiff**	**.574±.002**	**.517±.002**	**.531±.003**	**.537±.001**
	EHR-M-GAN	.998±.000	1.000±.000	1.000±.000	.977±.000
	DSPD-GP	.974±.001	.621±.002	.838±.004	.888±.000
	DSPD-OU	.927±.000	.804±.003	.886±.001	.971±.000
	CSPD-GP	.944±.001	.623±.002	.958±.002	.851±.001
	CSPD-OU	.967±.001	.875±.002	.947±.001	.982±.000
	GT-GAN	.995±.000	.910±.001	.990±.001	.981±.000
	TimeGAN	.997±.000	.974±.001	.643±.003	1.000±.000
	RCGAN	.983±.001	.999±.000	1.000±.000	1.000±.000
	Real Data	*.552±.002*	*.497±.002*	*.511±.006*	*.501±.002*
AA_{train} ($\sim$0.5)	**TimeDiff**	**.573±.002**	**.515±.002**	**.531±.002**	**.531±.002**
	EHR-M-GAN	.999±.000	1.000±.000	1.000±.000	.965±.002
	DSPD-GP	.968±.002	.620±.003	.851±.005	.888±.001
	DSPD-OU	.928±.001	.788±.003	.876±.002	.971±.000
	CSPD-GP	.940±.002	.629±.005	.965±.004	.852±.001
	CSPD-OU	.966±.001	.880±.003	.945±.002	.983±.000
	GT-GAN	.995±.001	.907±.002	.989±.001	.981±.000
	TimeGAN	.997±.000	.969±.003	.651±.004	1.000±.000
	RCGAN	.984±.001	.999±.000	1.000±.000	1.000±.000
	Real Data	*.286±.003*	*.268±.004*	*.327±.006*	*.266±.002*
NNAA (↓)	**TimeDiff**	.002±.002	.002±.002	.004±.003	.006±.002
	EHR-M-GAN	.000±.000	.000±.000	.000±.000	.012±.003
	DSPD-GP	.005±.003	.003±.003	.013±.007	.001±.001
	DSPD-OU	.001±.001	.016±.004	.010±.002	.000±.000
	CSPD-GP	.004±.002	.007±.005	.008±.004	.001±.001
	CSPD-OU	.001±.001	.005±.003	.002±.001	.001±.001
	GT-GAN	.001±.000	.004±.002	.001±.001	.000±.000
	TimeGAN	.000±.000	.005±.003	.008±.004	.000±.000
	RCGAN	.001±.000	.000±.000	.000±.000	.000±.000
	Real Data	*.267±.004*	*.229±.003*	*.184±.006*	*.235±.003*
MIR (↓)	**TimeDiff**	.191±.008	.232±.048	.236±.179	.227±.021
	EHR-M-GAN	.025±.007	.435±.031	.459±.161	.049±.006
	DSPD-GP	.032±.021	.050±.009	.106 ±.064	.000±.000
	DSPD-OU	.060±.032	.007±.006	.005±.005	.000±.000
	CSPD-GP	.060±.028	.034±.017	.004±.004	.000±.000
	CSPD-OU	.066±.046	.016±.020	.005±.003	.000±.000
	GT-GAN	.005±.002	.046±.013	.109±.057	.000±.000
	TimeGAN	.010±.002	.173±.020	.624±.006	.000±.000
	RCGAN	.013±.002	.277±.049	.063±.013	.000±.000
	Real Data	*.948±.090*	*.929±.005*	*.737±.011*	*.927±.001*

4.6 Appendix

4.6.1 Datasets

In this section, we provide further information on the datasets used in this study and the corresponding data processing procedures. Unless specified otherwise, all datasets are normalized by min-max scaling for model training, and the minimums and maximums are calculated feature-wise, i.e., we normalize each feature by its corresponding sample minimum and maximum, and this procedure is applied across all the features. For all EHR datasets, we extract the in-hospital mortality status as our class labels for TSTR and TSRTR evaluations.

Table 4.5: Dataset statistics.

Dataset	Sample Size	Number of Features	Sequence Length	Missing (%)	Mortality Rate (%)
Stocks	3,773	6	24	0	—
Energy	19,711	28	24	0	—
MIMIC-III	26,150	15	25	17.9	7.98
MIMIC-IV	21,593	11	72	7.9	23.67
HiRID	6,709	8	100	0	16.83
eICU	62,453	9	276	10.5	10.63

4.6.1.1 Stocks & Energy

We use the Stocks and Energy datasets for a fair comparison between TimeDiff and the existing GAN-based time-series generation methods. Both datasets can be downloaded from TimeGAN's official repository.

Stocks: The Stocks dataset contains daily Google stock data recorded between 2004 and 2019. It contains features such as volume, high, low, opening, closing, and adjusted closing prices. Each data point represents the value of those six features on a single day. The dataset is available online and can be accessed from the historical Google stock price on Yahoo.

Energy: The Energy dataset is from the UCI Appliances energy prediction data. It contains multivariate continuous-valued time-series and has high-dimensional, correlated, and periodic features. This dataset can be obtained from UCI machine learning repository.

To prepare both datasets for training and ensure consistency with previous approaches for a fair comparison, we use the same procedure as TimeGAN. We then apply training and testing splits for both datasets. For the Stocks dataset, we use 80% for training and 20% for testing. For the Energy dataset, we use 75% for training and 25% for testing.

4.6.1.2 MIMIC-III

The Medical Information Mart for Intensive Care-III (MIMIC-III) is a single-center database consisting of a large collection of EHR data for patients admitted to critical care units at Beth Israel Deaconess Medical Center between 2001 and 2012. The dataset contains information such as demographics, lab results, vital measurements, procedures, caregiver notes, and patient outcomes. It contains data for 38,597 distinct adult patients and 49,785 hospital admissions.

Variable Selection: In our study, we use the following vital sign measurements from MIMIC-III: heart rate (beats per minute), systolic blood pressure (mm Hg), diastolic blood pressure (mm Hg), mean blood pressure (mm Hg), respiratory rate (breaths per minute), body temperature (Celsius), and oxygen saturation (%). To ensure consistency and reproducibility, we adopt the scripts in official MIMIC-III repository for data pre-processing that selects the aforementioned features based on *itemid* and filters potential outliers[5]. We then extract records of the selected variables within the first

[5]For sake of reproducibility, the thresholds for the outliers are defined by the official repository.

24 hours of a patient's unit stay at one-hour intervals, where the initial measurement is treated as time step 0. This procedure gives us a multivariate time series of length 25 for each patient.

Cohort Selection: We select our MIMIC-III study cohort by applying the outlier filter criteria adopted by the official MIMIC-III repository. The filtering rules can be accessed here. We select patients based on the unit stay level using *icustay_id*. We only include patients who have spent at least 24 hours in their ICU stay.

We use 80% of the dataset for training and 20% for testing while ensuring a similar class ratio between the splits.

4.6.1.3 MIMIC-IV

The Medical Information Mart for Intensive Care-IV (MIMIC-IV) is a large collection of data for over 40,000 patients at intensive care units at the Beth Israel Deaconess Medical Center. It contains retrospectively collected medical data for 299,712 patients, 431,231 admissions, and 73,181 ICU stays. It improves upon the MIMIC-III dataset, incorporating more up-to-date medical data with an optimized data storage structure. In our study, we use vital signs for time-series generation. To simplify the data-cleaning process, we adopt scripts from the MIMIC Code Repository.

Variable Selection: We extracted five vital signs for each patient from MIMIC-IV. The selected variables are heart rate (beats per minute), systolic blood pressure (mm Hg), diastolic blood pressure (mm Hg), respiratory rate (breaths per minute), and oxygen saturation (%). We extract all measurements of each feature within the first 72 hours of each patient's ICU admission. Similar to MIMIC-III, we encode the features using the method described in Section 4.4.2 for model training.

Cohort Selection: Similar to MIMIC-III, we select our MIMIC-IV study cohort by

applying filtering criteria provided by the official MIMIC-IV repository. The criteria can be accessed here. We also select patients at the unit stay level and include those who stayed for at least 72 hours in ICU.

We use 75% for training and 25% for testing, and the class ratio is kept similar across the training and testing data.

4.6.1.4 eICU

The eICU Collaborative Research Database is a multi-center database with 200,859 admissions to intensive care units monitored by the eICU programs across the United States. It includes various information for the patients, such as vital sign measurements, care plan documentation, severity of illness measures, diagnosis information, and treatment information. The database contains 139,367 patients admitted to critical care units between 2014 and 2015.

Variable Selection: We select four vital sign variables from the *vitalPeriodic* table in our study: heart rate (beats per minute), respiratory rate (breaths per minute), oxygen saturation (%), and mean blood pressure (mm Hg). The measurements are recorded as one-minute averages and are then stored as five-minute medians. We extract values between each patient's first hour of the ICU stay and the next 24 hours for the selected variables. Since the measurements are recorded at 5-minute intervals, we obtain a multivariate time series of length 276 for each patient in our study cohort.

Cohort Selection: We select patients for our eICU study cohort by filtering the time interval. Specifically, we include patients who stay for at least 24 hours in their ICU stay, and the time series measurements are extracted. We did not use filtering criteria for time series in eICU. This is a design choice that allows us to evaluate TimeDiff when unfiltered time series are used as the input. We also select patients at the unit

stay level.

We use 75% for training and 25% for testing while ensuring the class ratio is similar between the two data splits.

4.6.1.5 HiRiD

The high time resolution ICU dataset (HiRID) is a publicly accessible critical care dataset consisting of data for more than 33,000 admissions to the Department of Intensive Care Medicine of the Bern University Hospital, Switzerland. It includes de-identified demographic information and 712 physiological variables, diagnostic tests, and treatment information between January 2008 to June 2016. The physiological measurements are recorded at 2-minute intervals.

Variable Selection: We consider seven variables in our study: heart rate (beats per minute), invasive systolic arterial pressure (mm Hg), invasive diastolic arterial pressure (mm Hg), invasive mean arterial pressure (mm Hg), peripheral oxygen saturation (%), ST elevation (mm), and central venous pressure (mm Hg). We selected the recorded data during the first 200 minutes of each patient's ICU stay.

Cohort Selection: We include patients who stayed for at least 200 minutes in our HiRID study cohort. Unlike all aforementioned EHR datasets, our HiRID study cohort only includes patients without missing values. This design choice allows us to evaluate the performance of TimeDiff in the absence of missing values on EHR datasets.

We use 80% of our study cohort as the training data and 20% as the testing data, and the mortality rate is kept similar between the splits.

4.6.2 Baselines

We reference the following source code for implementations of our baselines.

Table 4.6: Source code links for all baselines.

Method	Source Code Link
EHR-M-GAN [111]	LINK
DSPD/CSPD (GP or OU) [96]	LINK
GT-GAN [92]	LINK
TimeGAN [91]	LINK
RCGAN [87]	LINK
C-RNN-GAN [86]	LINK
T-Forcing [123, 124]	LINK
P-Forcing [125]	LINK

4.6.3 Model Training and Hyperparameter Selection

4.6.3.1 Neural Controlled Differential Equation

We attempted to use neural controlled differential equation (NCDE) [93] as our architecture for $\boldsymbol{s}_\theta$. We expect the continuous property of the NCDE to yield better results for time-series generation. NCDE is formally defined as the following:

Definition 1. *Suppose we have a time-series* $\boldsymbol{s} = \{(r_1, \boldsymbol{x}_1), ..., (r_n, \boldsymbol{x}_n)\}$ *and* D *is the dimensionality of the series. Let* $Y : [r_1, r_n] \to \mathbb{R}^{D+1}$ *be the natural cubic spline with knots at* $r_1, ..., r_n$ *such that* $Y_{t_i} = (\boldsymbol{x}_i, r_i)$. $\boldsymbol{s}$ *is often assumed to be a discretization of an underlying process that is approximated by* Y. *Let* $f_\theta : \mathbb{R}^h \to \mathbb{R}^{h \times (D+1)}$ *and* $\zeta_\theta : \mathbb{R}^{D+1} \to \mathbb{R}^h$ *be any neural networks, where* h *is the size of hidden states. Let* $z_{r_1} = \zeta_\theta(r_1, \boldsymbol{x}_1)$

The NCDE model is then defined to be the solution to the following CDE:

$$z_r = z_{r_1} + \int_{r_1}^{r} f_\theta(z_s)\mathrm{d}Y_s \quad \textit{for } r \in (r_1, r_n] \tag{4.17}$$

where the integral is a Riemann–Stieltjes integral.

However, we find that this approach suffers from high computational cost since it needs to calculate cubic Hermite spline and solve the CDE for every noisy sample input during training. It thus has low scalability for generating time-series data with long sequences. Nevertheless, we believe this direction is worth exploring for future research.

4.6.3.2 TimeDiff Training

The diffusion model is trained using $\mathcal{L}_{\text{train}}$ in Equation (4.15). We set λ to 0.01. We use cosine scheduling [132] for the variances $\left\{\beta^{(t)}\right\}_{t=1}^{T}$. We apply the exponential moving average to model parameters with a decay rate of 0.995. We use Adam optimizer [133] with a learning rate of 0.00008, $\beta_1 = 0.9$, and $\beta_2 = 0.99$. We set the total diffusion step T to be 1000, accumulate the gradient for every 2 steps, use 2 layers for the BRNN, and use a batch size of 32 across all our experiments.

4.6.3.3 Baselines

For a fair comparison, we use a 2-layer RNN with a hidden dimension size of four times the number of input features. We utilize the LSTM as our architecture whenever applicable. We use a hidden dimension size of 256 for the eICU dataset.

For deterministic models such as the T-Forcing and P-Forcing, we uniformly sample the initial data vector from the real training data. We subsequently use the initial data vector as an input to the deterministic models to generate the synthetic sequence by unrolling.

For stochastic process diffusion, we set *gp_sigma* to be 0.1 for Gaussian process (GP) and *ou_theta* to be 0.5 for Ornstein-Uhlenbeck (OU) process. For discrete diffusion, we set the total diffusion step at 1000. We use Adam optimizer with a learning rate of 0.00001 and batch size of 32 across all the experiments.

4.6.3.4 Software

We set the seed to 2023 and used the following software for our experiments.

Table 4.7: Software packages.

Method	Software
TimeDiff	PyTorch 2.0.1
EHR-M-GAN [111]	TensorFlow 1.14.0
DSPD/CSPD (GP or OU) [96]	PyTorch 2.0.1
GT-GAN [92]	PyTorch 2.0.0
TimeGAN [91]	TensorFlow 1.10.0
RCGAN [87]	TensorFlow 1.10.0
C-RNN-GAN [86]	PyTorch 2.0.1
T-Forcing [123, 124]	PyTorch 1.0.0
P-Forcing [125]	PyTorch 1.0.0

4.6.4 Evaluation Metrics

4.6.4.1 Discriminative and predictive scores

To ensure consistency with results obtained from TimeGAN and GT-GAN, we adopt the same source code from TimeGAN for calculating discriminative scores. We train a GRU time-series classification model to distinguish between real and synthetic samples, and $|0.5 - \text{Accuracy}|$ is used as the score.

For predictive scores, we use the implementation from GT-GAN, which computes the mean absolute error based on the next step *vector* prediction (see Appendix D of the GT-GAN paper [92]). For consistency, we compute the predictive scores for the Stocks

and Energy datasets by employing the implementation from TimeGAN that calculates the error for the next step *scalar* prediction. We apply standardization to the inputs of the discriminator and predictor and use linear activation for the predictor for all EHR datasets.

4.6.4.2 t-SNE

We perform hyperparameter search on the number of iterations, learning rate, and perplexity to optimize the performance of t-SNE [134]. We use 300 iterations, perplexity 30, and scaled learning rate [135]. We flatten the input data along the feature dimension, perform standardization, and then apply t-SNE directly to the data without using any summary statistics. We uniformly randomly select 2000 samples from the synthetic, real training, and real testing data for t-SNE visualizations on the eICU, MIMIC-III, MIMIC-IV, and Energy datasets. For the HiRID and Stocks dataset, we use 1000 and 700 samples, respectively, due to the limited size of real testing data.

4.6.4.3 In-hospital mortality prediction

Train on Synthetic, Test on Real (TSTR): We use the default hyperparameters for the six ML models using the scikit-learn software package. The models are trained using two input formats: (1) raw multivariate time-series data flattened along the feature dimension; (2) summary statistics for each feature (the first record since ICU admission, minimum, maximum, record range, mean, standard deviation, mode, and skewness). After training, the models are evaluated on real testing data in terms of AUC.

Train on Synthetic and Real, Test on Real (TSRTR): To evaluate the effect of the increased proportion of the synthetic samples for training on model performance, we uniformly randomly sample 2,000 real training data from our training set and use this

subset to train TimeDiff. After training of TimeDiff is complete, we subsequently add different amounts of the synthetic samples to the 2,000 real samples to train ML models for in-hospital mortality prediction. We set the synthetic percentages to be 0.1, 0.3, 0.5, 0.7, 0.9. In other words, the ML models are trained with at most 20,000 samples (18,000 synthetic and 2,000 real). This evaluation also simulates the scenario where synthetic samples from TimeDiff are used for data augmentation tasks in medical applications. Similar to computing the TSTR score, we train the ML models using either raw time-series data or summary statistics of each feature as the input. Results obtained using summary statistics as the input are presented in later sections.

4.6.4.4 NNAA Risk

We calculate the NNAA risk score [127] by using the implementation from this repository. Similar to performing t-SNE, we flatten the data along the feature dimension and apply standardization for preprocessing. The scaled datasets are then used to calculate the NNAA risk score.

For reference, we describe the components of the NNAA score below.

Definition 2. *Let* $S_T = \{x_T^{(1)}, ..., x_T^{(n)}\}$, $S_E = \{x_E^{(1)}, ..., x_E^{(n)}\}$ *and* $S_S = \{x_S^{(1)}, ..., x_S^{(n)}\}$ *be data samples with size n from real training, real testing, and synthetic datasets, respectively. The NNAA risk is the difference between two accuracies:*

$$NNAA = AA_{\text{test}} - AA_{\text{train}}, \tag{4.18}$$

$$AA_{\text{test}} = \frac{1}{2}\left(\frac{1}{n}\sum_{i=1}^{n}\mathbb{I}\big\{d_{ES}(i) > d_{EE}(i)\big\} + \frac{1}{n}\sum_{i=1}^{n}\mathbb{I}\big\{d_{SE}(i) > d_{SS}(i)\big\}\right), \tag{4.19}$$

$$AA_{\text{train}} = \frac{1}{2}\left(\frac{1}{n}\sum_{i=1}^{n}\mathbb{I}\big\{d_{TS}(i) > d_{TT}(i)\big\} + \frac{1}{n}\sum_{i=1}^{n}\mathbb{I}\big\{d_{ST}(i) > d_{SS}(i)\big\}\right), \tag{4.20}$$

where $\mathbb{I}\{\cdot\}$ *is the indicator function, and*

$$d_{TS}(i) = \min_j \left\| x_T^{(i)} - x_S^{(j)} \right\|, \quad d_{ST}(i) = \min_j \left\| x_S^{(i)} - x_T^{(j)} \right\|, \tag{4.21}$$

$$d_{ES}(i) = \min_j \left\| x_E^{(i)} - x_S^{(j)} \right\|, \quad d_{SE}(i) = \min_j \left\| x_S^{(i)} - x_E^{(j)} \right\|, \tag{4.22}$$

$$d_{TT}(i) = \min_{j,j\neq i} \left\| x_T^{(i)} - x_T^{(j)} \right\|, \quad d_{SS}(i) = \min_{j,j\neq i} \left\| x_S^{(i)} - x_S^{(j)} \right\|, \quad d_{EE}(i) = \min_{j,j\neq i} \left\| x_E^{(i)} - x_E^{(j)} \right\|. \tag{4.23}$$

In our experiments, there are instances where $AA_{\text{train}} > AA_{\text{test}}$. To consistently obtain positive values, we use $\text{NNAA} = |AA_{\text{test}} - AA_{\text{train}}|$ for our evaluations.

4.6.4.5 MIR

Our implementation of the MIR score [128] follows the source code in this repository. To keep a similar scale of the distance across different datasets, we apply normalization on the computed distances so that they are in the [0,1] range. We use a threshold of 0.08 for the MIMIC-IV, MIMIC-III, and HiRID datasets. We set the decision threshold to 0.005 for eICU. All the input data is normalized to the [0,1] range.

Bibliography

[1] Richard Bellman. "Dynamic programming". In: *Science* 153.3731 (1966), pp. 34–37.

[2] Richard S Sutton, Andrew G Barto, et al. "Introduction to reinforcement learning". In: (1998).

[3] Tianmeng Hu, Biao Luo, and Chunhua Yang. "Multi-objective optimization for autonomous driving strategy based on Deep Q Network". In: *Discover Artificial Intelligence* 1.1 (2021), pp. 1–11. DOI: 10.1007/s44163-021-00011-3. URL: https://doi.org/10.1007/s44163-021-00011-3.

[4] Chunming Liu, Xin Xu, and Dewen Hu. "Multiobjective reinforcement learning: A comprehensive overview". In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 45.3 (2014), pp. 385–398.

[5] Kristof Van Moffaert, Madalina M Drugan, and Ann Nowé. "Hypervolume-based multi-objective reinforcement learning". In: *International Conference on Evolutionary Multi-Criterion Optimization*. Springer. 2013, pp. 352–366.

[6] Leon Barrett and Srini Narayanan. "Learning all optimal policies with multiple criteria". In: *Proceedings of the 25th international conference on Machine learning*. 2008, pp. 41–47.

[7] Peter Vamplew et al. "Constructing stochastic mixture policies for episodic multiobjective reinforcement learning tasks". In: *Australasian joint conference on artificial intelligence*. Springer. 2009, pp. 340–349.

[8] Shahin Jabbari et al. "Fairness in reinforcement learning". In: *International conference on machine learning*. PMLR. 2017, pp. 1617–1626.

[9] Umer Siddique, Paul Weng, and Matthieu Zimmer. "Learning Fair Policies in Multi-Objective (Deep) Reinforcement Learning with Average and Discounted Rewards". In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 8905–8915. URL: https://proceedings.mlr.press/v119/siddique20a.html.

[10] Mridul Agarwal, Vaneet Aggarwal, and Tian Lan. "Multi-Objective Reinforcement Learning with Non-Linear Scalarization". In: *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*. AAMAS '22. Virtual Event, New Zealand: International Foundation for Autonomous Agents and Multiagent Systems, 2022, 9–17. ISBN: 9781450392136.

[11] Hervé Moulin. *Fair division and collective welfare*. MIT press, 2004.

[12] John F Nash Jr. "The bargaining problem". In: *Econometrica: Journal of the econometric society* (1950), pp. 155–162.

[13] R Duncan Luce and Howard Raiffa. *Games and decisions: Introduction and critical survey*. Courier Corporation, 1989.

[14] Ioannis Caragiannis et al. "The unreasonable fairness of maximum Nash welfare". In: *ACM Transactions on Economics and Computation (TEAC)* 7.3 (2019), pp. 1–32.

[15] Rick Durrett. *Probability: theory and examples*. Vol. 49. Cambridge university press, 2019.

[16] Zimeng Fan et al. "Welfare and fairness in multi-objective reinforcement learning". In: *arXiv preprint arXiv:2212.01382* (2022).

[17] Christopher John Cornish Hellaby Watkins. "Learning from delayed rewards". In: (1989).

[18] Richard S Sutton. "Learning to predict by the methods of temporal differences". In: *Machine learning* 3.1 (1988), pp. 9–44.

[19] Kristof Van Moffaert, Madalina M Drugan, and Ann Nowé. "Scalarized multi-objective reinforcement learning: Novel design techniques". In: *2013 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*. IEEE. 2013, pp. 191–199.

[20] Robert L McNamara et al. "Predicting in-hospital mortality in patients with acute myocardial infarction". In: *Journal of the American College of Cardiology* 68.6 (2016), pp. 626–635.

[21] Fred H Edwards et al. "Development and validation of a risk prediction model for in-hospital mortality after transcatheter aortic valve replacement". In: *JAMA Cardiology* 1.1 (2016), pp. 46–52.

[22] Gregg C Fonarow et al. “Risk stratification for in-hospital mortality in acutely decompensated heart failure: classification and regression tree analysis”. In: *JAMA* 293.5 (2005), pp. 572–580.

[23] Steven L Barriere and Stephen F Lowry. “An overview of mortality risk prediction in sepsis”. In: *Critical Care Medicine* 23.2 (1995), pp. 376–393.

[24] Ali A El-Solh et al. “Comparison of in-hospital mortality risk prediction models from COVID-19”. In: *PloS One* 15.12 (2020), e0244629.

[25] Sujoy Kar et al. “Multivariable mortality risk prediction using machine learning for COVID-19 patients at admission (AICOVID)”. In: *Scientific Reports* 11.1 (2021), p. 12801.

[26] William A Knaus et al. “APACHE—acute physiology and chronic health evaluation: a physiologically based classification system”. In: *Critical Care Medicine* 9.8 (1981), pp. 591–597.

[27] J L Vincent et al. *The SOFA (Sepsis-related Organ Failure Assessment) score to describe organ dysfunction/failure: On behalf of the Working Group on Sepsis-Related Problems of the European Society of Intensive Care Medicine (see contributors to the project in the appendix)*. 1996.

[28] Mervyn Singer et al. “The third international consensus definitions for sepsis and septic shock (Sepsis-3)”. In: *JAMA* 315.8 (2016), pp. 801–810.

[29] William A Knaus et al. “APACHE II: a severity of disease classification system.” In: *Critical Care Medicine* 13.10 (1985), pp. 818–829.

[30] Jean-Roger Le Gall et al. “A simplified acute physiology score for ICU patients.” In: *Critical Care Medicine* 12.11 (1984), pp. 975–977.

[31] Jack E Zimmerman et al. “Acute Physiology and Chronic Health Evaluation (APACHE) IV: hospital mortality assessment for today’s critically ill patients”. In: *Critical Care Medicine* 34.5 (2006), pp. 1297–1310.

[32] William S Cleveland. “Robust locally weighted regression and smoothing scatterplots”. In: *Journal of the American Statistical Association* 74.368 (1979), pp. 829–836.

[33] Alistair EW Johnson, Andrew A Kramer, and Gari D Clifford. “A new severity of illness scale using a subset of acute physiology and chronic health evalua-

tion data elements shows comparable predictive accuracy". In: *Critical Care Medicine* 41.7 (2013), pp. 1711–1718.

[34] Sourabh Katoch, Sumit Singh Chauhan, and Vijay Kumar. "A review on genetic algorithm: past, present, and future". In: *Multimedia Tools and Applications* 80 (2021), pp. 8091–8126.

[35] James Kennedy and Russell Eberhart. "Particle swarm optimization". In: *Proceedings of ICNN'95-International Conference on Neural Networks*. Vol. 4. IEEE. 1995, pp. 1942–1948.

[36] Min Hyuk Choi et al. "Mortality prediction of patients in intensive care units using machine learning algorithms based on electronic health records". In: *Scientific Reports* 12.1 (2022), p. 7180.

[37] Yasser El-Manzalawy et al. "OASIS+: leveraging machine learning to improve the prognostic accuracy of OASIS severity score for predicting in-hospital mortality". In: *BMC Medical Informatics and Decision Making* 21.1 (2021), p. 156.

[38] Scott Levin et al. "Machine-learning-based electronic triage more accurately differentiates patients with respect to clinical outcomes compared with the emergency severity index". In: *Annals of Emergency Medicine* 71.5 (2018), pp. 565–574.

[39] Maximiliano Klug et al. "A gradient boosting machine learning model for predicting early mortality in the emergency department triage: devising a nine-point triage score". In: *Journal of General Internal Medicine* 35 (2020), pp. 220–227.

[40] Woo Suk Hong, Adrian Daniel Haimovich, and R Andrew Taylor. "Predicting hospital admission at emergency department triage using machine learning". In: *PloS One* 13.7 (2018), e0201016.

[41] José A González-Nóvoa et al. "Using explainable machine learning to improve intensive care unit alarm systems". In: *Sensors* 21.21 (2021), p. 7125.

[42] Leo Breiman. "Random forests". In: *Machine Learning* 45 (2001), pp. 5–32.

[43] Tianqi Chen et al. "Xgboost: extreme gradient boosting". In: *R package version 0.4-2* 1.4 (2015), pp. 1–4.

[44] Feng Xie et al. “AutoScore: a machine learning–based automatic clinical score generator and its application to mortality prediction using electronic health records”. In: *JMIR Medical Informatics* 8.10 (2020), e21798.

[45] Berk Ustun and Cynthia Rudin. “Optimized risk scores”. In: *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*. 2017, pp. 1125–1134.

[46] Berk Ustun and Cynthia Rudin. “Learning Optimized Risk Scores.” In: *J. Mach. Learn. Res.* 20.150 (2019), pp. 1–75.

[47] Jiachang Liu et al. “FasterRisk: Fast and Accurate Interpretable Risk Scores”. In: *Advances in Neural Information Processing Systems*. Vol. 35. Curran Associates, Inc., 2022, pp. 17760–17773.

[48] Alistair EW Johnson et al. “MIMIC-III, a freely accessible critical care database”. In: *Scientific data* 3.1 (2016), pp. 1–9.

[49] Tom J Pollard et al. “The eICU Collaborative Research Database, a freely available multi-center database for critical care research”. In: *Scientific Data* 5.1 (2018), pp. 1–13.

[50] Jesse Davis and Mark Goadrich. “The relationship between Precision-Recall and ROC curves”. In: *Proceedings of the 23rd International Conference on Machine Learning*. 2006, pp. 233–240.

[51] Yin Lou, Rich Caruana, and Johannes Gehrke. “Intelligible models for classification and regression”. In: *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2012, pp. 150–158.

[52] Tianqi Chen and Carlos 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.

[53] Yoav Freund and Robert E Schapire. “A decision-theoretic generalization of on-line learning and an application to boosting”. In: *Journal of Computer and System Sciences* 55.1 (1997), pp. 119–139.

[54] Yingxiang Huang et al. “A tutorial on calibration measurements and calibration models for clinical prediction models”. In: *Journal of the American Medical Informatics Association* 27.4 (2020), pp. 621–633.

[55] Glenn W Brier. "Verification of forecasts expressed in terms of probability". In: *Monthly Weather Review* 78.1 (1950), pp. 1–3.

[56] Stanley Lemeshow and David W Hosmer Jr. "A review of goodness of fit statistics for use in the development of logistic regression models". In: *American Journal of Epidemiology* 115.1 (1982), pp. 92–106.

[57] Yizhao Zhou et al. "Missing data matter: an empirical evaluation of the impacts of missing EHR data in comparative effectiveness research". In: *Journal of the American Medical Informatics Association* (2023), ocad066.

[58] Stef Van Buuren and Karin Groothuis-Oudshoorn. "mice: Multivariate imputation by chained equations in R". In: *Journal of Statistical Software* 45 (2011), pp. 1–67.

[59] Wei-Chao Lin and Chih-Fong Tsai. "Missing value imputation: a review and analysis of the literature (2006–2017)". In: *Artificial Intelligence Review* 53 (2020), pp. 1487–1509.

[60] William A Knaus et al. "The APACHE III prognostic system: risk prediction of hospital mortality for critically III hospitalized adults". In: *Chest* 100.6 (1991), pp. 1619–1636.

[61] Jean-Roger Le Gall, Stanley Lemeshow, and Fabienne Saulnier. "A new simplified acute physiology score (SAPS II) based on a European/North American multicenter study". In: *JAMA* 270.24 (1993), pp. 2957–2963.

[62] Jean-Roger Le Gall et al. "The Logistic Organ Dysfunction system: a new way to assess organ dysfunction in the intensive care unit". In: *JAMA* 276.10 (1996), pp. 802–810.

[63] Roger C Bone et al. "Definitions for sepsis and organ failure and guidelines for the use of innovative therapies in sepsis". In: *Chest* 101.6 (1992), pp. 1644–1655.

[64] Trevor J Hastie and Robert J Tibshirani. *Generalized additive models*. Vol. 43. CRC press, 1990.

[65] Caroline Wang et al. "In pursuit of interpretable, fair and accurate machine learning for criminal recidivism prediction". In: *Journal of Quantitative Criminology* (2022), pp. 1–63.

[66] Benjamin Shickel et al. "Deep EHR: A Survey of Recent Advances in Deep Learning Techniques for Electronic Health Record (EHR) Analysis". In: *IEEE Journal of Biomedical and Health Informatics* 22.5 (2018), pp. 1589–1604.

[67] Benjamin Alan Goldstein et al. "Opportunities and challenges in developing risk prediction models with electronic health records data: a systematic review". In: *Journal of the American Medical Informatics Association* 24.1 (2017), pp. 198–208.

[68] Kathleen Benitez and Bradley A. Malin. "Evaluating re-identification risks with respect to the HIPAA privacy rule". In: *Journal of the American Medical Informatics Association: JAMIA* 17.2 (2010), pp. 169–177.

[69] Victor Janmey and Peter L Elkin. "Re-identification risk in HIPAA de-identified datasets: The MVA attack". In: *AMIA Annual Symposium Proceedings*. Vol. 2018. American Medical Informatics Association. 2018, p. 1329.

[70] Chao Yan et al. "A Multifaceted benchmarking of synthetic electronic health record generation models". In: *Nature Communications* 13 (2022), p. 7609.

[71] Jinsung Yoon et al. "EHR-Safe: generating high-fidelity and privacy-preserving synthetic electronic health records". In: *NPJ Digital Medicine* 6 (2023), p. 141.

[72] Aldren Gonzales, Guruprabha Guruswamy, and Scott R Smith. "Synthetic data in health care: a narrative review". In: *PLOS Digital Health* 2.1 (2023), e0000082.

[73] Melissa A. Haendel et al. "The National COVID Cohort Collaborative (N3C): Rationale, design, infrastructure, and deployment". In: *Journal of the American Medical Informatics Association: JAMIA* 28.3 (2020), pp. 427–443.

[74] Emily Herrett et al. "Data Resource Profile: Clinical Practice Research Datalink (CPRD)". In: *International Journal of Epidemiology* 44.3 (2015), pp. 827–836.

[75] Jie Gui et al. "A Review on Generative Adversarial Networks: Algorithms, Theory, and Applications". In: *IEEE Transactions on Knowledge and Data Engineering* 35.4 (2023), pp. 3313–3332.

[76] Xin Yi, Ekta Walia, and Paul S. Babyn. "Generative Adversarial Network in Medical Imaging: A Review". In: *Medical image analysis* 58 (2018), p. 101552.

[77] E. Choi et al. "Generating Multi-label Discrete Patient Records using Generative Adversarial Networks". In: *Proceedings of the 2nd Machine Learning for Healthcare Conference*. Vol. 68. 2017, pp. 286–305.

[78] Mrinal Kanti Baowaly et al. “Synthesizing electronic health records using improved generative adversarial networks”. In: *Journal of the American Medical Informatics Association* 26.3 (2019), pp. 228–241.

[79] Divya Saxena and Jiannong Cao. “Generative Adversarial Networks (GANs): Challenges, Solutions, and Future Directions”. In: *ACM Computing Surveys* 54.3 (2021), p. 63.

[80] Jonathan Ho, Ajay Jain, and Pieter Abbeel. “Denoising diffusion probabilistic models”. In: *Advances in neural information processing systems* 33 (2020), pp. 6840–6851.

[81] Alex Nichol and Prafulla Dhariwal. “Improved Denoising Diffusion Probabilistic Models”. In: *Proceedings of the 38th International Conference on Machine Learning (ICML)*. Vol. 139. 2021.

[82] Robin Rombach et al. “High-resolution image synthesis with latent diffusion models”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 10684–10695.

[83] Huan He et al. “MedDiff: Generating Electronic Health Records using Accelerated Denoising Diffusion Model”. In: *arXiv preprint arXiv:2302.04355* (2023).

[84] Hongyi Yuan, Songchi Zhou, and Sheng Yu. “EHRDiff: Exploring Realistic EHR Synthesis with Diffusion Models”. In: *arXiv preprint arXiv:2303.05656* (2023).

[85] Nicholas I-Hsien Kuo, Louisa R Jorm, and Sebastiano Barbieri. “Synthetic Health-related Longitudinal Data with Mixed-type Variables Generated using Diffusion Models”. In: *ArXiv* abs/2303.12281 (2023).

[86] Olof Mogren. *C-RNN-GAN: Continuous recurrent neural networks with adversarial training*. 2016. arXiv: `1611.09904 [cs.AI]`.

[87] Cristóbal Esteban, Stephanie L. Hyland, and Gunnar Rätsch. *Real-valued (Medical) Time Series Generation with Recurrent Conditional GANs*. 2017. arXiv: `1706.02633 [stat.ML]`.

[88] Andrew M. Dai and Quoc V. Le. *Semi-supervised Sequence Learning*. 2015. arXiv: `1511.01432 [cs.LG]`.

[89] Xinrui Lyu et al. *Improving Clinical Predictions through Unsupervised Time Series Representation Learning*. 2018. arXiv: `1812.00490 [cs.LG]`.

[90] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhutdinov. *Unsupervised Learning of Video Representations using LSTMs*. 2016. arXiv: 1502.04681 [cs.LG].

[91] Jinsung Yoon, Daniel Jarrett, and Mihaela van der Schaar. "Time-series Generative Adversarial Networks". In: *Advances in Neural Information Processing Systems*. Vol. 32. 2019.

[92] Jinsung Jeon et al. "GT-GAN: General Purpose Time Series Synthesis with Generative Adversarial Networks". In: *Advances in Neural Information Processing Systems*. Vol. 35. 2022, pp. 36999–37010.

[93] Patrick Kidger et al. "Neural controlled differential equations for irregular time series". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 6696–6707.

[94] Edward De Brouwer et al. "GRU-ODE-Bayes: Continuous modeling of sporadically-observed time series". In: *Advances in neural information processing systems* 32 (2019).

[95] Ruizhi Deng et al. *Modeling Continuous Stochastic Processes with Dynamic Normalizing Flows*. 2021. arXiv: 2002.10516 [cs.LG].

[96] Marin Biloš et al. "Modeling Temporal Data as Continuous Functions with Stochastic Process Diffusion". In: *Proceedings of the 40th International Conference on Machine Learning*. Ed. by Andreas Krause et al. Vol. 202. Proceedings of Machine Learning Research. PMLR, 2023, pp. 2452–2470. URL: https://proceedings.mlr.press/v202/bilos23a.html.

[97] Jascha Sohl-Dickstein et al. "Deep unsupervised learning using nonequilibrium thermodynamics". In: *International conference on machine learning*. PMLR. 2015, pp. 2256–2265.

[98] Shuyang Gu et al. "Vector Quantized Diffusion Model for Text-to-Image Synthesis". In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2022), pp. 10686–10696.

[99] Prafulla Dhariwal and Alex Nichol. "Diffusion Models Beat GANs on Image Synthesis". In: *Advances in Neural Information Processing Systems*. Ed. by A. Beygelzimer et al. 2021.

[100] Chitwan Saharia et al. “Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding”. In: *36th Conference on Neural Information Processing Systems*. 2022.

[101] Akim Kotelnikov et al. “TabDDPM: Modelling Tabular Data with Diffusion Models”. In: *ArXiv* abs/2209.15421 (2022).

[102] Ayan Das et al. “ChiroDiff: Modelling chirographic data with Diffusion Models”. In: *ArXiv* abs/2304.03785 (2023).

[103] Yang Song and Stefano Ermon. “Generative modeling by estimating gradients of the data distribution”. In: *Advances in neural information processing systems* 32 (2019).

[104] Yang Song and Stefano Ermon. “Improved techniques for training score-based generative models”. In: *Advances in neural information processing systems* 33 (2020), pp. 12438–12448.

[105] Sitan Chen et al. “Sampling is as easy as learning the score: theory for diffusion models with minimal data assumptions”. In: *The Eleventh International Conference on Learning Representations*. 2022.

[106] Chao Yan et al. “Generating Electronic Health Records with Multiple Data Types and Constraints”. In: *AMIA Annual Symposium Proceedings* 2020 (2020), pp. 1335–1344.

[107] Siddharth Biswal et al. “EVA: Generating Longitudinal Electronic Health Records Using Conditional Variational Autoencoders”. In: *Proceedings of the 6th Machine Learning for Healthcare Conference*. Vol. 149. 2021, pp. 260–282.

[108] Ahmed Ammar Naseer et al. “ScoEHR: Generating Synthetic Electronic Health Records using Continuous-time Diffusion Models”. In: (2023).

[109] Taha Ceritli et al. “Synthesizing Mixed-type Electronic Health Records using Diffusion Models”. In: *arXiv preprint arXiv:2302.14679* (2023).

[110] Heejoon Koo and To Eun Kim. “A Comprehensive Survey on Generative Diffusion Models for Structured Data”. In: *ArXiv* abs/2306.04139v2 (2023).

[111] Jin Li et al. “Generating synthetic mixed-type longitudinal electronic health records for artificial intelligent applications”. In: *NPJ Digital Medicine* 6.1 (2023), p. 98.

[112] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation". In: *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*. Springer. 2015, pp. 234–241.

[113] Diederik P. Kingma and Max Welling. "Auto-Encoding Variational Bayes". In: *ArXiv* abs/1312.6114 (2013).

[114] Emiel Hoogeboom et al. "Argmax Flows and Multinomial Diffusion: Learning Categorical Distributions". In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato et al. Vol. 34. Curran Associates, Inc., 2021, pp. 12454–12465. URL: https://proceedings.neurips.cc/paper_files/paper/2021/file/67d96d458abdef21792e6d8e590244e7-Paper.pdf.

[115] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[116] Yang Song et al. "Score-based generative modeling through stochastic differential equations". In: *arXiv preprint arXiv:2011.13456* (2020).

[117] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[118] Dan Hendrycks and Kevin Gimpel. "Gaussian error linear units (gelus)". In: *arXiv preprint arXiv:1606.08415* (2016).

[119] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. "Layer normalization". In: *arXiv preprint arXiv:1607.06450* (2016).

[120] Alistair E. W. Johnson et al. "MIMIC-IV, a freely accessible electronic health record dataset". In: *Scientific Data* 10 (2023). Article no. 1.

[121] Tom J. Pollard et al. "The eICU Collaborative Research Database, a freely available multi-center database for critical care research". In: *Scientific Data* 5.1 (2018), pp. 1–13.

[122] Stephanie L Hyland et al. "Early prediction of circulatory failure in the intensive care unit using machine learning". In: *Nature medicine* 26.3 (2020), pp. 364–373.

[123] Alex Graves. "Generating sequences with recurrent neural networks". In: *arXiv preprint arXiv:1308.0850* (2013).

[124] Ilya Sutskever, James Martens, and Geoffrey E Hinton. "Generating text with recurrent neural networks". In: *Proceedings of the 28th international conference on machine learning (ICML-11)*. 2011, pp. 1017–1024.

[125] Alex M Lamb et al. "Professor forcing: A new algorithm for training recurrent networks". In: *Advances in neural information processing systems* 29 (2016).

[126] Laurens Van der Maaten and Geoffrey Hinton. "Visualizing data using t-SNE." In: *Journal of machine learning research* 9.11 (2008).

[127] Andrew Yale et al. "Generation and evaluation of privacy preserving synthetic health data". In: *Neurocomputing* 416 (2020), pp. 244–255.

[128] Gaoyang Liu et al. "Socinf: Membership inference attacks on social media health data with machine learning". In: *IEEE Transactions on Computational Social Systems* 6.5 (2019), pp. 907–921.

[129] Reza Sadeghi, Tanvi Banerjee, and William Romine. "Early hospital mortality prediction using vital signals". In: *Smart Health* 9 (2018), pp. 265–274.

[130] Seyedmostafa Sheikhalishahi, Vevake Balaraman, and Venet Osmani. "Benchmarking machine learning models on eICU critical care dataset". In: *arXiv preprint arXiv:1910.00964* (2019).

[131] Jerome Friedman, Trevor Hastie, and Rob Tibshirani. "Regularization paths for generalized linear models via coordinate descent". In: *Journal of statistical software* 33.1 (2010), p. 1.

[132] Alexander Quinn Nichol and Prafulla Dhariwal. "Improved denoising diffusion probabilistic models". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 8162–8171.

[133] Diederik Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *International Conference on Learning Representations (ICLR)*. San Diega, CA, USA, 2015.

[134] Martin Wattenberg, Fernanda Viégas, and Ian Johnson. "How to use t-SNE effectively". In: *Distill* 1.10 (2016), e2.

[135] Anna C Belkina et al. "Automated optimized parameters for T-distributed stochastic neighbor embedding improve visualization and analysis of large datasets". In: *Nature communications* 10.1 (2019), p. 5415.