CSE 6250 Final Project

Team C3: Bo Fan, Surya Garikipati

Link to Github: https://github.com/BoFanDY/BD4H-C3-project

Link to Google Drive: https://drive.google.com/drive/folders/1MFlUNiHoK9KlXRnepsJrIzSrW6GdzZOJ

Please look for the presentation video and slides in the google drive (bd4h-Teamc3-presentation.mov) (bd4h-presentation.pptx)

I. Introduction

The paper *"Distilling Knowledge from Publicly Available Online EMR Data to Emerging Epidemic for Prognosis"* explores how knowledge distilled from electronic medical records (EMR) [1] can be leveraged for epidemic prognosis, particularly in settings with limited data availability. The authors present a novel framework using public EMR data to predict patient outcomes by training a robust prognostic model. This model has potential applications in real-world scenarios where timely prognosis during an epidemic can inform healthcare strategies and resource allocation.

In this project, we aim to reproduce the paper's results and assess the robustness and scalability of its methodology. Reproducing these results is crucial to verify the claims and evaluate the framework's feasibility for wider application. Additionally, this reproduction will provide insights into optimizing the model further, improving the generalizability of the approach to other datasets and use cases.

II. Scope of Reproducibility

The scope of this work centers on validating the accuracy and effectiveness of the proposed model by replicating the experimental setup and methodology described in the paper. Specifically, we aim to:

- Verify the model's ability to distill meaningful knowledge from EMR data, as measured by its performance metrics for epidemic prognosis.
- Reproduce the Mean Squared Error (MSE) and Mean Absolute Error (MAE) results on the TJH test dataset and compare these with the original paper's results.
- Evaluate model performance across varying training data volumes, comparing our findings with the paper's results.
- Reproduce the PCA analysis on post-processed data for clustering "alive" and "dead" patient groups, benchmarking against the visualizations presented in the paper.

Our primary objective is to achieve results comparable to the original paper's metrics while identifying challenges in replicating the methodology. These insights will help propose optimizations to improve the framework's robustness and scalability.

III. Methodology

A. Data Description

There were three main datasets used in the paper: the COVID-19 Target Dataset from Tongji Hospital (TJH) [2], the PhysioNet Source Dataset [3], and the COVID-19 Target Dataset from HM Hospital (HMH) [4]. These EMR datasets were collected by hospitals, and both the PhysioNet and TJH datasets are publicly available for download from their respective websites.

1) PhysioNet Source Dataset: This dataset was used to train both the teacher and student models. It includes 34 clinical features, 8 vital sign features, and 26 lab features. According to the paper, the dataset contains 40,336 unique patient records stored as .psv files. The label in this dataset is based on the Sepsis-3 clinical criteria. The teacher models were trained on the full dataset, while the student model was trained on a subset containing only features relevant to the target TJH dataset.

2) HMH Dataset: The HMH dataset is private and not publicly available. Since the paper primarily focuses on the TJH dataset for presenting results and proposed methods, we did not work on this dataset.

3) TJH Target Dataset: This is the primary dataset used in our reproducibility work, as all key results were based on predicting the Length of Stay (LOS) in this dataset. The TJH dataset consists of two files: a training file and a test file. The training dataset contains 6,120 record instances, 374 unique PATIENT_IDs, and 81 features, while the test dataset contains 757 record instances, 110 unique PATIENT_IDs, and 8 features.

A statistical summary of the LOS label in the training data is as follows: mean = 35.992, median = 21, standard deviation = 25.768, min = 0, and max = 70.

4) TJH X Dataset Post-Processing: This was a significant part of our work. Since only the raw data from the TJH dataset [2] was available, we developed a post-processing method to generate a time-series dataset for training and testing based on the description in the DistCare paper. First, we selected the relevant features based on the test data and calculated the target LOS using the columns `[Admission time, Discharge time, outcome]`, where `Outcome` is 0 for alive and 1 for dead. Next, we grouped by `PATIENT_ID` and sorted it by `RE_DATE` (the recorded date for this instance).

By defining the sequence length as the maximum `RE_DATE` length across all patients and padding the sequence with zeros, we created a sequence of feature vectors in the shape `(batch size, sequence length, feature size`

= 3) for each patient. Consequently, we modified the GRU layer as GRU(3, 32, batch size).

5) TJH Ground Truth y Dataset Post-Processing: The related three features shown in the test dataset were used as X data, and the LOS values were calculated according to the definition mentioned in the paper as:

$$y = \begin{cases} 35 - \text{remaining days}, & \text{if discharged from ICU} \\ 70 - 35, & \text{if died in ICU} \end{cases}$$

6) TJH Categorization: We categorized LOS into four groups (very low, low, high, and critical) in the same manner described in the paper for confusion matrix examination. Specifically, the categorized labels are defined as:

$$\text{label} = \begin{cases} \text{very low}, & \text{if } y < 7 \\ \text{low}, & \text{if } 7 \leq y < 35 \\ \text{high}, & \text{if } 35 \leq y < 63 \\ \text{critical}, & \text{if } y \geq 63 \end{cases}$$

B. Model Description

To reproduce the work in "Distilling Knowledge from Publicly Available Online EMR Data to Emerging Epidemic for Prognosis" [1], three models—teacher, student, and target—were employed to enable the transfer learning process described in the paper. Each model was constructed with three components: a feature extraction layer, a self-attention layer, and a prediction layer. The feature extraction layer captures underlying patterns and representations within the input data, producing feature embeddings that represent learned patterns for each medical feature. The self-attention layer enables the model to capture relationships between different feature steps and time steps by dynamically adjusting its focus. The prediction layer generates the model's final output, providing a prediction of the Length of Stay (LOS).

1) Teacher Model: The teacher model learns relative feature patterns from the large, non-target source dataset (PhysioNet dataset). It was trained on the PhysioNet dataset using all available features. The detailed architecture of the teacher model is illustrated in Figure 1(a). As shown, feature extraction is primarily accomplished through 34 GRU layers and 34 single-attention layers.

2) Student Model: The student model is an intermediate model that borrows extracted feature weights learned from the teacher model and modifies them for further training on the target dataset (TJH). The architecture of the student model is similar to the teacher model, as shown in Figure 1(b), but includes fewer feature extraction layers to align with the number of features in the target dataset. Specifically, the student model contains 18 GRU layers along with 18 single-attention layers. The student model was trained on a partial subset of the PhysioNet dataset that includes only features present in the target dataset.

3) Transfer from Teacher to Student Model: The student model was initialized with the weights from the teacher model, retaining only those weights for features relevant to the TJH dataset. The proposed distillation mechanism was applied by calculating the distillation loss (L_{dist}) as the KL divergence:

$$L_{dist} = D_{KL}(\text{softmax}(s_{tea}, s_{student})) \tag{1}$$

Here, s_{tea} is the feature representation output from the teacher's extraction layers. Additionally, the student model calculated a regular prediction loss (L_{pred}) using Mean Squared Error (MSE). The total loss function was defined as:

$$L_{total} = L_{dist} + L_{pred} \tag{2}$$

By minimizing L_{total}, the student model was trained.

4) Target Model: The target model predicts LOS on the target dataset (TJH). It shares the same architecture as the student model but is further fine-tuned on the TJH training dataset, starting from the trained student model's weights.

5) Transfer from Student Model to Target Model: To preserve the capacity of the trained student model, the GRU weights from the student model were transferred to the target model, while other parameters in the target model were randomly initialized. The target model was fine-tuned on the TJH training dataset and evaluated on the TJH test data. This fine-tuning was necessary to reproduce the results presented in the paper.

6) Customized GRU Transfer Model: We also implemented a customized GRU-based teacher model in this work. The teacher GRU model includes 4 GRU layers with a hidden size of 256, an attention mechanism, and two fully connected layers with a batch normalization layer in between. The model was initialized with pre-trained weights trained on the PhysioNet dataset and fine-tuned on the TJH training dataset before evaluation on the TJH test dataset.

Additionally, a GRU-based student model was developed with a hidden size of 128, 2 GRU layers, and two fully connected layers with a batch normalization layer. This student model was trained using knowledge distillation, leveraging soft targets from the teacher model alongside hard labels from the TJH training dataset. The student model achieved a balance between computational efficiency and predictive performance.

7) DistCare Model: The reproduced DistCare model used the same training objectives as described in the paper, with a total loss function defined as $L_{total} = L_{dist} + L_{pred}$. The Adam optimizer was employed for optimization.

8) GRU-Based Model: The GRU-based models were trained using a combined loss function: Mean Squared Error (MSE) as the hard loss and KL Divergence with temperature $T = 2.0$ as the soft loss, weighted equally ($\alpha = 0.5$). The Adam optimizer was used for optimization.

Finally, we transferred the weights from the pre-trained student model to the target model. The target model was then fine-tuned using the TJH training dataset and evaluated on the TJH test dataset.

C. Computational Implementation

All experiments were implemented using Python with the PyTorch framework. The experiments were conducted on both

(a). Teacher model

```
distcare_teacher(
  (PositionalEncoding): PositionalEncoding(
    (dropout): Dropout(p=0, inplace=False)
  )
  (GRUs): ModuleList(
    (0-33): 34 x GRU(1, 32, batch_first=True)
  )
  (LastStepAttentions): ModuleList(
    (0-33): 34 x SingleAttention(
      (tanh): Tanh()
      (softmax): Softmax(dim=None)
    )
  )
  (FinalAttentionQKV): FinalAttentionQKV(
    (W_q): Linear(in_features=32, out_features=32, bias=True)
    (W_k): Linear(in_features=32, out_features=32, bias=True)
    (W_v): Linear(in_features=32, out_features=32, bias=True)
    (W_out): Linear(in_features=32, out_features=1, bias=True)
    (dropout): Dropout(p=0.5, inplace=False)
    (tanh): Tanh()
    (softmax): Softmax(dim=None)
    (sigmoid): Sigmoid()
  )
  (MultiHeadedAttention): MultiHeadedAttention(
    (linears): ModuleList(
      (0-2): 3 x Linear(in_features=32, out_features=32, bias=True)
    )
    (final_linear): Linear(in_features=32, out_features=32, bias=True)
    (dropout): Dropout(p=0.5, inplace=False)
  )
  (SublayerConnection): SublayerConnection(
    (norm): LayerNorm()
    (dropout): Dropout(p=0.5, inplace=False)
  )
  (PositionwiseFeedForward): PositionwiseFeedForward(
    (w_1): Linear(in_features=32, out_features=64, bias=True)
    (w_2): Linear(in_features=64, out_features=32, bias=True)
    (dropout): Dropout(p=0.1, inplace=False)
  )
  (demo_proj_main): Linear(in_features=12, out_features=32, bias=True)
  (demo_proj): Linear(in_features=12, out_features=32, bias=True)
  (output): Linear(in_features=32, out_features=1, bias=True)
  (dropout): Dropout(p=0.5, inplace=False)
  (tanh): Tanh()
  (softmax): Softmax(dim=None)
  (sigmoid): Sigmoid()
  (relu): ReLU()
)
```

(b). student model

```
distcare_student(
  (PositionalEncoding): PositionalEncoding(
    (dropout): Dropout(p=0, inplace=False)
  )
  (GRUs): ModuleList(
    (0-17): 18 x GRU(1, 32, batch_first=True)
  )
  (LastStepAttentions): ModuleList(
    (0-17): 18 x SingleAttention(
      (tanh): Tanh()
      (softmax): Softmax(dim=None)
    )
  )
  (FinalAttentionQKV): FinalAttentionQKV(
    (W_q): Linear(in_features=32, out_features=32, bias=True)
    (W_k): Linear(in_features=32, out_features=32, bias=True)
    (W_v): Linear(in_features=32, out_features=32, bias=True)
    (W_out): Linear(in_features=32, out_features=1, bias=True)
    (dropout): Dropout(p=0.5, inplace=False)
    (tanh): Tanh()
    (softmax): Softmax(dim=None)
    (sigmoid): Sigmoid()
  )
  (MultiHeadedAttention): MultiHeadedAttention(
    (linears): ModuleList(
      (0-2): 3 x Linear(in_features=32, out_features=32, bias=True)
    )
    (final_linear): Linear(in_features=32, out_features=32, bias=True)
    (dropout): Dropout(p=0.5, inplace=False)
  )
  (SublayerConnection): SublayerConnection(
    (norm): LayerNorm()
    (dropout): Dropout(p=0.5, inplace=False)
  )
  (PositionwiseFeedForward): PositionwiseFeedForward(
    (w_1): Linear(in_features=32, out_features=64, bias=True)
    (w_2): Linear(in_features=64, out_features=32, bias=True)
    (dropout): Dropout(p=0.1, inplace=False)
  )
  (demo_proj_main): Linear(in_features=12, out_features=32, bias=True)
  (demo_proj): Linear(in_features=12, out_features=32, bias=True)
  (output): Linear(in_features=32, out_features=1, bias=True)
  (dropout): Dropout(p=0.5, inplace=False)
  (FC_embed): Linear(in_features=32, out_features=32, bias=True)
  (tanh): Tanh()
  (softmax): Softmax(dim=None)
  (sigmoid): Sigmoid()
  (relu): ReLU()
)
```

Fig. 1. The DistCare Framework. Left: Teacher model's healthcare representation learning on source dataset. Middle: Imitating teacher model's behavior on source dataset. Right: Transfer pre-trained parameters from the student model to the target model.

CPU and GPU machines. The CPU used was a 2 GHz 6-Core Intel Core i7, and the GPU used was an NVIDIA GeForce RTX 4070 Super.

The hyperparameters for the models are as follows:

- **DistCare model hyperparameters:** The GRU layers in the DistCare model were configured with a hidden dimension of 32 and an output dimension of 1. The model was trained with a batch size of 32 and a learning rate of 0.001 for 200 epochs. These hyperparameters were selected to closely align with those described in the original paper.
- **GRU transfer learning model hyperparameters:** The GRU teacher model used a hidden dimension of 256, while the student model used a hidden dimension of 128. Both models had an output dimension of 1 and were trained with a batch size of 32 and a learning rate of 0.001. The training process spanned 75 epochs for both models. Dropout rates of 0.4 and 0.2 were applied to the teacher and student models, respectively, to enhance regularization and prevent overfitting.

D. Code

In this work, we focused on data post-processing, transfer learning, fine-tuning the DistCare target model with the TJH dataset, developing a customized GRU transfer learning model, reproducing the results shown in the paper, and conducting analysis and discussion. The repository of the original paper can be found at https://github.com/ArthurLeoM/DistCare.

Our code implementation is available on the Georgia Tech GitHub repository. The repository includes three main notebook files to run the experiments:

- **main_distcare_reproduce.ipynb:** This notebook was developed to handle data processing, and to create the DistCare teacher, student, and target models. It performs

model transfer, training, and testing on the TJH dataset and generates all results reproduced using the same DistCare models proposed in the paper.

- **gru_transfer_distcare.ipynb:** This notebook is designed to create our customized GRU model using teacher-student architectures. It handles training and testing on the TJH dataset and generates the corresponding results.
- **mlp_distcare.ipynb:** This notebook creates a simple multi-layer perceptron (MLP) model. It performs training and testing on the TJH dataset and produces the corresponding results.

For additional details, please refer to the `README.md` file in the repository, which includes all necessary instructions and information.

IV. Results

There are four types of results shown in the paper, and we attempted to reproduce comparable results. Since all main results presented in the paper are based on the Mean Squared Error (MSE) and Mean Absolute Error (MAE) between the target Length of Stay (LOS), calculated by definition, and the LOS predicted by the target model, we focused on obtaining these metrics in our preliminary results. Further fine-tuning and additional results, such as confusion matrix analysis and model comparisons, were left for future work.

A. MSE and MAE Performance on the TJH Dataset

The MSE and MAE results on the TJH test dataset obtained with each trained model are shown in Figure 2. As illustrated in the figure, the reproduced DistCare model achieves an MSE of 305.8588 and an MAE of 16.4427 after hyperparameter tuning, which is worse than the results reported in the paper (MSE = 198.9287 and MAE = 9.7518). This discrepancy is preliminarily attributed to potential differences in data post-processing.

However, our customized GRU transfer learning model, leveraging the weights of the pre-trained DistCare model, produces results comparable to those in the paper (MSE = 218.8638 and MAE = 6.6734). Interestingly, our MLP model outperforms the others, achieving an MSE of 152.2419 and an MAE of 6.4463, demonstrating its ability to better capture the relationship between features and targets in our work.

B. MSE Performance on Test Dataset with Model Trained on Different Training Data Volumes

The paper shows the MSE performance on the test dataset with the model trained on the 90%, 80%, 50%, 20%, and 10% of the training data. Both the DistCare model and the GRU transfer learning model exhibit a clear trend: as the training data volume decreases, the MSE increases. When 90% or 80% of the training data is used, the models maintain relatively low MSE values, demonstrating effective generalization with sufficient data. However, as the training data volume reduces to 10%, the performance of the models deteriorates significantly. Notably, the DistCare model consistently outperforms the GRU transfer learning model across different data volumes,

	MSE	MAE
DistCare model results we reproduced	305.8588	16.4427
DistCare model results shown in the paper	198.9287	9.7518
GRU transfer learning model	218.8638	6.6734
MLP	152.2419	6.4463

Fig. 2. MSE and MAE performance of the reproduced DistCare model, GRU transfer learning model, and MLP model compared to the results reported in the paper. The table highlights the differences in predictive accuracy, demonstrating the performance of our custom models alongside the original DistCare model results.

underscoring the benefits of leveraging transfer learning and pre-trained weights in preserving predictive performance.

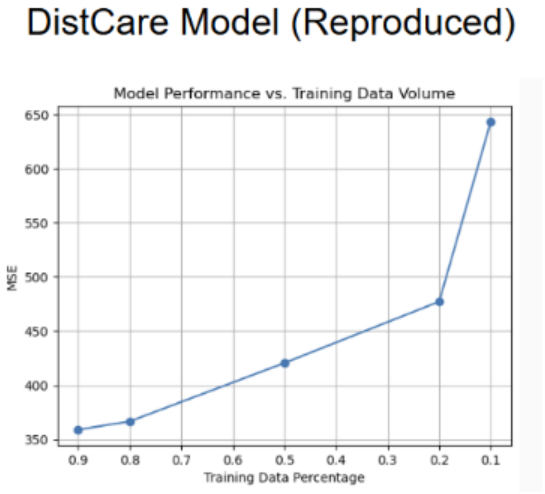

Fig. 3. MSE performance of the reproduced DistCare model on the test dataset with varying training data volumes. As training data volume decreases, MSE increases significantly, emphasizing the dependency on sufficient training data.

C. Confusion Matrix on the Prediction of Different Categories

The confusion matrix compares the performance of the GRU transfer learning model with the results from the DistCare model (as shown in the paper) in predicting different health risk categories. For the GRU transfer learning model, the highest accuracy is observed in the ”low” and ”high” categories, with diagonal values of 0.90 and 0.89, respectively, indicating strong predictive capability in these groups. The ”very low” and ”critical” categories also demonstrate acceptable performance, with accuracies of 0.72 and 0.74, though some misclassifications occur. In contrast, the DistCare model presented in the paper exhibits higher accuracy for the ”low” and ”critical” categories but struggles with the ”high” category, where predictions are more distributed. These results highlight the GRU transfer learning model’s generalization capabilities while reflecting areas where improvements are needed to better match the performance of the DistCare model.

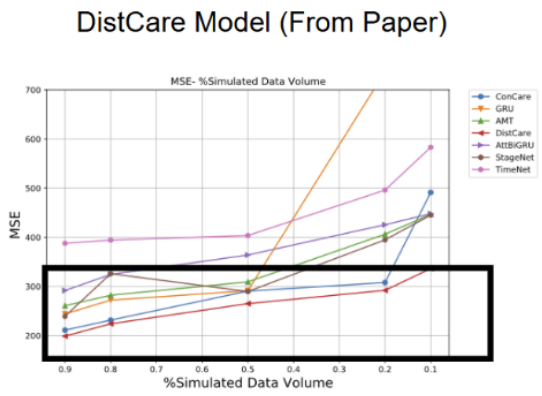

Fig. 4. MSE performance of the DistCare model as reported in the original paper, trained on varying percentages of simulated data volume. The black box highlights comparable ranges with reproduced results.

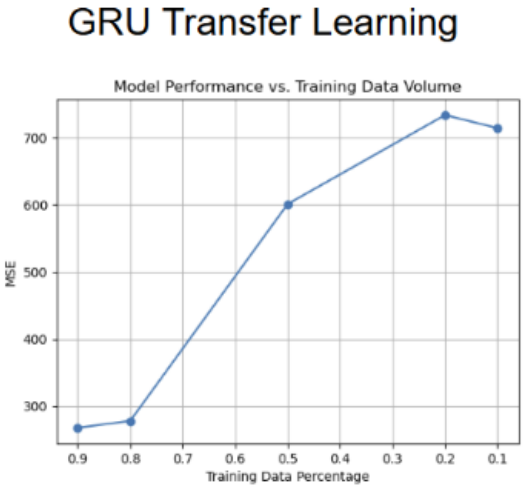

Fig. 5. MSE performance of the GRU transfer learning model on the test dataset with varying training data volumes. The performance deteriorates as training data decreases, showcasing the limitations of the GRU transfer learning model compared to the DistCare model.

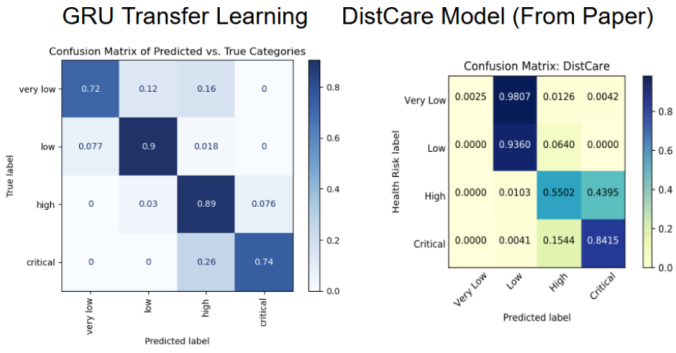

Fig. 6. Comparison of confusion matrices for the GRU transfer learning model and the results from the DistCare model (as shown in the paper) across different health risk categories.

D. PCA Clustering Results

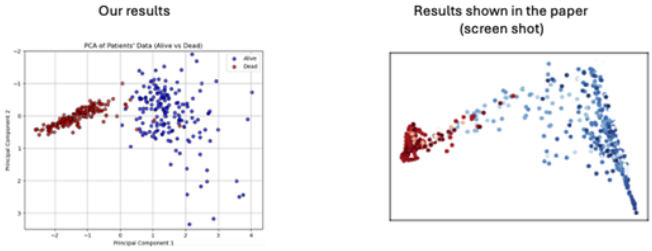

Fig. 7. Results for PCA clustering based on the post-processed features. Our result is shown on the left, and the screenshot of the paper's result is shown on the right. Red dots indicate patients who are dead, and blue dots indicate patients who are alive.

In the figure, the red dot cluster identified by PCA corresponds to patients who, as mentioned in the paper, unfortunately, died (clustered on the left side of the distribution), while the blue dot cluster represents patients who survived and were eventually discharged (clustered on the right side of the distribution). As shown on the right of Figure 7, we successfully obtained similar results to those reported in the paper by applying PCA to the post-processed TJH data. This demonstrates that akin to the paper's results, the data we post-processed also reveals the capacity to distinguish between dead and alive patients based solely on the generated features. Similar to the paper's findings, the red dot clusters indicate patients who unfortunately died (on the left), while the blue dots represent those who survived (on the right). The small discrepancy between our results and the paper's results may be attributed to potential differences in the data processing.

V. Discussion

We reproduced the DistCare model and partially reproduced the results shown in the paper. The MSE and MAE of the reproduced DistCare model are larger, even after hyperparameter tuning, indicating that our reproduced DistCare model performs slightly worse than the tuned version described in the paper. However, our post-processed data produces comparable results to those in the paper after applying PCA, suggesting that our post-processed data contains similar patterns to those presented in the paper. Additionally, we tested different models and developed a customized GRU-based model using a similar teacher-student transfer learning architecture as described in the paper, as well as an MLP model, achieving comparable MSE and MAE on the TJH dataset. These findings indicate that we partially reproduced the results shown in the paper, with even better outcomes in some aspects.

However, we acknowledge that our reproduced DistCare model, fine-tuned on the TJH training data, does not show comparable results, which may be due to potential differences in data processing details. During our reproduction work, we found that the easy parts included understanding and implementing the model and data processing. Additionally, the results were straightforward to interpret.

We also encountered some challenges and identified them as the difficult parts: the paper lacks some necessary details in the data processing section, making it challenging for us to determine the exact post-processing procedure used in the paper. Additionally, tuning many hyperparameters was time-consuming.

Our suggestion to the author is to provide more details on data post-processing and use variables instead of hardcoded numbers to make it easier for others to understand and reproduce the work.

VI. Conclusion

In this work, we reproduced the model used in the paper, performed data processing, conducted experiments, and obtained reproduced results. Additionally, we developed and implemented some customized models using similar ideas from the paper and achieved comparable results. The results were analyzed, and different aspects of the reproduction work were carefully discussed.

References

[1] Ma, Liantao, et al. "Distilling knowledge from publicly available online EMR data to emerging epidemic for prognosis." *Proceedings of the Web Conference 2021*, 2021.

[2] Yan, Li, et al. "An interpretable mortality prediction model for COVID-19 patients." *Nature Machine Intelligence*, vol. 2, no. 5, 2020, pp. 283–288.

[3] Reyna, Matthew A., et al. "Early prediction of sepsis from clinical data: the PhysioNet/Computing in Cardiology Challenge 2019." *Critical Care Medicine*, vol. 48, no. 2, 2020, pp. 210–217.

[4] Hospitales, H. M. "COVID data save lives." *HM Hospitales*, 2020.