CS6611 – CREATIVE AND INNOVATIVE PROJECT

Second Review, Team – 38

Team Members:
Shobia TR – 2021503051
Vinothini R – 2021503569
Dhivya PR - 2021503305

Project Guide: Dr. Ponsy R.K Sathia Bhama
Associate Professor,
Dept. of Computer Technology
MIT Campus, Anna University

Title:

Enhancing Mortality Prediction in ICU: A Deep Learning Framework forMultivariate Time Series Classification

Introduction:

In critical care settings like Intensive Care Units (ICUs), accurate and timely diagnosis and treatment decisions are vital for patient outcomes. However, relying solely on human diagnosis is time- consuming, prone to errors and potentially leading to treatment mistakes. Although severity scoring methods are available to assist physicians, they often rely on static snapshots of a patient's health during their ICU stay and overlook that. Recent advancements in survival analysis methods, many of these approaches primarily use (semi)parametric models with static covariates. These models fail to capitalize on the valuable information conveyed by time-varying Electronic Health Record (EHR) data, which captures changes in patient health conditions over time. The mortality prediction framework begins with preprocessing tasks aimed at ensuring data quality and robustness. Missing values in the EHR data are addressed using GAN-based imputation techniques, which generate plausible values based on the underlying distribution of the data. Additionally, to account for the temporal nature of ICU data, features are dynamically scaled using rolling window scaling, allowing for the adaptation to changing patient states over time.

The core of our framework lies in the application of TCNs and BiGRUs, which are capable of capturing complex temporal dependencies and interactions between clinical variables. TCNs, known for their ability to analyze sequential data with varying lengths, are employed to extract temporal patterns from ICU time-series data. Meanwhile, BiGRUs offer bidirectional processing capabilities, enabling the model to capture both forward and backward temporal dependencies in the data. By integrating these architectures, our framework seeks to harness the temporal dimension of patient trajectories, thereby enhancing the predictive capacity for mortality and treatment decisions in ICU settings.

Objective:

- To develop a real-time in-ICU mortality prediction model by using TCN-BiGRU Deep Learning technique.
- The TCN-Bi GRU model aims to leverage temporal dynamics, Learn the hierarchical features from diverse data sources
- Mitigate the impact of missing values in EHR data through the application of Generative Adversarial Network (GAN)-based imputation techniques, ensuring robustness and reliability of the predictive model.
- To better understand the factors contributing to mortality risk among ICU patients by extracting informative features from the data, facilitating insights into patient prognosis and clinical decision-making.

Dataset Description:

MIMIC-III

A popular, publicly available database called MIMIC-III (Medical Information Mart for Intensive Care III) holds de-identified health-related information for more than 40,000 patients who were admitted to Beth Israel Deaconess Medical Centre critical care units between 2001 and 2012. It contains details about demographics, vital signs, lab results, prescription drugs, surgeries, diagnoses, and survival rates.

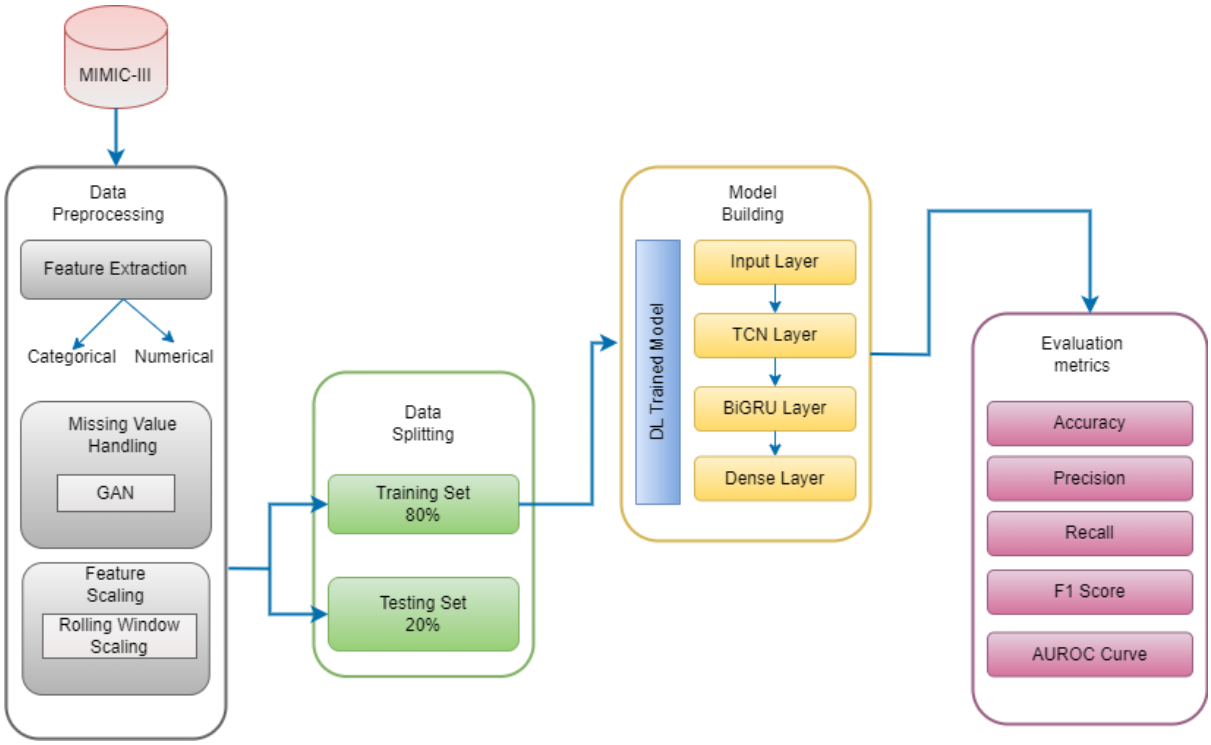
The MIMIC-III dataset is utilised by scholars and medical practitioners for many objectives such as conducting clinical research, creating prediction models, and launching training programmes. It is useful because it offers a sizable, varied, and accurate supply of clinical data for modelling and analysis.

The clearance of the institutional review boards (IRBs) of Beth Israel Deaconess and Massachusetts Institute of Technology (MIT) is necessary in order to access the MIMIC-III dataset.

Factors	**Description**
Age	Chronological order of individuals or entities in a dataset, crucial for demographic analysis.
Gender	Information on identity, often male, female, or non-binary, useful in social and demographic studies.
BMI	Body Mass Index, calculated from weight and height, indicating body fatness for health assessments.
Hypertensive	Dataset includes information on individuals diagnosed with high blood pressure, a chronic condition.
Atrial Fibrillation	Irregular and fast heartbeats originating in the atria, with associated clinical data.
CHD with no MI	Coronary Heart Disease without acute myocardial infarction signs, reflected in medical records.

Diabetes	Metabolic disorder with high blood sugar levels, including demographic and clinical data.
Deficiency Anaemias	Blood disorders with low levels of essential elements for red blood cell synthesis.
Depression	Mental illness characterized by persistent sadness, treatment history, and clinical assessments.
Hyperlipemia	Metabolic disorder with high blood lipid levels, linked to cardiovascular risks.
Renal Failure	Kidney disorders causing waste accumulation, including acute and chronic conditions.
COPD	Chronic respiratory disease causing airflow restriction, with demographic and medical history.
Heartbeats	Heart rate data measured in beats per minute, often collected through various technologies.
Systolic Blood Pressure	Maximum arterial pressure during heartbeats, crucial for cardiovascular health.
Diastolic Blood Pressure	Arterial pressure during heart's resting phase, recorded in mmHg.
Respiratory Rate	Breaths per minute, indicating respiratory health status.
Temperature Data	Measurements of atmospheric or body temperature, often with additional weather factors.
SpO2	Peripheral oxygen saturation, essential for monitoring oxygen levels in medical settings.
Urine Output	Volume of urine excreted over time, important for kidney function assessment.
Hematocrit	Percentage of red blood cells in blood volume, indicative of oxygen-carrying capacity.
Red Blood Cells	Essential blood cells carrying oxygen, with various parameters measured.
MCH	Mean haemoglobin quantity in each red blood cell, measured in picograms.
MCHC	Average haemoglobin concentration in red blood cells, in g/dL.
MCV	Average volume of red blood cells, measured in femtoliters.
RDW	Variation in red blood cell size, expressed as a percentage.
Leukocytes	White blood cells crucial for immune response, with differential counts included.
Platelets	Blood cell fragments essential for clotting, with counts and indices measured.
Neutrophils	White blood cells important for bacterial defense, with counts and percentages.
Basophils	White blood cells involved in inflammation and allergies, with counts measured.
Lymphocytes	Essential white blood cells for immune function, with subsets counted.

Block Diagram:



List of Modules:

1. **Data Preprocessing Module:**

- This module handles loading, cleaning, and preparing the dataset for modeling by replacing missing values, normalize the dataset and splitting it into training and testing sets.

  - **Missing Value Imputation Using GAN:**

    Missing value imputation is addressed using Generative Adversarial Networks (GANs). GANs leverage two neural networks - a generator and a discriminator - to impute missing values by generating plausible data instances and discerning between real and generated data. Through iterative training, the generator improves its imputation accuracy until realistic data instances are produced. This innovative approach offers a more nuanced and context-aware solution compared to traditional imputation methods.

  - **Normalization Using Rolling Window Scaling:**

    Rolling window scaling applies scaling techniques such as Min-Max scaling or Standardization within a rolling window context. This method preserves temporal

relationships within the data while ensuring that features are appropriately scaled for downstream analysis or modeling tasks. By adapting scaling to the local characteristics of the data, rolling window scaling enhances the robustness and accuracy of the preprocessing pipeline, especially in scenarios where the distribution of features varies over time.

2. **Classification Technique Module:**

   - This module defines and trains a classification model on the imputed dataset generated by the GAN,aiming to predict the target variable

   - ***Temporal Convolutional Network (TCN):*** TCN layers are utilized to capture temporal dependencies in the sequential data. TCN is known for its ability to efficiently model long-range dependencies in sequential data while maintaining parallelism and computational efficiency.

   - ***Bidirectional Gated Recurrent Unit (BiGRU):*** BiGRU layers are employed to capture bidirectional dependencies in the sequential data. GRU units are a type of recurrent neural network (RNN) layer that effectively captures sequential patterns in the data, while the bidirectional nature allows the model to learn from past and future information simultaneously.

   - ***Model Architecture:*** The architecture of the combined TCN and BiGRU model consists of multiple layers stacked together. These layers include TCN layers followed by BiGRU layers to capture both temporal and bidirectional dependencies in the data. The outputs of these layers are then concatenated to combine the information captured by each component.

   - ***Concatenation:*** After extracting features using TCN and BiGRU layers, the outputs are concatenated to merge the information learned from both temporal and bidirectional aspects of the data. This concatenation step allows the model to leverage complementary information from different sources.

   - ***Dense Layers:*** Following the concatenation, dense layers are added to the model architecture to further process the combined features and extract higher-level representations from the data. These dense layers may employ activation functions such as ReLU (Rectified Linear Unit) to introduce non-linearity into the model.

   - ***Output Layer:*** The output layer of the classification model typically consists of a single neuron with a sigmoid activation function. The sigmoid activation function squashes the output of the model into the range [0, 1], making it suitable for binary classification tasks where the target variable is binary (e.g., 0 or 1).

3. **Model Training Module:**

   - This module trains the classification model on the imputed dataset. TCN-BiGRU takes the input shape of the features and returns the compiled combined model.
   - The architecture includes layers followed by concatenation, dense layers, and an output layer with sigmoid activation.
   - During training, the model parameters are adjusted iteratively to minimize the chosen loss function, thereby improving the model's ability to classify instances correctly

4. Model Evaluation Module:

- This module evaluates the trained classification model on the test data to assess its performance using various metrics such as loss, accuracy, precision, recall, and F1-score.

    - Accuracy: Accuracy measures the ratio of correctly predicted instances to the total instances. It is a widely used metric for overall model performance.

$$Accuracy = \frac{Number\ of\ Correct\ Predictions}{Total\ Number\ of\ Predictions}$$

    - Precision: Precision is the ratio of correctly predicted positive observations to the total predicted positives. It is particularly relevant when the cost of false positives is high.

$$Precision = \frac{TP}{TP + FP}$$

    - Recall (Sensitivity): Recall is the ratio of correctly predicted positive observations to the all observations in the actual positive class. It is crucial when the cost of false negatives is high.

$$Recall = \frac{TP}{TP + FN}$$

    - F1 Score: The F1 Score is the harmonic mean of precision and recall. It provides a balanced measure that considers both false positives and false negatives.

$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

    - Confusion Matrix: The confusion matrix provides a detailed breakdown of the model's performance, showing the count of true positives, true negatives, false positives, and false negatives.

    - AUC (Area Under the Curve): AUC score quantifies the performance of a binary classification model by measuring the area under the receiver operating characteristic (ROC) curve, reflecting its ability to distinguish between the positive and negative classes.

    - NRMSE: It is a function that allows the user to calculate the normalized root mean square error (NRMSE) as absolute value between predicted and observed values using different type of normalization methods used to evaluate data imputation techniques.

$$\text{NRMSE} = \sqrt{\frac{mean(original\ value - imputed\ value)^2}{\max(original\ value) - \min(original\ value)}}$$

Algorithm:

1. **Data preprocessing**

Load Dataset:

- Load the dataset into a pandas Data Frame.

Handling missing values and normalization:

- Define the generator and discriminator models
- Include batch normalization layers after dense layers in both the generator and discriminator models.
- Generate synthetic samples using the trained generator.
- Impute missing values in the original dataset using the generated synthetic samples.

Split Dataset:

- Split the dataset into features (X) and the target variable (y).

Generative Adversial Network

Input:

Data01.csv: A CSV file containing the dataset with missing values.

Output:

A new CSV file ("complete_dataset.csv") containing the dataset with imputed values.
NRMSE scores for each column to assess imputation quality

Steps:

Load and Preprocess Data:

- Read the CSV file using pandas (pd.read_csv).
- Fill missing values (NA) with zeros (df.fillna(0)).

Split the data into two frames:

- complete_rows: Rows containing no missing values.
- incomplete_rows: Rows with missing values.

Define input and output shapes for the GAN:

- input_shape: Number of features excluding non-feature columns (assuming the first two columns are not features).
- output_shape: Same as input shape (predicting the missing features).

Build GAN Models:

- Generator Model (build_generator):
- Creates a neural network architecture to predict missing features.
- Uses Dense layers for feature learning.

Discriminator Model (build_discriminator):

- Another neural network architecture to differentiate between real (complete data) and generated data.
- Uses Dense layers for classification.

Train the Generator:

- Trains the generator model to minimize the difference between predicted missing values and the actual values in the complete data.
- Uses a process called adversarial training where the generator and discriminator compete.

Impute Missing Values:

- Once trained, use the generator model to predict missing values for the incomplete data.
- Replace missing entries in incomplete_rows with the predicted (imputed) values.

Combine and Evaluate:

- Concatenate complete_rows and the imputed incomplete_rows to form a complete dataset.
- Optionally, impute missing values using the mean of each column (complete_datasett).
- Calculate Normalized Root Mean Squared Error (NRMSE) to evaluate the quality of imputation on each column.
- Save the imputed dataset to a new CSV file.

Rolling Window Scaling

Input:

CSV file named "complete_dataset.csv" containing a dataset with numerical features.

Output:

CSV file named "norm_dataset.csv" containing the dataset with features scaled using a rolling window approach.

Steps:

Import Libraries:

- Import pandas (pd) for data manipulation.
- Import numpy (np) for numerical operations.

Define Rolling Window Scaling Function (rolling_window_scaling):

- This function takes two arguments:
  - data: A NumPy array representing the data for a single feature.
  - window_size: An integer representing the size of the rolling window.

- Initialize an empty NumPy array (scaled_data) with the same shape as data to store the scaled values.
- Iterate through the data using a loop, considering windows of size window_size:
  - For each window (slice of data with size window_size):
    - Calculate the absolute maximum value (max_abs_val) within the window.
    - To avoid division by zero, perform scaling only if max_abs_val is not zero.
    - In that case, divide each element in the window by max_abs_val.
    - Otherwise, keep the window values unchanged.
    - Store the scaled window values back into the corresponding positions of scaled_data.
    - Return the scaled_data array containing the scaled feature values.

Load Dataset:

- Read the CSV file "complete_dataset.csv" into a pandas DataFrame using pd.read_csv.

Preprocess Data:

- Drop rows with missing values using dataset.dropna(inplace=True). This ensures we only scale numerical features with valid values.

Select Features for Scaling:

- Define a list named numerical_columns containing the names of the numerical features you want to scale.

Apply Rolling Window Scaling:

- Define the window size (window_size) for the rolling window (e.g., 10 in this case).
- Iterate through each column name in numerical_columns:
  - Extract the data for that specific column using dataset[column].
  - Convert the pandas Series to a NumPy array and reshape it to have two dimensions (-1, 1). This ensures it can be used with the rolling_window_scaling function.
  - Apply the rolling_window_scaling function to the column data, passing the reshaped array and the window_size.
  - Assign the scaled data back to the corresponding column in the dataset (dataset[column] = ...).

Display and Save Results:

Print the first 10 rows of the scaled dataset using print(dataset.head(10)).
Save the scaled dataset to a new CSV file named "norm_dataset.csv" using dataset.to_csv("norm_dataset.csv", index=False).

Classification Model

TCN-Bi GRU

Input:

'data1fill_v2.csv' (assumed to contain timeseries data)

Output:

Trained model for binary classification on the timeseries data

Load libraries:

- pandas (for data manipulation), numpy (for numerical computations)
- sklearn.model_selection (for train-test split)
- sklearn.preprocessing (for data scaling)
- tensorflow.keras (for deep learning model building)
- tcn (custom library for Temporal Convolutional Networks)

Load data:

- Read the CSV file 'data1fill_v2.csv' into a pandas DataFrame df.

Preprocess data:

- Separate the features (X) from the target variable (y).
- Split the data into training and testing sets using train_test_split.

- Standardize the features using StandardScaler.
- Reshape the input data for the TCN layer (add time dimension).

Define the model architecture:

- TCN layer:
  - Uses temporal convolutions to extract features from sequential data.
  - Configured with specific parameters like number of filters, kernel size, etc.
- Bi-directional GRU layer:
  - Captures dependencies in both past and future directions of the sequence.
  - Uses dropout for regularization to prevent overfitting.
- Output layer:
  - Applies sigmoid activation for binary classification (0 or 1).
  - Compile the model:
  - Specifies the optimizer (adam), loss function (binary_crossentropy), and metrics (accuracy).

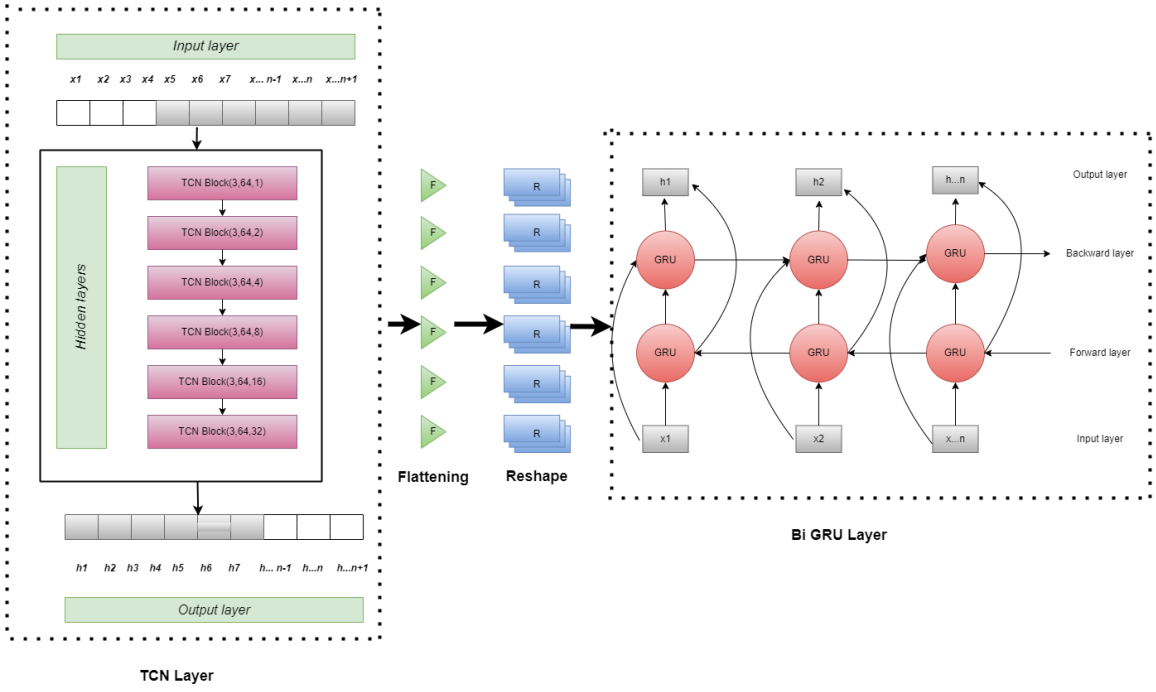
Train the model:

- Trains the model on the training data (X_train, y_train).
- Uses a batch size of 32 and trains for 10 epochs.
- Monitors validation loss during training using the testing data (X_test, y_test).
- Saves the best model based on minimum validation loss using ModelCheckpoint.

Evaluate the model:

- Loads the best saved model weights.
- Evaluates the model performance on the testing data and prints test loss and accuracy.

TCN-Bi GRU Architecture:



Implementation:

1. Missing Value imputation using GAN

Raw data with missing values.

	group	ID	outcome	age	gendera	BMI	hypertensive	\
0	1	125047	0.0	72	1	37.588179	0	
1	1	139812	0.0	75	2	NaN	0	
2	1	109787	0.0	83	2	26.572634	0	
3	1	117468	0.0	86	1	24.369642	1	
4	1	130587	0.0	43	2	83.264629	0	

	atrialfibrillation	CHD with no MI	diabetes	...	Systolic blood pressure	\
0	0	0	1	...	155.866667	
1	0	0	0	...	140.000000	
2	0	0	0	...	135.333333	
3	1	0	0	...	NaN	
4	0	0	0	...	126.400000	

	Diastolic blood pressure	Chloride	Anion gap	Magnesium ion	PH	\
0	68.333333	109.166667	13.166667	2.618182	7.230	
1	65.000000	98.444444	11.444444	1.887500	7.225	
2	61.375000	105.857143	10.000000	2.157143	7.268	
3	NaN	99.777778	8.666667	2.033333	7.430	
4	73.200000	92.071429	12.357143	1.942857	7.370	

After filling missing values.

	group	ID	outcome	age	gendera	BMI	hypertensive	\
0	1	125047	0.0	72	1	37.588179	0	
1	1	139812	0.0	75	2	30.188278	0	
2	1	109787	0.0	83	2	26.572634	0	
3	1	117468	0.0	86	1	24.369642	1	
4	1	130587	0.0	43	2	83.264629	0	

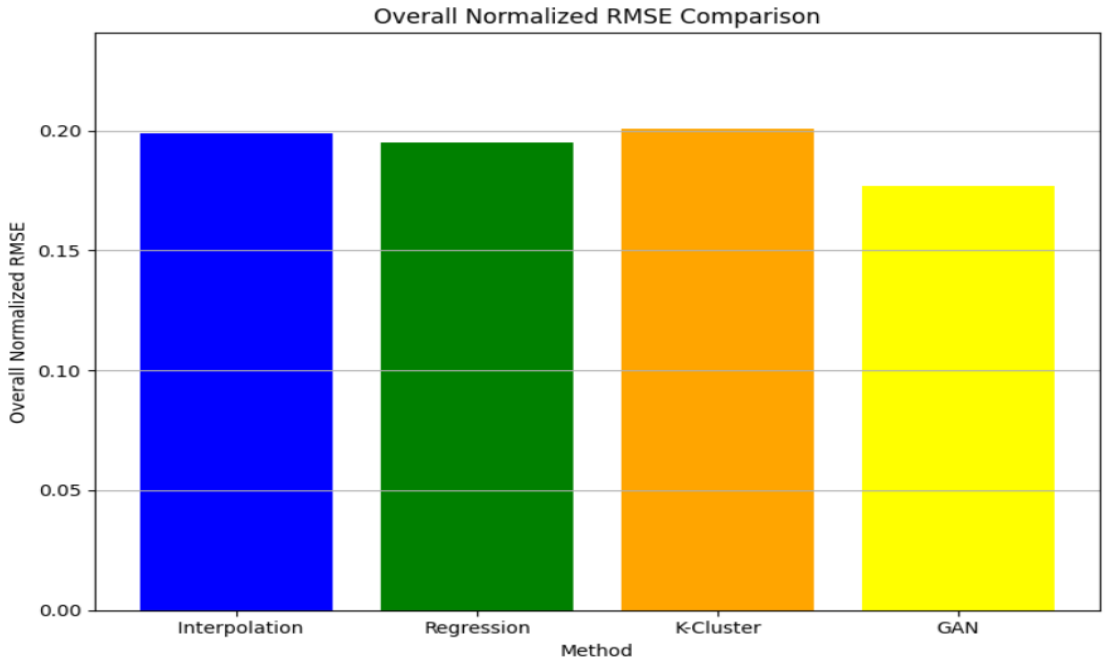
	atrialfibrillation	CHD with no MI	diabetes	...	Systolic blood pressure	\
0	0	0	1	...	155.866667	
1	0	0	0	...	140.000000	
2	0	0	0	...	135.333333	
3	1	0	0	...	117.995035	
4	0	0	0	...	126.400000	

	Diastolic blood pressure	Chloride	Anion gap	Magnesium ion	PH	\
0	68.333333	109.166667	13.166667	2.618182	7.230	
1	65.000000	98.444444	11.444444	1.887500	7.225	
2	61.375000	105.857143	10.000000	2.157143	7.268	
3	59.534497	99.777778	8.666667	2.033333	7.430	
4	73.200000	92.071429	12.357143	1.942857	7.370	

NRMSE Evaluation

```
Normalized rmse after GAN imputation 0.2043386396135714
```

2. NRMSE Evaluation for Missing Value Imputation



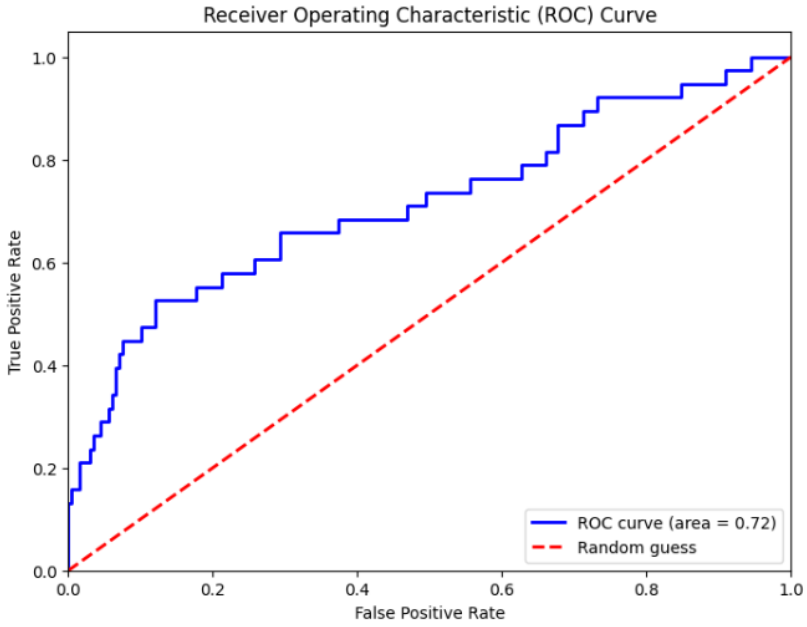
3. Rolling Window Scaling.

First 10 rows of the scaled dataset:

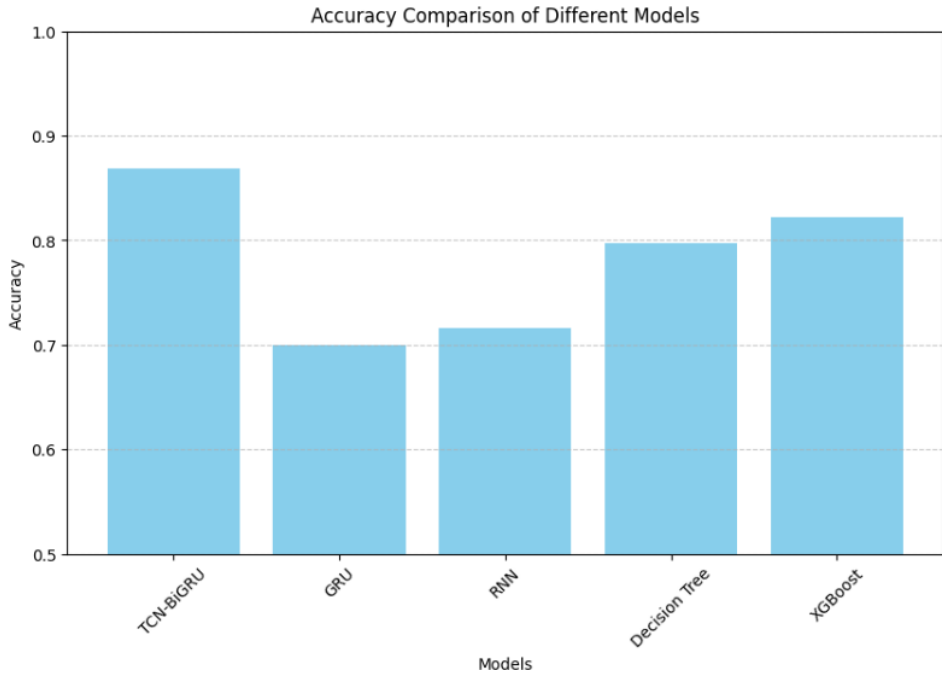
	group	ID	outcome	age	gendera	BMI	hypertensive	\
0	1	125047	0.0	0	1	0.451430	0	
1	1	139812	0.0	0	2	0.362558	0	
2	1	109787	0.0	0	2	0.319135	0	
3	1	117468	0.0	1	1	0.292677	1	
4	1	130587	0.0	0	2	1.000000	0	
5	1	138290	0.0	0	2	0.630679	1	
6	1	154653	0.0	0	1	0.480811	1	
7	1	194420	0.0	0	1	0.786097	1	
8	1	153461	0.0	0	2	0.442144	1	
9	1	113076	0.0	0	2	0.396190	1	

	atrialfibrillation	CHD with no MI	diabetes	...	Systolic blood pressure	\
0	0	0	1	...	0.995571	
1	0	0	0	...	0.894226	
2	0	0	0	...	0.864418	
3	1	0	0	...	0.753673	
4	0	0	0	...	0.807358	
5	0	0	0	...	1.000000	
6	1	0	0	...	0.792428	
7	0	0	0	...	0.667302	
8	1	0	1	...	0.883746	
9	1	0	1	...	0.616390	

4. AUROC score results for the classifier



5. Comparison results of other Models



6. Evaluation metrics for classification model

```
Precision: 0.8000
Recall: 0.9474
F1 Score: 0.8675
```

REFERENCES

1. J. Theis, W. L. Galanter, A. D. Boyd and H. Darabi, "Improving the In-Hospital Mortality Prediction of Diabetes ICU Patients Using a Process Mining/Deep Learning Architecture," in *IEEE Journal of Biomedical and Health Informatics*, vol. 26, no. 1, pp. 388-399, Jan. 2023.

2. T. Kondo *et al*., "Prediction of Short-Term Mortality of Cardiac Care Unit Patients Using Image-Transformed ECG Waveforms," in *IEEE Journal of Translational Engineering in Health and Medicine*, vol. 11, pp. 191-198, 2023.

3. S. Wang, H. Shen, Z. Liu and L. Xie, "TCN-Based Distal Force Feedback Strategy of a Vascular Interventional Surgery Robot," in *IEEE Sensors Journal*, vol. 24, no. 3, pp. 4120-4130, 1 Feb.1, 2024.

4. L. Tong, H. Ma, Q. Lin, J. He and L. Peng, "A Novel Deep Learning Bi-GRU-I Model for Real-Time Human Activity Recognition Using Inertial Sensors," in IEEE Sensors Journal, vol. 22, no. 6, pp. 6164-6174, 15 March15, 2022.

5. S. Ali, S. El-Sappagh, F. Ali, M. Imran and T. Abuhmed, "Multitask Deep Learning for Cost-Effective Prediction of Patient's Length of Stay and Readmission State Using Multimodal Physical Activity Sensory Data," in IEEE Journal of Biomedical and Health Informatics, vol. 26, no. 12, pp. 5793-5804, Dec. 2022

6. K. Psychogyios, L. Ilias, C. Ntanos and D. Askounis, "Missing Value Imputation Methods for Electronic Health Records," in IEEE Access, vol. 11, pp. 21562-21574, 2023

7. Yang, S., Cao, L., Zhou, Y. and Hu, C., 2023. A Retrospective Cohort Study: Predicting 90-Day Mortality for ICU Trauma Patients with a Machine Learning Algorithm Using XGBoost Using MIMIC-III Database. *Journal of Multidisciplinary Healthcare*.

8. J. Wu, X. Ye, C. Mou and W. Dai, "FineEHR: Refine Clinical Note Representations to Improve Mortality Prediction," *2023 11th International Conference on Digital Forensics and Security (ISDFS)*, Chattanooga, TN, USA, 2023, pp. 1-6

9. K. Zhang, K. Niu, Y. Zhou, W. Tai and G. Lu, "MedCT-BERT: Multimodal Mortality Prediction using Medical ConvTransformer-BERT Model," *2023 IEEE 35th International Conference on Tools with Artificial Intelligence (ICTAI)*, Atlanta, GA, USA, 2023, pp.

10. S. Wang, H. Shen, Z. Liu and L. Xie, "TCN-Based Distal Force Feedback Strategy of a Vascular Interventional Surgery Robot," in *IEEE Sensors Journal*, vol. 24, no. 3, pp. 4120-4130, 1 Feb.1, 2024