Data Analysis

On Cardiovascular Diseases

Samunuru Vaishnavi C3060843 MSC BIG DATA ANALYTICS

Contents

Contents 1

Introduction 3

Methodology 3

Object-Oriented Programming Implementation 3

1. Loading Module 3

2. EDA Module 3

3. Preprocessing Module 4

4. Feature Extraction Module 4

5. MLModels Module 4

6. GUI Module 4

Data Loading and Basic statistics 4

Exploratory Data Analysis 5

Basic statistics 5

Correlation heatmap 6

Frequency plot 6

Distribution plot 7

Bar plot 7

Scatterplot 7

Boxplot 7

Pie Chart 7

Data Preprocessing 8

Handling missing values 8

Encoding categorical variables 8

Data Scaling 8

Data Modeling and Fine Tuning 8

Data Class Imbalance 8

Modeling 8

User Interface Implementation 10

Execution Instructions 10

Pre-requisites: 10

Execution steps: 10

Reflection and Professional Development 10
  Learning and Contribution 10
  What Went Well 10
  Areas for Improvement 11
  Professional Development 11
References 11
Appendix 12

Data Analysis on Cardiovascular Diseases Using Machine Learning

Introduction

Cardiovascular disease (CVD) remains one of the leading causes of mortality globally. The thrust of this project is to leverage digital health innovations, particularly machine learning techniques, to analyze patient data for the prediction and prevention of cardiovascular-related fatalities. This report unveils the process and findings from the exploratory data analysis (EDA) to the implementation of predictive models using Python.

Methodology

The methodology or the approach for this whole assessment is as below:

- Data Loading
- Exploratory Data Analysis
- Data Preprocessing
- Modeling and Fine Tuning
- Graphical User Interface Implementation

Object-Oriented Programming Implementation

1. Loading Module

- Decision: A standalone loading module was established to encapsulate the functionality required to read and load datasets into the application.
- Justification: Encapsulating data loading processes in a separate module promotes a clean, maintainable architecture, enabling easier adaptation to changes or various data sources. The use of Python's `pandas` library provides powerful data structures and functions geared towards efficient data manipulation and accessibility.

2. EDA Module

- Decision: The EDA module was specifically dedicated to performing initial data examination, visualizations, and basic statistical analyses. Implemented inheritance concept in this module with base class in Loding module.
- Justification: Having a dedicated EDA module allows for a structured approach to understanding data patterns, distributions, and anomalies. Visualization libraries like `matplotlib` and `seaborn` offer illustrative and comprehensive plots to convey insights clearly, aiding in the exploratory analysis.

3. Preprocessing Module

- Decision: A preprocessing module was created to perform data cleansing, normalization, and splitting.
- Justification: Preprocessing is essential for preparing the data into a format suitable for machine learning. A separate module ensures that the preprocessing steps are modular and can be independently adjusted according to the needs of different datasets or ML models.

4. Feature Extraction Module

- Decision: The feature extraction module was designed to identify, compute, and select additional features that may improve model performance. But in the case of the above dataset, the data has less records of about 300 and only 13 attributes and also no new variables can be extracted from the given attributes. So we are not using feature extraction here.
- Justification: Feature extraction is critical for the success of machine learning algorithms. This module allows for the systematic investigation and inclusion of new features which can be crucial for enhancing predictive accuracy.

5. MLModels Module

- Decision: This module was constructed to house the machine learning algorithms for building predictive models, encompassing the training, testing, and evaluation phases.
- Justification: Centralizing all machine learning operations within a singular module streamlines the development process. It allows for a clear separation between model development and the rest of the application, facilitating easier updates and scalability in model management.

6. GUI Module

- Decision: The GUI module was developed to create a graphical user interface that enables end-users to interact with the application's functionalities in a user-friendly manner.
- Justification: The implementation of a GUI is instrumental for ensuring that the application is user-centric and accessible to individuals irrespective of their technical proficiency. Utilizing a framework like `streamlit` for Python makes the development of intuitive interfaces feasible, ultimately improving the user experience.

Data Loading and Basic statistics

The data loading is done using the pandas library[1]. The data contains 299 records in total with 13 attributes. The data description is

1. Age: Age of the patient.
2. Anaemia: Presence of anaemia (0: No, 1: Yes).
3. Creatinine Phosphokinase: Level of creatinine phosphokinase enzyme in the blood

(mcg/L).
4. Diabetes: Presence of diabetes (0: No, 1: Yes).
5. Ejection Fraction: Percentage of blood leaving the heart at each contraction.
6. High Blood Pressure: Presence of high blood pressure (0: No, 1: Yes).
7. Platelets: Platelet count in the blood (kiloplatelets/mL).
8. Serum Creatinine: Level of serum creatinine in the blood (mg/dL).
9. Serum Sodium: Level of serum sodium in the blood (mEq/L).
10. Sex: Gender of the patient (0: Female, 1: Male).
11. Smoking: Smoking status of the patient (0: No, 1: Yes).
12. Time: Follow-up period in days.
13. DEATH_EVENT: Survival status at the end of the follow-up period (0: Survived, 1: Not Survived).

Exploratory Data Analysis

The dataset provided was methodically explored to pinpoint missing values, outliers, and distributions of vital signs which could influence the predictive models. The pandas library facilitated data manipulation, and the matplotlib [2] and seaborn [3] libraries aided in the visualization of data distributions. Statistical computations, such as mean and standard deviation, illuminated the central tendencies and dispersion within the dataset.

In the Exploratory Data Analysis (EDA) phase, the `HeartDataEDA` class, a crucial component of our analytical application, was implemented to delve deeply into the dataset and unravel the intricate patterns and relationships between features.

Basic statistics

1. age:
   - Mean age is approximately 60.83 years.
   - The youngest individual is 40 years old, and the oldest is 95 years old.
2. anaemia:
   - Around 43.1% of individuals in the dataset have anemia (1 indicates presence, 0 indicates absence).
3. creatinine_phosphokinase:
   - The average level of creatinine phosphokinase is approximately 581.84.
   - The minimum value is 23, and the maximum value is 7861.
4. diabetes:
   - About 41.8% of individuals in the dataset have diabetes (1 indicates presence, 0 indicates absence).
5. ejection_fraction:
   - The average ejection fraction is approximately 38.08%.
   - The minimum ejection fraction is 14%, and the maximum is 80%.
6. high_blood_pressure:
   - Around 35.1% of individuals in the dataset have high blood pressure (1 indicates presence, 0 indicates absence).

7. platelets:
- The average platelet count is approximately 263,358.
- The minimum platelet count is 25,100, and the maximum is 850,000.

8. serum_creatinine:
- The average serum creatinine level is approximately 1.39.
- The minimum serum creatinine level is 0.5, and the maximum is 9.4.

9. serum_sodium:
- The average serum sodium level is approximately 136.63.
- The minimum serum sodium level is 113, and the maximum is 148.

10. sex:
- About 64.9% of individuals in the dataset are male (1 indicates male, 0 indicates female).

11. smoking:
- Around 32.1% of individuals in the dataset are smokers (1 indicates smoker, 0 indicates non-smoker).

12. time:
- The average follow-up period is approximately 130.26 days.
- The minimum follow-up period is 4 days, and the maximum is 285 days.

13. DEATH_EVENT:
- About 32.1% of individuals in the dataset had a death event (1 indicates death event, 0 indicates no death event).

These statistics provide a summary of the central tendency, variability, and distribution of the numerical features in the dataset.

Correlation heatmap

Fig. 7 depicts the correlation heatmap. This visualizes the relationships between different variables including creatinine_phosphokinase, ejection_fraction, age, platelets, serum_creatinine, serum_sodium, and time. The color intensity and the number in each cell represent the correlation coefficient between two variables. A value of 1 indicates a perfect positive correlation while values close to -1 indicate a strong negative correlation.The diagonal cells from top left to bottom right are red with a value of 1.00 indicating each variable's perfect positive correlation with itself. For the rest, the larger the number and darker the color, the higher the correlation between the two variables. Most of the off-diagonal cells have low absolute values indicating weak correlations.

Frequency plot

Fig. 3 shows the frequency plot representing the count of individuals with and without anaemia. The x-axis is labeled “anaemia” with two categories: 0 (no anaemia) and 1 (anaemia). The y-axis is labeled “count” and shows the number of individuals in each category. There are more individuals without anaemia (0) than there are with anaemia (1).

Distribution plot

Fig. 6 shows the distribution of age with a line plot overlaid to show the trend. The most common age range in this dataset appears to be 60-70 as indicated by the highest bar. The distribution is somewhat skewed to the right, indicating a larger number of older individuals.

Bar plot

Fig. 1 depicts a barplot between Death Event and Age. This is representing the ages of individuals at the time of a death event, categorized into two groups: 0 and 1. Both groups have similar average ages, around 60 with small variations as indicated by the error bars.

Fig. 2 depicts a bar plot between Death Event, platelets and diabetes. This is representing the relationship between platelet count, diabetes, and death events. There are two categories for diabetes (0 and 1) and two types of death events (0 and 1). The platelet count is higher for individuals without diabetes regardless of the death event type. However, there is a slight decrease in platelet count in the event of death.

Scatterplot

Fig. 8 depicts a scatterplot with Death Event, age and time. This scatterplot visualizes data points representing individuals, with their age plotted against time and color-coded based on a "DEATH_EVENT". Orange dots represent individuals who did not experience the event (0), while blue dots represent those who did (1). we can observe a pattern that when age is more people are dying when time is less.

Fig. 9 depicts a scatterplot with Death Event, platelets and creatinine. This visualizes the relationship between the levels of serum creatinine and platelets in individuals, color-coded by a binary death event (0 or 1). The majority of data points, especially those with lower serum creatinine levels, are associated with death event 0 (blue), indicating survival. A smaller number of data points, particularly those with higher serum creatinine levels, are associated with death event 1 (orange), indicating non-survival.

Boxplot

Fig. 4 shows a box plot. This plot is comparing the levels of creatinine phosphokinase in individuals with anaemia (1) and without anaemia (0). The median level of creatinine phosphokinase is slightly higher in individuals without anaemia. There are also several outliers in both groups, but especially in the group without anaemia, indicating that some individuals have exceptionally high levels of this enzyme.

Pie Chart

In Fig. 5 the chart has two segments: one blue and one orange. The blue segment constitutes 56.9% of the chart and is labeled with "0," while the orange segment makes up 43.1% of the chart and is labeled with "1".

The design and implementation of this EDA module evoke a crucial understanding of the dataset. It supports transparent, reproducible research practices by not only providing a visual examination but also saving each plot with a unique, randomly generated identifier that ensures traceability of analyses. These foundations cement the EDA module as a valuable asset for preliminary data assessment in the broader context of machine learning and predictive analytics in healthcare

Data Preprocessing

Handling missing values

The data preprocessing includes checking for missing data and handling it. The code for handling missing data has been included however the given dataset has no missing values.

Encoding categorical variables

Next important part of data preprocessing is encoding the categorical variables, the required code has been added, however the categorical variables in the given data are already encoded.

Data Scaling

Data scaling is important to ensure that features contribute equally to the outcome of machine learning algorithms and to improve the convergence speed of optimization algorithms.

Data Modeling and Fine Tuning

Data Class Imbalance

Data class imbalance occurs when the number of instances of one class significantly outnumbers another, potentially leading to biased predictions. The SMOTE algorithm creates synthetic examples of the minority class, improving balance and the performance of classifiers on imbalanced datasets.In the given data we have class imbalance for Anaemia, Death and High Blood Pressure. And we have adjusted it using Synthetic Minority Oversampling TEchnique i.e. SMOTE.

Modeling

The Python code embodies a comprehensive machine learning framework designed to address critical health-related predictions. Central to the process is the `HeartClassifier` class, which encapsulates the complete workflow from data handling, model training, to evaluation. Utilizing the `HeartDataLoader` for data ingestion, it seamlessly transitions from raw data to a pre-processed state, ready for analysis.

Upon loading and preprocessing the data, the classifier tackles the prediction of various outcomes such as 'anaemia', 'DEATH_EVENT', and 'high_blood_pressure'. For each outcome, it employs three distinct machine learning models[4][5] - Logistic Regression, Random Forest, and

Support Vector Machine (SVC) — each with a hyperparameter tuning process via `GridSearchCV` to ensure optimal model performance. The SMOTE technique is integrated to address class imbalance, demonstrating a commitment to robust predictive analysis.

Variable	Model	Precision (0)	Recall (0)	F1-Score (0)	Precision (1)	Recall (1)	F1-Score (1)	Accuracy
Anaemia	Logistic Regression	0.71	0.46	0.56	0.44	0.7	0.54	0.55
	Random Forest	0.59	0.43	0.5	0.36	0.52	0.43	0.47
	SVC	0.69	0.59	0.64	0.46	0.57	0.51	0.58
Death Event	Logistic Regression	0.89	0.78	0.83	0.64	0.8	0.71	0.78
	Random Forest	0.89	0.82	0.86	0.7	0.8	0.74	0.82
	SVC	0.87	0.68	0.76	0.55	0.8	0.65	0.72
High Blood Pressure	Logistic Regression	0.69	0.63	0.66	0.44	0.5	0.47	0.58
	Random Forest	0.75	0.71	0.73	0.54	0.59	0.57	0.67
	SVC	0.71	0.63	0.67	0.46	0.55	0.5	0.6

Table 1: Models' performance across various variables and classifiers

- **Anaemia**:
  - Logistic Regression achieves the highest overall accuracy of 55%.
  - Random Forest and SVC have lower accuracies of 47% and 58% respectively.
  - SVC shows the most balanced performance between precision and recall for both classes.

- **Death Event:**
  - Random Forest outperforms with the highest accuracy of 82%.
  - Logistic Regression follows closely with an accuracy of 78%.
  - SVC has a slightly lower accuracy of 72% but shows balanced performance for both classes.

- **High Blood Pressure:**
  - Random Forest achieves the highest accuracy of 67%.
  - Logistic Regression and SVC have accuracies of 58% and 60% respectively.
  - Random Forest exhibits the best balance between precision and recall for both classes.

Overall, Random Forest tends to perform the best across all variables, followed by Logistic Regression and then SVC. However, the choice of model may vary depending on the specific prediction task and the importance of precision, recall, and overall accuracy.

User Interface Implementation

In the user interface section of our application, built with Streamlit, we provide a seamless and interactive experience for end-users. Upon launching, users can select from five key functionalities: Load Data, Descriptive Stats, EDA (Exploratory Data Analysis), Data Preprocessing, and Modeling. This intuitive interface ensures that users can efficiently navigate through the different stages of data analysis and model development, fostering an accessible environment for both technical and non-technical users alike.

Execution Instructions

Pre-requisites:

- Python 3.10 or more.

Execution steps:

1. Extract the code zip and open the terminal in the extracted folder.
2. Install all the libraries in requirements.txt using “pip install -r requirements.txt” command.
3. Run the following command “streamlit run app.py” in terminal.
4. Open the link provided in the terminal in any browser to view the application.
5. Upload the input csv file to load the data and all other features follow.

The program's structure is logically segmented into modules for ease of understanding and maintenance. Instructions are documented for setting up the environment, loading the dataset, and running the application to extract predictions.

Reflection and Professional Development

Learning and Contribution

The project has been a valuable exercise in applying data science concepts to real-world problems in healthcare. The experience solidified my capability to manage complex datasets and leverage object-oriented programming for efficient software development. The tangible contribution of this application lies in its potential to advance digital health initiatives, assisting clinicians in patient monitoring and decision-making.

What Went Well

The modular architecture adopted was notably effective, fostering a clean separation of concerns. It facilitated easier testing, debugging, and provided a foundation for scaling the application. Moreover, the Streamlit-based user interface resonated well with users, delivering an enhanced, interactive experience that streamlined data exploration, paving the way for its adoption in clinical settings.

Areas for Improvement

In retrospect, the application's resilience could be fortified with more robust error handling to navigate the complexities of real-world usage more gracefully. Better error handling would also contribute to user trust and retention. Additionally, while the code is functionally compartmentalized, greater attention to in-depth documentation would undoubtedly aid future development efforts, ensuring an easier handover and onboarding for new contributors to the project.

Professional Development

Concluding this project, I have grown in my appreciation for best practices in software development and the design of user-centric interfaces in healthcare technology. I've identified valuable lessons in handling class imbalances in datasets, nuances of machine learning model tuning, and the importance of translating complex data insights into actionable intelligence. This growth will certainly serve me and the wider community in advancing healthcare analytics.

References

1. McKinney, W. (2017). Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython (2nd ed.). O'Reilly Media.
2. Matplotlib — Visualization with Python. https://matplotlib.org/.
3. seaborn: statistical data visualization — seaborn 0.13.2 documentation. https://seaborn.pydata.org/.
4. Géron, A. (2019). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems (2nd ed.). O'Reilly Media.
5. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Dubourg, V. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research, 12*(Oct), 2825-2830.

Appendix

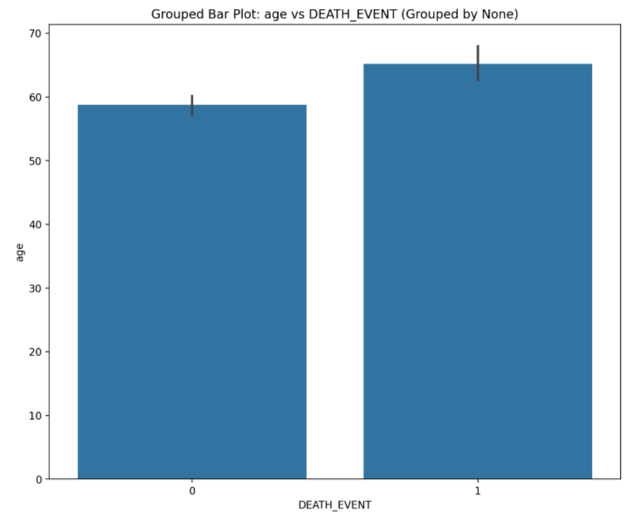


Fig. 1

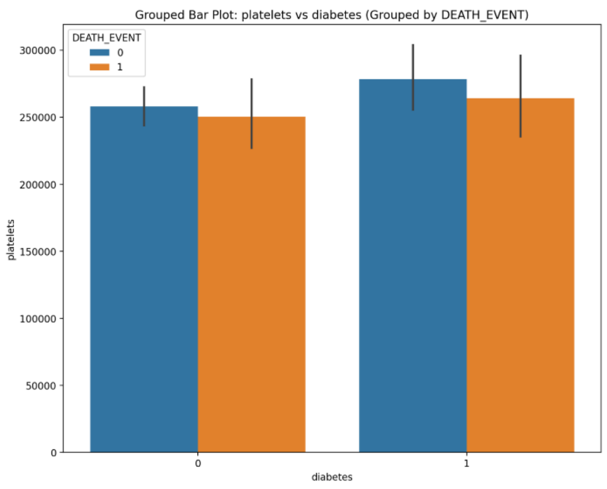


Fig. 2

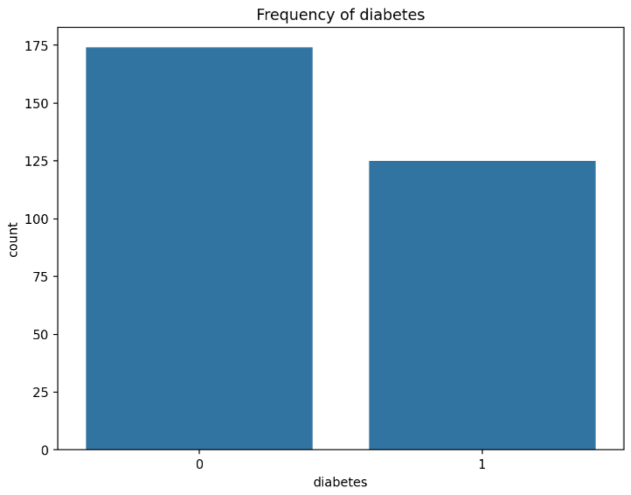


Fig. 3

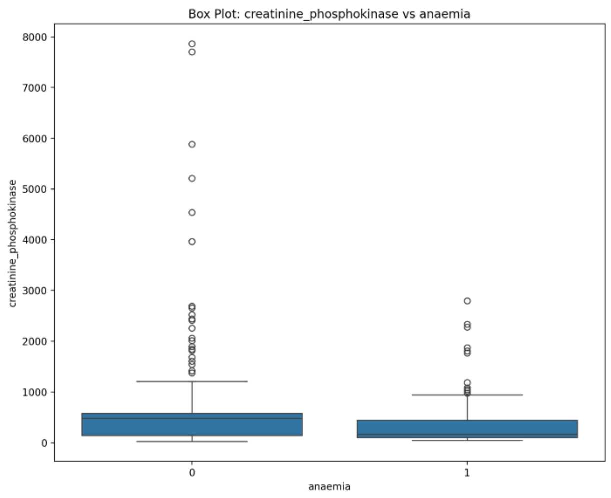


Fig. 4

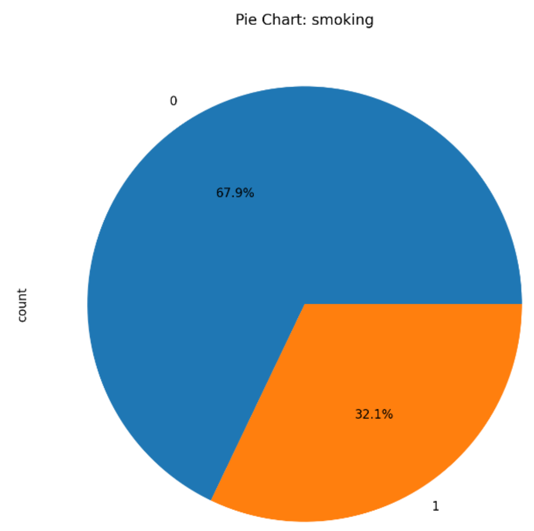


Fig. 5

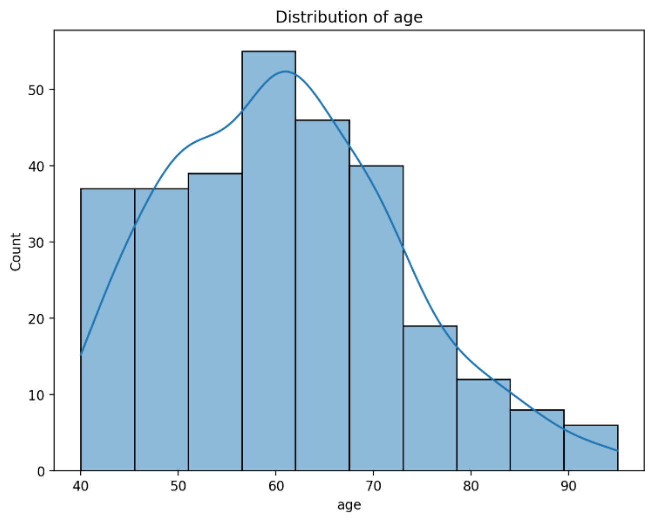
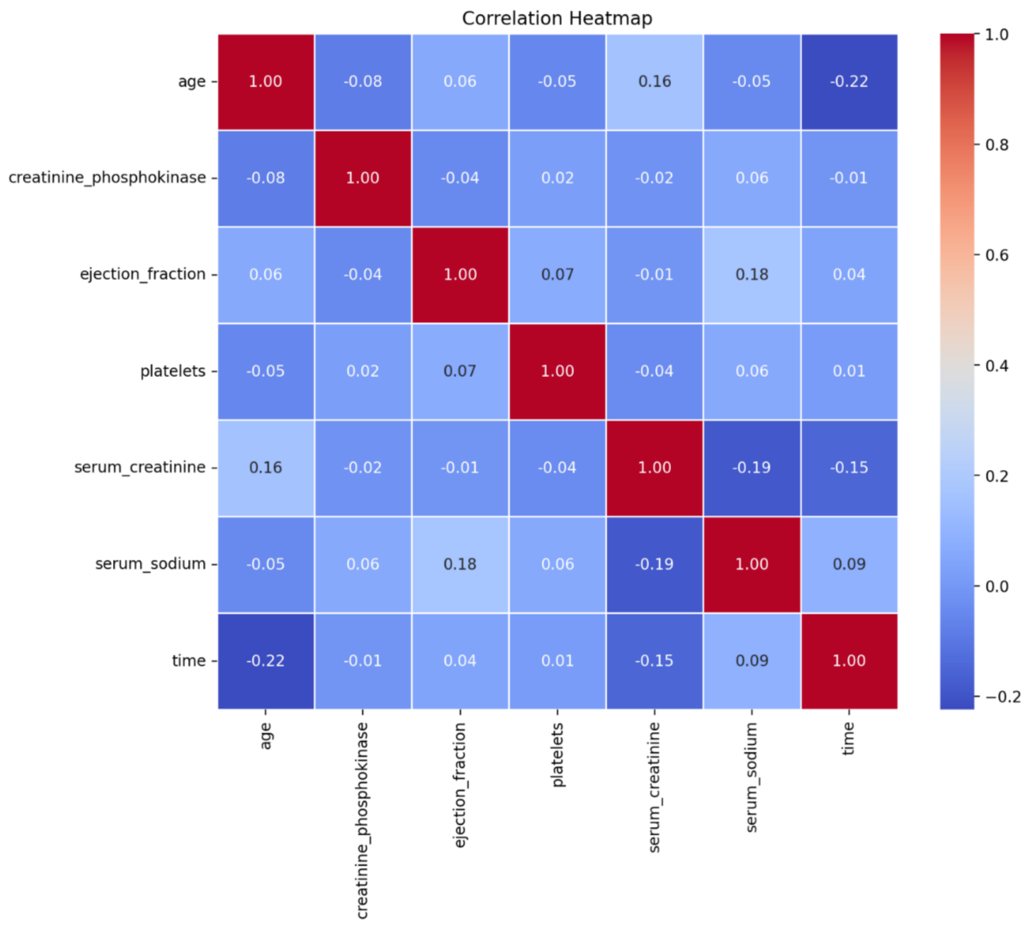


Fig. 6



	age	creatinine_phosphokinase	ejection_fraction	platelets	serum_creatinine	serum_sodium	time
age	1.00	-0.08	0.06	-0.05	0.16	-0.05	-0.22
creatinine_phosphokinase	-0.08	1.00	-0.04	0.02	-0.02	0.06	-0.01
ejection_fraction	0.06	-0.04	1.00	0.07	-0.01	0.18	0.04
platelets	-0.05	0.02	0.07	1.00	-0.04	0.06	0.01
serum_creatinine	0.16	-0.02	-0.01	-0.04	1.00	-0.19	-0.15
serum_sodium	-0.05	0.06	0.18	0.06	-0.19	1.00	0.09
time	-0.22	-0.01	0.04	0.01	-0.15	0.09	1.00

Fig. 7

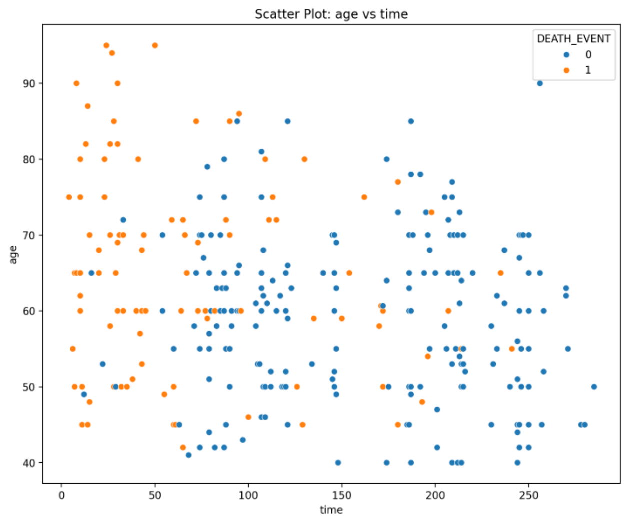

Scatter Plot: age vs time
DEATH_EVENT
0
1
age
time

Fig. 8

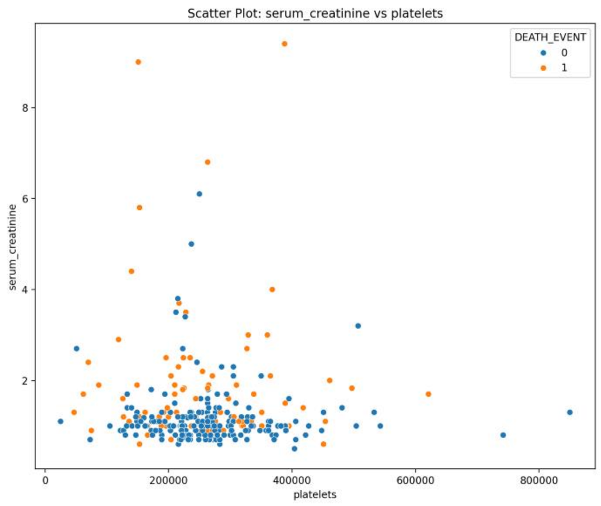

Scatter Plot: serum_creatinine vs platelets
DEATH_EVENT
0
1
serum_creatinine
platelets

Fig. 9