ECHOCARDIOGRAM ANALYSIS USING PYTHON

A PROJECT REPORT

Submitted by

Bhavya Singla (DL.AI.U4AID25063)

Daksh Kathuria (DL.AI.U4AID25066)

Deepanshi Jindal (DL.AI.U4AID25067)

Suyash Dubey (DL.AI.U4AID25097)

in partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY (B. Tech) IN ARTIFICIAL INTELLIGENCE AND DATA SCIENCE (AIDS)

Under the guidance of

Dr. Divyanshu Sinha

(Project Guide - CT)

Dr. Vivek Patel

(Project Guide – IBD)

Dr. Lakshmi Mohandas
Principal of School of AI

AMRITA VISHWA VIDYAPEETHAM

AMRITA SCHOOL OF ARTIFICIAL INTELLIGENCE

FARIDABAD – 121002

December 2025

AMRITA VISHWA VIDYAPEETHAM
DEEMED TO BE UNIVERSITY UNDER SECTION 3 OF UGC ACT, 1956

SCHOOL OF
ARTIFICIAL INTELLIGENCE
FARIDABAD

BONAFIDE CERTIFICATE

This is to certify that this project report entitled “**Echocardiogram Analysis using Python**” is the Bonafide work of Bhavya Singla (DL.AI.U4AID25063), Daksh Kathuria (DL.AI.U4AID25066), Deepanshi Jindal (DL.AI.U4AID25067) and Suyash Dubey (DL.AI.U4AID25097) who carried out the project work under my supervision.

Dr. Divyanshu Sinha
(Assistant Professor)
Dr. Vivek Patel
(Assistant Professor)
School of AI
Faridabad

AMRITA VISHWA VIDYAPEETHAM
DEEMED TO BE UNIVERSITY UNDER SECTION 3 OF UGC ACT, 1956

SCHOOL OF
ARTIFICIAL INTELLIGENCE
FARIDABAD

DECLARATION BY THE CANDIDATE

We declare that the report entitled "**Echocardiogram Analysis using Python**" submitted by us for the degree of Bachelor of Technology is the record of the project work carried out by us under the guidance of "**Dr. Divyanshu Sinha & Dr. Vivek Patel**" and this work has not formed the basis for the award of any degree, diploma, associateship, fellowship, titled in this or any other University or other similar institution of higher learning.

Bhavya Singla (DL.AI.U4AID25063)

Daksh Kathuria (DL.AI.U4AID25066)

Deepanshi Jindal (DL.AI.U4AID25067)

Suyash Dubey (DL.AI.U4AID25097)

ACKNOWLEDGEMENT

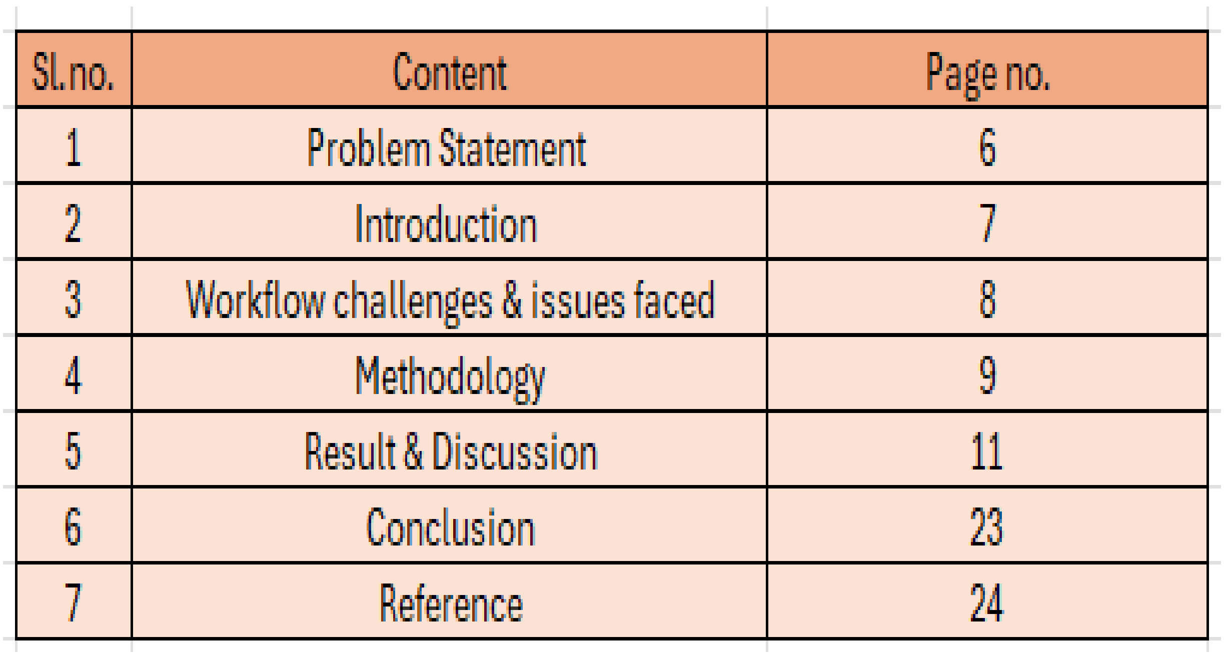
This project work would not have been possible without the contribution of many people. It gives us immense pleasure to express us profound gratitude to our honorable Chancellor **Sri Mata Amritanandamayi Devi**, for her blessings and for being a source of inspiration. We are indebted to extend our gratitude to our Principal, **Dr. Lakshmi Mohandas** Amrita School of Computing and Engineering, for facilitating us all the facilities and extended support to gain valuable education and learning experience.

We wish to express our sincere gratitude to our supervisors "**Dr. Divyanshu Sinha & Dr. Vivek Patel**" for their personal involvement and constant encouragement during the entire course of this work.

We are grateful to Project Supervisor, Review Panel Members and the entire faculty of the Department of Computer Science & Engineering, for their constructive criticisms and valuable suggestions which have been a rich source to improve the quality of this work.

Bhavya Singla (DL.AI.U4AID25063)
Daksh Kathuria (DL.AI.U4AID25066)
Deepanshi Jindal (DL.AI.U4AID25067)
Suyash Dubey (DL.AI.U4AID25097)

TABLE OF CONTENT



PROBLEM STATEMENT

Medical and clinical datasets, such as echocardiogram data, often suffer from data quality issues including missing values, inconsistent entries, categorical variables and logical ambiguities arising from limitations in data acquisition and patient follow-up. Utilizing such raw datasets directly for statistical analysis or machine learning tasks can lead to biased results, reduced predictive accuracy and unreliable clinical interpretations. Additionally, the presence of contradictory records, such as patients marked alive despite short survival durations, further compromises dataset integrity.

The problem addressed in this project is to design and implement a robust and systematic data preprocessing pipeline for a clinical echocardiogram dataset. The pipeline aims to accurately identify and handle missing values using appropriate imputation techniques, convert categorical and binary variables into numerical representations, normalize continuous features for consistent scaling and detect as well as remove or flag ambiguous patient records based on survival and alive indicators.

Furthermore, the preprocessing framework is intended to produce a clean, consistent and structured dataset that is suitable for downstream analytical tasks and machine learning model development. The final processed dataset should be logically consistent, free from missing and ambiguous values, computationally efficient and ready for direct use in predictive modelling without requiring additional preprocessing steps.

INTRODUCTION

The Echocardiogram dataset provides essential clinical information about patients who have experienced a heart attack, with the primary objective of predicting whether a patient will survive for at least one year after the event. The dataset includes a range of diagnostic and physiological features such as age at the time of the heart attack, fractional shortening, EPSs, left ventricular dimensions, wall-motion scores and the presence of pericardial effusion. These variables collectively capture the structural and functional condition of the heart and serve as critical indicators of patient prognosis.

In this project, the dataset is analysed with a focus on exploratory visualization. This includes handling missing and ambiguous records, transforming categorical variables into appropriate numerical formats and normalizing continuous measurements for improved model performance. Following preprocessing, exploratory data analysis is conducted to visualize feature distributions, identify trends, assess missing-data patterns and explore relationships between clinical attributes and survival outcomes. These visual insights help in understanding the underlying data structure and provide early indications of potential predictors of one-year survival.

Overall, this report presents a comprehensive workflow - from data cleaning to insightful visualization - designed to make the Echocardiogram dataset machine-learning ready, to support further predictive modelling related to post-heart-attack survival analysis and providing a strong foundation for academic analysis at the undergraduate level.

WORKFLOW CHALLENGES & ISSUES FACED

During the execution of this project, several practical challenges were encountered while preprocessing the clinical echocardiogram dataset. These challenges are commonly observed in real-world medical data and required careful analysis and systematic handling to ensure data quality, consistency and reliability for further analysis.

1. Presence of Missing Values in Clinical Features

The echocardiogram dataset contained missing values in several important clinical attributes such as age, fractional shortening, EPSs, left ventricular dimensions and wall motion scores. Since these variables play a crucial role in medical analysis, ignoring missing values could lead to biased results. To address this issue, appropriate imputation strategies were applied based on feature types, ensuring that the overall data distribution and clinical relevance were preserved.

2. Absence of Uniform Data Representation

Some variables in the dataset were recorded in categorical or binary formats, such as the presence or absence of pericardial effusion. These non-numerical representations posed challenges for computational analysis and machine learning models. This issue was resolved by converting categorical and binary variables into standardized numerical formats, enabling seamless mathematical processing and analysis.

3. Handling Ambiguous and Inconsistent Patient Records

The dataset included ambiguous records where patients were marked as alive despite having a survival duration of less than one year. Such logical inconsistencies can significantly affect the validity of analytical outcomes. These ambiguous records were systematically identified using survival time and alive-status features and were carefully removed to maintain the logical integrity of the dataset.

4. Feature Scaling and Normalization

The numerical features in the dataset existed on different scales, which could negatively impact statistical analysis and model performance. To address this issue, appropriate scaling and normalization techniques were applied to selected continuous features, ensuring uniformity across variables and improving computational stability during analysis.

5. Maintaining Data Consistency Throughout Preprocessing

Ensuring consistency in column names, data types and transformed features across different preprocessing stages was a critical challenge. Validation checks were implemented after each preprocessing step to prevent unintended data corruption and to confirm that the dataset remained structured and reliable for subsequent analytical tasks.

6. Preparing the Dataset for Analytical and Machine Learning Use

The final challenge involved structuring the processed dataset in a form that could be directly used for statistical analysis and machine learning applications without requiring additional preprocessing. This required careful coordination of all preprocessing steps to ensure the final dataset was clean, complete, logically consistent and computationally efficient.

METHODOLOGY

The methodology followed in this project was designed to systematically preprocess a clinical echocardiogram dataset and transform it into a clean, consistent and analysis-ready format suitable for statistical analysis and machine learning applications. Due to the presence of missing values, categorical variables and logical inconsistencies in the dataset, a structured and sequential preprocessing strategy was adopted to ensure data quality and reliability.

1. Data Loading and Initial Preparation

The echocardiogram dataset was first loaded into the Python environment using appropriate data-handling libraries. Initial preprocessing steps included examining the dataset dimensions, column names and data types to gain an overview of the dataset structure. This step ensured that all variables were correctly interpreted before further processing.

2. Initial Data Inspection

An initial inspection of the dataset was performed to understand the distribution and completeness of the data. Summary statistics were computed, and missing value counts were analysed across all features. This step helped identify critical clinical variables with missing or inconsistent entries and guided the selection of suitable preprocessing techniques.

3. Handling Missing Values

Missing values were handled using appropriate imputation techniques based on feature types. Continuous clinical variables such as age, fractional shortening, EPSs and left ventricular dimensions were imputed using mean values, while binary variables were imputed using mode values. This approach ensured minimal distortion of the original data distribution while maintaining dataset completeness.

4. Conversion of Categorical and Binary Variables

Certain features in the dataset such as the presence of pericardial effusion, were represented in categorical or binary form. To make these variables suitable for numerical analysis and machine learning models, they were converted into numerical representations using binary encoding, where absence and presence were mapped to 0 and 1, respectively.

5. Identification and Removal of Ambiguous Records

Logical consistency checks were performed to identify ambiguous patient records. In particular, records indicating patients as alive despite having survival durations of less than one year were identified as inconsistent. These ambiguous records were systematically removed using defined conditions based on survival time and alive status to preserve the logical integrity of the dataset.

6. Feature Scaling and Normalization

To ensure uniformity across numerical variables, selected continuous features were scaled and normalized. This step reduced the impact of varying numerical ranges among features and improved computational stability during statistical analysis and machine learning model development.

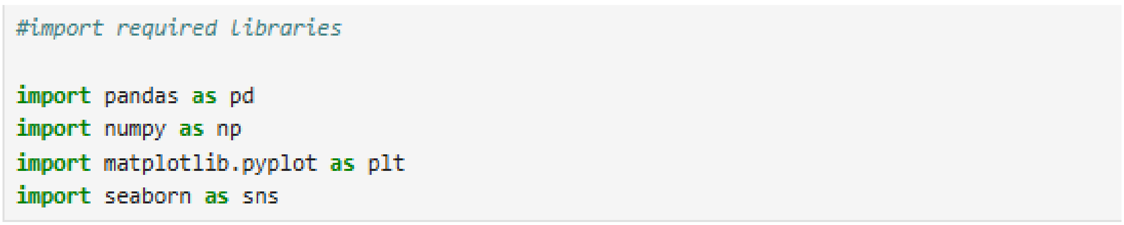
7. Data Visualization for Validation

Exploratory data visualizations were created to analyse the distribution of key clinical variables and to validate the effectiveness of preprocessing steps such as missing value imputation and normalization. These visualizations provided insights into data patterns and supported informed preprocessing decisions.

8. Final Dataset Generation

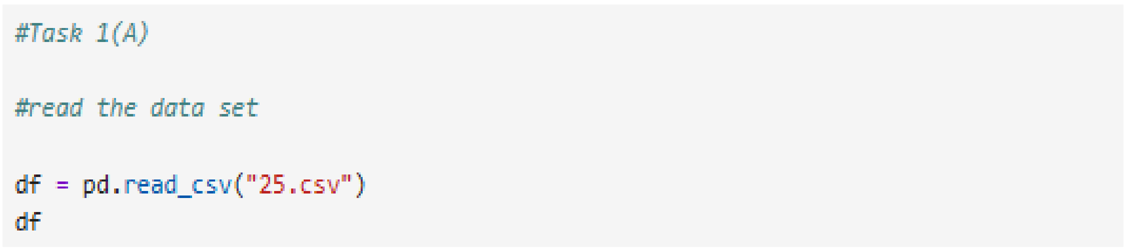
After completing all preprocessing steps, the cleaned and transformed dataset was finalized. The resulting dataset was free from missing values, categorical inconsistencies and logical ambiguities, and was structured in a format suitable for direct use in statistical analysis and machine learning workflows without requiring additional preprocessing.

RESULT AND DISCUSSION



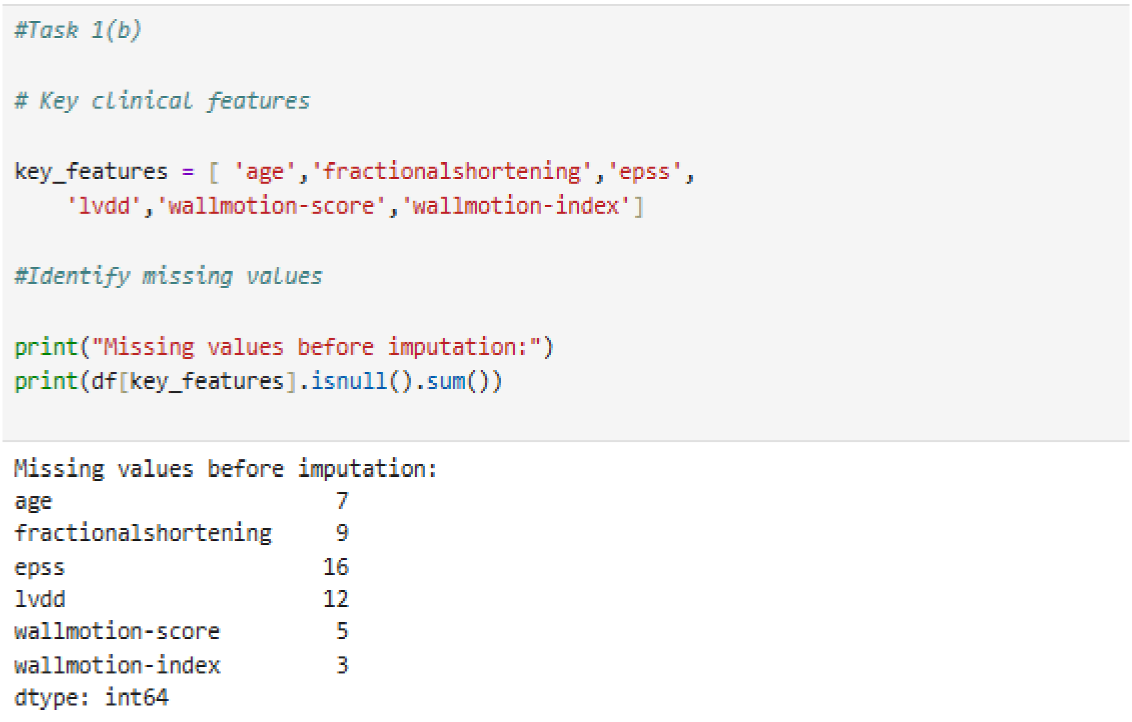
```
#import required libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

This cell imports the necessary Python libraries required for data analysis and visualization.



```
#Task 1(A)

#read the data set

df = pd.read_csv("25.csv")
df
```

The dataset is loaded into a pandas DataFrame named df for further preprocessing.



```
#Task 1(b)

# Key clinical features

key_features = [ 'age','fractionalshortening','epss',
    'lvdd','wallmotion-score','wallmotion-index']

#Identify missing values

print("Missing values before imputation:")
print(df[key_features].isnull().sum())
```

```
Missing values before imputation:
age                      7
fractionalshortening     9
epss                    16
lvdd                    12
wallmotion-score         5
wallmotion-index         3
dtype: int64
```

This step identifies missing values in key clinical features before imputation.

```
#Task 1(b)

#Handle missing values using median imputation

for feature in key_features:
    df[feature] = df[feature].fillna(df[feature].median())

# Verify missing values after imputation

print("\nMissing values after imputation:")
print(df[key_features].isnull().sum())
```

```
Missing values after imputation:
age                     0
fractionalshortening    0
epss                    0
lvdd                    0
wallmotion-score        0
wallmotion-index        0
dtype: int64
```

Median imputation replaces missing values in numerical features with the median of that feature to reduce the effect of outliers and preserve the data distribution.

```
#Task 1(c)

# Binary feature

binary_features = ['pericardialeffusion']

# Mode imputation

for feature in binary_features:
    df[feature] = df[feature].fillna(df[feature].mode()[0])

# Verify missing values after imputation

print("\nMissing values after imputation:")
print(df[binary_features].isnull().sum())
```

```
Missing values after imputation:
pericardialeffusion    0
dtype: int64
```

Binary feature imputation replaces missing values with the most frequent value (mode) to preserve the original class distribution.

```
#Task 1(d)

# Identify ambiguous records

ambiguous_records = (df['survival'] < 12) & (df['alive'] == 1)

# Number of ambiguous records

print("Ambiguous records found:", ambiguous_records.sum())

# Drop ambiguous records

df.drop(df[ambiguous_records].index, inplace=True)

print("Ambiguous records after removal:",
      ((df['survival'] < 12) & (df['alive'] == 1)).sum())
```

```
Ambiguous records found: 34
Ambiguous records after removal: 0
```

Ambiguous records were identified where patients had survival < 12 months but were still marked alive. Such records are logically inconsistent. A Boolean mask was created to detect these cases, 34 ambiguous records were found and they were removed from the dataset.

```
#Task 1(e)

# Drop columns only if they exist

df.drop(columns=['mult', 'group'], errors='ignore', inplace=True)

df
```

Redundant columns were removed to simplify analysis.

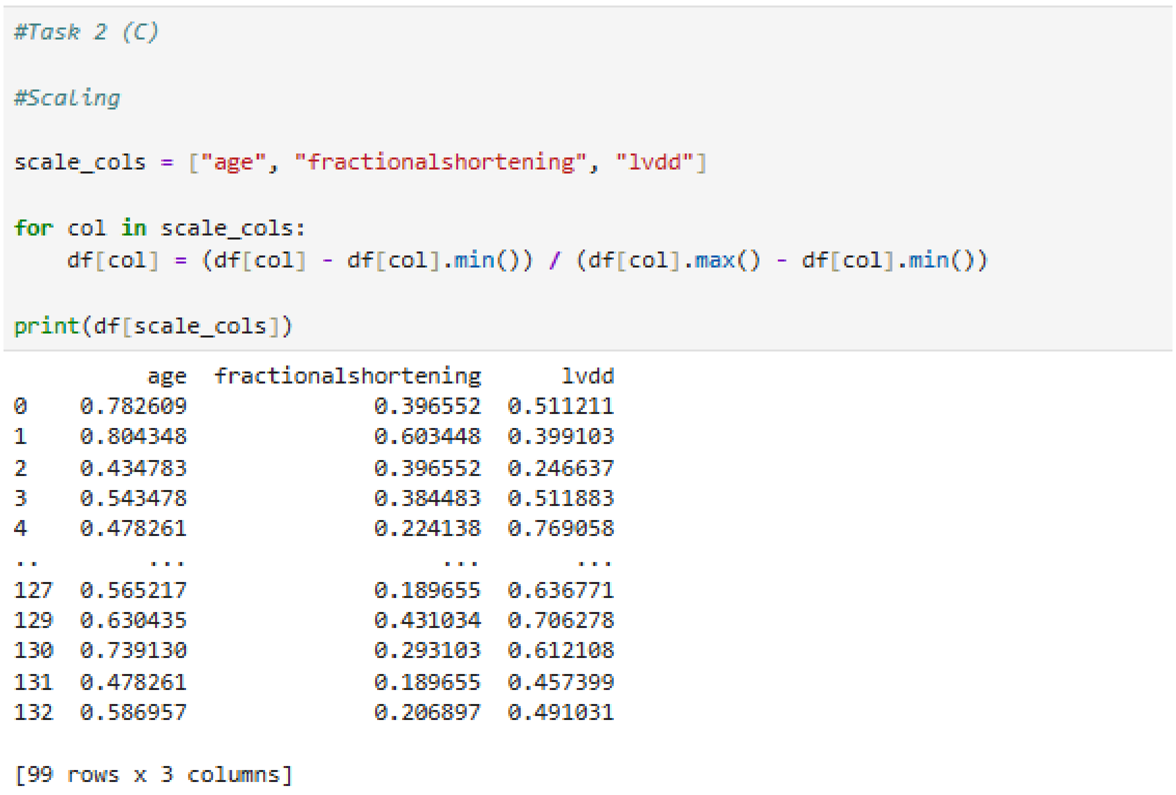
```
#Task 2 (A)

#The pericardial effusion variable was already encoded in binary form (0/1), therefore no additional transformation was required.
```

The pericardial effusion feature was already binary and required no preprocessing.

```
#Task 2 (B)

#Creating a target variable

df["alive_at_1"] = (df["survival"] >= 12).astype(int)
df
```

A

target variable alive_at_1 was created based on survival duration. Patients with survival time >= 12 months were labeled as 1 and those with survival < 12 months were labeled as 0. This binary variable represents one year survival status.

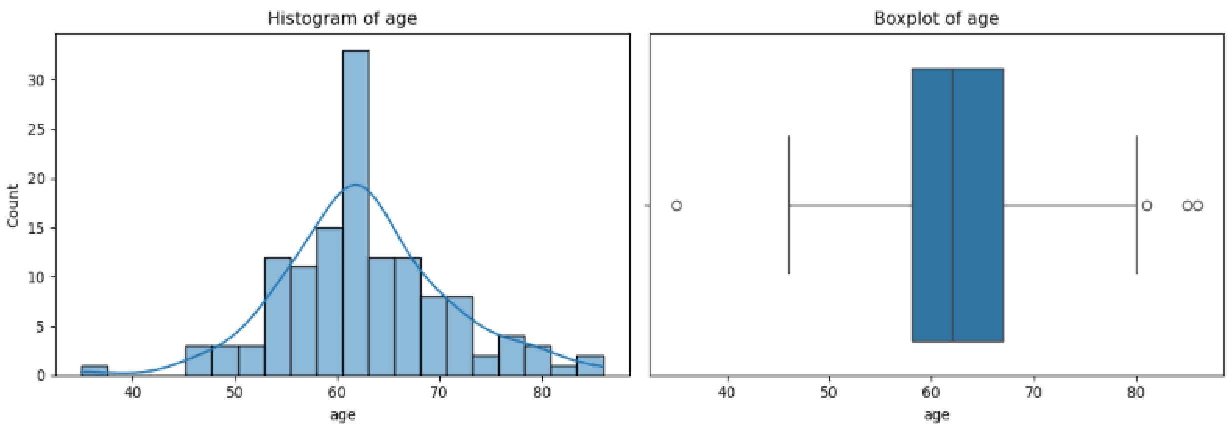


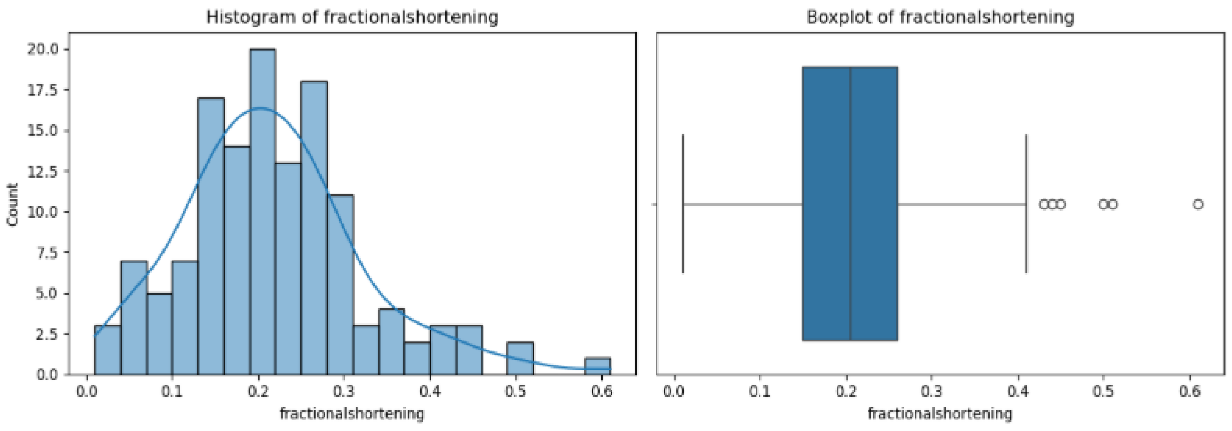
```python
#Task 2 (C)

#Scaling

scale_cols = ["age", "fractionalshortening", "lvdd"]

for col in scale_cols:
    df[col] = (df[col] - df[col].min()) / (df[col].max() - df[col].min())

print(df[scale_cols])
```

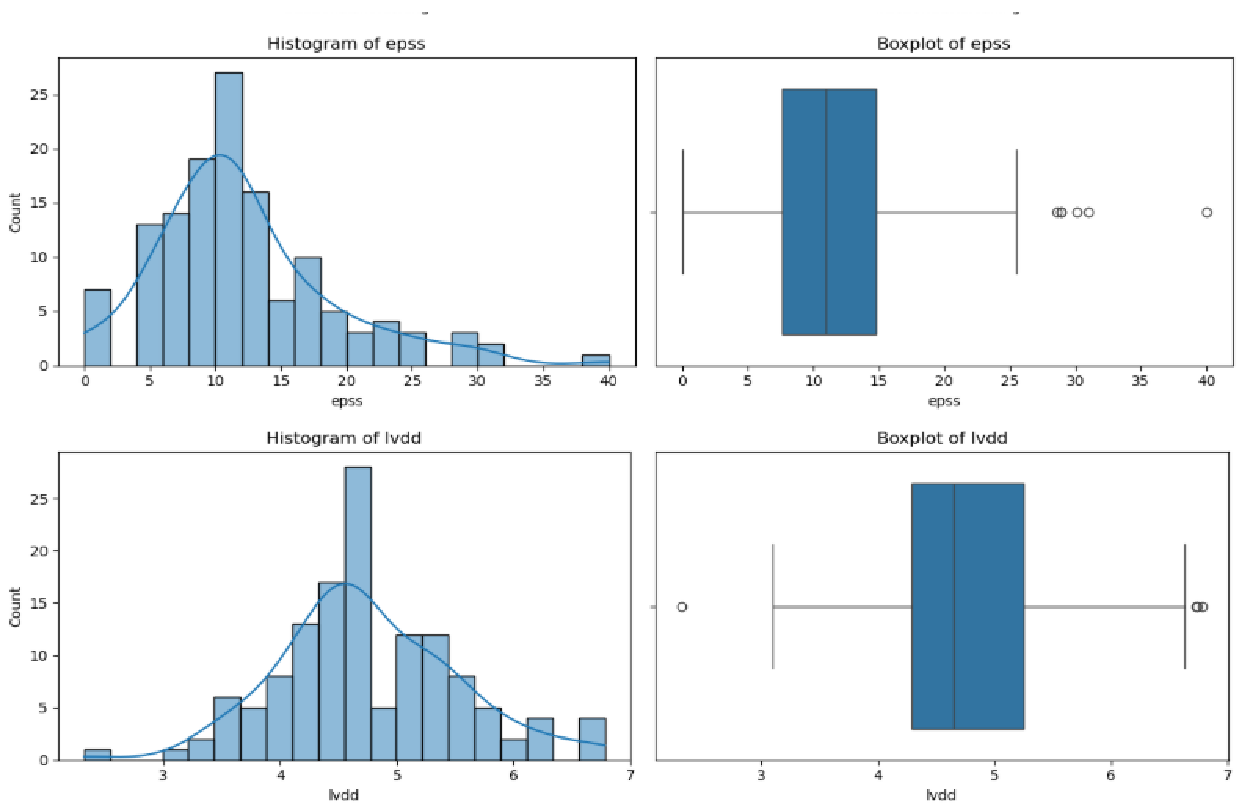
```
          age  fractionalshortening      lvdd
0    0.782609              0.396552  0.511211
1    0.804348              0.603448  0.399103
2    0.434783              0.396552  0.246637
3    0.543478              0.384483  0.511883
4    0.478261              0.224138  0.769058
..        ...                   ...       ...
127  0.565217              0.189655  0.636771
129  0.630435              0.431034  0.706278
130  0.739130              0.293103  0.612108
131  0.478261              0.189655  0.457399
132  0.586957              0.206897  0.491031

[99 rows x 3 columns]
```

Selected numerical features were scaled using min-max normalization. This method rescales values between 0 and 1 by subtracting the minimum and dividing by the range of each feature. Scaling was applied to ensure uniform contribution of features during analysis.

```python
#Task 3 (A)

#Plotting histograms and boxplots for continuous variables

cont_cols = ["age", "fractionalshortening", "epss", "lvdd"]

for col in cont_cols:
    plt.figure(figsize=(12,4))

    # Histogram
    plt.subplot(1,2,1)
    sns.histplot(df[col], bins=20, kde=True)
    plt.title(f"Histogram of {col}")

    # Boxplot
    plt.subplot(1,2,2)
    sns.boxplot(x=df[col])
    plt.title(f"Boxplot of {col}")

    plt.tight_layout()
    plt.show()
```

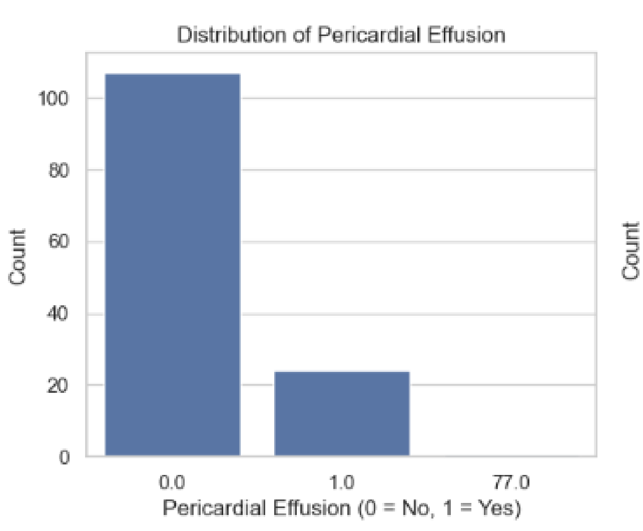


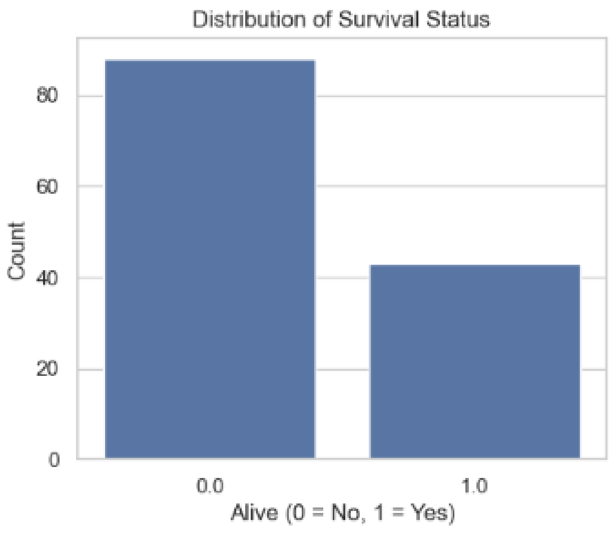




Histograms and boxplots were plotted for continuous variables such as age, fractional shortening, EPSS and LVDD. A loop was used to generate both plots for each variable. Histograms illustrate data distribution while boxplots help identify outliers and variability.

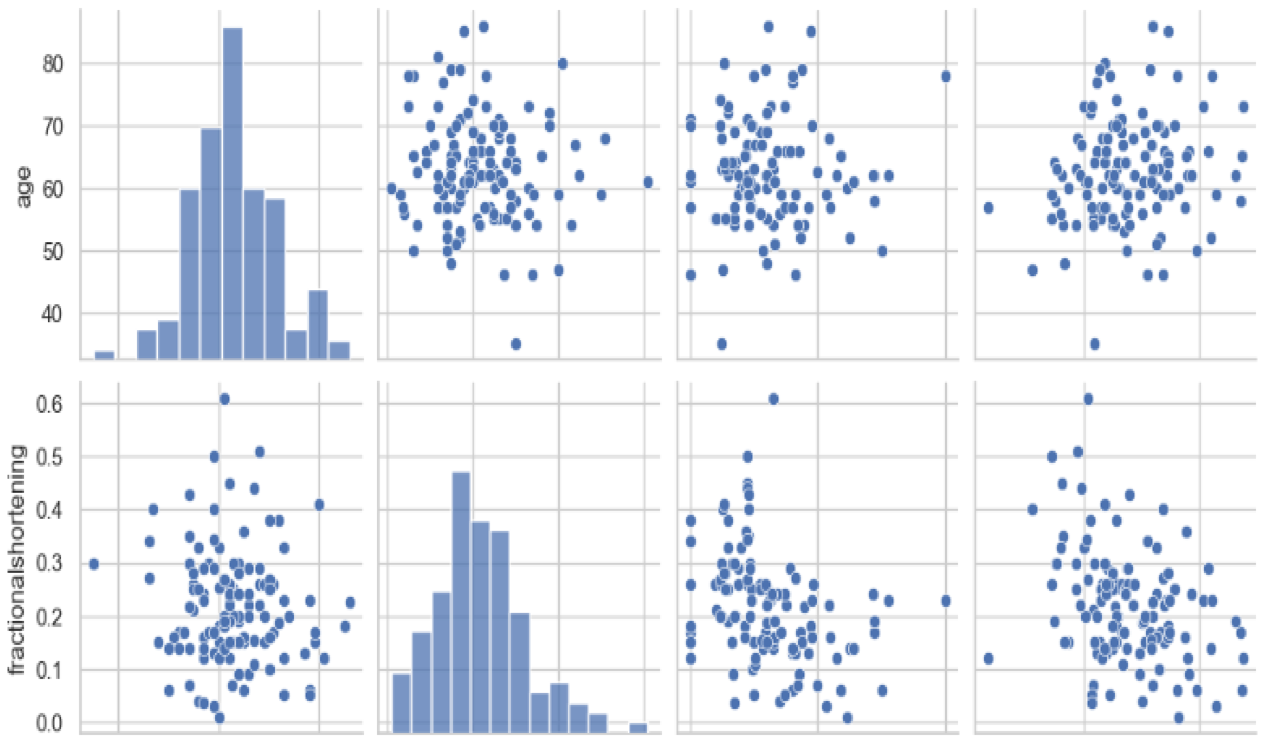
```
#Task 3 (B)

sns.set(style="whitegrid")

# Bar chart for Pericardial Effusion

plt.figure(figsize=(5,4))
sns.countplot(x="pericardialeffusion", data=df)
plt.title("Distribution of Pericardial Effusion")
plt.xlabel("Pericardial Effusion (0 = No, 1 = Yes)")
plt.ylabel("Count")
plt.show()

# Bar chart for Survival Status

plt.figure(figsize=(5,4))
sns.countplot(x="alive", data=df)
plt.title("Distribution of Survival Status")
plt.xlabel("Alive (0 = No, 1 = Yes)")
plt.ylabel("Count")
plt.show()
```

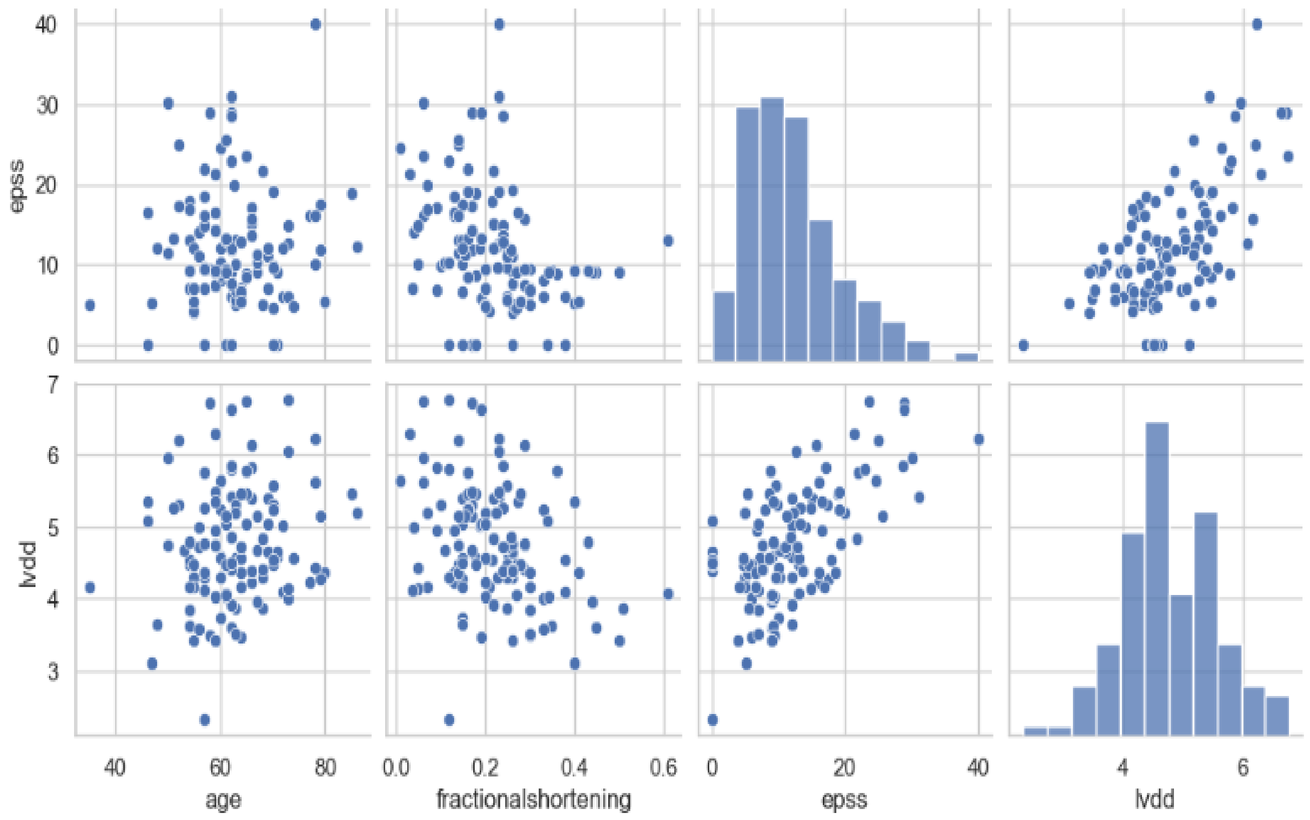




A categorical feature distribution plot shows the frequency of each category to understand class balance in the dataset.

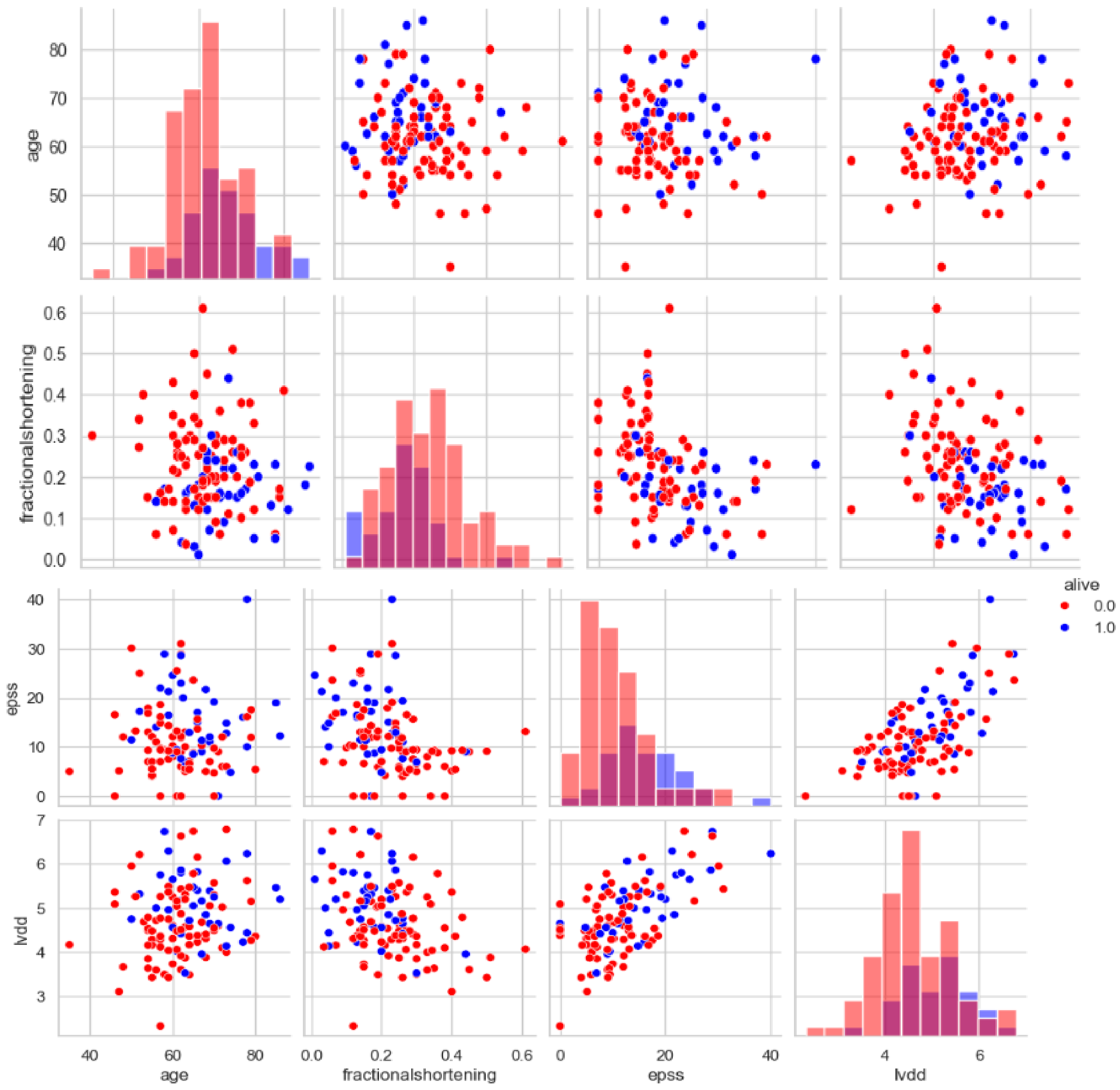
```
#Task 3 (C)

# Numerical variables

num_cols = ["age", "fractionalshortening", "epss", "lvdd"]

# Pair plot

sns.pairplot(df[num_cols], diag_kind="hist")
plt.show()
```





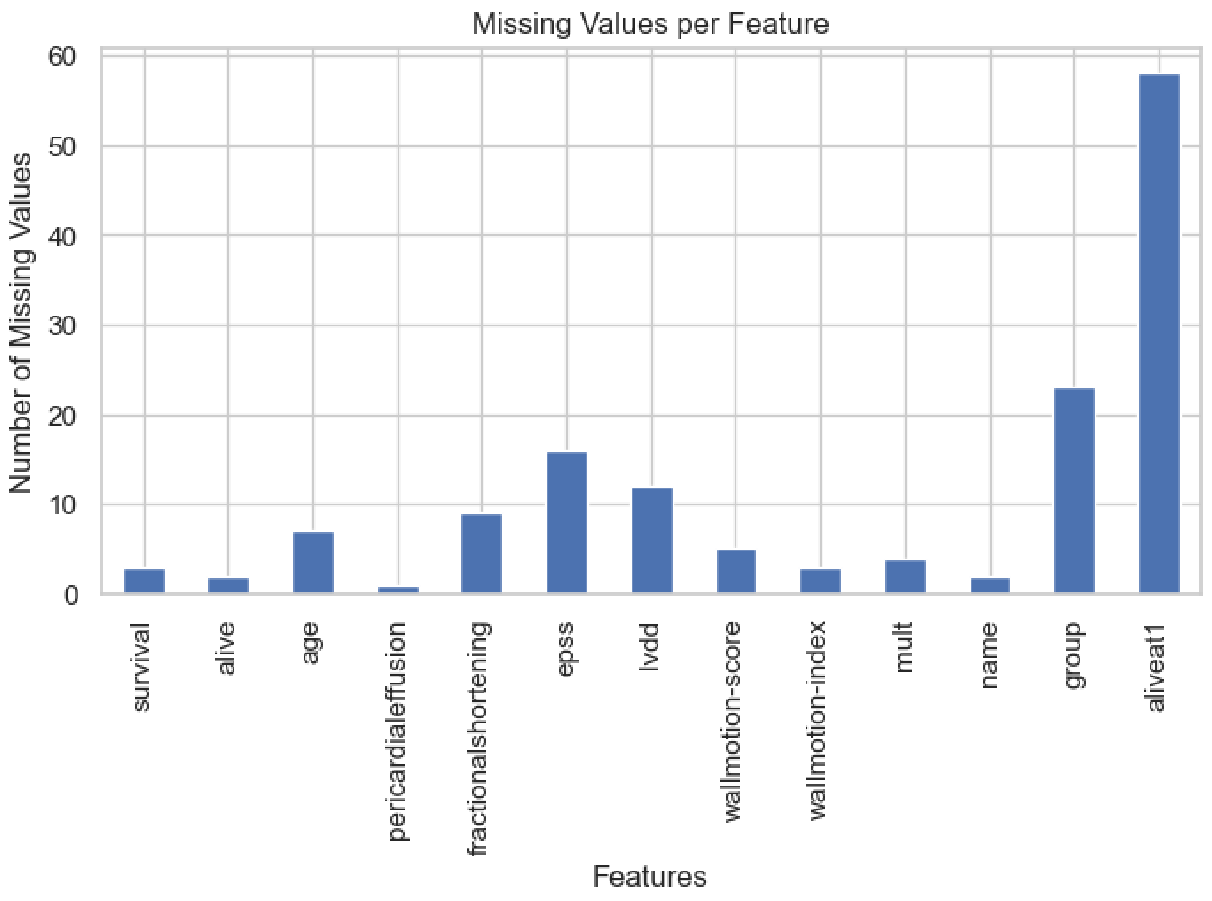
Pair plots help identify relationships between numerical features.

```
#Task 3 (C)

# Numerical features

num_cols = ["age", "fractionalshortening", "epss", "lvdd"]

# Pair plot with survival distinction

sns.pairplot( df, vars=num_cols, hue="alive",
    palette={0: "red", 1: "blue"}, diag_kind="hist")

plt.show()
```



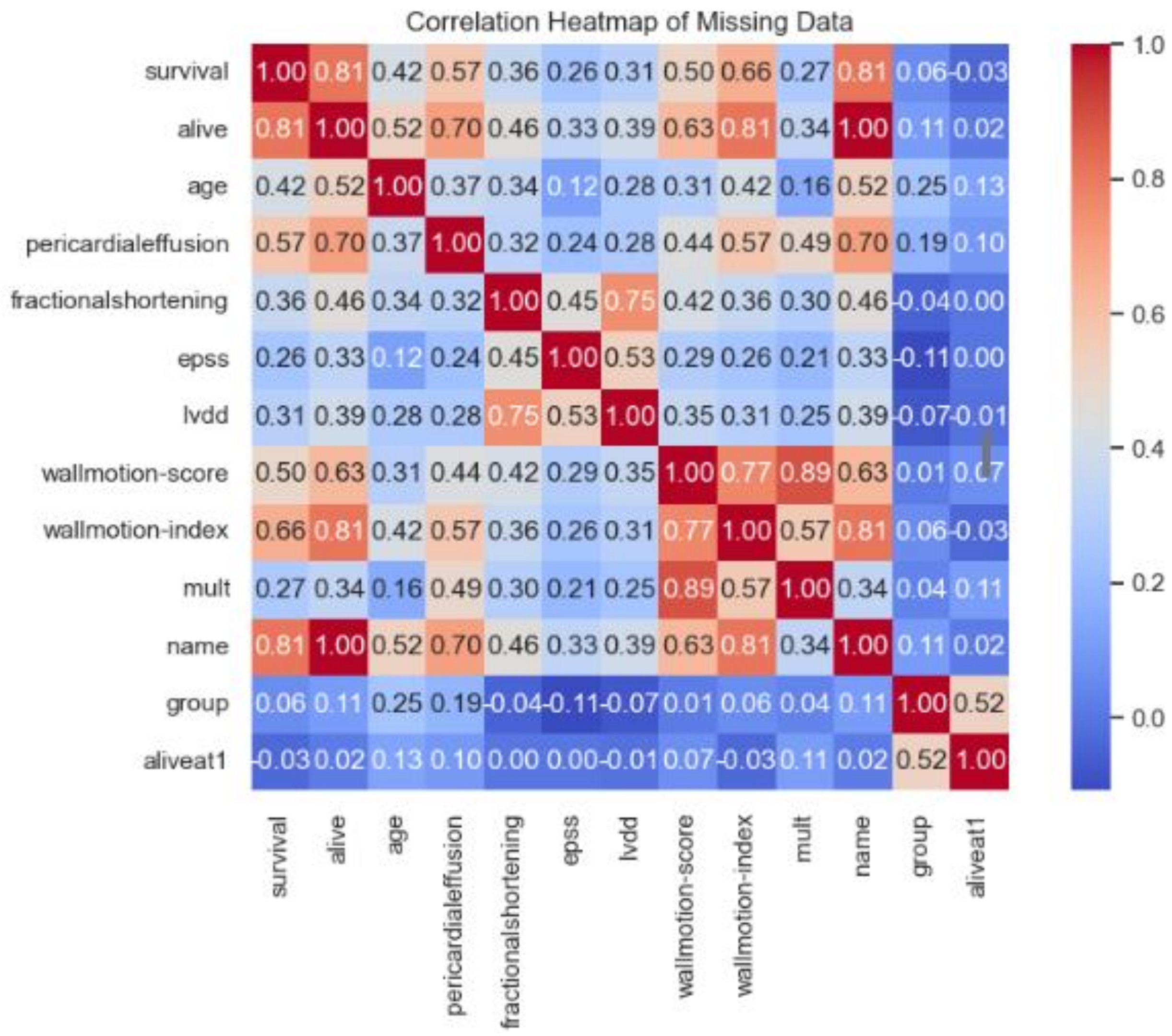
This visualization highlights how features differ based on survival outcome.

```
# Task 3 (D)

# Count missing values

missing_counts = df.isnull().sum()

plt.figure(figsize=(8,4))
missing_counts[missing_counts > 0].plot(kind='bar')
plt.title("Missing Values per Feature")
plt.xlabel("Features")
plt.ylabel("Number of Missing Values")
plt.show()
```



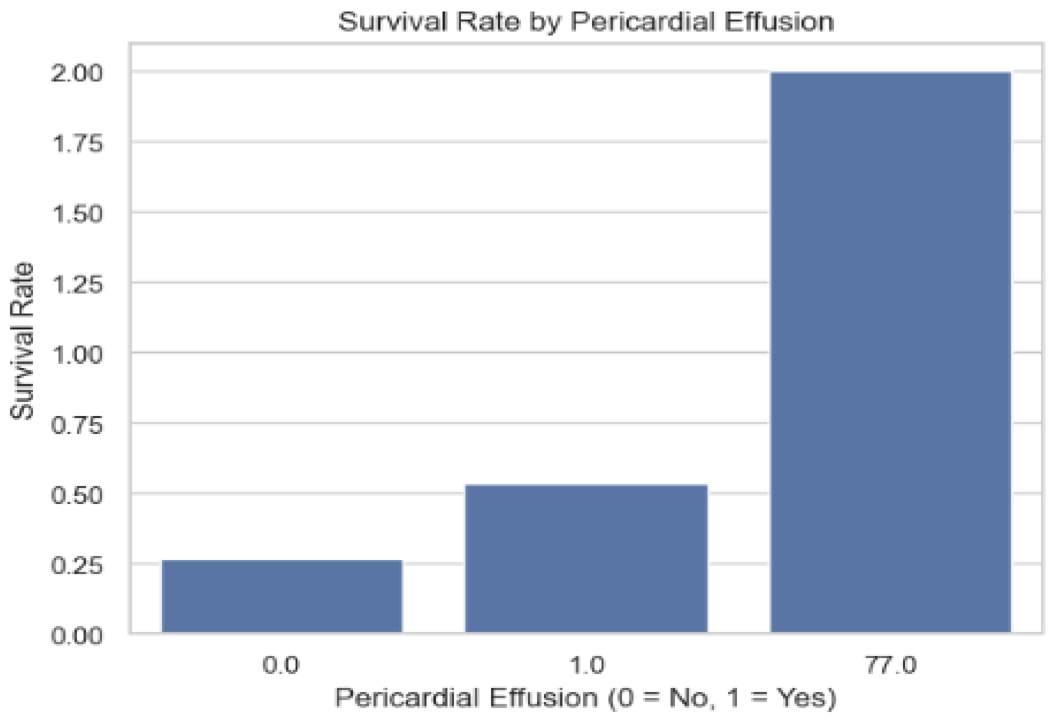
This plot identifies features with missing data.

```
#Task 3 (D)

# Correlation of missingness

missing_corr = df.isnull().corr()

# Plot heatmap

plt.figure(figsize=(10,6))
sns.heatmap( missing_corr, cmap="coolwarm", annot=True,fmt=".2f",square=True)

plt.title("Correlation Heatmap of Missing Data")
plt.show()
```



The heatmap shows relationships between missing values across features.

```
#Task 3 (E)

# Exploring survival rates

sns.barplot( x="pericardialeffusion", y="aliveat1",data=df,ci=None )

plt.xlabel("Pericardial Effusion (0 = No, 1 = Yes)")
plt.ylabel("Survival Rate")
plt.title("Survival Rate by Pericardial Effusion")
plt.show()
```



This plot analyzes the effect of pericardial effusion on survival.

```
#Task 4

#SUMMARISING

print("\n==== SUMMARY REPORT ====")
print("Final Dataset Shape:", df.shape)
print("\nSurvival Rate:")
print(df["alive"].value_counts(normalize=True))

print("\nKey Insights:")
print("""1. Continuous missing values handled with median imputation.
2. Binary missing values handled with mode imputation.
3. Ambiguous survival records removed.
4. New target variable alive-at-1 created based on survival ≥12 months.
5. Features like age, lvdd, fractional-shortening were scaled.
6. Visualizations show distributions, relationships, and survival patterns.""")
```

```
==== SUMMARY REPORT ====
Final Dataset Shape: (133, 13)

Survival Rate:
alive
0.0    0.671756
1.0    0.328244
Name: proportion, dtype: float64

Key Insights:
1. Continuous missing values handled with median imputation.
2. Binary missing values handled with mode imputation.
3. Ambiguous survival records removed.
4. New target variable alive-at-1 created based on survival ≥12 months.
5. Features like age, lvdd, fractional-shortening were scaled.
6. Visualizations show distributions, relationships, and survival patterns.
```

A summary report highlights preprocessing steps and key exploratory findings.

CONCLUSION

In this project, a systematic and structured data preprocessing pipeline was designed and implemented for a clinical echocardiogram dataset. The primary objective was to address common data quality issues such as missing values, categorical inconsistencies and logically ambiguous records, which often hinder reliable analysis and machine learning applications when left untreated.

The proposed methodology successfully identified and handled missing values using appropriate imputation techniques, converted categorical and binary variables into numerical form, and detected as well as removed ambiguous patient records based on survival and alive-status conditions. Feature scaling and normalization further enhanced data consistency, while validation through visualization confirmed the effectiveness of the preprocessing steps.

As a result, the final processed dataset was clean, logically consistent and structured in a format suitable for direct use in statistical analysis and machine learning models without requiring additional preprocessing. The outcomes of this project demonstrate the importance of systematic data preprocessing in improving data reliability, analytical accuracy and model readiness, particularly in real-world clinical datasets.

Overall, this project highlights how careful preprocessing plays a critical role in transforming raw medical data into meaningful and usable information, thereby forming a strong foundation for future predictive modelling and clinical decision-support systems.

REFERENCE

[1] *Histogram of Age at Heart Attack*. In **02_Python_Matplotlib_Exercise_1**, Section 1.0.6: Histogram, Cell: (8).

[2] *Histogram of Fractional Shortening*. In **02_Python_Matplotlib_Exercise_1**, Section 1.0.6: Histogram, Cell: (8).

[3] *Histogram of EPSs*. In **02_Python_Matplotlib_Exercise_1**, Section 1.0.6: Histogram, Cell: (8).

[4] *Histogram of Left Ventricular Dimension (LVDD)*. In **02_Python_Matplotlib_Exercise_1**, Section 1.0.6: Histogram, Cell: (8).

[5] *Boxplot of Age at Heart Attack*. In **011_Seaborn_Categorical_Box_Plot**, Section 1: Seaborn: Box Plot, Cell: (11)

[6] *Boxplot of Fractional Shortening*. In **011_Seaborn_Categorical_Box_Plot**, Section 1: Seaborn: Box Plot, Cell: (12)

[7] *Boxplot of EPSs*. In In **011_Seaborn_Categorical_Box_Plot**, Section 1: Seaborn: Box Plot, Cell: (11)

[8] *Boxplot of Left Ventricular Dimension (LVDD)*. In In **011_Seaborn_Categorical_Box_Plot**, Section 1: Seaborn: Box Plot, Cell: (12)

[9] *Bar Chart Showing Survival Status of Patients*. In **002_Python_Matplotlib_Exercise_1**, Section 1.0.3: Bar Chart, Cell: (5).

[10] *Scatter Plot of Fractional Shortening versus Survival*. In **006_Seaborn_Scatter_Plot_and_Joint_Plot**, Section 1: Scatter Plot and Joint Plot.

[11] *Scatter Plot of Left Ventricular Dimension versus Survival*. In **006_Seaborn_Scatter_Plot_and_Joint_Plot**, Section 1: Scatter Plot and Joint Plot.

[12] Numpy Notebook