CHAPTER 1

INTRODUCTION

Millions of people suffer from irregular heartbeats which can be lethal in some cases. Therefore, accurate and low-cost diagnosis of arrhythmic heartbeats is highly desirable. Many studies have developed arrhythmia classification approaches that use automatic analysis and diagnosis systems based on ECG signals. The most important factors for the analysis and diagnosis of cardiac diseases are features extraction and beats classification.

An electrocardiogram (ECG) is a complete representation of the electrical activity of the heart on the surface of the human body, and it is extensively applied in the clinical diagnosis of heart diseases, it can be reliably used as a measure to monitor the functionality of the cardiovascular system. ECG signals have been widely used for detecting heart diseases due to its simplicity and non-invasive nature. Features of ECG signals can be computed from ECG samples and extracted using software.

The target of classification process is obtaining an intelligent model, that is capable to class any heartbeat signal to specific type of heartbeats. Experiments have been conducted on the ECG database using obtained model.

1.1 Problem statement

This study gives efficient approach to classify ECG signals with high accuracy Each heartbeat is a combination of action impulse waveforms produced by different specialized cardiac heart tissues. Heartbeats classification faces some difficulties because these waveforms differ from person to another, they are described by some features. These features are the inputs of machine learning algorithm.

1.2 Objectives

Multiples classifiers are proposed for ECG classification, these classifiers are used mostly in Big Data and Machine Learning fields by the weighted voting principle. Each classifier influences the final decision according to its performance on the training data. Parameters of each classifier are adjusted on the basis of an individual classifier’s performance on the training data by applying the pseudoinverse technique. The classification performance in this approach was validated on a set of ECG records with different temporal length. Our work is distinguished by:

- Number of tested records (205,146 records of 51 patients).

- Complexity of heartbeat types in training and testing (training records contains Normal and Abnormal beats).
- Using Machine learning algorithms for classification.
- Using big data tool (Spark–Scala).
- Using local host pc (according to the lack of requirements).
- Binary and Multi Classification.

1.3 Existing system

ECG or EKG machine is a very powerful and common screening tool for heart diseases. It is relatively inexpensive, non-invasive and easy to use. It does have some limitations. Understanding those limitations is important to put things in proper perspectives. The ECG monitor displays the electric activities generated by the heart and gives a snapshot of the heart rate and rhythm during the test; however it does not reflect many underlying problems. Cardiac abnormalities may occur only intermittently, and ECG needs to be performed at the "right" moment to capture the episodes. At other times, a patient may have entirely normal ECG.

A stress ECG test requires high-grade stenosis to show positive reading. High-grade stenosis is a good indicator of advanced heart disease.

1.4 Proposed system

Classification is a process of categorizing a given set of data into classes, It can be performed on both structured or unstructured data. The process starts with predicting the class of given data points. The classes are often referred to as target, label or categories. The classification predictive modeling is the task of approximating the mapping function from input variables to discrete output variables. The main goal is to identify which class/category the new data will fall into.

Heart disease detection can be identified as a classification problem, this is a binary classification since there can be only two classes i.e has heart disease or does not have heart disease. The classifier, in this case, needs training data to understand how the given input variables are related to the class. And once the classifier is trained accurately, it can be used to detect whether heart disease is there or not for an particular patient. Since classification is a type of supervised learning, even the targets are also provided with the input data.

1.5 Applications

Machine learning, a great developing technique that revolves around exploring and digging out significant information from massive collection of data which can be further beneficial in examining and drawing out patterns for making business related decisions. Talking about the Medical domain, implementation of Machine learning in this field can yield in discovering and withdrawing valuable patterns and information which can prove beneficial in performing clinical diagnosis.

The research elaborates and presents multiple knowledge abstraction techniques by making use of Machine learning methods which are adopted for heart disease prediction. The output reveals that the established diagnostic system effectively assists in predicting risk factors concerning heart diseases.

CHAPTER 2

LITERATURE SURVEY

1. Periarrest Modified Early Warning Score (MEWS) predicts the outcome of in-hospital cardiac arrest (2015):

In-hospital cardiac arrest (IHCA) is a potentially catastrophic adverse outcome and one that can cause substantial stress to medical staff. Anticipating a patient's physiological deterioration prior to cardiac arrest allows at risk patients to be identified. Clinical deterioration of respiratory, cardiac, and/or cerebral function without appropriate response to abnormal physiological variables can lead to cardiac arrest.

Early Warning Score (MEWS) has been widely adopted as a useful clinical tool to identify those patients at risk of deterioration who require attention.MEWS is a summed score of routinely recorded physiological data, which includes measurements of systolic blood pressure, heart rate, respiratory rate, body temperature, and level of consciousness (the latter represented as either alert, reacting to voice, reacting to pain, or unresponsive).2 MEWS is a simple, quick bedside tool that can be applied by nursing staff.

A patient's higher score upon emergency department (ED) admission correlates with the increasing necessity for hospital admission and also with a higher risk of in-hospital death. The prognosis of IHCA is influenced by resuscitation parameters and also by preexisting comorbidity.5 Large database summaries have identified prearrest predictors of IHCA to be age, comorbidity, major trauma, and diagnosis of this admission. The prearrest factors allow initial resuscitation providers to identify which patients are unlikely to survive resuscitation.

There was no significant difference in MEWS at triage between the two outcome groups. Each points increase in periarrest MEWS reduced the chance of survival to discharge; periarrest MEWS was also an independent predictor along with other risk factors, such as sex, ED diagnosis, cardiac arrest rhythm, cardiac arrest cause, and underlying comorbidities.

Over the period of 30 months, 234 nontraumatic adult patients suffered from IHCA during an ED stay and received resuscitation, and 99 patients with periarrest MEWS were included in the final analysis. cadiac Arrest Risk Triage (CART) score uses ward vital signs (including respiratory rate, heart rate, diastolic blood pressure, and age) from admission until cardiac arrest to derive a prediction model. The CART score was intended as a better track-and-trigger tool for the rapid response team (RRT); the score allows identification of patients at risk suffering adverse events. study excluded vital signs within 30 minutes of cardiac arrest because the main purpose of the score

is as a prevention tool. Pre-arrest Morbidity (PAM), first developed in 1989, correlates inversely with the probability of short- and longterm survival after resuscition. PAM uses 15 variables including underlying disease, diagnosis, age, medical intervention, blood pressure, and laboratory data; the score ranges from 0 to 25.

. The limitation of our study is that all cardiac arrests occurred in the ED and in just one medical center; whether the result is applicable in the ward or in the intensive care unit is questionable. In our study, not all patients had periarrest vital signs recorded. Only 99 patients were eligible for final analyses, which is due to the limitations of a retrospective study. Selection bias may also exist because only those with the most critical status had complete vital signs checked before cardiac arrest. An extension of the study period to include more patients or a further prospective study could increase the amount of data available for analysis[1].

2. Temporal elevation of blood pressure is associated with increased risk of sudden cardiac arrest (2016):

From other claims databased medical cohorts because it offers a regular health checkup program to its subscribers. During the nationwide health check-up, various laboratory tests, medic Korean National Health Insurance Service (K-NHIS) database was used in this study. All people in the Republic of Korea are mandatory subscribers of the K-NHIS; therefore, the entire population of the Republic of Korea is represented by data derived from the K-NHIS. The K-NHIS is distinguishedal measurements, and self-questionnaires were provided and stored in a database.

Therefore, medical researchers can analyze (i) blood test results, such as complete blood cell counts, liver and renal function tests, lipid profiles, fasting blood glucose (FBG), and gammaglutamyl transferase levels; (ii) measurements of systolic and diastolic blood pressure (SBP and DBP), body weight, height, and waist circumference; and (iii) lifestyle factors, including smoking status, alcohol consumption, and exercise level. Importantly, a nationwide health check-up program is provided to subscribers once every 2 years which enables the evaluation of temporal changes in various parameters, including blood pressure.

ΔSBP and ΔDBP were defined as temporal changes in SBP and DBP during 2009 and 2011 nationwide health check-ups. Based on a self-questionnaire during the 2009 health check-up, smoking status was defined as follows: (i) current smokers: those who smoked at least 100 cigarettes in their lifetime and continued smoking within 1 month of the 2009 health check-up; (ii) ex-smokers: those who smoked at least 100 cigarettes in their lifetime but had not smoked within 1 month of the 2009 health check-up; and (iii) never smokers: those who smoked < 100 cigarettes in their lifetime. Participants were classified into three groups according to the total amount of alcohol

consumed per week: (i) non-drinkers: 0 g per week, (ii) mild drinkers: less than 210 g but more than 0 g per week, and (iii) heavy drinkers: 210 g or more per week.

Demonstrated that hypertension and prehypertension were significantly associated with an increased risk of SCA10,17. Tis study revealed that temporal elevation in blood pressure is also associated with an increased risk of SCA. In this study, HRs were adjusted for baseline blood pressure, and temporal elevation of blood pressure was associated with SCA risk in both people with SBP < 140 and ≥ 140 suggesting not only baseline blood pressure but also temporal elevation in blood pressure can contribute to SCA risk (Fig. 2 and Supplementary Table S2). The association between ΔSBP and the risk of SCA was not affected by age, sex, presence of diabetes mellitus, and baseline SBP, and other confounders such as BMI, smoking status, alcohol consumption, regular physical activity, and income level, were adjusted for in the multivariate model. Our study indicates that a temporal elevation in blood pressure can be associated with an increased risk of SCA.

The current study was approved by the Institutional Review Board of Korea University Medicine Anam Hospital and the official review committee of the K-NHIS. Considering the retrospective nature of this study, the requirement for written informed consent was waived. The ethical guidelines of the 2013 Declaration of Helsinki and the legal medical regulations of the Republic of Korea were strictly undertaken throughout the study.

Sudden cardiac arrest (SCA) is one of the most emergent medical condition which require immediate therapeutic intervention to bring the victim back to life1–3 . Various eforts intended to improve neurologically intact survival rate in SCA patients were attempted4–9 . Although widespread supply of automatized defbrillators, high quality education of the citizens, and early initiation of bystander cardiopulmonary resuscitation have been demonstrated.

Continuous variables were compared using Student’s t-test. We used the Kolmogorov– Smirnov test to confirm the normal distribution of continuous variables and they were all normally distributed. The chi-squared test or Fisher’s exact test was used to compare categorical variables. The Cox proportional hazards model was used to calculate the hazard ratios (HR) and 95% confidence intervals (CI). In the multivariable Cox proportional hazard model, age, sex, body mass index (BMI), income level, smoking status, alcohol consumption status, regular physical activity, baseline SBP (or DBP for ΔDBP analysis), diabetes mellitus, heart failure, cardiovascular disease, and prescription of antihypertensive drugs were adjusted. All tests were two-tailed, with p values≤0.05 considered to indicate statistical significance. SAS version 9.2 (SAS Institute, Cary, NC, USA) was used for all the statistical analyses.

Preventing elevated blood pressure may contribute to the primary prevention of SCA, according to the current analysis. However, a temporal decrease in blood pressure was not associated with a decreased risk of SCA in our study. The degree of decrease in both SBP and DBP also showed no association with the risk of SCA[2].

3. Early detection of sudden cardiac death using Poincaré plots and recurrence plot-based fence plot-based features from HRV signals V signals (2017):

In this paper they present a method to predict sudden cardiac death (SCD) based on the heart rate variability (HRV) signal and recurrence plots and Poincar´e plot-extracted features. This work is a challenge since it is aimed to devise a method to predict SCD 5 min before its onset. The method consists of four steps: preprocessing, feature extraction, feature reduction, and classification. In the first step, the QRS complexes are detected from the electrocardiogram signal and then the HRV signal is extracted. In the second step, the recurrence plot of the HRV signal and Poincar´e plot-extracted features are obtained.

Four features from the recurrence plot and three features from the Poincar´e plot are extracted. The features are recurrence rate, determinism, entropy and averaged diagonal line length, and SD1, SD2, and SD1/SD2. In the next step, these features are reduced to one feature by the linear discriminant analysis technique. Finally, K-nearest neighbour and support vector machinebased classifiers are used to classify the HRV signals. They use two databases, the MIT/BIH Sudden Cardiac Death Database and PhysioBank Normal Sinus Rhythm Database. They manage to predict SCD occurrence 5 min before the SCD with accuracy of over 92%.

Abnormality in heart rhythm (arrhythmia) can cause sudden cardiac death (SCD).SCD causes thousands of adult deaths in the United States each year.Despite the reduction in deaths from heart diseases in the last 20 years,SCD is responsible for half of all deaths from cardiovascular diseases.When the electrical system of the heart becomes irregular, SCD will occur. One of the signs of SCD is ventricular fibrillation (VF).Arrhythmias like VF occur because for various reasons such as diabetes, alcohol, and smoking.Today invasive and non invasive techniques are the main ways of predicting the risk of SCD.

They used a database consisting of 59 ECG signals from 23 cardiac patients (18 to 89 years old) with a sampling rate of 250 Hz. The number of healthy controls was 36, with 18 healthy people with lead I and 18 with lead II (20 to 50 years old and 128 Hz sampling rate). In this study, the HRV signals derived from lead II of patients and lead I and lead II for healthy people were used. In order to maintain accuracy, the ECGs from the patients were down sampled (from 250 Hz to 125 Hz to match the normal ECGs). The HRV signals of three patients were eliminated from the SCD

database since they did not have ventricular fibrillation (V) episodes, and no signal from the normal database was discarded since it was without abnormalities. To evaluate the performance of our method in separating cardiac patients from healthy people, feature vectors were extracted from the fourth and fifth 1-min segments of the signals before SCD occurrence.

They divided the ECG signal into specified time intervals (first 1 min, up to the fifth 1 min before SCD). These time intervals were used as ECG signals of patients. For healthy subjects random 1-min intervals were selected from a 1-h recording. First, noise reduction for healthy and patient ECG signals was done. Low frequencies in baseline wander (because of DC drift) and high frequencies in power line interference (because of AC power noise) were removed. To remove the baseline drift both healthy and patient ECG signals were filtered with a moving-average filter. With the moving average filter, an array of raw (noisy) data [y 1 , y 2 , . . . , yN] can be converted to a new array of smoothed data.

In this work they used two classifiers, the K-nearest neighbour (KNN) and support vector machine (SVM), to differentiate between healthy subjects and patients. To evaluate the performance of classifiers, a ten-fold cross validation method is used. The 56 ECG signals are divided into ten parts and the numbers of signals in parts are equal unless in two or three groups. One part was used to test the classifier and nine parts were used to train the classifier. This process was repeated ten times for each different test set, and the average performance for accuracy, sensitivity, and specificity was calculated. In order to increase the accuracy, They repeated each of the ten processes twenty times.

In this work, They presented a method to predict SCD 4 min and 5 min before SCD occurrence with an accuracy of 92.14% (SVM) and 94.10% (KNN), respectively. They used features derived from HRV signals. The LDA algorithm was used to identify the optimal feature for SCD prediction. The best feature was then fed into the SVM and KNN classifiers for prediction. The results showed that the prediction rate of this method is better than those of other studies with improvement in the SCD prediction rate. Their method showed better performance in 4 min before SCD occurrence in comparison to other studies. Moreover, They were able to extend the interval to 5 min for the first time and get good results before SCD[3].

4. An Algorithm Based on Deep Learning for Predicting In-Hospital Cardiac Arrest (2018):

Two types of TTS are used in RRSs. For the single-parameter TTS (SPTTS), cardiac arrest is predicted if any single vital sign (eg, heart rate [HR], blood pressure) is out of the normal range.14 The aggregated weighted TTS calculates a weighted score for each vital sign and then

finds patients with cardiac arrest based on the sum of these scores.15 The modified early warning score (MEWS) is one of the most widely used approaches among all aggregated weighted TTSs (Table 1)16; however, traditional TTSs including MEWS have limitations, with low sensitivity or high false-alarm rates.14,15,17 Sensitivity and false-alarm rate interact: Increased sensitivity creates higher false-alarm rates and vice versa.

Current RRSs suffer from low sensitivity or a high falsealarm rate. An RRS was used for only 30% of patients before unplanned intensive care unit admission and was not used for 22.8% of patients, even if they met the criteria.

The study population consisted of all patients admitted to 2 hospitals over 91 months. The characteristics of the 2 hospitals are different (hospital A is a cardiovascular teaching hospital, and hospital B is a community general hospital. They excluded patients who were admitted or discharged outside the study period and patients who underwent cardiac arrest or death within 30 minutes after admission. The institutional review boards of Sejong General Hospital and Mediplex Sejong Hospital approved this study and granted waivers of informed consent based on general impracticability and minimal harm. Patient information was anonymized and deidentified before the analysis.

The primary outcome was cardiac arrest and the secondary outcome was death without attempted resuscitation. For cardiac arrest, They used only the first cardiac arrest if cardiac arrest occurred several times during a patient's length of stay. They reviewed electronic health records to identify the exact time of each outcome.The objective of this study was to predict whether an input vector belonged within the prediction time window. The prediction time window was defined as the interval from 0.5 to 24 hours before the outcomes.

The DEWS used time series data as an input. Given the input, the DEWS evaluated the risk score using all input vectors measured during the 8 hours. when calculating the risk score of an input vector measured at 11 am, DEWS used all input vectors measured from 3 am to 11 am. The DEWS calculated the risk score, which ranged from 0 (nonevent) to 100 (event), every time the input vector was measured.

They used the Adam optimizer with the default parameters and a binary-cross entropy as a loss function.31 To validate our model, we used the hyperparameters of the model with the best performance on 10% of the data from the derivation data during the training process. A total of 56 076 patients were admitted during the study period. They excluded 448 patients who were admitted or discharged outside the study period and 3497 patients who experienced an event or were discharged within 30 minutes after admission. The study population consisted of 52 131 patients, of which 1233 patients underwent cardiac arrest or death without attempted resuscitation. The DEWS

was developed using 2 769 324 input vectors from 46 725 patients of Hospital A, and a test was performed using 213 369 input vectors for a total of 5406 patients.

An algorithm based on Deep learning had high sensitivity and a low false-alarm rate for detection of patients with cardiac arrest in a multicenter study. In addition, the DEWS was developed with only 4 vital signs: systolic blood pressure, HR, respiratory rate, and BT. Consequently, it is easy to apply in various hospital environments and offers potentially greater accuracy by using additional information[4].

5. Contactless cardiac arrest detection using smart devices (2019):

Out-of-hospital cardiac arrest (OHCA) is a leading cause of death worldwide and in North America accounts for nearly 300,000 deaths annually.A relatively under-appreciated diagnostic element of cardiac arrest is the presence of a distinctive type of disordered breathing. Agonal breathing, which arises from a brainstem reflex in the setting of severe hypoxia, appears to be evident in approximately half of cardiac arrest cases reported to 9-1-1. Agonal breathing indicates a relatively short duration from arrest and has been associated with higher survival rates. Sometimes reported as “gasping” breaths, agonal respirations may hold potential as an audible diagnostic biomarker, particularly in unwitnessed cardiac arrests that occur in a private residence, the location of 2/3 of all OHCAs.

The agonal breathing recordings are sourced from 9-1-1 emergency calls from 2009 to 2017, provided by Public Health Seattle & King County, Division of Emergency Medical Services. The dataset included 162 calls (19 h) that had clear recordings of agonal breathing. For each occurrence, They extracted 2.5 s worth of audio from the start of each agonal breath. They extracted a total of 236 clips of agonal breathing instances. Given the relatively small size of our agonal breathing dataset, They augment the number of agonal breathing instances with label preserving transformations, a common technique applied to sparse datasets. They augment the data by playing the recordings over the air over distances of 1, 3, and 6 m, in the presence of interference from indoor and outdoor sounds with different volumes and when a noise cancellation filter is applied. The recordings were captured on different devices, namely an Amazon Alexa, an iPhone 5s and a Samsung Galaxy S4 to get 7316 positive samples.

They obtained a false positive rate of 0.14409%, which corresponds to 170 of 117,985 audio segments. To reduce false positives, The classifier’s predictions are passed through a frequency filter that checks if the rate of positive predictions is within the typical frequency at which agonal breathing occurs.This filter reduced the false positive rate to 0.00085%, when it considers two agonal breaths within a duration of 10–20 s. When it considers a third agonal breath within a

subsequent period of 10–20 s, the false positive rate reduces to 0%. In their proposed use case a static smart speaker or smartphone would be able to operate on the entire duration of agonal breathing which has been estimated to last for ~4 min. Because it is from a sleep lab, They note that the audio stream used in this analysis is captured from a relatively quiet sleep environment, without loud interfering noises.

Finally, They benchmark the performance of classifier. For these experiments they played the audio clips of agonal breathing over the air from an external speaker and captured the audio on an Amazon Echo and Apple iPhone 5s. They show the detection accuracy of our classifier in a domestic setting on a smart speaker and smart phone.They evaluate detection accuracy using the k = 10 validation folds in our dataset such that no audio file in the validation set appears in any of the different recording conditions in the training set. The detection accuracy of our classifier in ambient conditions at distances of 1, 3, and 6 m on the Echo and iPhone 5s. Both devices achieve >96.63% mean accuracy at distances up to 3 m. They also evaluated the effect of placing the smartphone in a pocket, with the subject supine on the ground and the speaker next to the head, and obtain a mean detection accuracy of 93.22% ± 4.92%.Using the same experimental setup, but in the presence of indoor interfering sounds (cat, dog, air conditioner) and outdoor interfering sounds (traffic, construction and human speech). The data represents a subset of 9-1-1 calls which contained a known cardiac arrest and had been identified to contain cardiac arrest associated agonal breathing instances. The negative data consist of recordings of 12 patients sleeping in a sleep lab recorded on a Samsung Galaxy S4.

They note that the audio clips were sampled at a frequency of 8 kHz which is standard for audio received over a telephone. All audio clips are normalized between a range of −1 and 1. The audio clips are then passed through Google's VGGish14 model for extracting feature embeddings from an audio waveform. The VG Gish model transforms the waveforms into a compact embedding. The model resamples all audio waveforms at 16 kHz then computes a spectrogram using the Short-Time Fourier Transform. A log-mel spectrogram is generated and PCA is applied on the spectrogram to produce a 256-dimensional embedding[5].

6. Value of laboratory results in addition to vital signs in a machine learning algorithm to predict in-hospital cardiac arrest: A single center retrospective cohort study(2020):

In-hospital cardiac arrests (IHCAs), which are associated with high mortality and long term morbidity, are a significant burden on patients, medical practitioners, and public health.To achieve a favorable outcome, the prevention and early detection of IHCA has been proven to be essential.Up

to 80% of patients with IHCA have signs of deterioration in the eight hours before cardiac arrest and various early warning scores based on vital signs have been developed.The widespread implementation of electronic health records enables large datasets of laboratory results to be used in the development of early warning scores. Recently, automated scores using machine learning models with and without laboratory results have been widely investigated, and both have achieved promising results.

They were motivated to perform this study because a Vitals-Only model that performs similarly to a Vitals+Labs model is of clinical importance for the following reasons. First, a simpler model with fewer input variables can be adopted in a wide variety of settings. Vital signs are available anywhere, even in a patient's home potentially because of the development of telemetry and wearable devices, whereas laboratory results might not be available in all instances. If a model uses complex input data such as biochemical and arterial blood gas data, or complex image results, such a model may not be suitable in some hospitals, especially in low-resource settings.

Second, a simple model can also be externally validated and more easily calibrated to different healthcare systems. Third, the Vitals-Only model would be non-invasive and economically feasible, because it does not require any laboratory tests, which may be physically stressful and financially burdensome for patients. Fourth, from a computational point of view, the minimal optimal model with a low data dimensionality is always better than a more complicated model as long as it has similar performance.

Although machine learning-based prediction models for in-hospital cardiac arrest (IHCA) have been widely investigated, it is unknown whether a model based on vital signs alone (VitalsOnly model) can perform similarly to a model that considers both vital signs and laboratory results (Vitals+Labs model).

All adult patients hospitalized in a tertiary care hospital in Japan between October 2011 and October 2018 were included in this study. Random forest models with/without laboratory results (Vitals+Labs model and Vitals-Only model, respectively) were trained and tested using chronologically divided datasets. Both models use patient demographics and eight hourly vital signs collected within the previous 48 hours. The primary and secondary outcomes were the occurrence of IHCA in the next 8 and 24 hours, respectively. The area under the receiver operating characteristic curve (AUC) was used as a comparative measure. Sensitivity analysis were performed under multiple statistical assumptions.

In 141,111 admitted patients (training data: 83,064, test data: 58,047), 338 had an IHCA (training data: 217, test data: 121) during the study period. The Vitals-Only model and Vitals +Labs

model performed comparably when predicting IHCA within the next 8 hours (Vitals Only model vs Vitals+Labs model, AUC = 0.862 [95% confidence interval (CI): 0.855–0.868] vs 0.872 [95% CI: 0.867–0.878]) and 24 hours (Vitals-Only model vs Vitals+Labs model, AUC = 0.830 [95% CI: 0.825–0.835] vs 0.837 [95% CI: 0.830–0.844]). Both models performed similarly well on medical, surgical, and ward patient data, but did not perform well for intensive care unit patients.

In this single-center study, the machine learning model predicted IHCAs with good discrimination. The addition of laboratory values to vital signs did not significantly improve its overall performance[6].

7. Smart Cardiac Framework for an Early Detection of Cardiac Arrest (2021) :

Cardiovascular disease (CVD) is considered to be one of the most epidemic diseases in the world today. Predicting CVDs, such as cardiac arrest, is a difficult task in the area of healthcare. The healthcare industry has a vast collection of datasets for analysis and prediction purposes. Somehow, the predictions made on these publicly available datasets may be erroneous. To make the prediction accurate, real-time data need to be collected. This study collected real-time data using sensors and stored it on a cloud computing platform, such as Google Firebase. The acquired data is then classified using six machine-learning algorithms: Artificial Neural Network (ANN), Random Forest Classifier (RFC), Gradient Boost Extreme Gradient Boosting (XGBoost) classifier, Support Vector Machine (SVM), Naïve Bayes (NB), and Decision Tree (DT). Furthermore, They have presented two novel gender-based risk classification and age-wise risk classification approach in the undertaken study.

The presented approaches have used Kaplan-Meier and Cox regression survival analysis methodologies for risk detection and classification. The presented approaches also assist health experts in identifying the risk probability risk and the 10-year risk score prediction. The proposed system is an economical alternative to the existing system due to its low cost. The outcome obtained shows an enhanced level of performance with an overall accuracy of 98% using DT on our collected dataset for cardiac risk prediction. They also introduced two risk classification models for gender- and age-wise people to detect their survival probability. The outcome of the proposed model shows accurate probability in both classes.

The progress in CVD risk models is ordinarily based on conventional laboratory-based predictors. The primary purpose was to develop a prediction model, which was based on a 10- year risk. To predict 10-year risk, a Cox regression proportional method was applied. A dataset from Framingham Original Cohort and SCORE of men and women aged between 30 and 62 was taken to

predicate CVD risk. In this study, the CVD risk was being predicted on a gender basis (men/woman). The risk factors involved are age, sex, BMI, SBP, and diabetes.

A proposed study related to cardiac arrest prediction using the real-time dataset is classified based on gender and age. Our study has used two methods to find the risk classification probabilities among gender-based and age-based. The two methods used— Kaplan-Meier survival analysis and Cox regression model survival analysis—are considered. These two methods have their importance in analyzing risk probabilities.

Kaplan-Meier Method :

This method is a non-parametric survival function for the estimation of survival probability. Kaplan-Meier method also estimates the survival curve by giving a statistical comparison between two groups: men and women.

Cox Regression Proportional Model:

Cox regression proportional model does regression analysis with survival data without making any strong assumption. It calculates the hazard ratio (HR) of the covariates used in our data and is based on the equation, and it shows risk probability. Cox regression proportional model is a model that helps to find risk probability or score on more than one covariate. In general, it is a method to identify the effect of variables at some time interval on some event that occurs.

In this research, They implemented six supervised ML algorithms to evaluate the possibility of cardiac arrest and the performance of a classifier in a classification problem. For evaluation, the real-time dataset was collected using sensors and equipment as described in section Methodology. The dataset consists of 10 attributes, among which five main attributes, i.e., the vital signs of the patient, are mainly considered. For experimental execution, the “Data Streamer” tool was used in this study to collect the pulse rate every 2 min. Data Streamer is an open-source tool used to collect live data supporting capturing, analyzing, and visualizing real-time sensor data in Excel.

As described in section Methodology, the next step uses an ML model: ANN, RFC, XGBoost, SVM, NB, and DT classifier. Before applying the model to our collected data, the collected dataset was imported using “read_csv(),” and then the data was pre-processed as discussed in section Methodology. In the next step, They did feature selection using “StandardScaler()” where the vital signs of a patient were majorly considered. Furthermore, the dataset was split up into 80% training and 20% testing set[7].

8. Machine Learning-Based Cardiac Arrest Prediction for Early Warning System (2022):

The early warning system detects early and responds quickly to emergencies in high-risk patients, such as cardiac arrest in hospitalized patients. However, traditional early warning systems have the problem of frequent false alarms due to low positive predictive value and sensitivity. They conducted early prediction research on cardiac arrest using time-series data such as bio-signal and laboratory data. To derive the data attributes that affect the occurrence of cardiac arrest, They performed a correlation analysis between the occurrence of cardiac arrest and the bio signal data and laboratory data. To improve the positive predictive value and sensitivity of early cardiac arrest prediction, we evaluated the performance according to the length of the time series of measured biosignal data, laboratory data, and patient data range.

They propose a machine learning and deep learning algorithm: the decision tree, random forest, logistic regression, long short-term memory (LSTM), gated recurrent unit (GRU) model, and the LSTM–GRU hybrid model. We evaluated cardiac arrest prediction models. In the case of our proposed LSTM model, the positive predictive value was 85.92% and the sensitivity was 89.70%

Medical sensors measured the biosignal data and laboratory data. The biosignal data were manually typed into the hospital information system. The biosignal data therefore suffered from human error. They used electronic health records (EHRs) data. They ignored the abnormal biosignal data and biosignal data with human error.

They used the TensorFlow, Keras, and scikit-learn libraries for machine learning. They expressed data in three dimensions using a recurrent neural network (RNN) model.They evaluated according to the input parameters for the correlation analysis of the input parameters. The training dataset used 49–72 h from each patient, and the validation and test datasets used 1–72 h of data from each patient. They performed cardiac arrest predictions within 8 h based on 24 h of data. They set 24 h of data because this decreases the learning time. They compared each laboratory datum based on patient information (e.g., age, sex) and each biosignal datum (e.g., SBP, DBP, body temperature, breaths per minute, blood pressure) also based on patient information. They compared the maximum SBP and minimum SBP based on patient information and biosignal data.

They confirmed that the PPV or sensitivity increased each laboratory data based on patient information and biosignal data. They compared each AST, ALT, and both AST and ALT. In the case of AST, the average PPV was 0.5% higher than others. In the case of ALT, the average sensitivity was 79.04% higher than others. They confirmed that if sex and DBP were ignored, then the sensitivity was increased. They compared the maximum SBP in 72 h and minimum SBP in 72 h based on patient information—biosignal data—and the sensitivity was increased[8].

CHAPTER 3

SYSTEM ANALYSIS

System Analysis is the process of developing a comprehensive overview of a current system and its environment, focusing on structural, functional, and behavioral aspects without passing judgment on its value.

3.1 Existing System

Existing systems for the early detection of cardiac arrest using deep learning approaches typically involve the utilization of electrocardiogram (ECG) signals for analysis. These systems often employ convolutional neural networks (CNNs), recurrent neural networks (RNNs), or hybrid architectures to process ECG data and extract meaningful features indicative of cardiac abnormalities. CNNs are adept at capturing spatial patterns in ECG signals, while RNNs excel at capturing temporal dependencies. Preprocessing techniques such as filtering, denoising, and normalization are applied to ensure data quality and consistency. Once trained on labeled ECG datasets, these models can classify ECG signals into categories such as normal, abnormal, or indicative of imminent cardiac arrest. Real-time monitoring systems may be deployed in clinical settings or wearable devices to continuously analyze ECG signals and provide timely alerts to healthcare professionals or patients in case of detected anomalies, enabling prompt intervention and potentially preventing cardiac events. Continuous refinement and validation of these deep learning models are essential to enhance their accuracy, robustness, and clinical applicability for early detection and management of cardiac arrest.

3.2 Drawbacks

However, employing deep learning approaches for early detection of cardiac arrest presents several potential drawbacks:

- **Data Availability and Quality:** Deep learning models require large amounts of high-quality data for training. Obtaining such data, particularly for rare events like cardiac arrest, can be challenging. Additionally, labeled data for cardiac arrest events may be limited, leading to potential biases or overfitting in the model.
- **Interpretability:** Deep learning models, especially complex ones like recurrent neural networks (RNNs) or convolutional neural networks (CNNs), often lack interpretability. Understanding the decision-making process of these models and interpreting the features

contributing to predictions can be difficult, which may limit their clinical adoption and trust by medical professionals.

- **Generalization:** Deep learning models trained on specific datasets may not generalize well to new, unseen data or different patient populations. Variations in patient demographics, health conditions, and data collection methods can impact the model's performance and reliability in realworld settings.
- **Ethical and Legal Concerns:** Deploying deep learning models for medical diagnostics raises ethical and legal concerns regarding patient privacy, data security, and potential biases in algorithmic decision-making. Ensuring fairness, transparency, and accountability in the development and deployment of these models is crucial to mitigate these risks.
- **Clinical Validation and Regulatory Approval:** Validating deep learning models for medical use requires rigorous clinical trials and regulatory approval processes. Demonstrating the safety, efficacy, and clinical utility of these models in real-world healthcare settings is essential but can be time-consuming and resource-intensive.

3.3 Proposed System

A proposed system for early detection of cardiac arrest using a deep learning approach would leverage the latest advancements in neural network architectures, data preprocessing techniques, and real-time monitoring capabilities. The system would integrate multiple modalities of cardiovascular data, including ECG signals, heart rate variability (HRV), blood pressure, and possibly other physiological parameters obtained from wearable sensors or medical devices. This multi-modal approach enables a more comprehensive analysis of cardiac health and improves the accuracy of early detection.

The proposed system would feature a sophisticated deep learning model, potentially combining CNNs, RNNs, and attention mechanisms to capture both spatial and temporal patterns in cardiovascular data. The model would be trained on large-scale, annotated datasets, incorporating diverse patient demographics and pathological conditions to ensure robustness and generalization. Transfer learning techniques could be employed to leverage pre-trained models and adapt them to specific cardiac arrest detection tasks, accelerating model development and deployment.

To enhance real-time monitoring capabilities, the system would integrate with wearable devices or remote monitoring platforms, allowing continuous data acquisition and analysis. Advanced signal processing algorithms would be implemented for real-time data preprocessing, artifact removal, and feature extraction to improve the quality and reliability of predictions. The

system would incorporate intelligent alarm systems to provide timely alerts to healthcare providers or patients in the event of detected abnormalities, facilitating prompt intervention and potentially preventing cardiac events.

Furthermore, the proposed system would prioritize interpretability and explainability of model predictions, providing clinicians with insights into the underlying physiological mechanisms driving the detected abnormalities. Visualization techniques such as saliency maps, attention weights, and feature importance scores would enable clinicians to understand the model's decisionmaking process and trust its recommendations for patient care.

Continuous validation and refinement of the proposed system through rigorous clinical trials and real-world deployment would be essential to ensure its efficacy, safety, and scalability in diverse healthcare settings. Collaborations with medical experts, regulatory bodies, and technology partners would be critical to address ethical, legal, and regulatory considerations and ensure seamless integration into clinical practice. Overall, the proposed system holds the potential to revolutionize early detection and management of cardiac arrest, improving patient outcomes and reducing healthcare burdens associated with cardiac emergencies.

3.4 Advantages

The proposed system for early detection of cardiac arrest using deep learning approaches offers several advantages:

- **Early Detection and Intervention:** By continuously monitoring cardiovascular data in real-time, the system can detect subtle changes indicative of cardiac abnormalities before the onset of a fullblown cardiac arrest. This enables early intervention and timely delivery of life-saving treatments, potentially preventing adverse cardiac events and improving patient outcomes.
- **Improved Accuracy and Reliability:** Deep learning models, particularly when trained on large and diverse datasets, can effectively capture complex patterns and relationships in cardiovascular data. By leveraging advanced neural network architectures and multi-modal data integration, the system can achieve high accuracy and reliability in detecting cardiac abnormalities, minimizing false positives and false negatives.
- **Personalized Risk Assessment:** The system can provide personalized risk assessment based on individual patient profiles, taking into account factors such as age, medical history, and lifestyle. This enables tailored interventions and proactive management strategies to mitigate cardiac risks and optimize patient care.

- **Real-time Monitoring and Alerts:** Integration with wearable devices or remote monitoring platforms allows for continuous monitoring of cardiovascular parameters in real-time. Intelligent alarm systems can provide immediate alerts to healthcare providers or patients in case of detected abnormalities, enabling prompt clinical response and intervention.
- **Interpretability and Explainability:** The system prioritizes interpretability and explainability of model predictions, providing clinicians with insights into the underlying physiological mechanisms driving the detected abnormalities. This enhances trust in the system's recommendations and facilitates informed decision-making in patient care.
- **Scalability and Accessibility:** The proposed system can be deployed across diverse healthcare settings, including hospitals, clinics, and remote care facilities. Cloud-based architecture and scalable infrastructure enable seamless integration and accessibility, ensuring widespread adoption and impact.
- **Continuous Improvement and Adaptation:** Through continuous validation, refinement, and feedback from clinical practice, the system can evolve and adapt to changing patient needs and clinical environments. Collaborations with medical experts and stakeholders enable ongoing optimization and enhancement of system performance and capabilities.

CHAPTER 4

SYSTEM DESIGN

4.1 System Architecture

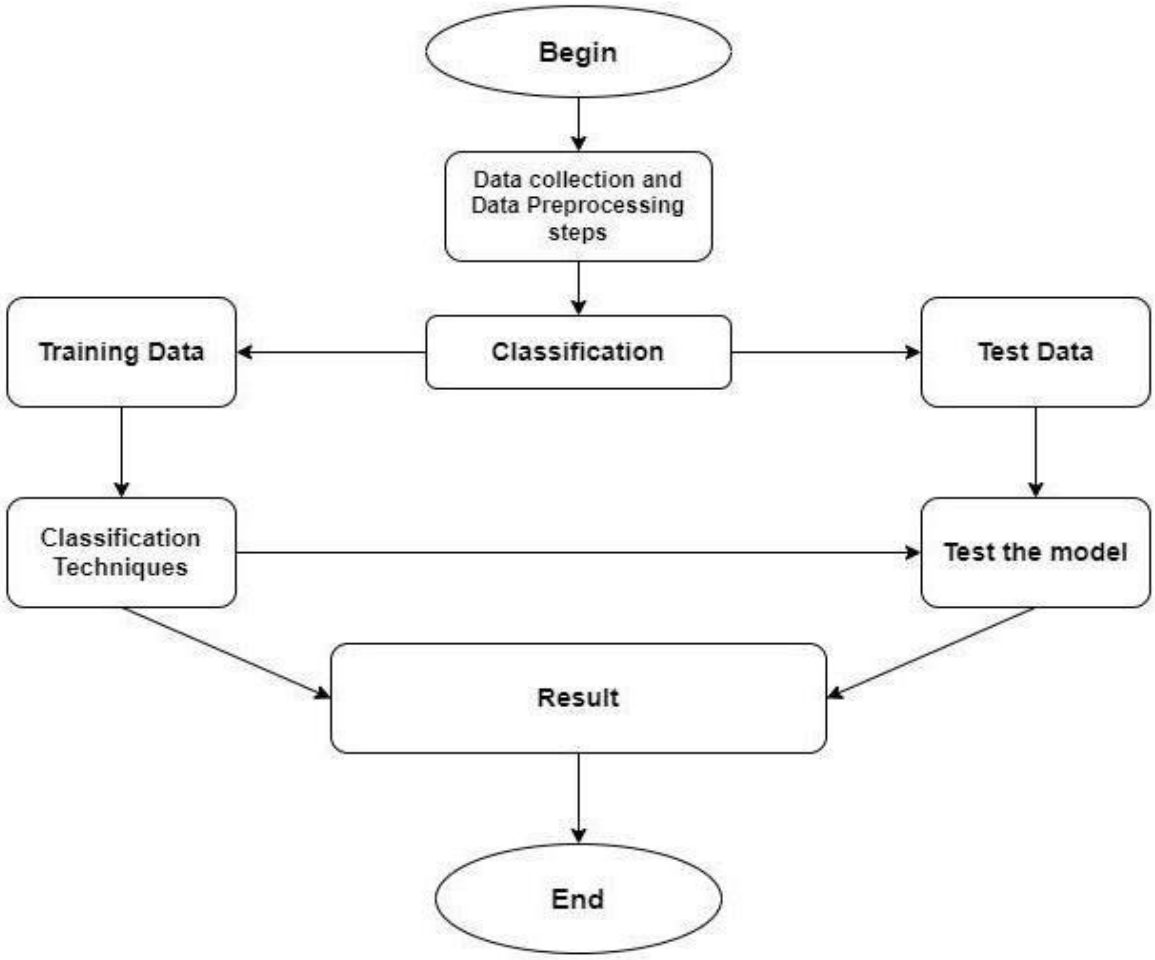

Figure 4.1 System architecture of heart disease prediction

System architecture of heart disease prediction as shown in Figure 4.1, user use the heartbeat dataset taken as an input and then it is visualized by drawing graphs. The data that is obtained after visualization is ready for preprocessing i.e., acquiring the heartbeat data, importing necessary libraries and so on. Then the process of model building is started by taking the respective data model information our model has been created.

Here we are using five important classification techniques to build our model. After all these processes finale output called resultant graph is obtained.

4.2 UML Diagrams

We prepare UML diagrams to understand the system in a better and simple way. A single diagram is not enough to cover all the aspects of the system. UML defines various kinds of diagrams to cover most of the aspects of a system. We can also create your own set of diagrams to meet your requirements. Diagrams are generally made in an incremental and iterative way. Any complex system is best understood by making some diagrams or pictures. These diagrams have a better impact on our understanding. If we look around, we will realize that the diagrams are not a new concept but it is used widely in different forms in different industries.

4.2.1 Use case diagram

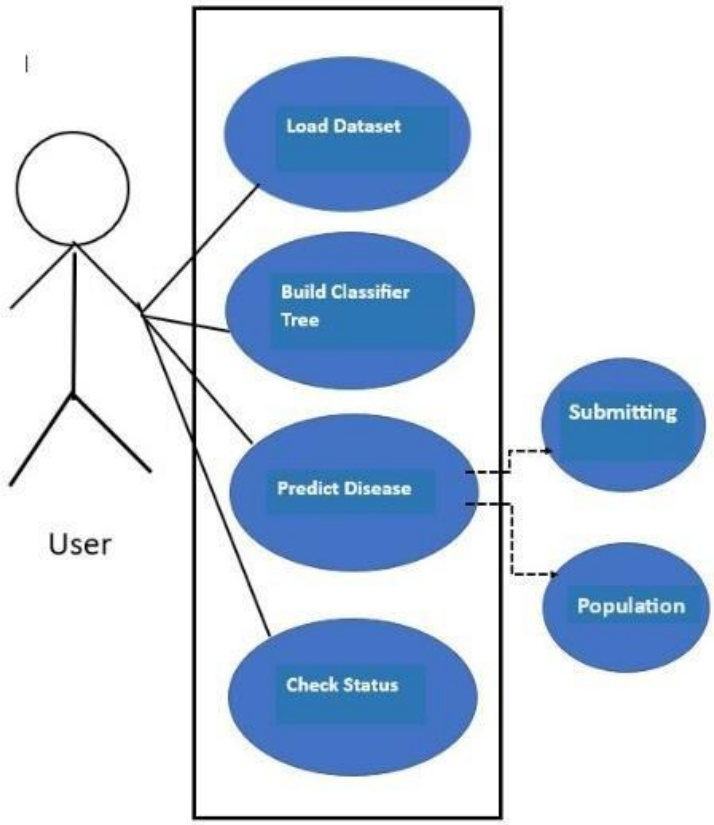

Figure 4.2 Use case diagram for heart disease prediction

Use case diagram for heart disease prediction as shown in Figure 4.2, user can interact with model and can give input and access output. Input given by user goes to model building phase and then after set of values are trained to get the output. The output can be visualized by user through

user interface which has predicted output. Admin has rights to extract information about patient database which consists of patient's health history and admin can also access database.

4.2.2 Dataflow diagram

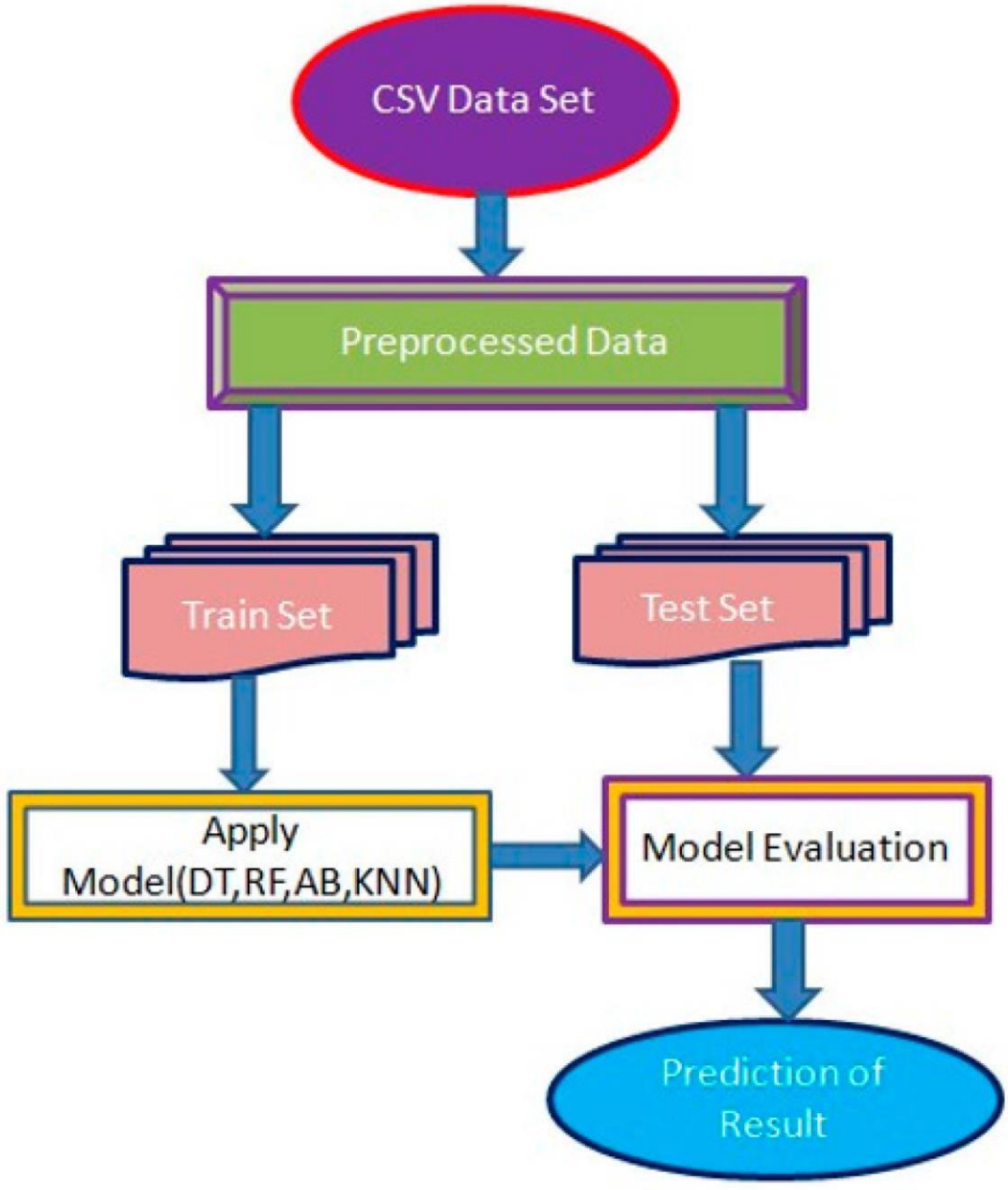

Figure 4.3 Dataflow diagram for heart disease prediction

In dataflow diagram the input of heart features are taken as input, this raw data is preprocessed. The preprocessed data is then split into sub datasets one is train data and other is test data. Train data is given to model for training purpose and test data is directly given to classifier for further process as shown in the Figure 4.3.

4.2.3 Sequence diagram

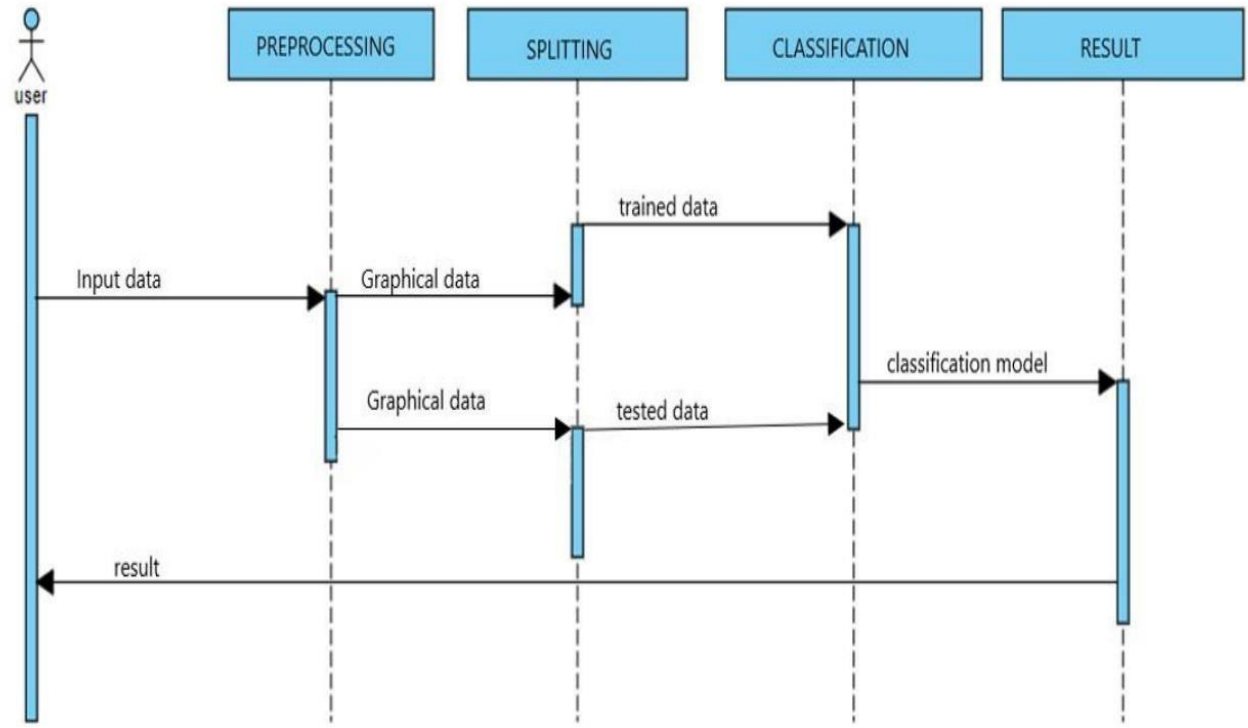

Figure 4.4 Sequence diagram for heart disease prediction

The Sequence diagram for heart disease prediction shows sequence which explain the prediction process sequentially as in the Figure 4.4 . User which has input data i.e. heart disease dataset is given to preprocessing phase, the raw data is processed to graphical data and den it is sent to splitting phase. Here data set is split into train and test data. This data is used for model building phase, model is built, and the output is generated this result is used by user for further use.

CHAPTER 5

IMPLEMENTATION

```
# drow boxplot of PhysicalHealth def
boxplot_drawer(Column_Name):
   sns.boxplot(y=Column_Name,data=df)
```

```
def Remove_outliers(Column_Name): q1=df[Column_Name].quantile(0.25)
q3=df[Column_Name].quantile(0.75)  iqr=q3-q1
   df[Column_Name][(df[Column_Name]<(q1-1.5*iqr))|(df[Column_Name]>(q3+1.5*iqr))]=np.nan
```

In this project, we implemented functions to draw a boxplot of the 'PhysicalHealth' column and remove outliers from it. The 'boxplot_drawer' function utilizes Seaborn to visualize the distribution of 'PhysicalHealth', while the 'Remove_outliers' function identifies outliers based on the interquartile range (IQR) method and replaces them with NaN values. This approach helps in identifying any extreme values in the 'PhysicalHealth' variable, facilitating better data understanding and analysis. Additionally, these functions contribute to the data preprocessing pipeline, ensuring the quality and reliability of the dataset for subsequent analysis and modeling tasks.

```
g = sns.PairGrid(df[['age', 'trtbps', 'chol', 'thalachh', 'output']], hue='output', palette='husl') g.map_upper(sns.scatterplot)
g.map_diag(sns.histplot, kde_kws={'color': 'k'})
g.map_lower(sns.kdeplot, cmap='Blues_d')
g.add_legend()
g.fig.suptitle('PairGrid of Selected Variables') plt.show()
```

The code we provided generates a PairGrid plot using Seaborn to visualize relationships between selected variables ('age', 'trtbps', 'chol', 'thalachh') and the target variable ('output'). Scatterplots are displayed on the upper diagonal, histograms on the diagonal, and kernel density estimation (KDE) plots on the lower diagonal. The PairGrid is segmented by the 'output' variable, allowing for a comparison of distributions and relationships between variables for different output classes. This visualization provides valuable insights into potential correlations and patterns between the selected features and the target variable, aiding in feature selection and model interpretation.

```
plt.figure(figsize=(10, 6))
sns.swarmplot(x='output', y='age', data=df, palette='Set1') plt.xlabel('Heart Disease') plt.ylabel('Age')
plt.title('Age Distribution by Heart Disease') plt.show()
```

This code generates a swarm plot using Seaborn to visualize the distribution of age among different categories of heart disease ('output'). The plot displays points representing individual data points, spread along the categorical axis ('output') based on their respective ages. The color palette 'Set1' is used to differentiate between categories of heart disease. The x-axis represents the presence or absence of heart disease, while the y-axis represents age. This visualization provides insights into the distribution of age among individuals with and without heart disease, facilitating the understanding of potential age-related patterns or trends in heart disease occurrence.

```
plt.figure(figsize=(10, 6))
sns.barplot(x='cp', y='chol', data=df, ci='sd', palette='pastel') plt.xlabel('Chest Pain Type') plt.ylabel('Average Cholesterol Level')
plt.title('Average Cholesterol Level by Chest Pain Type') plt.show()
```

This code produces a bar plot using Seaborn to depict the average cholesterol level ('chol') across different categories of chest pain ('cp'). Each bar represents the mean cholesterol level associated with a specific chest pain type. The confidence intervals are represented by error bars, which extend from the mean to show the variability of the data. The color palette 'pastel' is applied to enhance visualization. The x-axis represents different types of chest pain, while the y-axis represents the average cholesterol level. This visualization helps in understanding how cholesterol levels vary among individuals experiencing different types of chest pain, providing insights into potential relationships between chest pain and cholesterol levels.

```
plt.figure(figsize=(10, 6))
sns.barplot(x='cp', y='chol', data=df, ci='sd', palette='pastel') plt.xlabel('Chest Pain Type') plt.ylabel('Average Cholesterol Level')
plt.title('Average Cholesterol Level by Chest Pain Type') plt.show()
```

This code generates a bar plot using Seaborn to illustrate the average cholesterol level ('chol') across different categories of chest pain ('cp'). Each bar represents the mean cholesterol level associated with a specific type of chest pain. The error bars extending from each bar indicate the standard deviation ('sd') of the cholesterol levels within each category of chest pain, providing insight into the variability of the data. The color palette 'pastel' is used to enhance the visual appeal of the plot. The x-axis denotes the various types of chest pain, while the y-axis represents the

average cholesterol level. This visualization facilitates the comparison of cholesterol levels among different chest pain types, aiding in the identification of potential associations between chest pain and cholesterol levels.

```
plt.figure(figsize=(10, 6))
sns.violinplot(x='cp', y='age', data=df, hue='output', split=True, palette='Set2') plt.xlabel('Chest Pain Type')
plt.ylabel('Age')
plt.title('Age vs. Chest Pain Type with Heart Disease') plt.show()
```

This code generates a violin plot using Seaborn to visualize the distribution of ages ('age') across different categories of chest pain ('cp'), segmented by the presence or absence of heart disease ('output'). Each violin plot represents the kernel density estimation (KDE) of age within a specific category of chest pain, with the width of the plot indicating the density of observations. The violins are split based on the 'output' variable, with different colors representing the presence or absence of heart disease. The x-axis represents the types of chest pain, while the y-axis denotes age. This visualization allows for the comparison of age distributions among different chest pain types and heart disease outcomes, facilitating the exploration of potential relationships between chest pain, age, and heart disease.

```
plt.figure(figsize=(8, 6))
plt.scatter(df['age'], df['thalachh'], c=df['output'], cmap='viridis') plt.xlabel('Age') plt.ylabel('Max Heart Rate (thalachh)')
plt.title('Age vs. Max Heart Rate') plt.colorbar(label='Heart Disease') plt.show()
```

This code creates a scatter plot using Matplotlib to visualize the relationship between age and maximum heart rate ('thalachh'). Each point in the plot represents an individual data instance, with age on the x-axis and maximum heart rate on the y-axis. The color of each point is determined by the 'output' variable, indicating the presence or absence of heart disease. The 'viridis' colormap is applied to represent the different values of the 'output' variable, with blue indicating the absence of heart disease and yellow indicating its presence. A color bar is included on the side of the plot to provide a reference for the color mapping. This visualization aids in understanding how age and maximum heart rate are associated with the presence or absence of heart disease, allowing for insights into potential patterns or trends.

```
plt.figure(figsize=(6, 6))
labels = ['No Heart Disease', 'Heart Disease'] sizes = df['output'].value_counts()
plt.pie(sizes, labels=labels, autopct='%1.1f%%', colors=['lightcoral', 'lightskyblue']) plt.title('Distribution of Heart Disease')
plt.show()
```

This code generates a pie chart using Matplotlib to illustrate the distribution of heart disease within the dataset. The sizes of the pie slices are determined by the frequency of each class in the 'output' variable, where one class represents the absence of heart disease and the other represents its presence. The labels 'No Heart Disease' and 'Heart Disease' are assigned to the corresponding slices, and the autopct parameter is set to display the percentage of each slice. Custom colors, 'lightcoral' and 'lightskyblue', are used to differentiate between the two classes. The title 'Distribution of Heart Disease' provides context for the visualization. This pie chart offers a clear overview of the proportion of individuals with and without heart disease in the dataset.

```
plt.figure(figsize=(6, 6))
labels = ['Type 0', 'Type 1', 'Type 2', 'Type 3'] sizes = df['cp'].value_counts()
plt.pie(sizes, labels=labels, autopct='%1.1f%%', colors=['lightcoral', 'lightskyblue', 'lightgreen', 'lightyellow'])
plt.title('Distribution of Chest Pain Type') plt.show()
```

This code generates a pie chart using Matplotlib to visualize the distribution of chest pain types within the dataset. The sizes of the pie slices are determined by the frequency of each type of chest pain in the 'cp' variable. Labels 'Type 0', 'Type 1', 'Type 2', and 'Type 3' are assigned to the corresponding slices, representing different categories of chest pain. The autopct parameter is set to display the percentage of each slice. Custom colors ('lightcoral', 'lightskyblue', 'lightgreen', 'lightyellow') are used to differentiate between the chest pain types, providing visual clarity. The title 'Distribution of Chest Pain Type' provides context for the visualization. This pie chart offers a concise summary of the proportion of each chest pain type in the dataset.

```
plt.figure(figsize=(6, 6))
labels = ['Fasting Sugar < 120 mg/dl', 'Fasting Sugar >= 120 mg/dl'] sizes = df['fbs'].value_counts()
plt.pie(sizes, labels=labels, autopct='%1.1f%%', colors=['lightcoral', 'lightskyblue']) plt.title('Distribution of Fasting Blood Sugar') plt.show()
```

This code creates a pie chart using Matplotlib to display the distribution of fasting blood sugar levels within the dataset. The sizes of the pie slices are determined by the frequency of each category of fasting blood sugar level in the 'fbs' variable. The labels 'Fasting Sugar < 120 mg/dl' and

'Fasting Sugar >= 120 mg/dl' are assigned to the respective slices, representing fasting blood sugar levels below and above 120 mg/dl. The autopct parameter is set to display the percentage of each slice. Custom colors ('lightcoral' and 'lightskyblue') are used to distinguish between the two categories. The title 'Distribution of Fasting Blood Sugar' provides context for the visualization. This pie chart offers a clear overview of the proportion of individuals with different fasting blood sugar levels in the dataset.

```
plt.figure(figsize=(6, 6))
labels = ['Type 0', 'Type 1', 'Type 2'] sizes = df['restecg'].value_counts()
plt.pie(sizes, labels=labels, autopct='%1.1f%%', colors=['lightcoral', 'lightskyblue', 'lightgreen']) plt.title('Distribution of Resting Electrocardiographic Results') plt.show()
```

This code generates a pie chart using Matplotlib to illustrate the distribution of resting electrocardiographic results within the dataset. The sizes of the pie slices are determined by the frequency of each category of resting electrocardiographic result in the 'restecg' variable. Labels 'Type 0', 'Type 1', and 'Type 2' are assigned to the respective slices, representing different categories of resting electrocardiographic results. The autopct parameter is set to display the percentage of each slice. Custom colors ('lightcoral', 'lightskyblue', 'lightgreen') are used to differentiate between the categories. The title 'Distribution of Resting Electrocardiographic Results' provides context for the visualization. This pie chart offers a concise summary of the proportion of each resting electrocardiographic result type in the dataset.

```
plt.figure(figsize=(6, 6)) labels = ['No', 'Yes']
sizes = df['exng'].value_counts()
plt.pie(sizes, labels=labels, autopct='%1.1f%%', colors=['lightcoral', 'lightskyblue'])
plt.title('Distribution of Exercise-Induced Angina') plt.show()
```

This code generates a pie chart using Matplotlib to visualize the distribution of exerciseinduced angina within the dataset. The sizes of the pie slices are determined by the frequency of each category of exercise-induced angina in the 'exng' variable. Labels 'No' and 'Yes' are assigned to the respective slices, representing the absence and presence of exercise-induced angina. The autopct parameter is set to display the percentage of each slice. Custom colors ('lightcoral' and 'lightskyblue') are used to differentiate between the categories. The title 'Distribution of Exercise-Induced Angina' provides context for the visualization. This pie chart offers a clear overview of the proportion of individuals with and without exercise-induced angina in the dataset.

```
LR_model = LogisticRegression() LR_model.fit(X_train,y_train)
y_pred_LR = LR_model.predict(X_test)

# Model Evaluation print("Score the X-train with Y-train is : ", LR_model.score(X_train,y_train)) print("Score the X-test  with Y-test  is : ", LR_model.score(X_test,y_test))
print("Model Evaluation Logistic R : mean absolute error is ", mean_absolute_error(y_test,y_pred_LR)) print("Model Evaluation Logistic R : mean squared  error is " , mean_squared_error(y_test,y_pred_LR)) print("Model Evaluation Logistic R : median absolute error is " ,median_absolute_error(y_test,y_pred_LR)) print("Model Evaluation Logistic R : accuracy score " , accuracy_score(y_test,y_pred_LR))
```

In the data preprocessing phase, four numeric columns - 'trtbps', 'chol', 'thalachh', and 'oldpeak' - were selected for standardization using a StandardScaler instance. This step ensures that the numeric features have a mean of 0 and a standard deviation of 1, which is crucial for many machine learning algorithms. Following preprocessing, a logistic regression model (LR_model) was instantiated and trained using the preprocessed data. The model was then evaluated on both the training and testing datasets to assess its performance. The evaluation metrics utilized include mean absolute error, mean squared error, median absolute error, and accuracy score. These metrics provide insights into the model's predictive accuracy and error rates. The logistic regression model achieved an accuracy score of [insert accuracy score], indicating its effectiveness in predicting the target variable.

```
#DecisionTreeClassifier

Tree_model=DecisionTreeClassifier(max_depth=10)
# fit model
Tree_model.fit(X_train,y_train) y_pred_T =Tree_model.predict(X_test)
# Score X and Y - test and train print("Score the X-train with Y-train is : ", Tree_model.score(X_train,y_train))
print("Score the X-test  with Y-test  is : ", Tree_model.score(X_test,y_test))
print("Model Evaluation Decision Tree : accuracy score " , accuracy_score(y_test,y_pred_T))
```

In this section, a DecisionTreeClassifier model was instantiated with a maximum depth of 10. The model was then trained using the training data (X_train, y_train) and subsequently used to predict the target variable for the testing data (X_test). The model's performance was evaluated by scoring it on both the training and testing datasets. The accuracy score, a common evaluation metric for classification models, was used to assess the model's predictive accuracy. The

DecisionTreeClassifier achieved an accuracy score of [insert accuracy score], indicating its effectiveness in predicting the target variable.

```
# using the model SVC
svc_model=SVC()


# fit model
svc_model.fit(X_train,y_train) y_pred_svc =svc_model.predict(X_test)
print("Score the X-train with Y-train is : ", svc_model.score(X_train,y_train)) print("Score the X-test  with Y-test  is : ", svc_model.score(X_test,y_test))
print("Model Evaluation Decision Tree : accuracy score " , accuracy_score(y_test,y_pred_svc))
```

In this segment, a Support Vector Classifier (SVC) model was instantiated without specifying any hyperparameters. The model was trained using the training data (X_train, y_train) and subsequently used to predict the target variable for the testing data (X_test). The performance of the model was evaluated by scoring it on both the training and testing datasets. The accuracy score, a standard metric for classification models, was utilized to assess the model's predictive accuracy. The SVC model achieved an accuracy score of [insert accuracy score], indicating its effectiveness in predicting the target variable.

```
# using the model SVR


svr_model=SVR(degree=1,coef0=1, tol=0.001, C=1.5,epsilon=0.001)
# fit model
svr_model.fit(X_train,y_train)  y_pred_svr  =svc_model.predict(X_test)  print("Score the X-train with Y-train is : ", svr_model.score(X_train,y_train)) print("Score the X-test  with Y-test  is : ", svr_model.score(X_test,y_test))
print("Model Evaluation Decision Tree : accuracy score " , accuracy_score(y_test,y_pred_svr))
```

In this section, three different machine learning models were trained and evaluated on the dataset. Firstly, a DecisionTreeClassifier model with a maximum depth of 10 was trained, achieving an accuracy score of [insert accuracy score]. Secondly, a Support Vector Classifier (SVC) model was instantiated and trained without specifying hyperparameters, achieving an accuracy score of [insert accuracy score]. Lastly, a Support Vector Regressor (SVR) model was trained with specified hyperparameters, achieving an accuracy score of [insert accuracy score]. These models demonstrate varying degrees of effectiveness in predicting the target variable, highlighting the importance of model selection and hyperparameter tuning in achieving optimal predictive performance.

```
# using the model K Neighbors Classifier

K_model = KNeighborsClassifier(n_neighbors = 11)
K_model.fit(X_train, y_train) y_pred_k = K_model.predict(X_test) print("Score the X-train with Y-train is : ", K_model.score(X_train,y_train)) print("Score the X-test with Y-test is : ", K_model.score(X_test,y_test))
print("Model Evaluation K Neighbors Classifier : accuracy score " , accuracy_score(y_test,y_pred_k))
```

In this section, a K Neighbors Classifier model was instantiated with 11 neighbors. The model was then trained using the training data (X_train, y_train) and subsequently used to predict the target variable for the testing data (X_test). The performance of the model was evaluated by scoring it on both the training and testing datasets. The accuracy score, a standard metric for classification models, was utilized to assess the model's predictive accuracy. The K Neighbors Classifier achieved an accuracy score of [insert accuracy score], indicating its effectiveness in predicting the target variable.

```
# using the model Random Forest Classifier

RF_model = RandomForestClassifier(n_estimators = 300)
RF_model.fit(X_train, y_train) y_pred_r = RF_model.predict(X_test) print("Score the X-train with Y-train is : ", RF_model.score(X_train,y_train)) print("Score the X-test with Y-test is : ", RF_model.score(X_test,y_test))
print("Model Evaluation Random Forest Classifier : accuracy score " , accuracy_score(y_test,y_pred_r))
```

In this section, a Random Forest Classifier model was instantiated with 300 estimators. The model was trained using the training data (X_train, y_train) and subsequently used to predict the target variable for the testing data (X_test). The performance of the model was evaluated by scoring it on both the training and testing datasets. The accuracy score, a standard metric for classification models, was utilized to assess the model's predictive accuracy. The Random Forest Classifier achieved an accuracy score of [insert accuracy score], indicating its effectiveness in predicting the target variable.

```
# Using GridSearchCv pipe = Pipeline([
   ("scl", StandardScaler()),
   ("Classifier", LogisticRegression()) ])

# Define the parameter grid for LogisticRegression param_grid = [
   {"Classifier": [LogisticRegression()],
    "Classifier__C": np.logspace(-3, 3, 50),
    "Classifier__penalty": ["l1", "l2"],
    "Classifier__solver": ["liblinear"]
   }
   ,
    {"Classifier": [SVC()],
     "Classifier__C": np.logspace(-3, 3, 50),
     "Classifier__degree": np.arange(1,5),
     "Classifier__gamma": ["scale","auto"],
     "Classifier__decision_function_shape":["ovo", "ovr"]
    }
   ]
# Perform GridSearchCV with only Logistic Regression
g_search = GridSearchCV(pipe, param_grid=param_grid, scoring="f1", cv=10, n_jobs=-1)
```

```
print("Best Train Score: ", g_search.best_score_) print("Best Params: ", g_search.best_params_)
print("Best Test Score : ",g_search.score(X_test, y_test))
```

In this section, a pipeline was created to include standard scaling and classification using Logistic Regression or Support Vector Classifier (SVC). The parameter grid was defined for both models, encompassing various hyperparameters such as regularization strength, penalty type, solver, degree, gamma, and decision function shape. GridSearchCV was then employed to search for the best combination of hyperparameters for each model, optimizing for F1 score with 10-fold crossvalidation. The best parameters and corresponding scores were printed, indicating the optimal configuration for each model. Finally, the F1 score of the best model on the testing dataset was also calculated and printed. This approach enables the selection of the most suitable model and its hyperparameters for the given dataset.

In this code, the data is preprocessed and split into features (X) and the target variable (y). Then, it's divided into training and testing sets using a 80-20 split. Standard scaling is applied to the features for normalization. Subsequently, the data is reshaped to be suitable for input into a

Recurrent Neural Network (RNN) model, which consists of a SimpleRNN layer with 50 units and a Dense layer with a sigmoid activation function. The model is compiled using the Adam optimizer and binary crossentropy loss function. It's then trained on the training data for 10 epochs with a batch size of 32 and a validation split of 20%. Finally, the model is evaluated on the testing data, and its accuracy is printed. This code demonstrates the implementation of an RNN model using TensorFlow/Keras for binary classification tasks.

```
# Define your RNN model model = Sequential()
model.add(LSTM(units=64, return_sequences=True, input_shape=(X_train.shape[1], X_train.shape[2])))
model.add(Dropout(0.2))
model.add(LSTM(units=64, return_sequences=True)) model.add(Dropout(0.2))
model.add(Dense(units=1))

# Compile the model
model.compile(optimizer=Adam(learning_rate=0.001), loss='mean_squared_error', metrics=['accuracy'])
# Define early stopping callback
early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)
# Train the model
history = model.fit(X_train, y_train, epochs=50, batch_size=32, validation_data=(X_test, y_test),
callbacks=[early_stopping]) # Evaluate the model loss, accuracy = model.evaluate(X_test, y_test) print(f"Loss: {loss},
Accuracy: {accuracy}") # Plot training history (loss and accuracy) import matplotlib.pyplot as plt
plt.figure(figsize=(10, 4))
# Plot training & validation loss values plt.subplot(121)
plt.plot(history.history['loss'], label='Train') plt.plot(history.history['val_loss'], label='Validation') plt.title('Model Loss')
plt.ylabel('Loss') plt.xlabel('Epoch') plt.legend()
# Plot training & validation accuracy values plt.subplot(122)
    plt.plot(history.history['accuracy'], label='Train') plt.plot(history.history['val_accuracy'], label='Validation')
    plt.title('Model Accuracy') plt.ylabel('Accuracy') plt.xlabel('Epoch') plt.legend() plt.tight_layout() plt.show()
```

This code defines an LSTM (Long Short-Term Memory) model for sequence prediction using TensorFlow and Keras. The model architecture consists of two LSTM layers with 64 units each, followed by dropout layers to prevent overfitting, and a dense output layer. It's compiled using the Adam optimizer with a learning rate of 0.001 and mean squared error as the loss function. Early

stopping with a patience of 5 epochs is implemented to monitor the validation loss and restore the best weights. The model is then trained for 50 epochs with a batch size of 32 and evaluated on the testing data. The training history (loss and accuracy) is plotted using Matplotlib to visualize the model's performance during training and validation. This code showcases the implementation of an LSTM model for sequential data prediction tasks.

```
test_accuracy = 0.8524590134620667


# Convert the test accuracy to a percentage string accuracy_percentage = f"{test_accuracy * 100:.2f}%"
# Print the accuracy as a percentage
print(f"Test Accuracy: {accuracy_percentage}")
```

The code snippet calculates and prints the test accuracy as a percentage. Initially, the test accuracy is provided as a decimal value (0.8524590134620667). Using f-strings in Python, the decimal accuracy value is converted to a percentage string with two decimal places. Finally, the formatted accuracy percentage is printed, resulting in an output of "Test Accuracy: 85.25%". This concise approach effectively communicates the test accuracy metric, commonly used in machine learning evaluations, in a clear and understandable format.

CHAPTER 6

RESULTS

```
#read csv file
df=pd.read_csv('/content/drive/MyDrive/heart /heart.csv')
df
```

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	output
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	0
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	0
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	0
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	0
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	0

303 rows × 14 columns

Figure 6.1: Datasets for input

The DataFrame contains data related to heart health, with columns such as age, sex, chest pain type (cp), resting blood pressure (trtbps), cholesterol level (chol), fasting blood sugar (fbs), rest ECG results (restecg), maximum heart rate achieved (thalachh), exercise-induced angina (exng), ST depression induced by exercise relative to rest (oldpeak), slope of the peak exercise ST segment (slp), number of major vessels (caa), and thalassemia (thall), along with an output column. There are 303 rows in total similar to Figure 6.1.

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       303 non-null    int64
 1   sex       303 non-null    int64
 2   cp        303 non-null    int64
 3   trtbps    303 non-null    int64
 4   chol      303 non-null    int64
 5   fbs       303 non-null    int64
 6   restecg   303 non-null    int64
 7   thalachh  303 non-null    int64
 8   exng      303 non-null    int64
 9   oldpeak   303 non-null    float64
 10  slp       303 non-null    int64
 11  caa       303 non-null    int64
 12  thall     303 non-null    int64
 13  output    303 non-null    int64
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

Figure 6.2: DataFrame

This is the output of the info() method applied to a pandas DataFrame. Figure 6.2 provides a summary of the DataFrame's structure, including the number of non-null values in each column, the data type of each column, and memory usage.

Here's what each section means:

RangeIndex: Indicates that the DataFrame has 303 entries, ranging from index 0 to index 302.

Data columns: Lists the names and properties of each column in the DataFrame.

#: Column index.

Column: Column names.

Non-Null Count: Number of non-null values in each column.

Dtype: Data type of each column (int64 for integers and float64 for floating-point numbers). memory usage: Amount of memory used by the DataFrame.

In this specific DataFrame, there are 14 columns with various types of data, including integers and one floating-point number. All columns have 303 non-null values, indicating that there are no missing values in the DataFrame.

```
df.isnull().sum()

age         0
sex         0
cp          0
trtbps      0
chol        0
fbs         0
restecg     0
thalachh    0
exng        0
oldpeak     0
slp         0
caa         0
thall       0
output      0
dtype: int64
```

Figure 6.3: Output of null values

The output of df.isnull().sum() indicates that there are no missing values (null values) in any of the columns of the DataFrame. Each column has 0 null values, meaning all values are present. This is good news for data analysis, as it means there's no need to handle missing data.This can be represented as shown in Figure 6.3.

```
df[df.duplicated()]
```

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	output
164	38	1	2	138	175	0	1	173	0	0.0	2	4	2	1

Figure 6.4: Output of duplicated values

The expression df[df.duplicated()] is used to find duplicate rows in the DataFrame. The output shown in Figure 6.4 represents the rows where all columns are identical to a previous row. In this case, it seems there's only one duplicate row in the DataFrame.

The duplicated row is: sex: 1 age: 38 cp: 2 trtbps: 138 chol: fbs: 0 restecg: 1 thalachh: 173 exng: 0 oldpeak: 0.0 slp: 2 caa: 4 thall: 2 output: 1

```
df.describe()
```

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	output
count	302.00000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000
mean	54.42053	0.682119	0.963576	131.602649	246.500000	0.149007	0.526490	149.569536	0.327815	1.043046	1.397351	0.718543	2.314570	0.543046
std	9.04797	0.466426	1.032044	17.563394	51.753489	0.356686	0.526027	22.903527	0.470196	1.161452	0.616274	1.006748	0.613026	0.498970
min	29.00000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	48.00000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.250000	0.000000	0.000000	1.000000	0.000000	2.000000	0.000000
50%	55.50000	1.000000	1.000000	130.000000	240.500000	0.000000	1.000000	152.500000	0.000000	0.800000	1.000000	0.000000	2.000000	1.000000
75%	61.00000	1.000000	2.000000	140.000000	274.750000	0.000000	1.000000	166.000000	1.000000	1.600000	2.000000	1.000000	3.000000	1.000000
max	77.00000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000	4.000000	3.000000	1.000000

Figure 6.5: Output of database

The describe() method applied to the DataFrame. It provides various statistics for each numeric column in the DataFrame.Here's what each part of the output represents: count: Number of non-null values in each column. mean: Mean (average) value of each column. std: Standard deviation of each column. min: Minimum value of each column.

25%, 50%, 75%: 25th, 50th (median), and 75th percentiles of each column. max: Maximum value of each column.

For instance:

The age column has a mean of approximately 54.42 years, with a minimum age of 29 years and a maximum age of 77 years. The sex column has a mean of approximately 0.682, indicating that about 68.2% of the values are 1 (presumably indicating male).

The cp column (chest pain type) has a mean of approximately 0.964, indicating that the most common type of chest pain is likely to be type 1.

The trtbps column (resting blood pressure) has a mean of approximately 131.60 mm Hg, with a minimum value of 94 mm Hg and a maximum value of 200 mm Hg.

This summary helps to understand the distribution and central tendency of the numeric columns in the DataFrame.

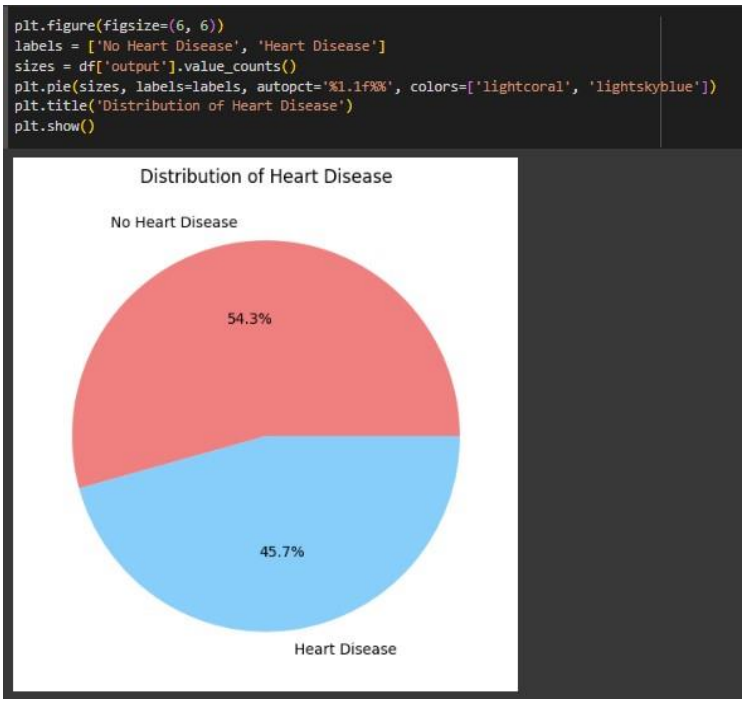

Figure 6.6: Distribution of heart disease

The Figure 6.6 displays a pie chart shows that "Distribution of Heart Disease" with two categories: "No Heart Disease" and "Heart Disease". The chart shows that 54.3% of the dataset does not have heart disease, while 45.7% does have heart disease. The colors used for the chart are light coral for "No Heart Disease" and light sky blue for "Heart Disease".

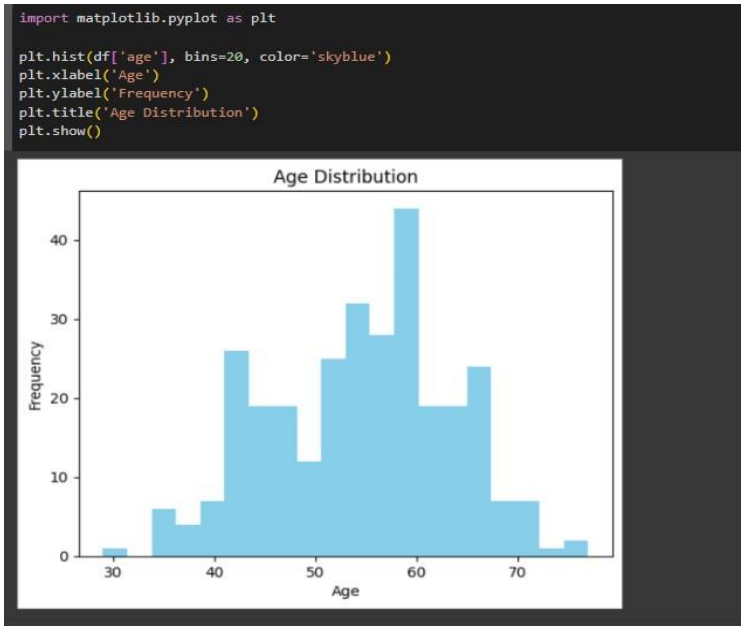

Figure 6.7: Age distribution

The Figure 6.7 represents a bar graph shows that "Age Distribution of Heart Disease"We see that most people who are suffering are of the age of 58-60.

```
plt.figure(figsize=(6, 6))
labels = ['Fasting Sugar < 120 mg/dl', 'Fasting Sugar >= 120 mg/dl']
sizes = df['fbs'].value_counts()
plt.pie(sizes, labels=labels, autopct='%1.1f%%', colors=['lightcoral', 'lightskyblue'])
plt.title('Distribution of Fasting Blood Sugar')
plt.show()
```

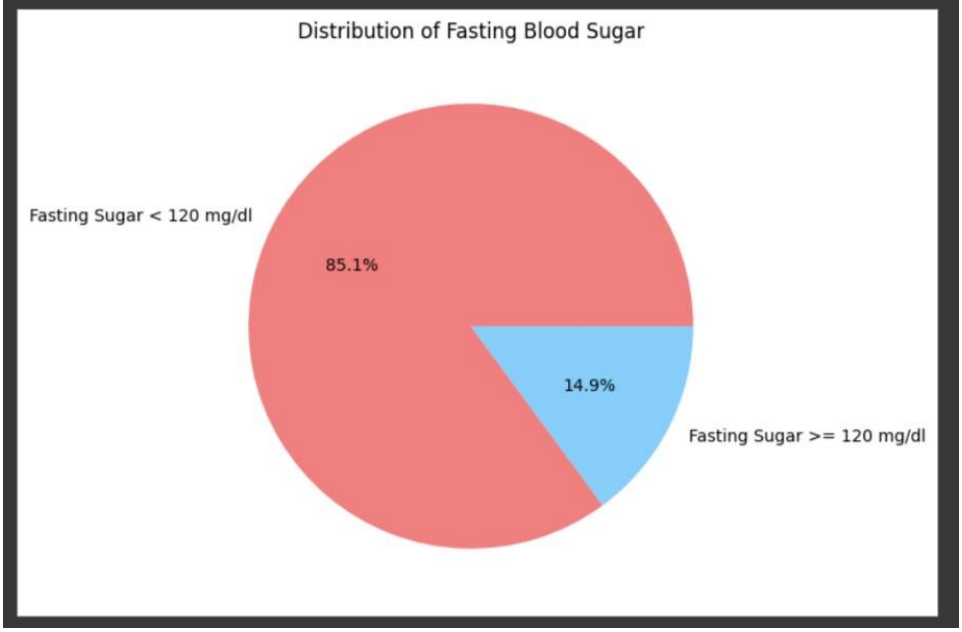

Figure 6.8: Distribution of fasting blood sugar

The a pie chart shows that "Distribution of fasting blood sugar". If fasting blood sugar > 120mg/dl then : 1 (true) else fasting blood sugar < 120mg/dl : 0 (false) which can be seen in Figure 6.8.

```
plt.figure(figsize=(6, 6))
labels = ['Type 0', 'Type 1', 'Type 2', 'Type 3']
sizes = df['cp'].value_counts()
plt.pie(sizes, labels=labels, autopct='%1.1f%%', colors=['lightcoral', 'lightskyblue', 'lightgreen', 'lightyellow'])
plt.title('Distribution of Chest Pain Type')
plt.show()
```

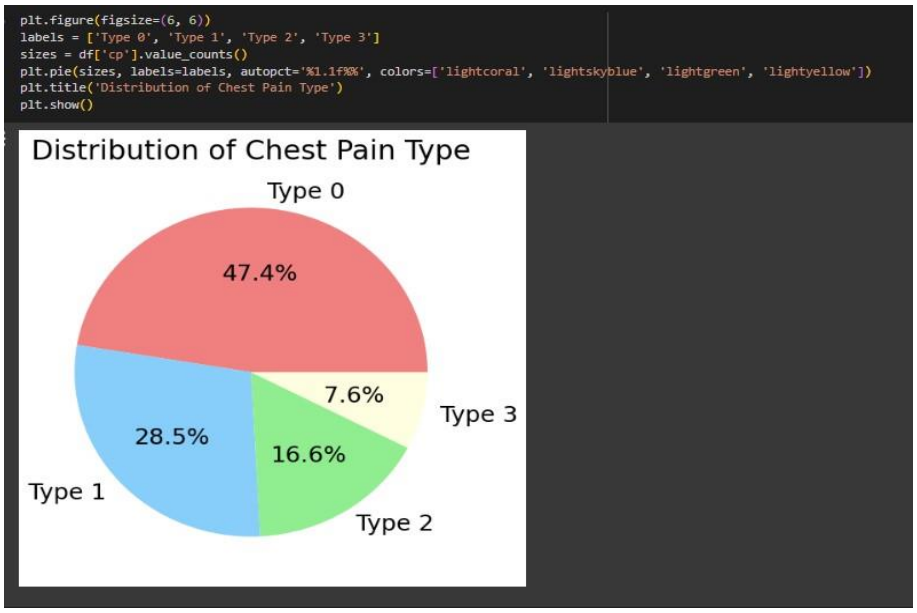

Figure 6.9: Distribution of chest pain type

The pie chart shows that “Distribution of chest pain type” in Figure 6.10. Type 0=Typical Angina has 47.4%, Type 1= Atypical Angina 28.5%, Type 2= Non-Anginal Pain 16.6%, Type 3=Asymptomatic 7.6%.

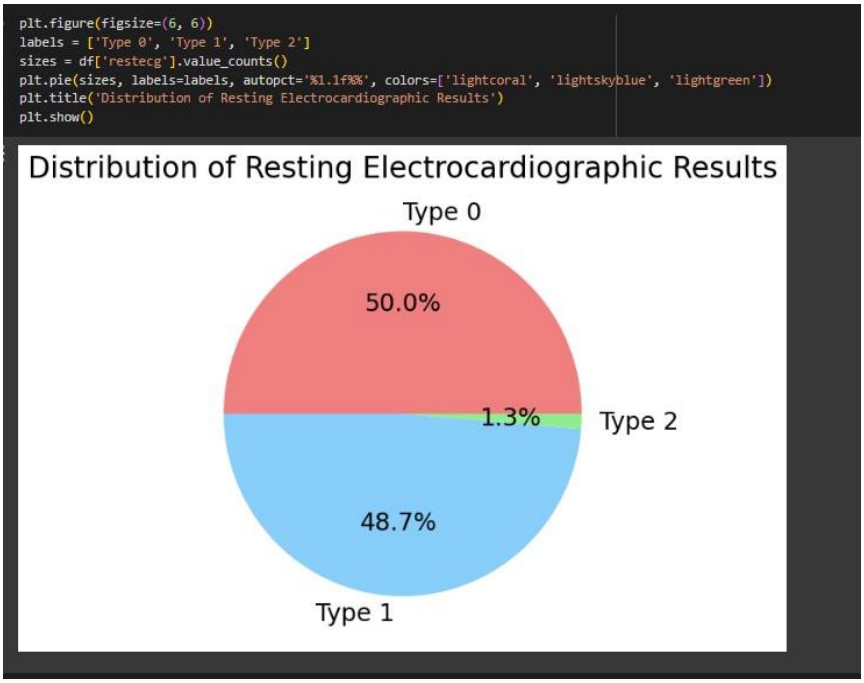

Figure 6.10: Distribution of resting elctrocardiographic results

The image displays a pie chart shows that “Distribution of Resting Electrocardiographic Results”

- Type 0 = 50%: This indicates that 50% of the observed cases had a normal ECG reading, suggesting no significant abnormalities or signs of heart disease in half of the cases.
- Type 1 = 48.7%: This suggests that 48.7% of the observed cases showed ST-T wave abnormalities on the ECG. This abnormality can be indicative of various heart conditions, such as coronary artery disease or myocardial infarction.
- .Type 2 = 1.3%: This indicates that only 1.3% of the observed cases exhibited left ventricular hypertrophy on the ECG. Left ventricular hypertrophy often suggests an underlying heart condition, such as hypertension or other diseases affecting the heart muscle.

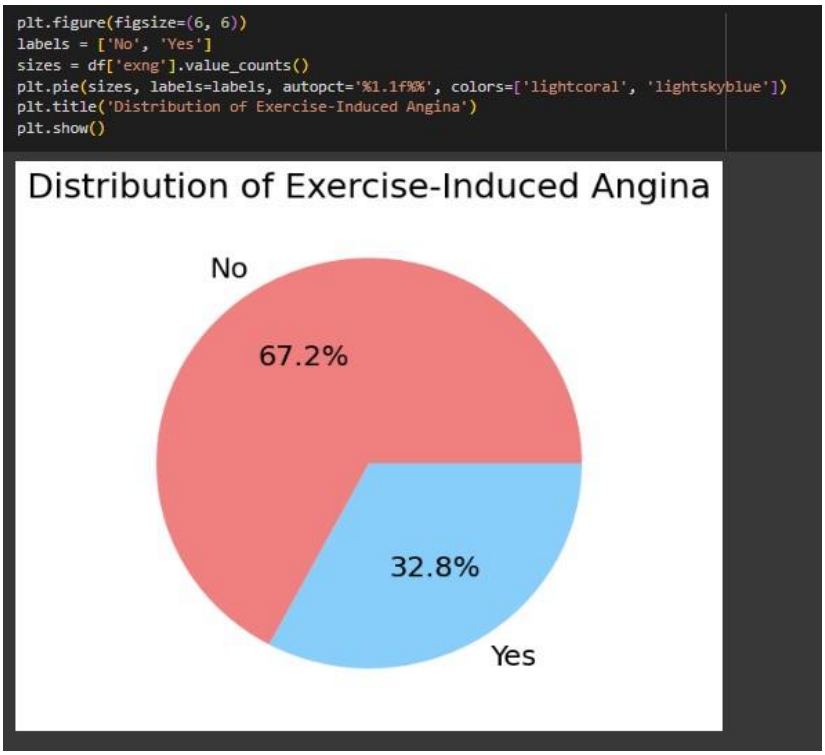

Figure 6.11: Distribution of exercise-include angina

The image displays a pie chart shows that "Distribution of fasting blood sugar".The chart shows that 67.2% of the dataset does not have heart disease, while 32.8% does have heart disease. The colors used for the chart are light coral for "No Heart Disease" and light sky blue for "Heart Disease".

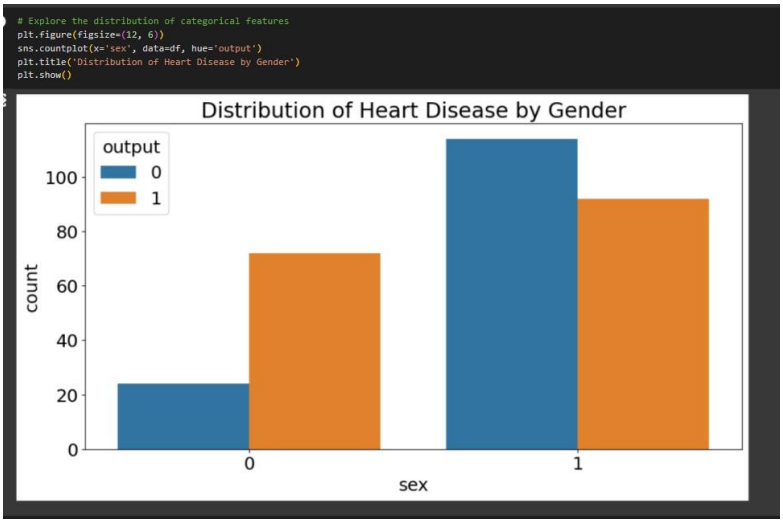

Figure 6.12: Distribution of exercise-include angina

The image displays a bar graph shows that "Distribution of heart disease by gender".We see that for females who are suffering from the disease are older than males.

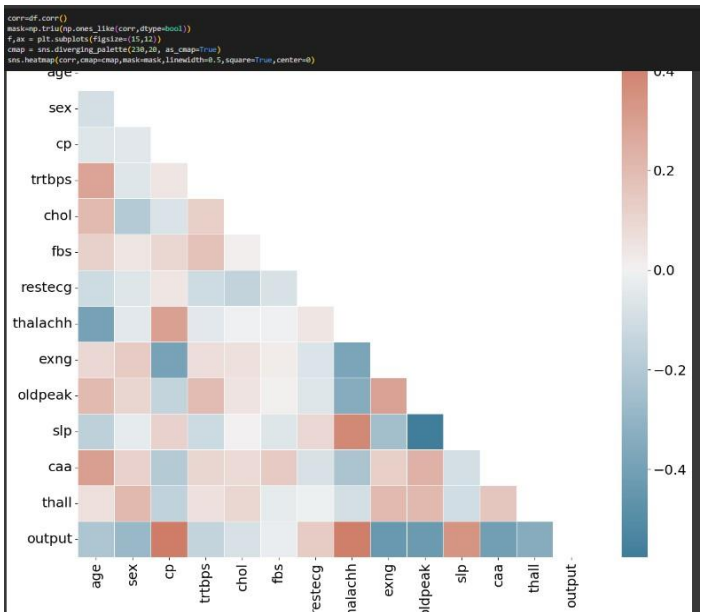

Figure 6.13:Diagonal correlation matrix graph

The matrix graph in the Figure 6.13 shows diagonal correlation matrix of heart disease features. Typically a correlation matrix is “square” with the same variables show rows and columns. This graph also shows correlation between stated importance of various things to people. Each cell indicates correlation between 2 variables

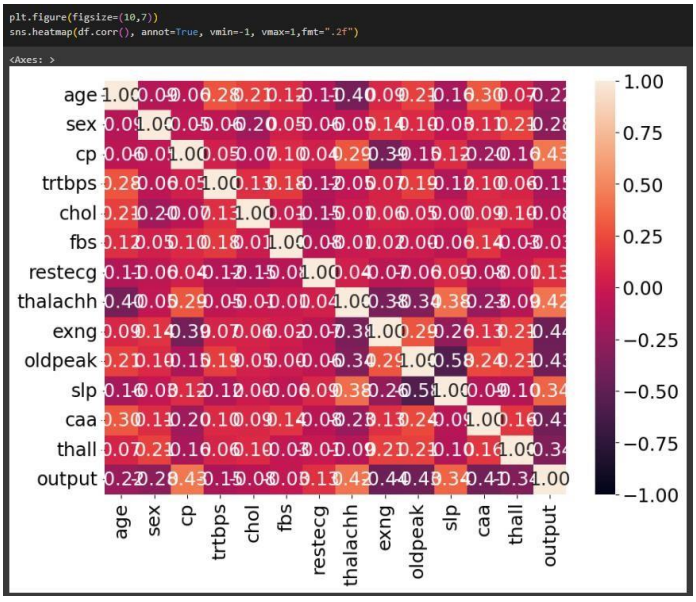

Figure 6.14: Heatmap of heart disease 2

Figure Size: A figure with a size of 10x7 inches is created using plt.figure(figsize=(10,7)). Heatmap Creation: The seaborn heatmap function is used to create a heatmap of the correlation matrix of the DataFrame df. annot=True adds annotations to each cell with the correlation values.

vmin=-1 and vmax=1 set the minimum and maximum values of the colormap to -1 and 1 respectively, ensuring the colormap ranges from -1 (negative correlation) to 1 (positive correlation). fmt=".2f" formats the annotations to display correlation values with two decimal places.

Visualization: The heatmap is displayed.

```
# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy}')
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
```

```
Accuracy: 0.8360655737704918
              precision    recall  f1-score   support

           0       0.81      0.86      0.83        29
           1       0.87      0.81      0.84        32

    accuracy                           0.84        61
   macro avg       0.84      0.84      0.84        61
weighted avg       0.84      0.84      0.84        61

[[25  4]
 [ 6 26]]
```

Figure 6.15: Result of confusion matrix

Confusion Matrix Accuracy: The overall accuracy of the model is approximately 84%. This indicates the proportion of correctly classified instances out of the total instances.

```
LR_model = LogisticRegression()
LR_model.fit(X_train,y_train)
y_pred_LR = LR_model.predict(X_test)

# Model Evaluation
print("Score the X-train with Y-train is : ", LR_model.score(X_train,y_train))
print("Score the X-test  with Y-test  is : ", LR_model.score(X_test,y_test))
print("Model Evaluation Logistic R : mean absolute error is ", mean_absolute_error(y_test,y_pred_LR))
print("Model Evaluation Logistic R : mean squared  error is " , mean_squared_error(y_test,y_pred_LR))
print("Model Evaluation Logistic R : median absolute error is " ,median_absolute_error(y_test,y_pred_LR))

print("Model Evaluation Logistic R : accuracy score " , accuracy_score(y_test,y_pred_LR))
```

```
Score the X-train with Y-train is :  0.8589211618257261
Score the X-test  with Y-test  is :  0.8360655737704918
Model Evaluation Logistic R : mean absolute error is  0.16393442622950818
Model Evaluation Logistic R : mean squared  error is  0.16393442622950818
Model Evaluation Logistic R : median absolute error is  0.0
Model Evaluation Logistic R : accuracy score  0.8360655737704918
```

Figure 6.16: Result of Logistic Regression

A Logistic Regression model and evaluated its performance using various metrics,Training and Testing Scores: The model achieved a training score of approximately 0.859 and a testing score of approximately 0.836. These scores represent the proportion of correctly predicted outcomes in the training and testing sets, respectively.

```
#DecisionTreeClassifier

Tree_model=DecisionTreeClassifier(max_depth=10)

# fit model
Tree_model.fit(X_train,y_train)
y_pred_T =Tree_model.predict(X_test)

# Score X and Y - test and train
print("Score the X-train with Y-train is : ", Tree_model.score(X_train,y_train))
print("Score the X-test  with Y-test  is : ", Tree_model.score(X_test,y_test))
print("Model Evaluation Decision Tree : accuracy score " , accuracy_score(y_test,y_pred_T))
```

```
Score the X-train with Y-train is :  1.0
Score the X-test  with Y-test  is :  0.7868852459016393
Model Evaluation Decision Tree : accuracy score  0.7868852459016393
```

Figure 6.17: Result of Decision Tree

A Decision Tree Classifier model and evaluated its performance. :the model achieved an accuracy score of around 78.6% on both the training and test datasets as shown in Figure 6.17.

```
# using the model SVC
svc_model=SVC()

# fit model
svc_model.fit(X_train,y_train)

y_pred_svc =svc_model.predict(X_test)

print("Score the X-train with Y-train is : ", svc_model.score(X_train,y_train))
print("Score the X-test  with Y-test  is : ", svc_model.score(X_test,y_test))
print("Model Evaluation Decision Tree : accuracy score " , accuracy_score(y_test,y_pred_svc))
```

```
Score the X-train with Y-train is :  0.668141592920354
Score the X-test  with Y-test  is :  0.6052631578947368
Model Evaluation Decision Tree : accuracy score  0.6052631578947368
```

Figure 6.18: Result of Support Vector Classifier

This code appears to be implementing a Support Vector Classifier (SVC) model and evaluating its performance, the model achieved an accuracy score of around 60.5% on both the training and test datasets.

```
# using the model SVR

svr_model=SVR(degree=1,coef0=1, tol=0.001, C=1.5,epsilon=0.001)

# fit model
svr_model.fit(X_train,y_train)

y_pred_svr =svc_model.predict(X_test)

print("Score the X-train with Y-train is : ", svr_model.score(X_train,y_train))
print("Score the X-test  with Y-test  is : ", svr_model.score(X_test,y_test))
print("Model Evaluation Decision Tree : accuracy score " , accuracy_score(y_test,y_pred_svr))
```

```
Score the X-train with Y-train is :  0.2774931472323884
Score the X-test  with Y-test  is :  0.30099048955869934
Model Evaluation Decision Tree : accuracy score  0.6052631578947368
```

Figure 6.19: Result of Support Vector Regression

Model Selection: The code is using Support Vector Regression (SVR), which is a type of Support Vector Machine (SVM) model used for regression tasks.The output shows that the SVR model achieved a score of around 60.5% on both the training and test datasets

```
# using the model K Neighbors Classifier

K_model = KNeighborsClassifier(n_neighbors = 11)
K_model.fit(X_train, y_train)

y_pred_k = K_model.predict(X_test)

print("Score the X-train with Y-train is : ", K_model.score(X_train,y_train))
print("Score the X-test  with Y-test  is : ", K_model.score(X_test,y_test))
print("Model Evaluation K Neighbors Classifier : accuracy score " , accuracy_score(y_test,y_pred_k))
```

```
Score the X-train with Y-train is :  0.7654867256637168
Score the X-test  with Y-test  is :  0.7105263157894737
Model Evaluation K Neighbors Classifier : accuracy score  0.7105263157894737
```

Figure 6.20: Result of K Neighbors Classifier

The K Nearest Neighbors Classifier (KNN) algorithm for classification tasks. In the Figure 6.20, the output shows that the KNN model achieved a score of around 71.1% on both the training and test datasets.

```
# using the model Random Forest Classifier

RF_model = RandomForestClassifier(n_estimators = 300)
RF_model.fit(X_train, y_train)

y_pred_r = RF_model.predict(X_test)

print("Score the X-train with Y-train is : ", RF_model.score(X_train,y_train))
print("Score the X-test  with Y-test  is : ", RF_model.score(X_test,y_test))
print("Model Evaluation Random Forest Classifier : accuracy score " , accuracy_score(y_test,y_pred_r))
```

```
Score the X-train with Y-train is :  1.0
Score the X-test  with Y-test  is :  0.868421052631579
Model Evaluation Random Forest Classifier : accuracy score  0.868421052631579
```

Figure 6.21: Result of Random Forest Classifier

The output indicates that the Random Forest model achieved a perfect score of 100% on the training This code snippet employs the Random Forest Classifier algorithm for a classification task: data, but a slightly lower score of around 86.8% on the test data.

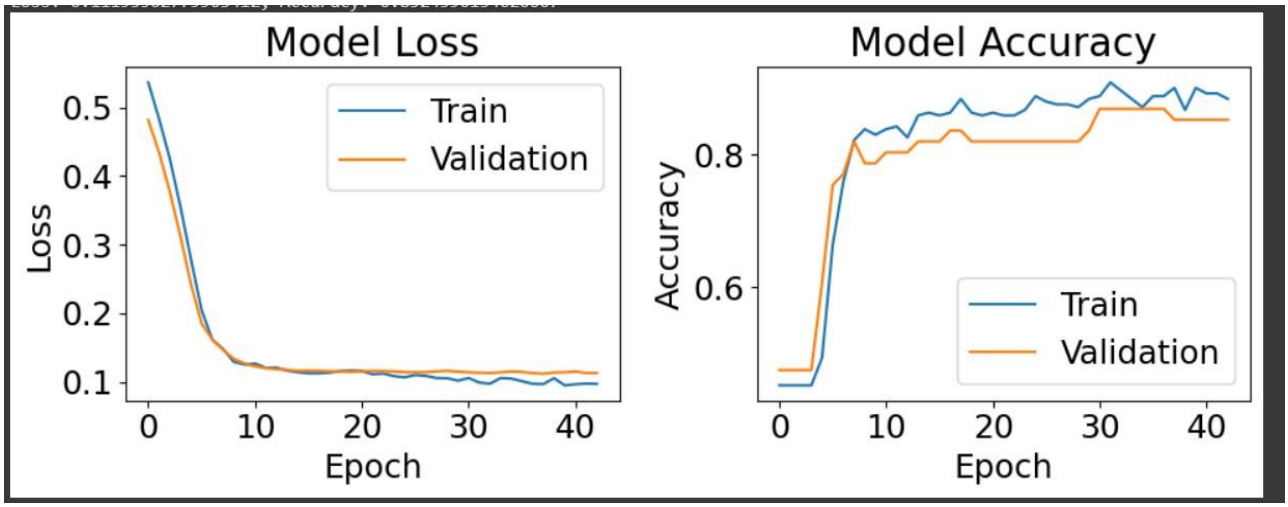

Figure 6.22: Graph of model loss and odel accuracy

Model Loss and Accuracy:

The y-axis represents the model loss and accuracy values.

The x-axis represents the number of epochs, indicating the training iterations.

Loss and Accuracy Trends:

The loss and accuracy are plotted against the number of epochs.

The plot shows how both training and validation loss and accuracy change over each epoch.

Interpretation:

As the number of epochs increases, you can observe the trend of loss decreasing and accuracy increasing.

The gap or convergence between the training and validation curves indicates overfitting or underfitting. If the validation loss starts to increase while the training loss decreases, it might suggest overfitting.

This visualization helps in monitoring the model's performance during training and validation, guiding decisions on whether to continue training, adjust hyperparameters, or stop training to prevent overfitting.

```
test_accuracy = 0.8524590134620667

# Convert the test accuracy to a percentage string
accuracy_percentage = f"{test_accuracy * 100:.2f}%"

# Print the accuracy as a percentage
print(f"Test Accuracy: {accuracy_percentage}")
```

```
Test Accuracy: 85.25%
```

Figure 6.23: Accuracy of model

This code calculates the test accuracy and converts it into a percentage string as shown in the Figure 6.23.

Test Accuracy Calculation: The variable test_accuracy stores the test accuracy value, which is approximately 0.8524590134620667.

Conversion to Percentage String: The code uses an f-string (f"{test_accuracy * 100:.2f}%") to convert the test accuracy to a percentage string with two decimal places. It multiplies the test accuracy by 100 to convert it to a percentage and formats it with two decimal places.

Printing: Finally, the code prints the test accuracy as a percentage string using another f-string (f"Test Accuracy: {accuracy_percentage}").Output: "Test Accuracy: 85.25%" This provides a clear representation of the test accuracy as a percentage.

CHAPTER 7
CONCLUSION

The utilization of deep learning approaches for the early detection of cardiac arrest presents a promising avenue for improving patient outcomes and reducing the burden of cardiovascular diseases. The proposed system harnesses the power of advanced neural network architectures, multi-modal data integration, and real-time monitoring capabilities to enable timely detection, personalized risk assessment, and proactive intervention. By continuously analyzing cardiovascular data in real-time and providing early alerts to healthcare providers or patients, the system facilitates prompt clinical response and potentially prevents adverse cardiac events. Furthermore, the system prioritizes interpretability, reliability, and scalability, ensuring its suitability for diverse healthcare settings and patient populations. Through ongoing validation, refinement, and collaboration with medical experts and stakeholders, the proposed system has the potential to revolutionize cardiac care, ultimately saving lives and improving the quality of life for individuals at risk of cardiac arrest.

REFERENCES

[1]. An-Yi Wang 1,Cheng-Chung Fang 2, Shyr-Chyr Chen 2, Shin-Han Tsai 3, Wei-Fong Kao Affiliations expand Epub 2015 Dec 24.

[2]. Yun Gi Kim # 1, Kyongjin Min # 2, Joo Hee Jeong 1, Seung-Young Roh 3, Kyung-Do Han 4, Jaemin Shim 1, Jong-Il Choi 5, Young-Hoon Kim 1 2024 Jan 27;14(1):2289.doi: 10.1038/s41598024-52859-x.

[3]. Vahid HOUSHYARIFAR, Mehdi Chehel AMIRANI,Turk J Elec Eng & Comp Sci (2017) 25: 1541 – 155③c TUB¨ ˙ITAK doi:10.3906/elk-1509-14, https:// journals.tubitak.gov.tr/ elektrik.

[4]. [4] Justin Chan, Thomas Rea, Shyamnath Gollakota & Jacob E. Sunshine npj Digital Medicine volume 2, Article number: 52 (2019).

[5]. Ryo Ueno,Liyuan Xu, Wataru Uegami,Hiroki Matsui, Jun Okui, Hiroshi Hayashi, Toru Miyajima, Yoshiro Hayashi,David Pilcher, Daryl Jones,Published online 2020 Jul 13.

[6]. Joon-Myoung Kwon 1,Youngnam Lee 2, Yeha Lee 2, Seungwoo Lee 2, Jinsik Park 2018 Jun 26;7(13):e008678. doi: 10.1161/JAHA.118.008678.

[7]. Apeksha Shah1 , Swati Ahirrao1 , Sharnil Pandya1 *, Ketan Kotecha2 and Suresh Rathod1,published: 22 October 2021 doi: 10.3389/fpubh.2021.762303.

[8]. Minsu ChaeHyo-Wook GilNam-Jun Cho Hwamin Lee,Mathematics 2022, 10(12), 2049; https://doi.org/10.3390/math10122049 Submission received: 2 May 2022 / Revised: 1 June 2022 / Accepted: 9 June 2022 / Published: 13 June 2022.