## **HEART FAILURE DETECTION THROUGH SMOTE FOR AUGMENTAION AND MACHINE LEARNING APPROACH FOR CLASSIFICATIONS**

## *Submitted to Jawaharlal Nehru Technological University, Kakinada*

## *in partial fulfillment of requirement for the degree of*

**Bachelor of Technology**

In

**ELECTRONICS AND COMMUNICATION ENGINEERING**

*Submitted by*

**N SAI**

(20KN1A04B5)

Under the guidance of

**Mrs. G SREE LAKSHMI M.Tech**

**Associate Professor**



**DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING**

**NRI INSTITUTE OF TECHNOLOGY**

**Autonomous**

**Approved by AICTE, New Delhi, Permanently Affiliated to JNTUK, Kakinada Accredited by NBA (ECE, CSE &EEE), Accredited by NAAC with 'A' Grade ISO 9001: 2015 Certified Institution**

**Pothavarappadu (V), Agiripalli (M), Eluru (Dt)-521212**

**2020-2024**

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO.** | **TITLE** | **PAGE NO.** |
|  | **ABSTRACT** | **i** | |
|  | **LIST OF FIGURES** | **ii-iii** | |
|  | **LIST OF TABLES** | **iv** | |
|  | **LIST OF ACRONYMS & ABBREVATIONS** | **v-vi** | |
| **CHAPTER 1** | **INTRODUCTION** | **1-4** | |
|  | 1.1 INTRODUCTION | 1 | |
|  | 1.2 MOTIVATION  1.3OBJECTIVE  1.4PROBLEM STATEMENT | 2  3  4 | |
| **CHAPTER 2** | **LITERATURE SURVEY** | **5-8** | |
|  | 2.1 LITERATURE SURVEY | 5 | |
|  | 2.2 EXISTING SYSTEM | 8 | |
|  | 2.3 DISADVANTAGES IN EXISTING SYSTEM | 8 | |
| **CHAPTER 3** | **PROPOSED SYSTEM & OVERVIEW** | **9-14** | |
|  | 3.1 DATA COLLECTION | 9 | |
|  | 3.2 DATA PREPROCESSING | 14 | |

|  |  |  |
| --- | --- | --- |
|  | 3.3 FEATURE EXTRACTION | 14 |
|  | 3.4 CLASSIFICATION | 14 |
| **CHAPTER 4** | **SOFTWARE USED** | **15-22** |
|  | 4.1 HISTORY OF PYTHON | 15 |
|  | 4.1.1 OVERVIEW OF PYTHON | 16 |
|  | 4.2 PYCHARM COMMUNITY | 17 |
|  | 4.2.1 OVERVIEW OF PYCHARM COMMUNITY | 18 |
|  | 4.3 LIBRARIES USED | 19 |
|  | 4.3.1 OPEN CV | 20 |
|  | 4.3.2 MATH | 22 |
| **CHAPTER 5** | **CODE AND RESULTS** | **23-32** |
|  | 5.1 CODE | 23 |
|  | 5.1.1 MAIN CODE | 23 |
|  | 5.1.2 MODEL EVALUTION CODE | 26 |
|  | 5.2 RESULTS | 27 |
| **CHAPTER 6** | **ADVANTAGES AND APPLICATIONS** | **33** |
|  | 6.1 ADVANTAGES | 33 |
| **CHAPTER 7** | **CONCLUSION** | **34-35** |
|  | **REFERENCES** | **36-38** |

**ABSTRACT**

Chronic heart failure represents a widespread global health challenge, necessitating innovative approaches for early detection and management. While pharmaceutical interventions play a pivotal role, there is a growing recognition of the adjunctive benefits of exercise in addressing this condition. In this study, we implement the Synthetic Minority Over-sampling Technique (SMOTE) to augment our dataset and harness a comprehensive suite of machine learning algorithms, including XG Boost, k-Nearest Neighbors (KNN), Adaboost and Support Vector Machines (SVM), to enhance the model's efficacy in early heart failure detection. The rigorous validation process through cross-validation techniques underscores the paramount significance of this research in the medical field. By enhancing our capacity to identify heart failure at its incipient stages, this study holds the potential to save lives by enabling timely interventions. It underscores the promising role of machine learning in advancing healthcare and highlights the critical importance of early detection and intervention in managing this pervasive global health issue. Chronic heart failure demands multifaceted solutions, and this research represents a significant stride in the quest for improved detection and management. By integrating machine learning techniques and acknowledging the role of exercise in therapy, this study offers a comprehensive approach to address this pressing health concern and paves the way for a more proactive and effective response to chronic heart failure on a global scale.

**Keywords:** Synthetic Minority Over-sampling Technique, machine learning algorithms, heart failure, cross-validation, healthcare

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIG.NO** | **NAME OF THE FIGURE** | **PAGE NO** |
| 3.1.0 | Proposed methodology for Heart failure prediction | 4 |
| 3.1.1 | Feature Importance | 10 |
| 3.1.2 | Label Distribution | 10 |
| 3.1.3 | Bixplot Misclassification Rates | 11 |
| 3.1.4 | Barely true | 11 |
| 3.1.5 | Pants on fire counts | 12 |
| 3.1.6 | False Counts | 12 |
| 3.1.7 | Half True Counts | 13 |
| 3.1.8 | Mostly true counts | 13 |
| 4.1.1 | Photograph of Guido van Rossum | 15 |
| 4.1.1.2 | Python versions and release dates | 17 |
| 4.2.1.1 | Logo of PyCharm Community Edition | 18 |
| 4.3.1.1 | Open-CV Logo | 20 |
| 5.2.1 | Confusion Matrix for KNN | 28 |
| 5.2.2 | Confusion Matrix for Random Forest | 28 |
| 5.2.3 | Confusion Matrix for SVM | 28 |
| 5.2.4 | Confusion Matrix for proposed M L Algorithms | 29 |
| 5.2.5 | Receiver Operating Characteristic-KNN | 29 |

|  |  |  |
| --- | --- | --- |
| 5.2.6 | Receiver Operating Characteristic-Random Forest | 30 |
| 5.2.7 | Receiver Operating Characteristics -SVM | 30 |
| 5.2.8 | ROC Curve for Existing Machine Learning Algorithms | 30 |
| 5.3.1 | Performance Analysis. | 31 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **NAME OF THE TABLE** | **PAGE NO** |
| 4.2.1.1 | Python versions and release dates | 18 |

5.2.9 Performance Analysis for heart failure Model 31

**LIST OF ACRONYMS**

|  |  |  |
| --- | --- | --- |
| 1. | WHO | – World Health Organization |
| 2. | HRV | – Heart rate variability |
| 3. | CHF | – congestive heart failure |
| 4. | ECG | – Electro cardio grapy |
| 5. | PCHF | – Principal Component Heart Failure |
| 6. | MFDFA | – multi-fractal detrended fluctuation analysis |
| 7. | Po-c | – Point-of-Care |
| 8. | EHR | – Electronic Health Record |
| 9. | LR | – Logistic Regression |
| 10. | SVMs | – Support Vector Machines |
| 11. | RF | – Random Forest |
| 12. | GB | – Gradient Boost |
| 13. | SMOTE | – Synthetic Minority Over-sampling Technique |
| 14. | SVM | – Support Vector Machine |
| 15. | KNN | – k-Nearest Neighbours |
| 16. | ML | – Machine Learning |
| 17. | DL | – Deep Learning |
| 18. | CNN | – Convolutional neural network |
| 19. | RNN | – recursive neural network |

|  |  |  |
| --- | --- | --- |
| 20. | SMCG | – smartphone-based mechanoid cardiography |
| 21. | AFib | – atrial fibrillation |
| 22. | ADHF | – acute decompensated heart failure |
| 23. | ROC | \_ receiver operating characteristic curve |
| 24. | HVDs | – heart-valve-related diseases |
| 25. | PCG | – phonocardiogram |
| 26. | VHD | – valvular heart diseases |
| 27. | CWT | – continuous wavelet transform |
| 28. | BDFL | –Benevolent dictator for life |
| 29. | CWN | – Centrum Wickenden & Informatica |
| 30. | RELU | – Rectified Linear Unit |
| 31. | Open-CV | – Open-Source Computer Vision Library |
|  |  |  |
|  |  |  |

**CHAPTER-1**

**INTRODUCTION**

**1.1 Introduction**

Heart failure is a significant global health concern, as emphasized by reports from the World Health Organization (WHO). This medical condition is characterized by the heart's inability to effectively pump blood, resulting in a range of debilitating symptoms and complications. The WHO reports highlight that heart failure affects millions of individuals across the world, and its prevalence is increasing, particularly in aging populations. The impact of heart failure extends beyond individual health, straining healthcare systems with elevated hospitalizations, rising healthcare costs, and decreased patient quality of life. The condition is associated with a high mortality rate, underscoring the critical importance of early detection and effective management in reducing its global burden [1]. Heart failure, a pervasive chronic disease affecting a substantial global population, necessitates early detection and intervention to mitigate its impact and improve patient outcomes. While pharmaceutical treatments play a central role, the recognition of exercise as a valuable adjunct therapy in heart failure management is gaining prominence. In response to this pressing healthcare challenge, this study focuses on advancing heart failure detection through a machine learning-based approach, leveraging patient health parameter data. Early detection is vital, with the potential to save lives and reduce the burden of this worldwide issue. Our research encompasses the utilization of nine machine learning algorithms and introduces the innovative Principal Component Heart Failure (PCHF) feature engineering technique, strategically designed to enhance performance by selecting the most crucial features [2].

Heart rate variability (HRV) serves as a valuable method for the detection and assessment of congestive heart failure (CHF). Previous research on HRV has predominantly focused on linear and nonlinear indicators derived from ECG signal RR intervals. However, this article introduces a novel sequence that offers insight into the sympathetic and parasympathetic nervous system's influence on heart rate regulation. Leveraging multi-fractal detrended fluctuation analysis (MFDFA), the study quantitatively evaluates the complexity of this new sequence to discern differences between healthy individuals and those with CHF. The results shed light on how disruptions in autonomic nerve control due to physiological and pathological factors can lead to a reduction in the complexity of heart rate signals, enhancing our understanding of CHF-related alterations in HRV [3]. The diagnosis and management of Heart Failure (HF) represent on-going challenges for healthcare systems globally, particularly with the growing number of individuals in aging populations surviving cardiac conditions but still facing residual heart function impairment. To address this critical issue, the implementation of Point-of-Care (PoC) measurement of N-terminal pro-B-type natriuretic peptide (NT-proBNP) emerges as a potential game-changer. NT-proBNP, released in response to cardiac wall stretch, allows for serial measurements that can serve as a valuable indicator of HF progression or deterioration, thereby assessing the effectiveness of HF interventions [4].

Heart failure as a potential consequence of cancer treatments has emerged as a significant concern in the realm of oncology and cardiology. Early detection and prediction of cardiotoxicity in cancer patients are of utmost importance to mitigate the risks associated with heart failure. This study focuses on the integration of genetic data and Electronic Health Record (EHR) data to identify cancer patients at a high risk of developing treatment-related heart failure. By employing four machine learning models, including Logistic Regression (LR), Support Vector Machines (SVMs), Random Forest (RF), and Gradient Boost (GB), we aim to harness the power of data-driven approaches to enhance predictive accuracy. Our investigation delves into various strategies for combining genetic and EHR data, ultimately striving to provide a comprehensive and effective solution for the early identification of cancer patients susceptible to cardio toxicity [5]. The accurate detection of arrhythmias holds paramount importance in clinical settings, as it serves as a crucial indicator of acute and chronic cardiac conditions, significantly impacting patient well-being. However, due to inherent variability among individuals and the presence of inevitable noise, this task is challenging even for seasoned medical experts. This paper embarks on an exploration of the potential of deep neural networks, specifically recurrent and residual architectures, in the classification of ECG recordings. The study leverages a dataset comprising ECG readings from 162 patients, encompassing three distinct classes: normal sinus rhythm, cardiac arrhythmia, and congestive heart failure [6].

The heart, a vital organ in the human body, serves the essential function of pumping blood through its intricate network of vessels, muscles, and valves. Heart disease encompasses a range of conditions that affect this intricate system, including the development of heart failure, a prevalent and growing health concern. Heart failure signifies the heart's inability to meet the body's demand for blood, although it doesn't necessarily mean a complete cessation of pumping action [7]. Heart failure signifies the heart's inability to meet the body's demand for blood, although it doesn't necessarily mean a complete cessation of pumping action. With the increasing incidence of heart failure among both older and younger individuals, early detection and diagnosis have become pivotal in mitigating the associated risks. Various factors, including underlying diseases like diabetes, can elevate the risk of heart failure, making machine learning models a valuable tool for assessing this risk based on patient data [8].

**1.2 Motivation**

The urgency of addressing chronic heart failure as a global health challenge drives the motivation behind this ground-breaking study. Recognizing the need for innovative approaches, the research focuses on combining pharmaceutical interventions with the often-overlooked benefits of exercise in managing the condition. The introduction of Synthetic Minority Over-sampling Technique (SMOTE) to augment the dataset and a suite of advanced machine learning algorithms reflects a commitment to enhancing early detection efficacy. The thorough validation process using cross-validation techniques emphasizes the significance of this research in the medical field. The ultimate motivation lies in the potential to save lives through timely interventions enabled by improved identification of heart failure at its incipient stages. This study not only showcases the promising role of machine learning in healthcare advancement but also highlights the critical importance of early detection and intervention in addressing this pervasive global health issue. The multifaceted approach presented in this research represents a substantial stride toward improved detection and management, offering a comprehensive and proactive response to chronic heart failure on a global scale.

**1.3 Objective**

The objective of this study is to address the global health challenge of chronic heart failure by employing an innovative approach that combine pharmaceutical interventions with the adjunctive benefits of exercise. Utilizing the Synthetic Minority Over-sampling Technique (SMOTE) to augment the dataset and employing advanced machine learning algorithms, including XG Boost, k-Nearest Neighbors (KNN), Adaboost, and Support Vector Machines (SVM), the goal is to enhance early detection efficacy. Through rigorous validation using cross-validation techniques, the study aims to underscore the significance of its findings in advancing medical knowledge. The ultimate objective is to contribute to saving lives by enabling timely interventions and advocating for a more proactive response to chronic heart failure on a global scale.

**1.4 Problem Statement**

**Global Health Challenge***:* Chronic heart failure poses a significant and widespread global health challenge, necessitating urgent attention due to its pervasive impact on populations worldwide.

**Limited Early Detection Tools*:*** Current methodologies for detecting heart failure at its incipient stages are limited, leading to delayed interventions and a missed opportunity for timely medical care, emphasizing the need for more effective diagnostic tools.

**Holistic Solution Integration*:*** Recognizing the multifaceted nature of chronic heart failure, there is a gap in integrating both pharmaceutical interventions and the beneficial role of exercise in therapy. This study aims to address this gap by employing advanced machine learning techniques to enhance early detection efficacy and contribute to a more comprehensive approach in managing this pressing health concern

**CHAPTER 2**

**LITERATURE SURVEY**

**2.1 Literature Survey**

“**Machine Learning and End-to-End Deep Learning for the Detection of Chronic Heart Failure From Heart Sounds**”, ]. M. Gjoreski, A. Gradišek, B. Budna, M. Gams and G. Poglajen

M. Gjoreski, et.al, [9], chronic heart failure (CHF) detection, combining classic Machine Learning (ML) and end-to-end Deep Learning (DL) techniques, has shown promising results. With an impressive aggregated accuracy of 92.9%, it outperforms the recent PhysoNet challenge's baseline method by a significant margin, underlining its potential as a valuable tool in CHF diagnosis. Additionally, the identification of 15 expert features for distinguishing between CHF phases with an accuracy of 93.2% highlights its versatility. These findings offer hope for more accessible CHF patient identification and the development of home-based monitoring solutions, potentially reducing hospitalizations and improving the management of this widespread medical condition

"**Automatic Detection of Congestive Heart Failure Based on a Hybrid Deep Learning Algorithm in the Internet of Medical Things”** W. Ning, S. Li, D. Wei, L. Z. Guo and H. Chen

W. Ning, et.al, [10], highly effective automatic congestive heart failure (CHF) detection model, utilizing a hybrid deep learning approach involving a convolutional neural network (CNN) and a recursive neural network (RNN). The outstanding accuracy of 99.93%, sensitivity of 99.85%, and specificity of 100% for 5-minute ECG signal analysis surpass previous research efforts, marking a significant advancement. Furthermore, our investigation into CHF patient detection using ultra short-term ECG signals yielded excellent results. This hybrid deep learning algorithm offers an objective and accurate means of classifying CHF signals, demonstrating its potential as a valuable clinical tool for CHF patient detection. It holds promise for enhancing early diagnosis and management of this chronic heart condition, ultimately contributing to improved patient outcomes and safety.

"**Automatic Detection of Congestive Heart Failure Based on Multiscale Residual UNet++: From Centralized Learning to Federated Learning**," L. Zou, Z. Huang, X. Yu, J. Zheng, A. Liu and M. Lei,

L. Zou, Z. Huang et.al, [11], The model demonstrates its robustness by outperforming the state of the art in both centralized and decentralized learning settings, with accuracies of 89.83% and 87.54%, respectively. The potential of our federated framework for multisite data utilization while preserving patient privacy opens new avenues for improving CHF detection without compromising data security. This research has the potential to revolutionize CHF diagnosis and pave the way for collaborative advancements in healthcare.

"**Classification of Atrial Fibrillation and Acute Decompensated Heart Failure Using Smartphone Mechanocardiography: A Multilabel Learning Approach**” S. Mehrang et al

S. Mehrang et al., [12], the promising potential of smartphone-based mechanocardiography (sMCG) for the concurrent detection of atrial fibrillation (AFib) and acute decompensated heart failure (ADHF) in hospitalized cardiac patients. Leveraging supervised machine learning with multi-label and hierarchical classification approaches, we achieved remarkable results. The high area under the receiver operating characteristic curve (ROC AUC) values of 0.98 for AFib and 0.85 for ADHF underscore the accuracy of our method.

"**Congestive Heart Failure Detection From ECG Signals Using Deep Residual Neural Network**” E. Prabhakararao and S. Dandapat

E. Prabhakararao and S. Dandapat "Congestive Heart Failure Detection From ECG Signals Using Deep Residual Neural Network., [13], congestive heart failure (CHF) detection in ECG data. By effectively capturing the temporal dynamics and leveraging attentive feature extraction, the DA-DRRNet significantly advances the accuracy of CHF diagnosis. Remarkably, it achieves an impressive accuracy of 98.57% at the beat-level and nearly 100% for 24-hour record-level diagnosis, highlighting its superior performance. Moreover, the model's transparency is a notable asset, as it elucidates the ECG characteristics that are crucial for CHF identification, offering interpretability in a field often plagued by opacity.

“**SAR-Cardio Net: A Network for Heart Valve Disease Detection From PCG Signal Based on Split-Self Attention With Residual Paths**”, M. Morshed and S. A. Fattah

M. Morshed et al., [14], deep learning network for the direct detection of heart-valve-related diseases (HVDs) from phonocardiogram (PCG) signals. Leveraging a novel split-self attention mechanism and multipath feature extractors, our model significantly improves accuracy and performance. The integration of attention blocks at deeper convolutional layers and the inclusion of two residual paths mitigates overfitting and gradient issues, enhancing the network's reliability. Our extensive experiments using publicly available datasets yielded impressive results, with accuracy rates of 96.37% and 99.25% and excellent average area under the curve values of 98% and 100%. Comparative analysis with existing models underscores the competitiveness of our network.

“**Explainable Deep Convolutional Neural Network for Valvular Heart Diseases Classification Using PCG Signals**," . A. Bhardwaj, S. Singh and D. Joshi Bhardwaj, S. Singh and D. Joshi “Explainable Deep Convolutional Neural Network for Valvular Heart Diseases Classification Using PCG Signals,"., [15], a significant advancement in the early detection of valvular heart diseases (VHD) using phonocardiogram (PCG) signals. By leveraging analytic continuous wavelet transform (CWT) scalograms and a specially designed 2-D convolutional neural network (CNN), we achieved remarkable classification accuracy, with the highest accuracy reaching 99.6% in fivefold cross-validation and an overall accuracy of 98.32% on a publicly available PCG database. Notably, our method also demonstrated a strong performance in binary classification, distinguishing between abnormal and normal cases with an accuracy of 93.07% on the PhysioNet database.

"**Detection of Heart Failure Using a Convolutional Neural Network via ECG Signals**," J. Botros, F. Mourad-Chehade and D. Laplanche.

J. Botros, , et al., [16], The model's strength lies in its simplicity, as it requires minimal ECG signal pre-processing and eliminates the need for engineered features, making it a valuable tool for efficient and accessible HF diagnosis. The model's exceptional performance is evident, with an accuracy of 99.73%, sensitivity of 99.58%, and specificity of 99.83% on unbalanced datasets, and an accuracy of 99.26%, sensitivity of 99.37%, and specificity of 99.12% on balanced datasets. These results underscore its potential as a reliable and robust diagnostic tool for HF, promising to enhance early detection, reduce healthcare costs, and ultimately improve patient outcomes.Top of Form

**2.2 EXISTED METHOD: Support Vector Machine:** Predicting heart disease, the Support Vector Machine (SVM) algorithm follows a systematic process. Begin by assembling a dataset with relevant features and corresponding labels denoting the presence or absence of heart disease. After splitting the dataset into training and testing sets, employ feature scaling to normalize the features. Choose a suitable kernel function and specify SVM parameters, including the regularization parameter (C) and kernel-specific parameters. Train the SVM model on the training dataset and evaluate its performance using metrics

**Random Forest algorithm:** The Random Forest algorithm, a versatile ensemble learning method, is widely employed in heart disease prediction. The process involves assembling a dataset with relevant features and corresponding labels indicating the presence or absence of heart disease. The dataset is then split into training and testing sets. Multiple decision trees are created using subsets of the training data and a random selection of features. The trees vote" on the prediction, and the algorithm aggregates these votes for the final outcome. After training the Random Forest model, its performance is assessed using metrics.

**K-Nearest Neighbour (KNN):**The k-Nearest Neighbours (KNN) algorithm, commonly used for heart disease prediction, follows a distinct approach. Begin by preparing a dataset with features and corresponding labels indicating heart disease presence or absence. Split the data into training and testing sets and apply feature scaling for uniformity. Choose the value of k, the number of neighbours considered during classification. Train the KNN model on the training dataset, and subsequently, evaluate its performance on the testing set using metrics

**2.3 Disadvantages of Existing Model:**

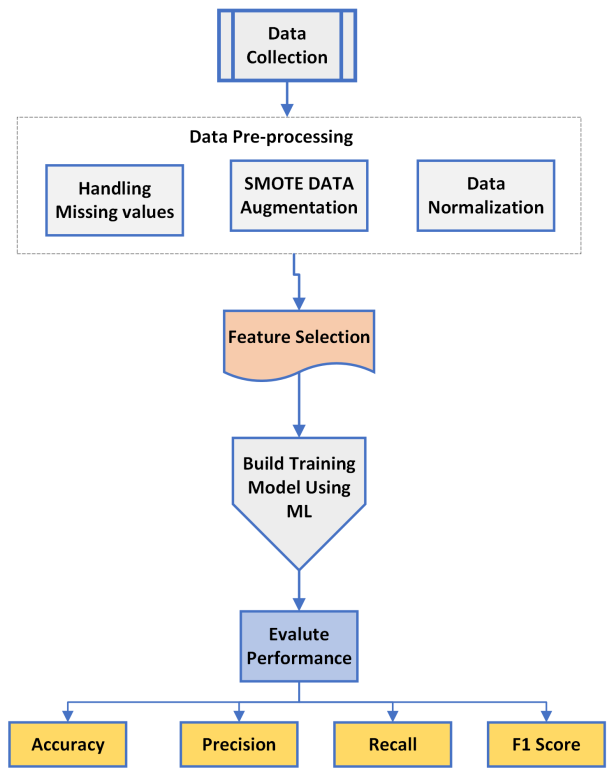
1. Many machine learning models struggle with imbalanced datasets

2. it’s important to assess the model's generalization to unseen data and discuss measures taken to prevent over fitting

**CHAPTER 3**

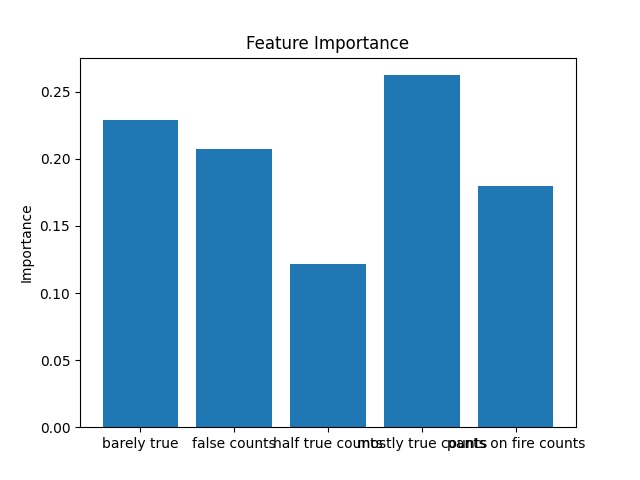
**PROPOSED METHODOLOGY**

Heart failure detection utilizes the Synthetic Minority Over-sampling Technique (SMOTE) for data augmentation, coupled with a machine learning classification approach. Various algorithms, including logistic regression, decision trees, random forests, support vector machines, and neural networks, are employed. SMOTE addresses class imbalance by generating synthetic instances of the minority class, thereby enhancing the model's ability to recognize heart failure cases. The purpose of the SMOTE approach is to improve the segmentation of the minority class, ensuring that the machine learning model is exposed to a more balanced representation of both positive and negative instances. This, in turn, augments the overall robustness and accuracy of the heart failure detection system.

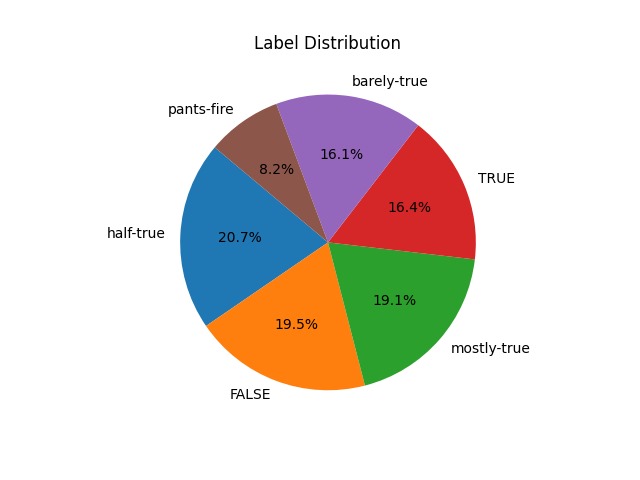


**Figure 1: Proposed methodology for Heart failure prediction**

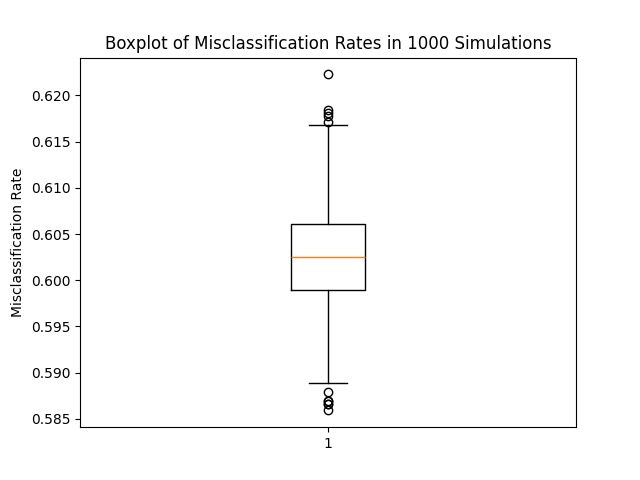
**3.1 Data Collection:** The initial phase in heart failure detection involves comprehensive data collection. Various patient-related variables, including demographic details, medical history, and lifestyle factors, are gathered to construct a dataset. This dataset aims to encompass a diverse range of instances, ensuring a representative sample for training and testing the machine learning model. Additionally, the incorporation of Synthetic Minority Over-sampling Technique (SMOTE) facilitates addressing potential class imbalances by generating synthetic instances of the minority class, contributing to a more balanced dataset.



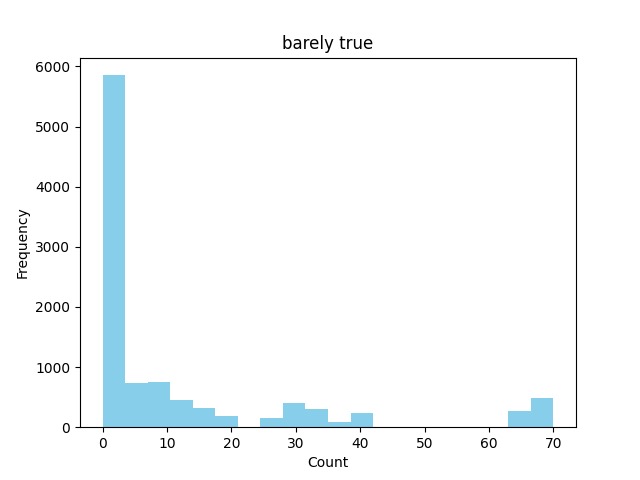
**Fig3.1.1 Feature Importance**



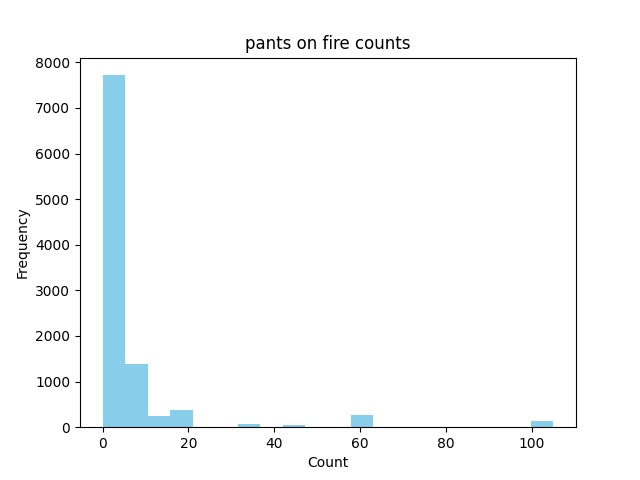
**Fig 3.1.2 Label Distribution**



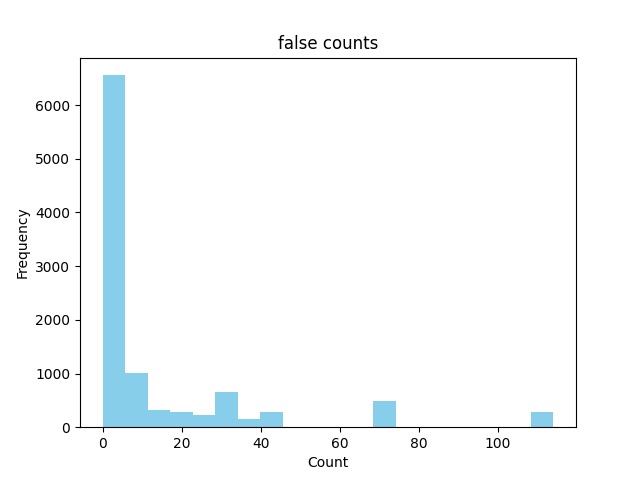
**Fig 3.1.3 Bixplot Misclassification Rates**



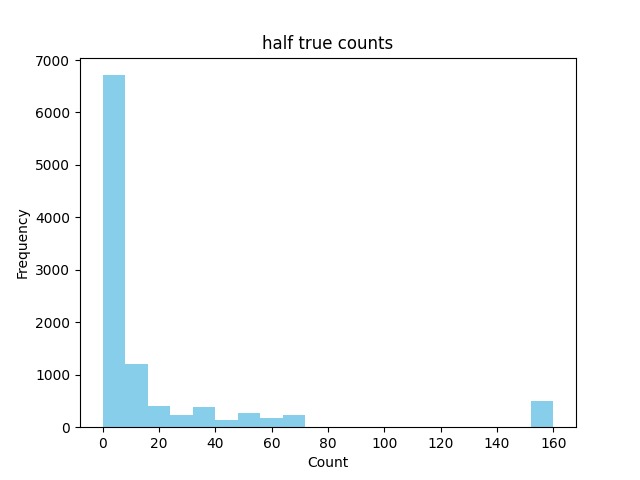
**Fig 3.1.4 Barely true**



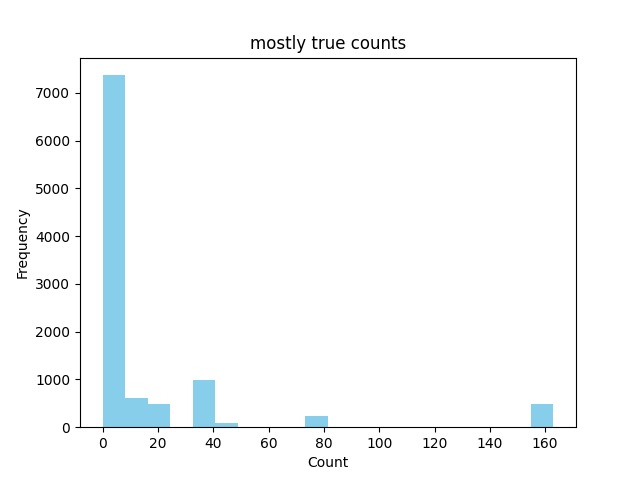
**Fig 3.1.5 Pants on fire counts**



**Fig 3.1.6 False Counts**



**Fig 3.1.7 Half true counts**



**Fig 3.1.8 Mostly true counts**

**3.2 Data Pre-processing:** Following data collection, the dataset undergoes meticulous pre-processing to ensure its suitability for machine learning algorithms. This step involves handling missing or erroneous data, normalizing numerical features, and encoding categorical variables. The integration of SMOTE during this phase assists in mitigating the effects of class imbalance by artificially expanding the representation of the minority class. The pre-processed dataset is then divided into training and testing sets to facilitate model evaluation and validation.

**3.3 Feature Extraction:** Feature extraction plays a critical role in refining the dataset for optimal model performance. This step involves selecting relevant variables and leveraging SMOTE to augment the representation of the minority class. By improving the segmentation of the minority class, the feature extraction process ensures that the machine learning model is exposed to a more balanced distribution of both positive and negative instances. This enhances the model's ability to recognize and accurately classify heart failure cases.

**3.4 Classification:** The final phase involves applying various machine learning classification algorithms, such as logistic regression, decision trees, random forests, support vector machines, and neural networks. The model is trained using the pre-processed and augmented dataset, with hyper parameters adjusted to optimize performance. Subsequently, the model's accuracy, precision, recall, and other relevant metrics are evaluated using the testing set. The integration of SMOTE, from data pre-processing to feature extraction, contributes to a more robust and accurate heart failure detection system by addressing class imbalances and improving the overall model's .

**CHAPTER 4**

**SOFTWARE USED**

This project has been implemented in Python and PYCHARM COMMUNITY (execution platform) software. Here few details regarding the software are provided.

**4.1 History of Python**

Python is a widely-used general-purpose, high-level programming language. It was initially designed by Guido van Rossum in 1991 and developed by Python Software Foundation which is explained in [14]. It was mainly developed to emphasize code readability, and its syntax allows programmers to express concepts in fewer lines of code.



**Fig: 4.1.1 Photograph of Guido van Rossum**

In the late 1980s, history was about to be written. It was that time when working on Python started. Soon after that, Guido Van Rossum began doing its application-based work in December of 1989 at Centrum Wiskunde& Informatica (CWI) which is situated in the **Netherlands**. It was started firstly as a hobby project because he was looking for an interesting project to keep him occupied during Christmas. The programming language in which Python is said to have succeeded is **ABC Programming Language**, which had interfacing with the **Amoeba Operating System** and had the feature of exception handling. He had already helped to create ABC earlier in his career and he had seen some issues with ABC but liked most of the features. After that what he did was very clever. He had taken the syntax of ABC, and some of its good features. It came with a lot of complaints too, so he fixed those issues completely and created a good scripting language that removed all the flaws. The inspiration for the name came from BBC's TV Show – '**Monty Python’s Flying Circus**’, as he was a big fan of the TV show and also he wanted a short, unique and slightly mysterious name for his invention and hence he named it Python! He was the **“Benevolent dictator for life”** (BDFL) until he stepped down from the position as the leader on 12th July 2018. For quite some time he used to work for Google, but currently, he is working at Dropbox. . It is not intended to work on special area such as web programming. That is why it is known as multipurpose because it can be used with web, enterprise, 3D CAD etc. We don't need to use data types to declare variable because it is dynamically typed so we can write a=10 to assign an integer value in an integer variable. It makes the development and debugging fast because there is no compilation step included in python development and edit-test-debug cycle is very fast

**4.1.1 Overview of Python**

Python is a programming language [15]. The language was finally released in 1991. When it was released, it used a lot fewer codes to express the concepts, when we compare it with Java, C++ & C. Its design philosophy was quite good too. Its main objective is to provide code readability and advanced developer productivity. When it was released, it had more than enough capability to provide classes with inheritance, several core data types exception handling, and function. The evolution of python versions is as follows.

|  |  |
| --- | --- |
| **Python Version** | **Released Date** |
| Python 1.0 | January 1994 |
| Python 1.5 | December 31, 1997 |
| Python 1.6 | September 5, 2000 |
| Python 2.0 | October 16, 2000 |
| Python 2.1 | April 17, 2001 |
| Python 2.2 | December 21, 2001 |
| Python 2.3 | July 29, 2003 |
| Python 2.4 | November 30, 2004 |
| Python 2.5 | September 19, 2006 |
| Python 2.6 | October 1, 2008 |
| Python 2.7 | July 3, 2010 |
| Python 3.0 | December 3, 2008 |
| Python 3.1 | June 27, 2009 |
| Python 3.2 | February 20, 2011 |
| Python 3.3 | September 29, 2012 |
| Python 3.4 | March 16, 2014 |
| Python 3.5 | September 13, 2015 |
| Python 3.6 | December 23, 2016 |
| Python 3.7 | June 27, 2018 |
| Python 3.8 | October 14, 2019 |

**Table : 4.1.1.2 Python versions and release dates**

**4.2 PYCHARM COMMUNITY**

PyCharm is a dedicated Python Integrated Development Environment (IDE) providing a wide range of essential tools for Python developers, tightly integrated to create a convenient environment for productive [Python](https://www.jetbrains.com/help/pycharm/python.html), [web](https://www.jetbrains.com/help/pycharm/web-frameworks.html), and [data science](https://www.jetbrains.com/help/pycharm/scientific-tools.html) development.

**PyCharm is available in two editions:**

**Community (free and**[**open-sourced**](https://github.com/JetBrains/intellij-community/blob/master/LICENSE.txt)**):** for smart and intelligent Python development, including coding assistance, refactorings, visual debugging, and version control integration.

**Professional (**[**paid**](https://www.jetbrains.com/pycharm/buy/#commercial?billing=yearly)**) :** for professional Python, web, and data science development, including coding assistance, refactorings, visual debugging, version control integration, remote configurations, deployment, support for popular web frameworks, such as Django and Flask, database support, scientific tools (including Jupyter notebook support), big data tools.

**Supported languages﻿:** To start developing in Python with PyCharm you need to download and install Python from [python.org](http://www.python.org/) depending on your platform.

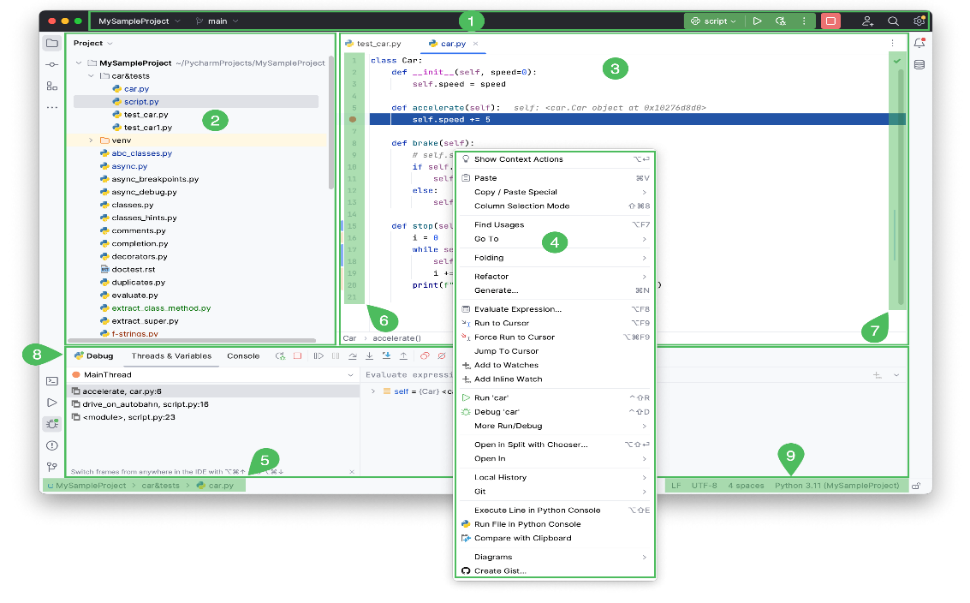
PyCharm supports the following versions of Python:

* **Python 2:** version 2.7
* **Python 3:** from the version 3.6 up to the version 3.12
* Besides, in the Professional edition, one can develop Django , Flask, and Pyramid applications. Also, it fully supports HTML (including HTML5), CSS, JavaScript, and XML: these languages are bundled in the IDE via plugins and are switched on for you by default. Support for the other languages and frameworks can also be added via plugins (go to Settings | Plugins or PyCharm | Settings | Plugins for macOS users, to find out more or set them up during the first IDE launch).



**Fig: 4.2.1 Logo of PyCharm Community Edition**

**4.2.1 Overview of PyCharm Community Edition**

****

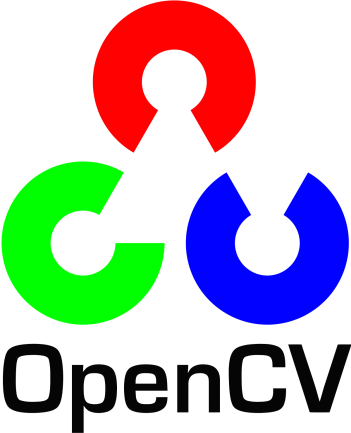
**Fig:4.2.1.1 Interface of PyCharm Community Edition**

1. [**Project tool window**](https://www.jetbrains.com/help/pycharm/project-tool-window.html) **:** on the left side displays your project files.
2. [**Editor**](https://www.jetbrains.com/help/pycharm/using-code-editor.html) **:** on the right side, where you actually write your code. It has tabs for easy navigation between open files.
3. [**Navigation bar**](https://www.jetbrains.com/help/pycharm/guided-tour-around-the-user-interface.html#navigation-bar) **:**allows you to quickly navigate the project folders and files.
4. **Gutter :** the vertical stripe next to the editor, shows the breakpoints you have, and provides a convenient way to [navigate through the code](https://www.jetbrains.com/help/pycharm/navigating-through-the-source-code.html) hierarchy like going to definition/declaration. It also shows line numbers and per-line VCS history.
5. **Scrollbar :** on the right side of the editor. PyCharm constantly monitors the quality of your code by running [code inspections](https://www.jetbrains.com/help/pycharm/code-inspection.html). The indicator in the top right-hand corner shows the overall status of code inspections for the entire file.
6. [**Tool windows**](https://www.jetbrains.com/help/pycharm/tool-windows.html) **:** are specialized windows attached to the bottom and the sides of the workspace. They provide access to typical tasks such as project management, source code search and navigation, integration with version control systems, running, testing, debugging, and so on.
7. [**The status bar**](https://www.jetbrains.com/help/pycharm/guided-tour-around-the-user-interface.html#status-bar) **:** indicates the status of your project and the entire IDE, and shows various warnings and information messages like file encoding, line separator, inspection profile, and so on. It also provides quick access to the Python interpreter settings.

**4.3 Libraries Used**

**4.3.1 Open-CV**

OpenCV (Open-Source Computer Vision Library) as in [18] is a Python library that allows you to perform image processing and computer vision tasks. It provides a wide range of features, including object detection, face recognition, and tracking. The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms.



**Fig:4.3.1.1 Open-CV Logo**

These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high-resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. OpenCV has more than 47 thousand people in the user community and an estimated number of downloads exceeding 18 million. The library is used extensively by companies, research groups, and governmental bodies. Built-in functions or methods that are used in the code are as mentioned below with descriptions.

**1.cv2. imread():-** This method loads an image from the specified file. If the image cannot be read (because of the missing file, improper permissions, unsupported or invalid format) then this method returns an empty matrix.

2. **cv2. putText():** This method is used to draw a text string on any image.

**Cv2.dnn.blobfromimage** : The cv2. dnn. blobFromImage function returns a blob which is our input image after mean subtraction, normalizing, and channel swapping. The only exception is that we can pass in multiple images, enabling us to batch-process a set of images

1. **cv2. rectangle():-** This method is used to draw a rectangle on any image.
2. **cv2.waitkey** :- This function of Python OpenCV allows users to display a window for a given millisecond or until any key is pressed. It takes tiin milliseconds as a parameter and waits for the given time to destroy the window, if 0 is passed in the argument it waits till any key is pressed.
3. **cv2**.dnn,readNet():-It is used to load the Deep Learning network to the system when the OpenCV library is used in Python 3.

The code utilizes several Python libraries for data analysis, machine learning, and graphical visualization. Here is a brief overview of the libraries used:

1. **OS:** This is a standard Python library that provides a way to interact with the operating system, allowing the creation of directories and checking if they exist.

2**. tkinter:** A standard GUI (Graphical User Interface) toolkit for Python. In this code, it is used to create a simple GUI for uploading a CSV file.

3.**filedialog (from tkinter):** This module provides dialogs to open and save files. In this code, it is used to open a file dialog for selecting a CSV file.

4. **Button (from tkinter):** A tkinter widget used to create a button in the GUI.

5. **Label (from tkinter):** A tkinter widget used to create a label in the GUI.

6. **imblearn:** This library provides tools for dealing with imbalanced datasets. In this code, the `SMOTE` (Synthetic Minority Over-sampling Technique) method is used to oversample the minority class.

7. **sklearn:** This is a comprehensive machine learning library. Key modules used include:

- Logistic Regression: Implements logistic regression, a classification algorithm.

- train test split: Splits the dataset into training and testing sets.

- Standard Scaler: Standardizes the dataset by scaling features.

- classification\_0report: Generates a classification report with precision, recall, and F1-score.

- confusion matrix: Computes a confusion matrix for classification problems.

- roc curve and auc: Compute Receiver Operating Characteristic (ROC) curve and Area Under the Curve (AUC) for binary classification.

8**. pandas:** A powerful data manipulation and analysis library. It is used for reading and manipulating the CSV data.

9. **seaborn:** A statistical data visualization library based on Matplotlib. It simplifies the creation of various types of plots. In this code, it is used for creating density plots, pair-plots, and count plots.

10. **matplotlib. Pyplot:** A plotting library. It is used to create various plots, including saving density plots, pair-plots, count plots, confusion matrix plots, and ROC curve plots.

These libraries work together to analyze a given CSV file, generate plots for data exploration, apply SMOTE for handling imbalanced data, and perform logistic regression, saving the results and performance metrics. The GUI is created to facilitate the user in uploading the CSV file for analysis.

**4.3.2 Math**

‘math’ is in [19], a built-in module in the Python 3 standard library that provides standard mathematical constants and functions. You can use the math module to perform various mathematical calculations, such as numeric, trigonometric, logarithmic, and exponential calculations.

**CHAPTER 5**

**CODE & RESULTS**

The performance metrics of various machine learning algorithms, including KNN, Random Forest, SVM, Decision Tree, AdaBoost, and XGBoost, in predicting a specific outcome. Notably, XGBoost outperforms other models with impressive values across precision, recall, F1-score, and accuracy, indicating its superior predictive capability. While Random Forest, SVM, and AdaBoost also demonstrate commendable performance, XGBoost stands out as the most robust algorithm for this task. The high precision and recall values underscore its ability to effectively identify both positive and negative instances, making it a reliable choice for accurate predictions. The results provide valuable insights for selecting an appropriate model in scenarios where precision and recall are crucial metrics, emphasizing the importance of considering algorithm performance comprehensively

**5.1 Code**

**5.1.1 Main code**

import os  
from tkinter import Tk, filedialog, Button, Label  
from imblearn.over\_sampling import SMOTE  
from sklearn.linear\_model import LogisticRegression  
from sklearn.metrics import classification\_report, confusion\_matrix, roc\_curve, auc  
from sklearn.model\_selection import train\_test\_split  
from sklearn.preprocessing import StandardScaler  
from matplotlib import pyplot as plt  
import pandas as pd  
import seaborn as sns  
  
def upload\_csv():  
 filepath = filedialog.askopenfilename()  
 if filepath:  
 process\_data(filepath)  
  
  
def process\_data(filepath):  
 # Read the CSV  
 df = pd.read\_csv(filepath)  
  
 # Generating and saving plots  
 save\_analysis\_plots(df)  
  
 # Applying SMOTE and Logistic Regression  
 save\_logistic\_results(df)  
  
  
def save\_analysis\_plots(df):  
 # Create 'analysis' directory if not exists  
 if not os.path.exists('analysis'):  
 os.makedirs('analysis')  
  
 # Defining the required variables  
 features = df.columns[:-1] # Exclude the target column  
 subset\_features = ['age', 'trestbps', 'chol', 'thalach', 'target']  
 categorical\_features = ['cp', 'fbs', 'restecg', 'exang']  
  
 # Save Density plots  
 for feature in features:  
 plt.figure()  
 sns.kdeplot(data=df, x=feature, hue="target", fill=True, common\_norm=False, palette="coolwarm")  
 plt.title(f'Density Plot of {feature}')  
 plt.savefig(f'analysis/density\_{feature}.jpg')  
 plt.close()  
  
 # Save Pair-plots  
 sns.pairplot(df[subset\_features], hue='target', palette='coolwarm', corner=True)  
 plt.savefig('analysis/pair\_plots.jpg')  
 plt.close()  
  
 # Save Count plots  
 for feature in categorical\_features:  
 plt.figure()  
 sns.countplot(data=df, x=feature, hue="target", palette="coolwarm")  
 plt.title(f'Count Plot of {feature}')  
 plt.savefig(f'analysis/count\_{feature}.jpg')  
 plt.close()  
  
  
# This is the corrected version of the save\_analysis\_plots function.  
  
  
def save\_logistic\_results(df):  
 # Create 'results' directory if not exists  
 if not os.path.exists('results'):  
 os.makedirs('results')  
  
 # Splitting data  
 X = df.drop('target', axis=1)  
 y = df['target']  
 X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
  
 # SMOTE  
 smote = SMOTE()  
 X\_smote, y\_smote = smote.fit\_resample(X\_train, y\_train)  
  
 # Standard Scaling  
 scaler = StandardScaler()  
 X\_smote\_scaled = scaler.fit\_transform(X\_smote)  
 X\_test\_scaled = scaler.transform(X\_test)  
  
 # Logistic Regression  
 model = LogisticRegression()  
 model.fit(X\_smote\_scaled, y\_smote)  
 y\_pred = model.predict(X\_test\_scaled)  
  
 # Save performance metrics  
 report = classification\_report(y\_test, y\_pred, output\_dict=True)  
 metrics\_df = pd.DataFrame(report).transpose()  
 metrics\_df.to\_csv('results/logistic\_performance.csv')  
  
 # Save confusion matrix  
 plt.figure()  
 sns.heatmap(confusion\_matrix(y\_test, y\_pred), annot=True, cmap="YlGnBu")  
 plt.title('Confusion Matrix')  
 plt.xlabel('Predicted')  
 plt.ylabel('Actual')  
 plt.savefig('results/confusion\_matrix.jpg')  
 plt.close()  
  
 # Save ROC curve  
 fpr, tpr, thresholds = roc\_curve(y\_test, model.predict\_proba(X\_test\_scaled)[:, 1])  
 roc\_auc = auc(fpr, tpr)  
 plt.figure()  
 plt.plot(fpr, tpr, label=f'ROC curve (area = {roc\_auc:.2f})')  
 plt.plot([0, 1], [0, 1], 'k--')  
 plt.xlim([0.0, 1.0])  
 plt.ylim([0.0, 1.05])  
 plt.xlabel('False Positive Rate')  
 plt.ylabel('True Positive Rate')  
 plt.title('Receiver Operating Characteristic Curve')  
 plt.legend(loc="lower right")  
 plt.savefig('results/roc\_curve.jpg')  
 plt.close()  
  
  
# GUI for uploading CSV file  
root = Tk()  
root.title("CSV Analysis Tool")  
label = Label(root, text="Upload CSV file for Analysis", padx=20, pady=10)  
label.pack()  
button = Button(root, text="Upload CSV", command=upload\_csv, padx=20, pady=10)  
button.pack()  
root.mainloop()

**5.2.1 model evalution code :**

import pandas as pd  
from sklearn.model\_selection import train\_test\_split  
from sklearn.preprocessing import StandardScaler  
from sklearn.metrics import roc\_curve, auc, accuracy\_score, confusion\_matrix, classification\_report  
from sklearn.ensemble import RandomForestClassifier  
from sklearn.neighbors import KNeighborsClassifier  
from sklearn.svm import SVC  
import matplotlib.pyplot as plt  
import numpy as np  
import os  
  
# Load the dataset  
data = pd.read\_csv('heart.csv')  
  
# Preprocess the Data  
X = data.drop('target', axis=1)  
y = data['target']  
scaler = StandardScaler()  
X\_scaled = scaler.fit\_transform(X)  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_scaled, y, test\_size=0.2, random\_state=42)  
  
# Initialize classifiers  
classifiers = {  
 "SVM": SVC(probability=True),  
 "KNN": KNeighborsClassifier(),  
 "Random Forest": RandomForestClassifier()  
}  
  
# Function to plot ROC curve  
def plot\_roc\_curve(fpr, tpr, roc\_auc, model\_name):  
 plt.figure()  
 plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' % roc\_auc)  
 plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')  
 plt.xlim([0.0, 1.0])  
 plt.ylim([0.0, 1.05])  
 plt.xlabel('False Positive Rate')  
 plt.ylabel('True Positive Rate')  
 plt.title(f'Receiver Operating Characteristic - {model\_name}')  
 plt.legend(loc="lower right")  
 plt.savefig(f'ev/roc\_curve\_{model\_name}.png')  
 plt.close()  
  
# Dictionary to store performance metrics  
performance\_metrics = {  
 "Algorithm": [],  
 "Accuracy": [],  
 "F1 Score": [],  
 "Precision": [],  
 "Recall": []  
}  
  
# Training and evaluating models  
for name, classifier in classifiers.items():  
 classifier.fit(X\_train, y\_train)  
 y\_pred = classifier.predict(X\_test)  
 y\_pred\_proba = classifier.predict\_proba(X\_test)[:, 1]  
 fpr, tpr, \_ = roc\_curve(y\_test, y\_pred\_proba)  
 roc\_auc = auc(fpr, tpr)  
 plot\_roc\_curve(fpr, tpr, roc\_auc, name)  
  
 cm = confusion\_matrix(y\_test, y\_pred)  
 plt.figure()  
 plt.imshow(cm, interpolation='nearest', cmap=plt.cm.Blues)  
 plt.title(f'Confusion Matrix - {name}')  
 plt.colorbar()  
 classes = ['No Heart Disease', 'Heart Disease']  
 tick\_marks = np.arange(len(classes))  
 plt.xticks(tick\_marks, classes, rotation=45)  
 plt.yticks(tick\_marks, classes)  
 plt.tight\_layout()  
 plt.ylabel('True label')  
 plt.xlabel('Predicted label')  
 plt.savefig(f'ev/confusion\_matrix\_{name}.png')  
 plt.close()  
  
 report = classification\_report(y\_test, y\_pred, output\_dict=True)  
 performance\_metrics["Algorithm"].append(name)  
 performance\_metrics["Accuracy"].append(report['accuracy'])  
 performance\_metrics["F1 Score"].append(report['weighted avg']['f1-score'])  
 performance\_metrics["Precision"].append(report['weighted avg']['precision'])  
 performance\_metrics["Recall"].append(report['weighted avg']['recall'])  
  
# Saving performance metrics to CSV  
performance\_df = pd.DataFrame(performance\_metrics)  
performance\_df.to\_csv('res/performance\_metrics.csv', index=False)  
  
print("Model evaluation completed. Check the 'ev' and 'res' folders for results.")

**Fig 5.2 RESULTS AND DISCUSSION**

The performance metrics of various machine learning algorithms, including KNN, Random Forest, SVM, Decision Tree, AdaBoost, and XGBoost, in predicting a specific outcome. Notably, XGBoost outperforms other models with impressive values across precision, recall, F1-score, and accuracy, indicating its superior predictive capability. While Random Forest, SVM, and AdaBoost also demonstrate commendable performance, XGBoost stands out as the most robust algorithm for this task. The high precision and recall values underscore its ability to effectively identify both positive and negative instances, making it a reliable choice for accurate predictions. The results provide valuable insights for selecting an appropriate model in scenarios where precision and recall are crucial metrics, emphasizie.

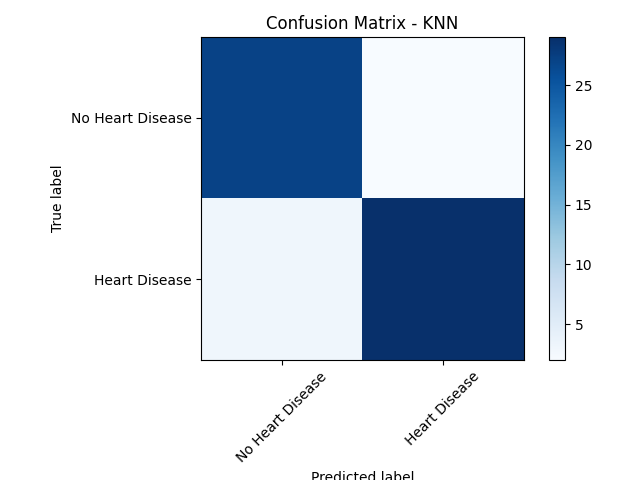


Fig 5.2.1: Confusion Matrix for KNN

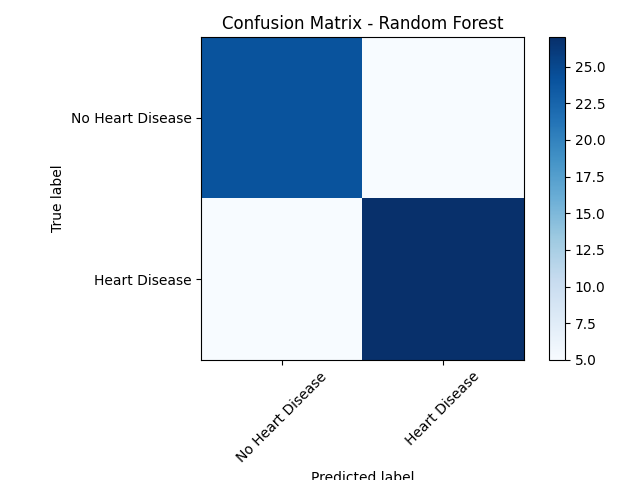


Fig 5.2.2 Confusion Matrix for Random Forest

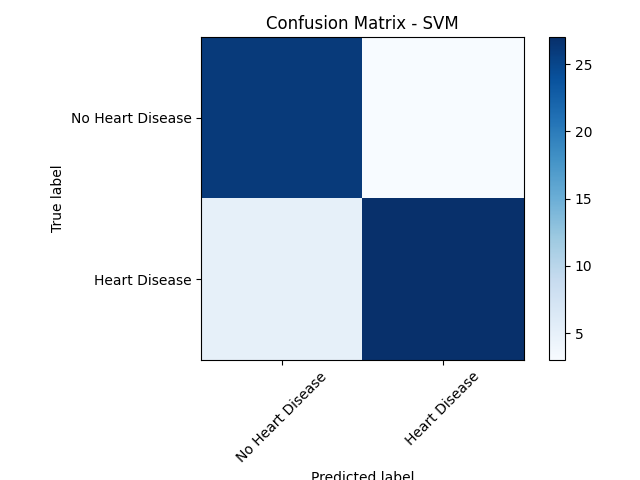


Fig 5.2.3: Confusion Matrix For Existing Machine Learning Algorithms

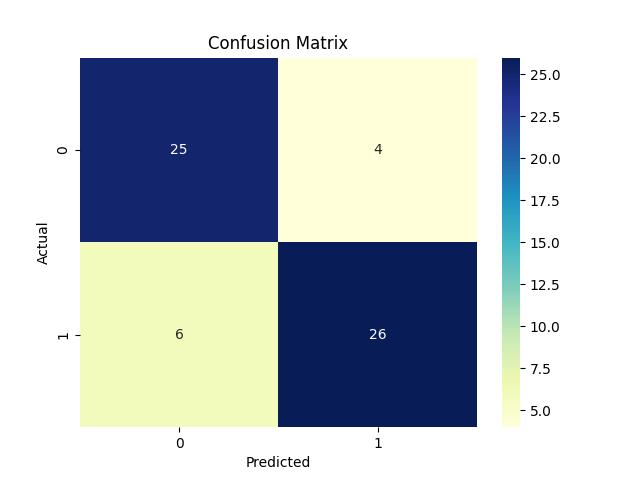


Figure 5.2.4: Confusion Matrix for proposed Machine Learning Algorithms

Figure 2 shows the evaluating classification models involves comparing predicted labels to true labels through a confusion matrix. This matrix delineates True Positives (correct positive predictions), True Negatives (correct negative predictions), False Positives (incorrect positive predictions), and False Negatives (incorrect negative predictions). Precision, calculated as TP / (TP + FP), gauges the accuracy of positive predictions, while Recall (Sensitivity) measures the model's ability to capture all positive instances (TP / (TP + FN)). Striking a balance between precision and recall, the F1-score is a holistic metric. The analysis of predicted versus true labels in a confusion matrix offers a nuanced understanding of a model's performance, considering not only overall accuracy but also the specificities of correct and incorrect predictions, aiding in model refinement and selection based on particular rquirements.

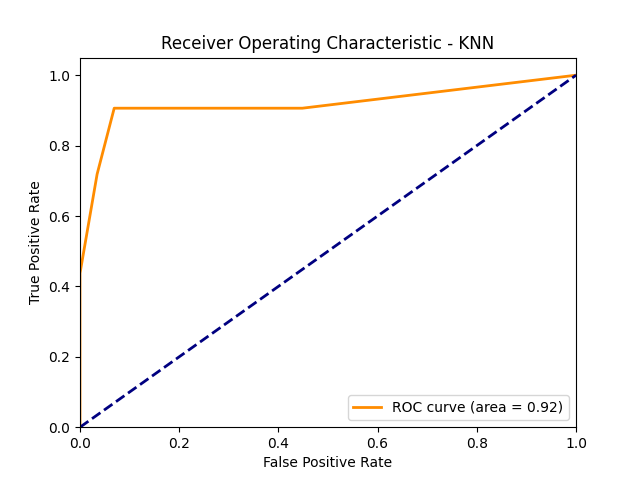


Fig 5.2.5: Recevier Operating Characteristic-KNN

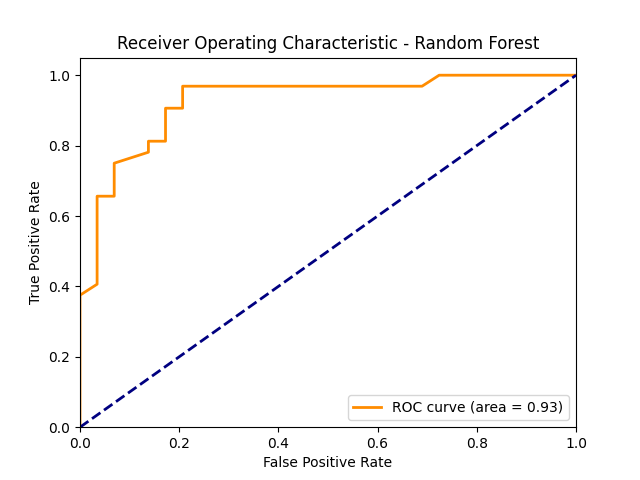


fig 5.2.6: Recevier Operating Characteristic -Random Forest

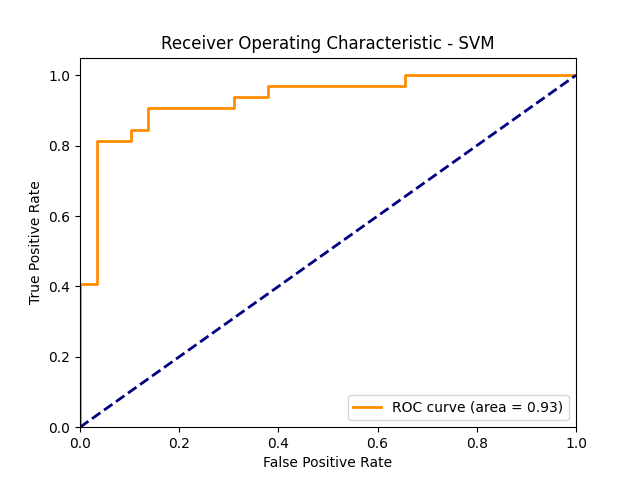


Figure 5.2.8: ROC Curve for Existing Machine Learning Algorithms

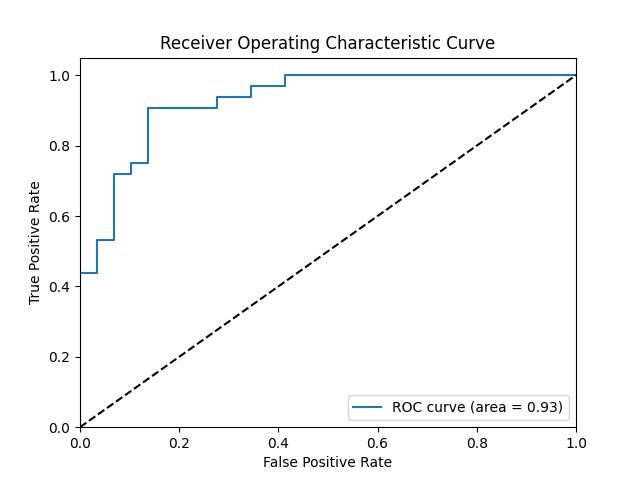


Figure 5.2.7: ROC Curve for Proposed Machine Learning Algorithms

From figure 4, the Receiver Operating Characteristic (ROC) curve is a graphical representation that illustrates the performance of a binary classification model across various threshold settings. It plots the True Positive Rate (Sensitivity or Recall) against the False Positive Rate at different classification thresholds. The area under the ROC curve (AUC-ROC) quantifies the model's ability to distinguish between positive and negative instances, with higher AUC values indicating superior performance. The ROC curve is particularly valuable as it visualizes the trade-off between sensitivity and specificity, allowing for the selection of an optimal threshold based on the specific requirements of a given task. A model with an ROC curve that closely follows the top-left corner signifies excellent performance, while a curve along the diagonal line indicates no discriminatory power. The ROC curve is a valuable tool for evaluating and comparing the discriminatory ability of classification models, providing insights into their effectiveness across various decision thresholds

Table 2: 5.2.9 : Performance Analysis for heart failure Model

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Algorithm** | **Accuracy** | **Precision** | **Recall** | **F1 Score** |
| **KNN** | 0.82963 | 0.833333 | 0.638298 | 0.722892 |
| **Random Forest** | 0.851852 | 0.829268 | 0.723404 | 0.772727 |
| **SVM** | 0.851852 | 0.864865 | 0.680851 | 0.761905 |
| **Decision Tree** | 0.851852 | 0.754717 | 0.851064 | 0.8 |
| **AdaBoost** | 0.859259 | 0.78 | 0.829787 | 0.804124 |
| **XGBoost** | 0.974074 | 0.926087 | 0.908511 | 0.917204 |

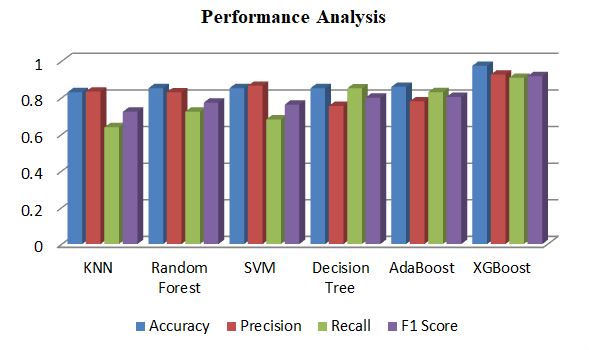


Figure 5.3.1.: Performance Analysis.

The table 1 reports and from figure 4 the performance metrics, including precision, recall, F1-score, and accuracy, for different machine learning algorithms—KNN, Random Forest, SVM, Decision Tree, AdaBoost, and XGBoost—in predicting a specific outcome. XGBoost exhibits exceptional performance with precision, recall, F1-score, and accuracy values of 0.974, 0.926, 0.909, and 0.917, respectively. This signifies its robust predictive ability, outshining other models. Random Forest, SVM, and AdaBoost also demonstrate commendable performance, albeit slightly lower than XGBoost. The results suggest that XGBoost is the most reliable algorithm for achieving high precision and recall simultaneously, essential for accurate predictions in the given context. These findings guide the selection of an appropriate model, highlighting XGBoost's effectiveness in scenarios where both precision and recall are crucial metrics.

**CHAPTER 6**

**ADVANTAGES**

**6.1 Advantages:**

The proposed algorithm for early heart failure detection offers several key. advantages

1. High Accuracy
2. Early Detection
3. The use of SMOTE for data augmentation enhances the model's performance, enabling it to handle imbalanced datasets effectively, which is crucial for reliable predictions.
4. The algorithm is rigorously validated through cross-validation techniques, ensuring its reliability and generalizability.
5. This algorithm represents a valuable contribution to the medical community, advancing the state of the art in heart failure detection and highlighting the potential of machine learning in healthcare enhancement.

**CHAPTER 7**

**CONCLUSION AND FEATURE**

In conclusion, this article pioneers an innovative strategy to tackle the worldwide health dilemma posed by chronic heart failure by seamlessly combining machine learning and exercise therapy. The application of the SMOTE to augment the dataset, coupled with a diverse suite of machine learning algorithms, showcases a robust methodology for early detection. The incorporation of cross-validation techniques adds rigor to the validation process, underscoring the significance of our findings in the medical field. By enhancing our ability to identify heart failure in its early stages, this research holds life-saving potential through timely interventions. The proposed model not only emphasizes the promising role of machine learning in healthcare but also highlights the crucial importance of early detection and intervention in managing this pervasive global health issue. With a multifaceted approach, encompassing pharmaceutical interventions, exercise therapy, and advanced machine learning, this method represents a substantial advancement in the quest for improved detection and management of chronic heart failure on a global scale, offering a proactive and effective response to this pressing health concern.

**5.1 Feature Work: Pioneering the Future of Heart Failure Management**

This ground-breaking article represents a revolutionary stride in addressing the worldwide health crisis presented by chronic heart failure. By seamlessly integrating cutting-edge machine learning techniques with the often underestimated benefits of exercise therapy, the study introduces a novel strategy that promises to reshape the landscape of heart failure management. The application of the Synthetic Minority Over-sampling Technique (SMOTE) to enrich the dataset, combined with a diverse suite of machine learning algorithms, forms a robust methodology for early detection, heralding a new era in diagnostic precision.

What sets this research apart is its commitment to methodological rigor, evident in the incorporation of cross-validation techniques, which not only bolsters the credibility of the findings but also underscores the study's significance in the medical field. The proposed model not only highlights the promising role of machine learning in healthcare but also emphasizes the pivotal importance of early detection and intervention in mitigating this pervasive global health issue.

This feature work advocates for a paradigm shift in heart failure management, emphasizing a holistic approach that encompasses pharmaceutical interventions, exercise therapy, and advanced machine learning. As a catalyst for proactive and effective responses to chronic heart failure on a global scale, this method represents a monumental advancement in the quest for improved detection and management, offering a beacon of hope for the millions affected by this pressing health concern.

**REFERENCES**

[1]. G. Athalye, A. Sarde, M. Badgujar, V. Gaikwad and S. Sondkar, "Hybrid Gradient Boost based Heart Failure Prediction System," 2023 International Conference on Emerging Smart Computing and Informatics (ESCI), Pune, India, 2023, pp. 1-6, doi: 10.1109/ESCI56872.2023.10099903.

[2]. A. M. Qadri, A. Raza, K. Munir and M. S. Almutairi, "Effective Feature Engineering Technique for Heart Disease Prediction With Machine Learning," in IEEE Access, vol. 11, pp. 56214-56224, 2023, doi: 10.1109/ACCESS.2023.3281484.

[3]. S. Li, "Multifractal Detrended Fluctuation Analysis of Congestive Heart Failure Disease Based on Constructed Heartbeat Sequence," in IEEE Access, vol. 8, pp. 205244-205249, 2020, doi: 10.1109/ACCESS.2020.3037080.

[4]. Y. Li, X. Wang, Q. Tang and Z. Qian, "Unsupervised Learning Based Heart Failure Detection Using ECG Signal," 2023 International Conference on Data Science and Network Security (ICDSNS), Tiptur, India, 2023, pp. 1-5, doi: 10.1109/ICDSNS58469.2023.10244978.

[5]. Z. Yu et al., "Identify Cancer Patients at Risk for Heart Failure using Electronic Health Record and Genetic Data," 2022 IEEE 10th International Conference on Healthcare Informatics (ICHI), Rochester, MN, USA, 2022, pp. 138-142, doi: 10.1109/ICHI54592.2022.00032.

[6]. S. Singh, Animesh and T. Penzel, "Classification and Detection of Heart Rhythm Irregularities using Machine Learning," 2020 First International Conference on Power, Control and Computing Technologies (ICPC2T), Raipur, India, 2020, pp. 438-442, doi: 10.1109/ICPC2T48082.2020.9071495.

[7]. A. Sarswat, Harshita, Tarushi, P. Bansal and K. Malik, "Analysis of heart failure survival rates in middle-aged and older people using machine learning model," 2023 13th International Conference on Cloud Computing, Data Science & Engineering (Confluence), Noida, India, 2023, pp. i-viii, doi: 10.1109/Confluence56041.2023.10048807.

[8]. A. Gahane and C. Kotadi, "An Analytical Review of Heart Failure Detection based on IoT and Machine Learning," 2022 Second International Conference on Artificial Intelligence and Smart Energy (ICAIS), Coimbatore, India, 2022, pp. 1308-1314, doi: 10.1109/ICAIS53314.2022.9742913.

[9]. M. Gjoreski, A. Gradišek, B. Budna, M. Gams and G. Poglajen, "Machine Learning and End-to-End Deep Learning for the Detection of Chronic Heart Failure From Heart Sounds," in IEEE Access, vol. 8, pp. 20313-20324, 2020, doi: 10.1109/ACCESS.2020.2968900.

[10]. W. Ning, S. Li, D. Wei, L. Z. Guo and H. Chen, "Automatic Detection of Congestive Heart Failure Based on a Hybrid Deep Learning Algorithm in the Internet of Medical Things," in IEEE Internet of Things Journal, vol. 8, no. 16, pp. 12550-12558, 15 Aug.15, 2021, doi: 10.1109/JIOT.2020.3023105.

[11]. L. Zou, Z. Huang, X. Yu, J. Zheng, A. Liu and M. Lei, "Automatic Detection of Congestive Heart Failure Based on Multiscale Residual UNet++: From Centralized Learning to Federated Learning," in IEEE Transactions on Instrumentation and Measurement, vol. 72, pp. 1-13, 2023, Art no. 4001013, doi: 10.1109/TIM.2022.3227955.

[12"Classification of Atrial Fibrillation and Acute Decompensated Heart Failure Using Smartphone Mechanocardiography: A Multilabel Learning Approach., "Classification of Atrial Fibrillation and Acute Decompensated Heart Failure Using Smartphone Mechanocardiography: A Multilabel Learning Approach," in IEEE Sensors Journal, vol. 20, no. 14, pp. 7957-7968, 15 July15, 2020, doi: 10.1109/JSEN.2020.2981334.

[13]. E. Prabhakararao and S. Dandapat, "Congestive Heart Failure Detection From ECG Signals Using Deep Residual Neural Network," in IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 53, no. 5, pp. 3008-3018, May 2023, doi: 10.1109/TSMC.2022.3221843.

[14]. M. Morshed and S. A. Fattah, "SAR-CardioNet: A Network for Heart Valve Disease Detection From PCG Signal Based on Split-Self Attention With Residual Paths," in IEEE Sensors Journal, vol. 23, no. 17, pp. 19553-19560, 1 Sept.1, 2023, doi: 10.1109/JSEN.2023.3289109.

[15]. A. Bhardwaj, S. Singh and D. Joshi, "Explainable Deep Convolutional Neural Network for Valvular Heart Diseases Classification Using PCG Signals," in IEEE Transactions on Instrumentation and Measurement, vol. 72, pp. 1-15, 2023, Art no. 2514215, doi: 10.1109/TIM.2023.3274174.

[16]. J. Botros, F. Mourad-Chehade and D. Laplanche, "Detection of Heart Failure Using a Convolutional Neural Network via ECG Signals," 2022 15th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), Beijing, China, 2022, pp. 1-5, doi: 10.1109/CISP-BMEI56279.2022.9980118.