IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

# Early Detection For Monitoring Cardiac Arrest

*Healthcare Analytics for using IoT & Machine Learning*

**S Vinay[1], Nisarga BG[2], Sainath M[3], Sachin KP[4], Mrs. Dr. Chetana R[5]**

[1,2,3,4]Student, Dept of ECE, SJB Institute of Technology, Bengaluru, Karnataka, India

[5]Assistance Professor, Dept of ECE, SJB Institute of Technology, Bengaluru, Karnataka, India

**Abstract:** Cardiac arrest remains a critical medical emergency with high mortality rates, often occurring without warning and outside clinical settings. Traditional healthcare lacks continuous, real-time monitoring and primarily focuses on post-event treatment. So we integrated sensors like ECG, PULSE RATE, TEMPERATURE sensors to a microcontroller esp32 to get real-time cardiac arrest prediction using machine learning algorithms like Random Forest, K-Nearest Neighbour (KNN), Weighted KNN.

***Index Terms*** – Cardiac arrest, ECG, KNN, ThingSpeak cloud, Remote Healthcare, low cost, Real-time monitoring

## I. INTRODUCTION

The human heart normally follows a rhythmic electrical conduction pattern. When this rhythm becomes irregular, too fast, or abnormally slow, the condition is clinically referred to as cardiac arrhythmia. In many cases, individuals may not initially experience noticeable symptoms, while others may report discomfort such as dizziness, chest strain, breathlessness, or occasional palpitations. Although some forms of arrhythmia are harmless, severe cases can disrupt the normal pumping efficiency of the heart, which may eventually lead to stroke, sudden cardiac arrest, or death. According to global health studies, cardiac disorders are among the leading contributors to mortality, especially in ageing populations and individuals with pre-existing health complications.

## II. PROBLEM STATEMENT

- Many people suffer heart attacks without knowing it until it's too late. There is no easy way to keep track of heart health all the time. Heart attacks often happen without warning, and many people don't get help in time. We need a small, low-cost device that can monitor the heart and quickly alert someone if there's a problem.
- Existing solutions primarily focus on post-event treatment rather than prevention or early detection. While wearable devices and biosensors can collect valuable physiological data, they often operate in isolation and lack the intelligence needed to interpret complex patterns that precede cardiac arrest.
- There is a critical need for a solution that combines IoT-enabled biosensing technology with machine learning algorithms to provide early warnings, reduce emergency response time, and improve survival outcomes for individuals at risk of sudden cardiac arrest.

## III. MOTIVATION

***Personalized Healthcare***

Traditional healthcare often relies on periodic check-ups, which may miss early warning signs of cardiac arrest. our device ensures that each individual's heart activity is continuously monitored in real-time.

***Psychological Assurance***

Knowing that their heart health is being tracked continuously provides patients with peace of mind. This reassurance improves compliance and encourages healthier behaviour, which is often a challenge in conventional healthcare systems.

***Cost Efficiency & Accessibility***

Personalised monitoring reduces unnecessary diagnostic tests and hospitalisations. For patients in remote and rural, resource-limited areas, low-cost IoT devices make advanced cardiac monitoring accessible without requiring regular physical appearance in hospitals

## IV. OBJECTIVES

***Early Detection of Cardiac Arrest***

To identify the early signs of Cardiac arrest through continuous monitoring of vital health parameters.

***High Accuracy***

To ensure reliable detection using advanced machine learning models and high-quality medical data.

***Real-Time Alert***

- To provide instant notifications to patients and healthcare providers for timely medical response.
- Apply machine learning algorithms to analyse physiological data, identify high-risk patterns, and generate accurate alerts for potential cardiac arrest events before they occur.

## V. SOFTWARE AND HARDWARE REQUIREMENTS

***Software requirements***

- Anaconda Navigator Jupiter Notebook, Python 3.7
- Python 3.3 or 2.7 or higher
- Libraries: Matplotlib, Seaborn, Numpy, Pandas, Keras, Pillow, SK Learn, OpenCV OS
- Thing speak cloud

***Hardware requirements***

- ESP32 Microcontroller

  The ESP32 is a low-cost Wi-Fi microcontroller with built-in TCP/IP networking software and microcontroller capability, produced by Espressif Systems.
  The ESP32 Wi-Fi module is primarily used for enabling Wi-Fi connectivity in various applications, particularly in the Internet of Things (IOT) space. It's a cost-effective solution for connecting microcontrollers to the internet, allowing for remote control, data transfer, and web server hosting.

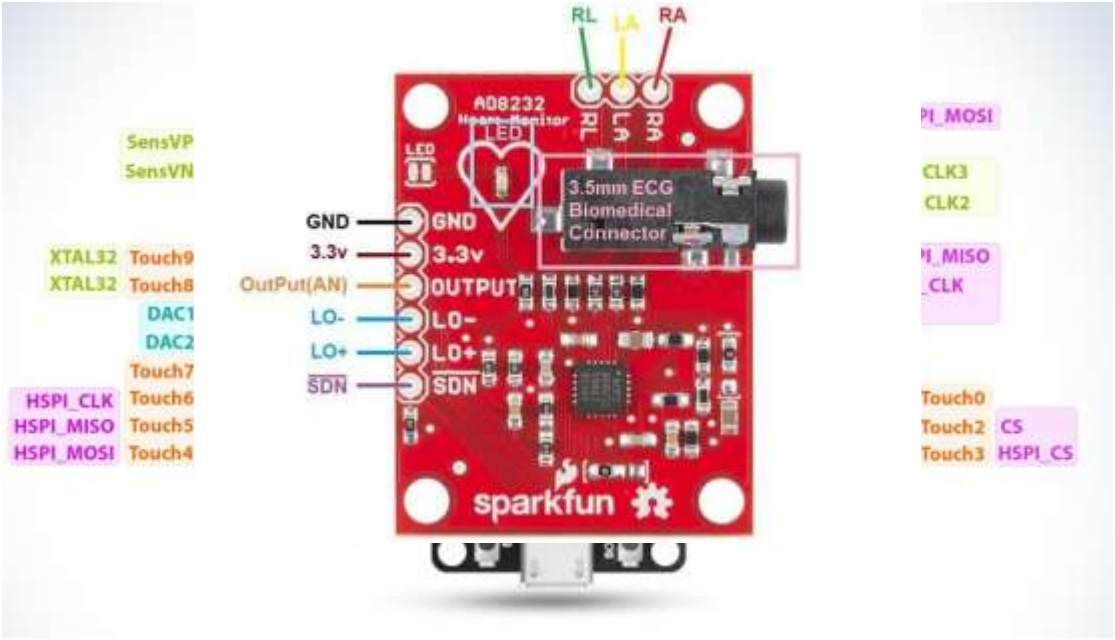


*Fig1.ESP32*

- Pulse rate sensor

  A pulse rate monitor is an instrument that calculates heart rate based on changes in blood volume in the arteries, usually optically. The tool commonly used for this task is Photo Plethysmography (PPG), in which light, generally green-coloured (approximately 550 nm), is shone on the skin and then back-scattered by blood.

*Fig2.Pulse rate sensor*

- ECG sensor-AD8232

  The **AD8232** is a compact analog front-end chip built specifically for ECG and other biopotential monitoring tasks. Its role is to capture the faint electrical signals produced by the body, strengthen them, and clean out unwanted noise — whether that noise comes from body movement or electrodes placed farther apart. By conditioning the signal in this way, the AD8232 makes it straightforward for a low-power ADC or a microcontroller to read and process the data, enabling reliable heart activity monitoring in portable or wearable devices.

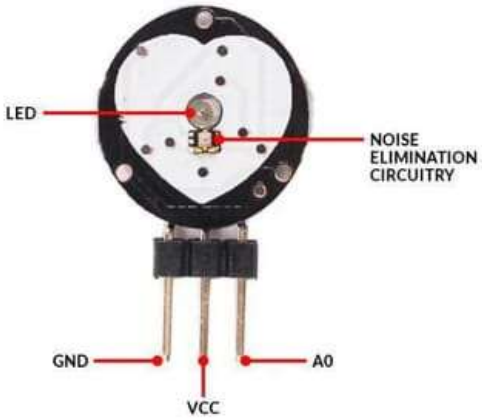


*Fig3.ECG-AD8232*

- Temperature sensor-DS18B20

  The DS18B20 is a popular 1-wire digital temperature sensor known for its accuracy, ease of use, and digital output,making it perfect for microcontrollers like the ESP32.
  The sensor operates over a wide temperature range, from -55℃ to +125℃ (-67°F to +257°F), making it suitable for diverse applications such as monitoring accurate, hydroponic systems, boilers, and environmental projects. Its accuracy is specified as ± 0.5℃ within the range of -10℃ to +85℃ (14°F to 185°F).


*Fig4.Temperature sensor-DS18B20*

***Functional requirements***

- User-facing GUI programmes
  We plan to design interactive applications for both desktop and mobile platforms. These interfaces will allow clients to easily connect with and navigate our network services.
- Data Pre-processing Module
  A dedicated unit will handle the preparation of datasets obtained from the UCI repository. This includes cleaning unlabeled records, applying stemming and lemmatization, and performing other transformations to ensure the data is ready for analysis.

- A comparison of algorithms
  multiple algorithms will be compared to determine which approach delivers the most reliable results.

## VI. PROPOSED MODEL

DaCollecting data forms the foundation of any system's processing workflow, and for this purpose we rely on the UCI Machine Learning Repository. The datasets available there have been thoroughly validated, both by the UCI curators and by numerous independent researchers, ensuring their reliability for academic and practical use.

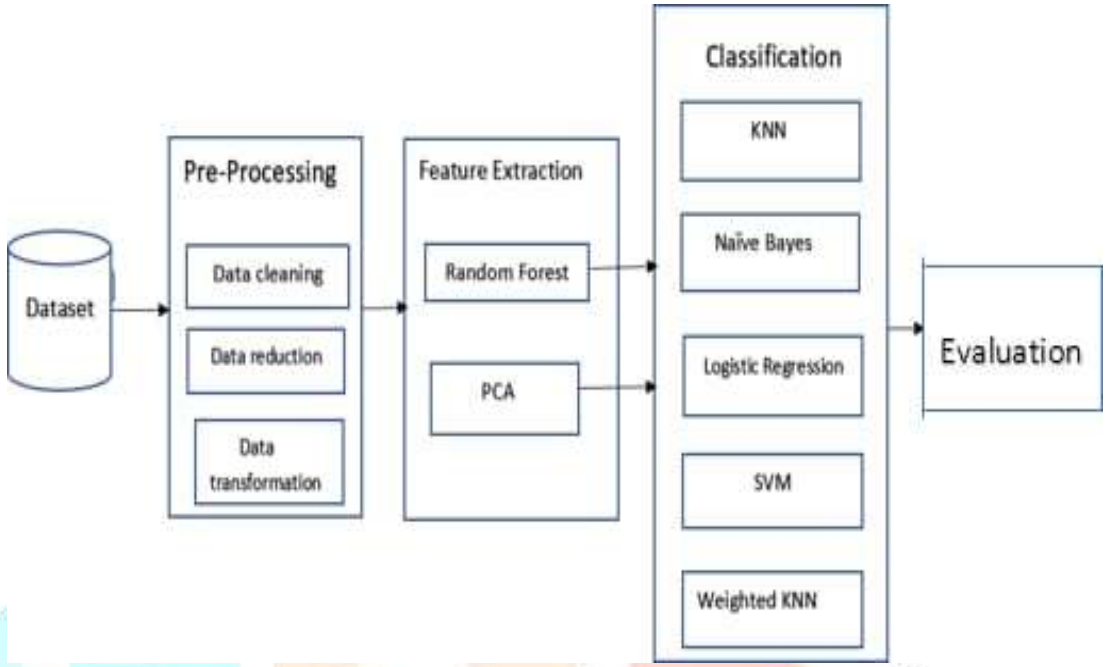


*Fig5. block diagram of System Architecture*

The above figure (Fig5), Information extraction in this study begins with a structured dataset derived from ECG signals. Key parameters such as heart rate, R-R interval, deflection count, amplitude, gender, and other relevant attributes are taken into account. For reliability, the dataset was sourced from the UCI Machine Learning Repository, a widely recognized platform for validated data. The collected information was then organized into a CSV file format, ensuring it could be efficiently stored and processed for subsequent analysis

### *Preprocessing*

The dataset initially contained missing entries and irregularities, making it unsuitable for direct classification tasks. To refine the data, variables that remained constant across all subjects were eliminated, as they contributed no meaningful distinction. Invariant features were identified through statistical checks such as variance and standard deviation. For the attributes with incomplete values, mean imputation was applied, ensuring the dataset was consistent and ready for further analysis.

### *Feature Extraction*

Feature selection in this work was carried out using two approaches: Random Forest and Principal Component Analysis (PCA). Since the preprocessed dataset contained a large number of attributes, the classification model required significant computational resources. Selecting the most relevant features was therefore essential, both to reduce processing time and to highlight the variables most strongly linked to the output class. Within the dataset, certain records were duplicated or represented repeated cases of the same condition. To address this redundancy, the Random Forest algorithm was applied, which not only performed classification but also helped in minimizing unnecessary data.

### *Classification*

The third stage of the process is classification, which represents a crucial step in building the machine learning model. At this point, five algorithms were applied: K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Naïve Bayes, Logistic Regression, and Random Forest. Before classification, feature reduction techniques were used to refine the dataset, which was then stored in CSV format. Using this optimized data, evaluation metrics—including accuracy, precision, recall, and the F1-score—were calculated to assess the performance of each algorithm.

***Evaluation***

The effectiveness of each method is carefully evaluated and the results are presented. The features selected during this process are then used as inputs for the five classification algorithms applied in the subsequent stage

## VII. IMPLEMENTATION

Machine Learning is widely recognized as one of the most effective approaches for testing, as it relies on both training and evaluation phases. Artificial Intelligence (AI), a broader discipline, aims to replicate human capabilities through computational systems, with Machine Learning serving as a specialized subset. When combined, these technologies contribute to what is often referred to as machine intelligence. Unlike conventional systems, Machine Learning models are designed to analyze data and extract meaningful patterns for practical use.

In this research, five algorithms—Naïve Bayes, Logistic Regression, K-Nearest Neighbors (KNN), Support Vector Machine (SVM), and Weighted KNN—are evaluated in terms of accuracy.

***Methodology***

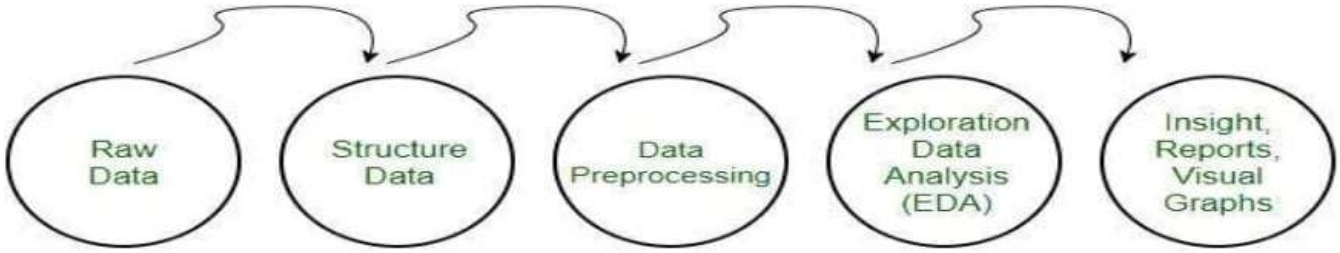


*Fig6.methodology*

***Importing Libraries***

- **NumPy** is a fundamental Python library designed for computing. It provides efficient tools for numerical operations and will be used extensively in this project. The package is imported with the alias **`np`**.
- **Pandas** is an open-source library licensed under BSD that specializes in data manipulation and analysis. It offers flexible data structures and powerful methods for handling datasets. In this work, it is imported as **`pd`**.
- **Matplotlib (pyplot)** supplies a wide range of plotting functions, enabling Python to operate in a style similar to MATLAB. For visualization tasks, it is commonly imported as **`plt`**.
- **Seaborn** builds on Matplotlib to produce visually appealing and informative statistical graphics. It simplifies the creation of complex plots and will be used to enhance the clarity of data visualizations.

***Data Pre-Processing***

- Data preprocessing refers to the set of adjustments applied to raw datasets before they are analyzed or used in machine learning models. This stage is essential because information collected from multiple sources often arrives in an unstructured form, making direct analysis difficult. Preprocessing transforms such messy data into a clean, organized format that algorithms can interpret effectively.
- A critical aspect of preprocessing is the identification and treatment of missing values, commonly represented as *NaN* (Not a Number). Even a small number of NaNs can distort statistical measures or reduce the accuracy of predictive models, so checking for them is a necessary step.
- Forward filling
- Backward filling

### *Data Analysis*

Data analysis is the process of dissecting, sanitising, modifying, and modelling data with the aim of revealing relevant information, guiding deductions, and assisting in decision- making.
Data analysis has many different components and steps, including a variety of methods with different names that are applied in a number of business, scientific, and social science fields. Because it helps businesses to operate more efficiently and make more scientific judgments, data analysis is essential in today's business environment.

### *Feature Extraction*

Feature extraction is the process of transforming raw data into measurable attributes that can be utilized by machine learning algorithms. This step preserves the essential characteristics of the original dataset while converting them into a numerical form suitable for computation. Compared to applying algorithms directly on unprocessed data, feature extraction typically yields more accurate and meaningful results, since the model is trained on well-defined inputs.
During model training, the importance of each feature can be evaluated by examining its contribution to reducing impurity in the dataset. A feature that consistently lowers impurity is considered more influential in guiding the model's decisions. In the case of random forests, this measure of importance is calculated by averaging the impurity reduction contributed by each feature across all decision trees. The resulting score provides a reliable estimate of the variable's overall significance within the model.

### *Train and Test dataset*

Once the dataset has been cleaned, explored, and visualized, the next step is to fit the first machine learning model. To ensure that the model is both effective and generalizable, the data is typically divided into two distinct subsets: a **training set** and a **test set**.

### *Prediction and Accuracy*

Machine learning algorithms can be trained to predict a customer's smartphone choice based on observed preferences and behavioral patterns. This predictive capability is highly valuable for smartphone manufacturers, as it allows them to identify the features and attributes that most influence consumer decisions. By understanding these factors, companies can refine their product designs and marketing strategies to better align with customer expectations.
The effectiveness of such predictive models is often measured using accuracy, which indicates how well the algorithm correctly classifies each observation. In simple terms, accuracy reflects the proportion of predictions that match the actual outcomes. A higher accuracy score suggests that the model is reliably capturing the underlying decision-making process, whereas lower accuracy highlights the need for further refinement of features or model parameter

## ALGORITHM

- Random Forest Algorithm

The Random Forest algorithm is widely used for classification tasks due to its ability to handle complex datasets and identify the most influential attributes. In this project, Random Forest is applied to determine the principal features that contribute most to the predictive model. Since the dataset may contain duplicate entries or incorrect values, it is essential to remove such inconsistencies to ensure reliability.

- K- Nearest Neighbor

the K-Nearest Neighbours (KNN) algorithm is applied to evaluate the performance of the dataset. The process begins by importing the KNN library and then dividing the dataset into two subsets: a training set and a test set. The training set is used to fit the model, while the test set provides an independent evaluation of its predictive ability.

To measure performance, the accuracy score is calculated, which reflects the proportion of correctly classified instances in the test data. In this case, the model achieved an accuracy of 58.67%, indicating that just over half of the predictions matched the actual outcomes. This result provides a baseline understanding of the model's effectiveness and highlights the need for further optimization, such as tuning hyperparameters or refining feature selection, to improve predictive accuracy

- SVM classifier

the Support Vector Machine (SVM) algorithm is employed to evaluate the dataset's classification performance. The process begins by importing the SVM library and dividing the dataset into two subsets: a training set, used to fit the model, and a test set, used to assess its predictive ability.
The model's effectiveness is then measured using the accuracy score, which represents the proportion of correctly classified instances in the test data. In this case, the SVM achieved an accuracy of 94.09%, demonstrating a strong ability to generalize and correctly predict outcomes. This high accuracy suggests that the SVM model is well-suited for the dataset and performs significantly better compared to baseline approaches, making it a reliable choice for classification in this project.

- Logistic Regression

The Logistic Regression algorithm is applied to evaluate the dataset's classification performance. The process begins by importing the Logistic Regression library and dividing the dataset into two subsets: a training set, which is used to fit the model, and a test set, which is used to assess its predictive ability. The model's effectiveness is then measured using the accuracy score**,** which represents the proportion of correctly classified instances in the test data. In this case, Logistic Regression achieved an accuracy of 55.71%**,** indicating that slightly more than half of the predictions matched the actual outcomes. While this accuracy provides a baseline measure of performance, it also suggests that the model may require further optimization such as feature engineering, parameter tuning, or the use of more advanced algorithms to achieve stronger predictive results.

- Naive Bayes

The Naïve Bayes algorithm is applied to evaluate the dataset's classification performance. The process begins by importing the Naïve Bayes library and dividing the dataset into two subsets: a training set, which is used to fit the model, and a test set, which is used to assess its predictive ability. model's performance is then measured using the accuracy score, which represents the proportion of correctly classified instances in the test data. In this case, Naïve Bayes achieved an accuracy of 15.86%.

- Weighted KNN

The Weighted K-Nearest Neighbours (KNN) algorithm was applied to evaluate the classification performance on the dataset. To begin, the necessary library for Weighted KNN was imported. The dataset was then partitioned into training and testing subsets to ensure reliable model assessment. After training the classifier on the training set, predictions were generated for the test set. The model's effectiveness was quantified using the accuracy score**,** which compares predicted labels against the true labels. The experiment demonstrated that the Weighted KNN achieved an impressive accuracy of 98.89%**,** highlighting its strong predictive capability for this dataset

- Decision Tree

The Decision Tree classifier was employed to evaluate the dataset's predictive performance. Initially, the required library was imported, and the dataset was systematically divided into training and testing subsets to ensure unbiased model assessment. The classifier was trained using the training data, after which predictions were generated for the test set. To measure performance,

the accuracy score was calculated by comparing predicted labels with the actual outcomes. The results showed that the Decision Tree achieved an accuracy of 97.78%**,** confirming its effectiveness in handling this dataset.

```
scores = [score_knn,score_svm,score_lr,score_nb,score_wknn]
algorithms = ["KNN","Support Vector Machine","Logistic regression","naviebayes","Weigthed - knn"]

for i in range(len(algorithms)):
    print("The accuracy score achieved using "+algorithms[i]+" is: "+str(scores[i])+" %")
```

Python

```
The accuracy score achieved using KNN is: 0.5867158671586716 %
The accuracy score achieved using Support Vector Machine is: 0.940959409594096 %
The accuracy score achieved using Logistic regression is: 0.5571955719557196 %
The accuracy score achieved using naviebayes is: 0.15867158671586715 %
The accuracy score achieved using Weigthed - knn is: 0.988929889298893 %
```

*Fig7. showing weighted-KNN achieved highest accuracy*

## VIII. Integration of Hardware and Software

To determine the network's breakpoint in our system, stress testing andmanual testing were also conducted. Stress testing was carried out manually utilising hundreds of nodes that were rented from an internet server, while manual testing was carried out using the Selenium programme. The dataset was first tested during data preprocessing, the first module, to make sure there were no unknown or missing values. Data cleaning is carried out effectively using the original CSV file as input. To lessen the dimensionality of the dataset, the second and third tests are carried out in the second module, Feature Extraction. To obtain the reduced feature dataset, the preprocessed csv file is obtained, and PCA and random forest are successfully applied independently. This bothersome structural exam is predicated on prior structural understanding. Unit tests are used to validate a particular business process, application, or system configuration at the component level. Unit tests ensure that every step of a business process conforms with established standards and has clearly defined inputs and outputs. To determine the network's breakpoint in our system, stress testing and manual testing were also conducted. The Table1 below provides a summary of the test's outcomes.

| MODULE | INPUT | EXPECTED OUTPUT | ACTUAL OUTPUT | RESULT |
|---|---|---|---|---|
| Pre-processing | Original CSV file | Successfully cleansed data | Sent IP address of master/UDP message UMSTR | PASS |
| Feature Extraction (Random Forest) | Preprocessed CSV file | Reduced features/attributes | Reduced features/attributes | PASS |
| Feature Extraction (Principal Component Analysis) | Preprocessed CSV file | Reduced features/attributes | Reduced features/attributes | PASS |
| Classification (KNN and weighted-KNN) | CSV file with reduced features | Accuracy and classification according to KNN | Accuracy and classification according to KNN | PASS |
| Classification (Logistic Regression) | CSV file with reduced features | Accuracy and Classification according to Logistic Regression | Accuracy and Classification according to Logistic Regression | PASS |
| Classification (Naive-Bayes) | CSV file with reduced features | Accuracy and Classification according to Naive-Bayes | Accuracy and Classification according to Naive-Bayes | PASS |
| Classification (Support Vector Machine) | CSV file with reduced features | Accuracy and Classification according to SVM | Accuracy and Classification according to SVM | PASS |

Table1.*Tests and Nodes along with Results*

**Importing Libraries**

```
import pandas as pd
import numpy
from sklearn import svm
from sklearn.linear_model import LogisticRegression
import warnings
warnings.filterwarnings('ignore')
```

```
data = pd.read_csv('data_arrhythmia.csv')
```

```
data.head()
```

| | age | sex | height | weight | qrs_duration | p-r_interval | q-t_interval | t_interval | p_interval | qrs | ... | KY | KZ | LA | LB | LC | LD | LE | LF | LG | diagnosis |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 75 | 0 | 190 | 80 | 91 | 193 | 371 | 174 | 121 | -16 | ... | 0.0 | 9.0 | -0.9 | 0.0 | 0 | 0.9 | 2.9 | 23.3 | 49.4 | 8 |
| 1 | 56 | 1 | 165 | 64 | 81 | 174 | 401 | 149 | 39 | 25 | ... | 0.0 | 8.5 | 0.0 | 0.0 | 0 | 0.2 | 2.1 | 20.4 | 38.8 | 6 |
| 2 | 54 | 0 | 172 | 95 | 138 | 163 | 386 | 185 | 102 | 96 | ... | 0.0 | 9.5 | -2.4 | 0.0 | 0 | 0.3 | 3.4 | 12.3 | 49.0 | 10 |
| 3 | 55 | 0 | 175 | 94 | 100 | 202 | 380 | 179 | 143 | 28 | ... | 0.0 | 12.2 | -2.2 | 0.0 | 0 | 0.4 | 2.6 | 34.6 | 61.6 | 1 |
| 4 | 75 | 0 | 190 | 80 | 88 | 181 | 360 | 177 | 103 | -16 | ... | 0.0 | 13.1 | -3.6 | 0.0 | 0 | -0.1 | 3.9 | 25.4 | 62.8 | 7 |

5 rows × 280 columns

*Fig8.explains how the libraries are imported using the pandas, numpy,sklearn.Then using the pandas library an CSV file is read and then the dataset is described.*

| | Age | Sex | Height | Weight | qrs_duration | p-r_interval | q-t_interval | t_interval | p_interval | qrs | T |
|---|---|---|---|---|---|---|---|---|---|---|---|
| count | 1356.000000 | 1356.000000 | 1356.000000 | 1356.000000 | 1356.000000 | 1356.000000 | 1356.000000 | 1356.000000 | 1356.000000 | 1356.000000 | 1332.000000 |
| mean | 46.471239 | 0.550885 | 166.188053 | 68.170354 | 88.920354 | 155.152655 | 367.207965 | 169.949115 | 90.004425 | 33.676991 | 36.150901 |
| std | 16.454474 | 0.497587 | 37.142898 | 16.578554 | 15.353051 | 44.809176 | 33.360774 | 35.606765 | 25.807576 | 45.397893 | 57.814769 |
| min | 0.000000 | 0.000000 | 105.000000 | 6.000000 | 55.000000 | 0.000000 | 232.000000 | 108.000000 | 0.000000 | -172.000000 | -177.000000 |
| 25% | 36.000000 | 0.000000 | 160.000000 | 59.000000 | 80.000000 | 142.000000 | 350.000000 | 148.000000 | 79.000000 | 3.750000 | 14.000000 |
| 50% | 47.000000 | 1.000000 | 164.000000 | 68.000000 | 86.000000 | 157.000000 | 367.000000 | 162.000000 | 91.000000 | 40.000000 | 41.000000 |
| 75% | 58.000000 | 1.000000 | 170.000000 | 79.000000 | 94.000000 | 175.000000 | 384.000000 | 179.000000 | 102.000000 | 66.000000 | 63.250000 |
| max | 83.000000 | 1.000000 | 780.000000 | 176.000000 | 188.000000 | 524.000000 | 509.000000 | 381.000000 | 205.000000 | 169.000000 | 179.000000 |

*Fig9.describes the complete dataset as for the each attribute it will counthow many entries are there then it will calculate the mean, std, minimum, maximumvalue.*

| | Age | Sex | Height | Weight | qrs_duration | p-r_interval | q-t_interval | t_interval | p_interval | qrs | T | P | QRST | J | heart_rate | q_wave | r_wave |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 75 | 0 | 190 | 80 | 91 | 193 | 371 | 174 | 121 | -16 | 13.0 | 64.0 | -2.0 | -13.592105 | 63.000000 | 0 | 52 |
| 1 | 56 | 1 | 165 | 64 | 81 | 174 | 401 | 149 | 39 | 25 | 37.0 | -17.0 | 31.0 | -13.592105 | 53.000000 | 0 | 48 |
| 2 | 54 | 0 | 172 | 95 | 138 | 163 | 386 | 185 | 102 | 96 | 34.0 | 70.0 | 66.0 | 23.000000 | 75.000000 | 0 | 40 |
| 3 | 55 | 0 | 175 | 94 | 100 | 202 | 380 | 179 | 143 | 28 | 11.0 | -5.0 | 20.0 | -13.592105 | 71.000000 | 0 | 72 |
| 4 | 75 | 0 | 190 | 80 | 88 | 181 | 360 | 177 | 103 | -16 | 13.0 | 61.0 | 3.0 | -13.592105 | 74.463415 | 0 | 48 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 95 | 55 | 0 | 185 | 105 | 87 | 292 | 406 | 192 | 175 | 19 | 58.0 | 18.0 | 51.0 | -13.592105 | 64.000000 | 20 | 36 |
| 96 | 33 | 1 | 150 | 55 | 102 | 143 | 364 | 168 | 82 | 33 | 17.0 | 46.0 | 25.0 | -13.592105 | 79.000000 | 24 | 44 |
| 97 | 37 | 1 | 157 | 62 | 85 | 145 | 397 | 176 | 88 | 24 | 47.0 | 57.0 | 36.0 | -13.592105 | 73.000000 | 20 | 60 |
| 98 | 52 | 1 | 155 | 104 | 84 | 188 | 450 | 193 | 89 | 22 | -5.0 | 7.0 | 5.0 | -13.592105 | 66.000000 | 0 | 56 |
| 99 | 36 | 1 | 160 | 70 | 78 | 118 | 241 | 152 | 68 | 26 | -165.0 | 43.0 | 30.0 | 175.000000 | 72.000000 | 0 | 72 |

*Fig10.explains the sample of the arrhythmia dataset where the attributesincluded is age, sex, height, weight ,q r s duration, pr interval ,qt interval ,t interval , p interval, q r s, T , P , QRST , J , heart rate, q wave , r wave and so on.*

## XI. RESULTS

The performance of the proposed system was evaluated using multiple machine learning algorithms applied to the preprocessed cardiac dataset. Each model was trained and tested to assess its ability to accurately classify high-risk cardiac conditions. The comparative analysis highlights the strengths and limitations of different approaches, providing insight into which algorithms are most suitable for real-time cardiac arrest prediction. Accuracy values, precision, recall, and F1 scores were computed to ensure a fair evaluation, and the outcomes are summarized by using the Table2.

Table2.Accuracy Comparison of Algorithms

| | Accuracy | Precision | Recall | F1-score |
|---|---|---|---|---|
| KNN | 58.67 | 88 | 63 | 71 |
| Support Vector Machine | 94.09 | 99 | 98 | 98 |
| Logistic Regression | 56.00 | 84 | 83 | 84 |
| Naïve Bayes | 18.87 | 74 | 70 | 71 |
| Weight KNN | 98.89 | 99 | 99 | 99 |

**Key Observations**

- Weighted KNN achieved the highest accuracy (98.89%), among all other models.
- SVM provided high accuracy (94.09%), proving its resilience in handling complex, non-linear data.
- KNN and Logistic Regression showed moderate performance (~55–58%).
- Naïve Bayes performed poorly (18.87%), thus it unsuitable for datasets where features exists strong corelations

**Output samples:**

Patient 1

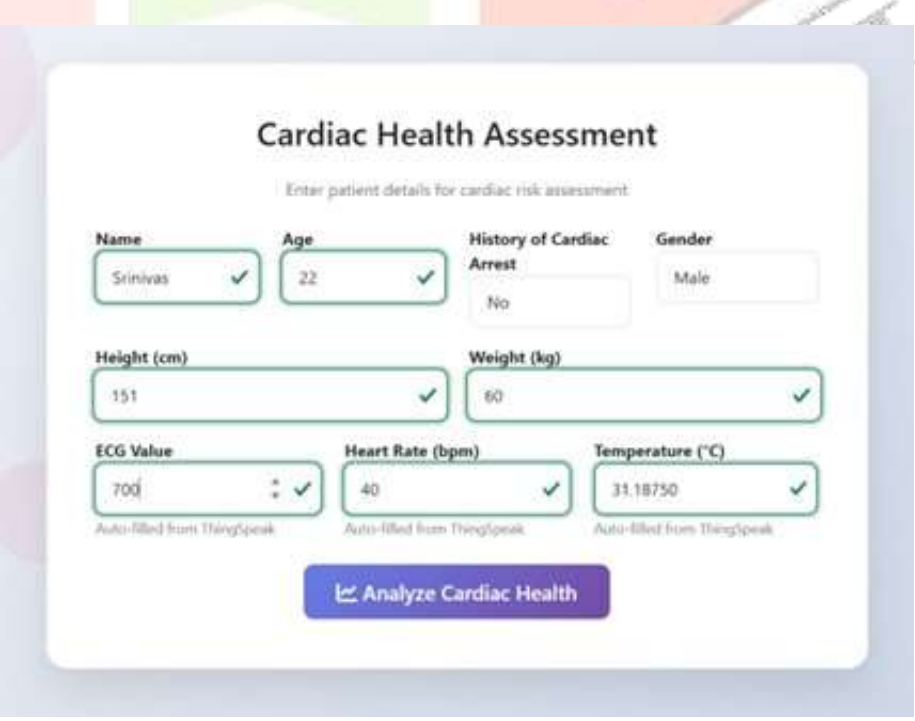


*Fig11.1 The above figure explains about the online platform of the Cardiac Detection. It is the welcome page of the system, where the input values are taken from Thing speak cloud*

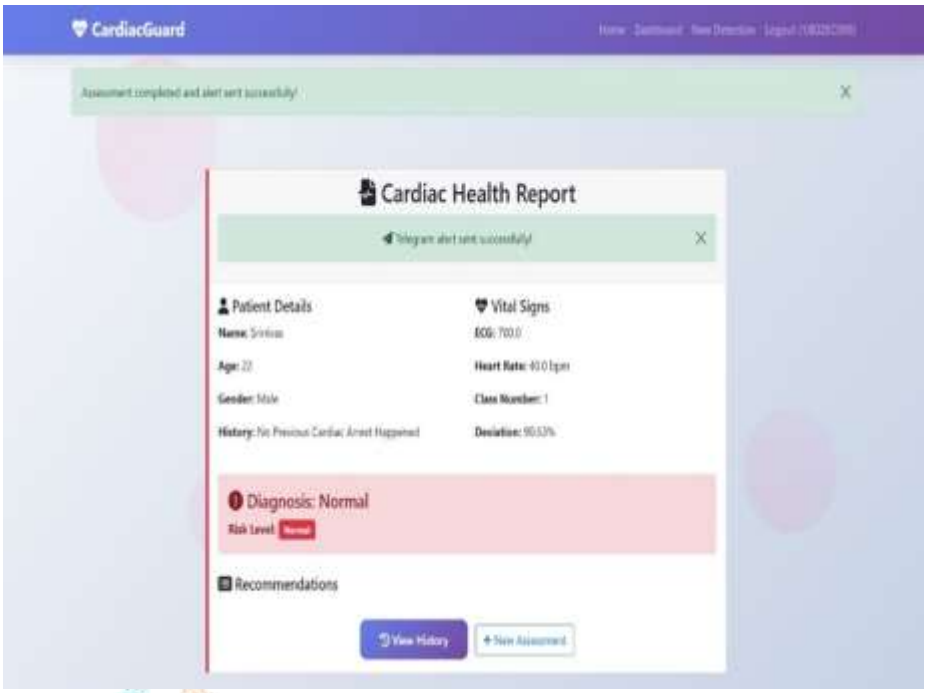


*Fig11.2 shows the Result page of the Cardiac Detection System. It depicts the output after computing the values that are read from Thing Speak Cloud. predicting normal*

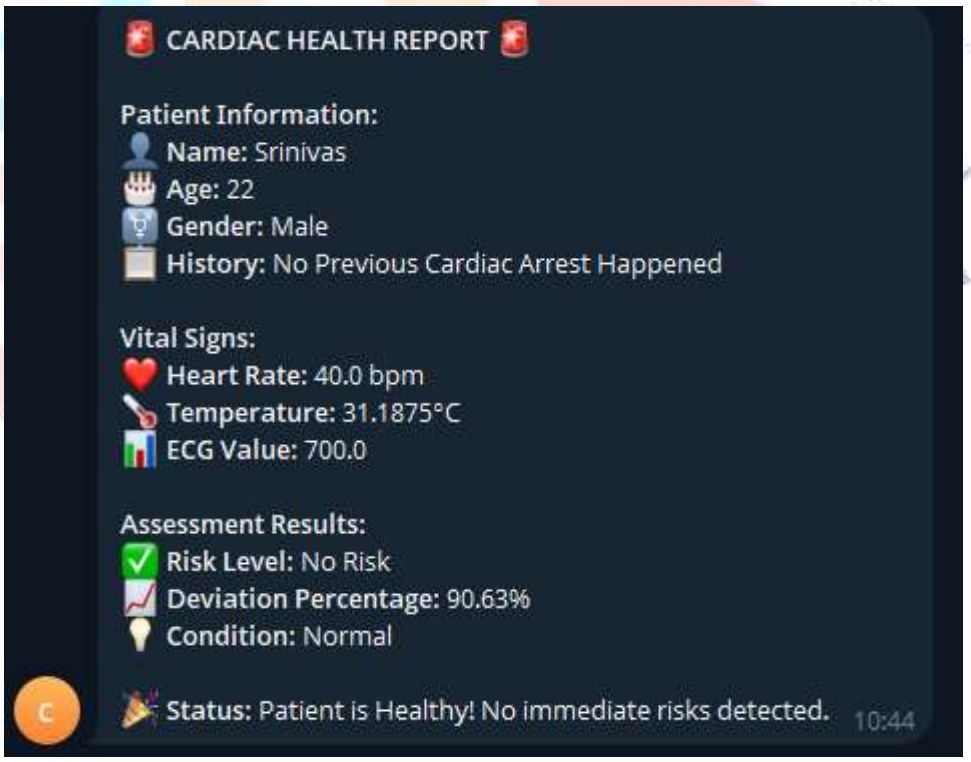


*Fig11.3 Result displayed in telegram bot*

Patient 2

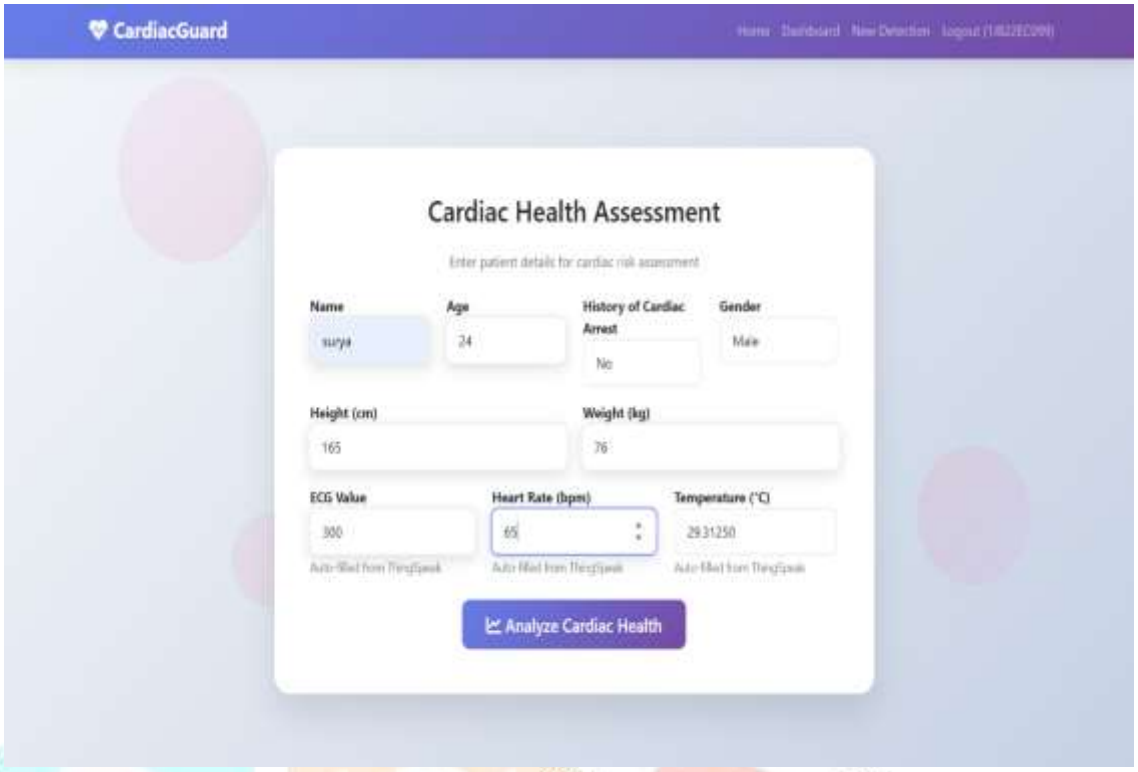


*Fig12.1 The above figure explains about the online platform of the Cardiac Detection. It is the welcome page of the system, where the input values are taken from Thing speak cloud*

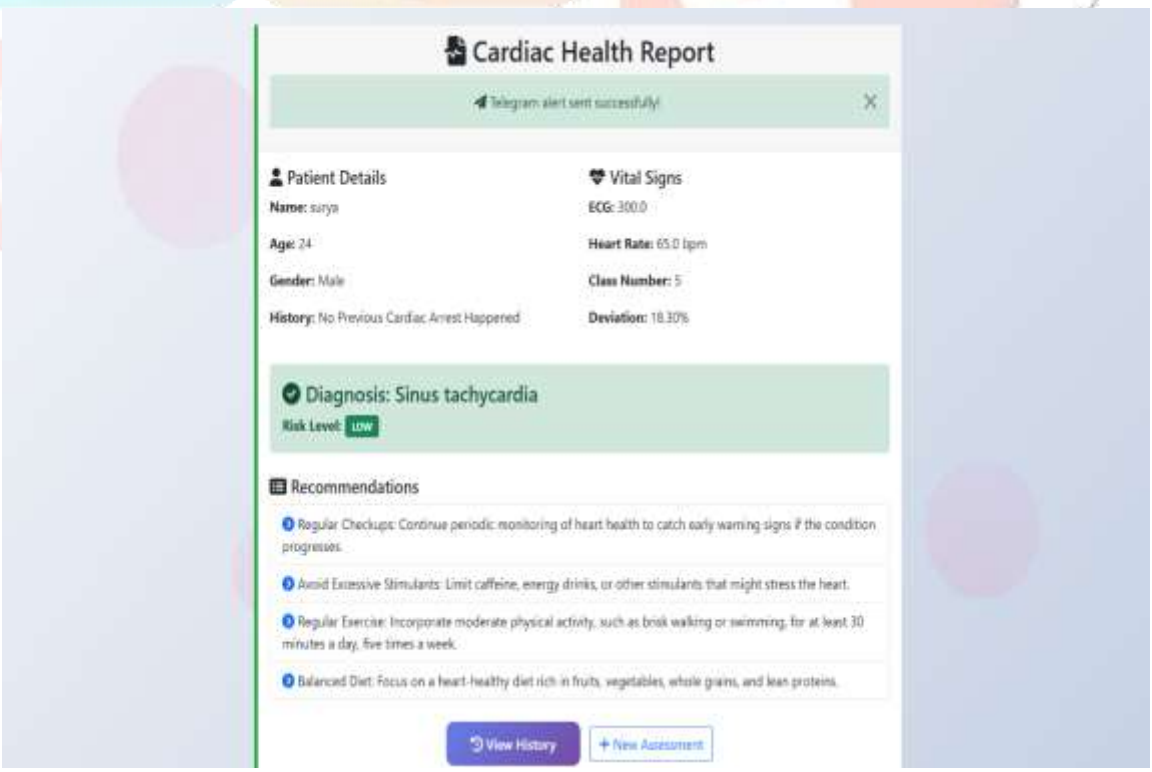


*Fig12.2 shows the Result page of the Cardiac Detection System. It depicts the output after computing the values that are read from Thing Speak Cloud. Predicting low risk*


*Fig12.3 Result displayed in telegram bot*

Patient 3

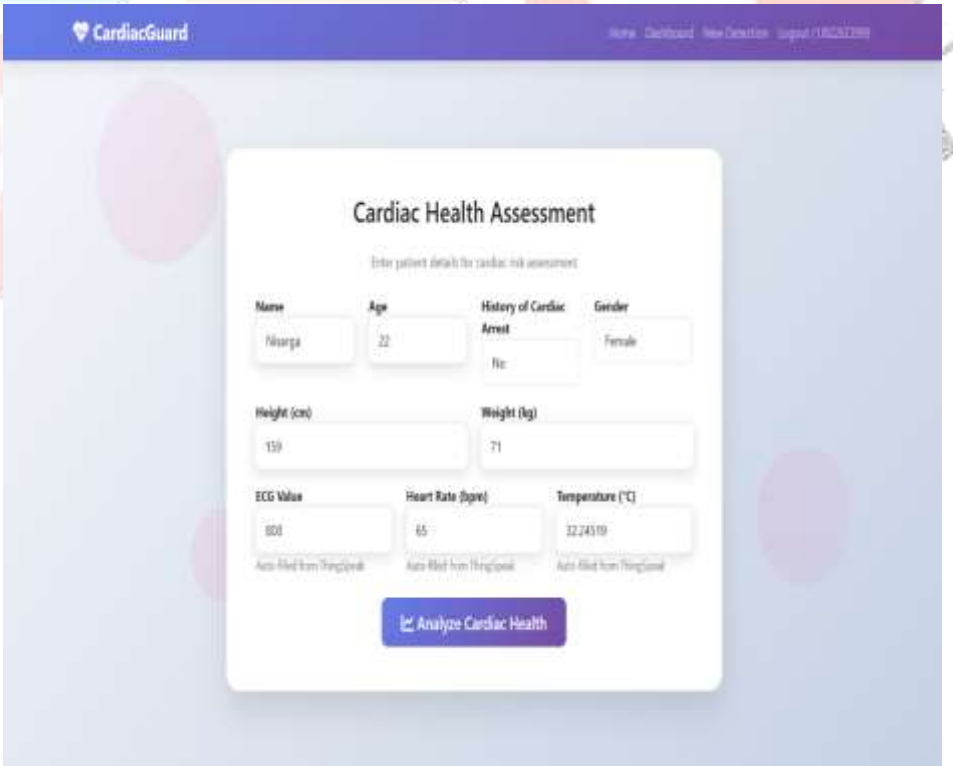


*Fig13.1 explains about the online platform of the Cardiac Detection. It is the welcome page of the system, where the input values are taken from Thing speak cloud*

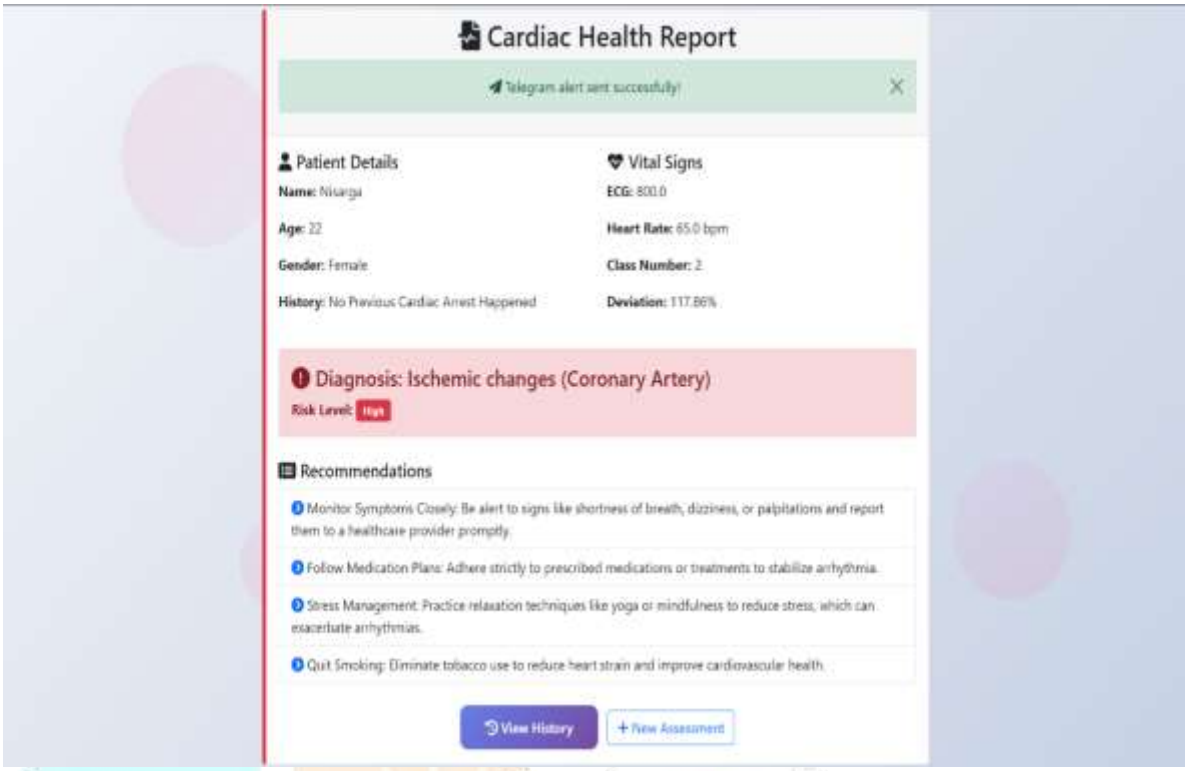


*Fig13.2 shows the Result page of the Cardiac Detection System. It depicts the output after computing the values that are read from Thing Speak Cloud. Predicting High risk and giving recommendations*

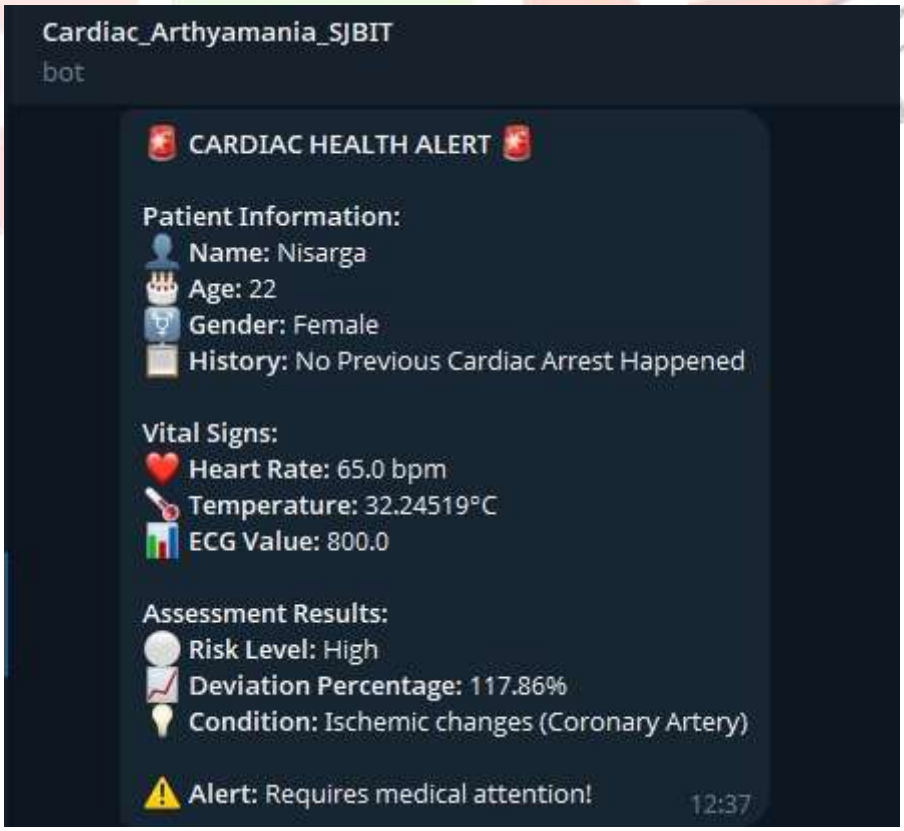


*Fig13.3 Result displayed in telegram bot*

## XII. Conclusion and Future Work

### Conclusion

The findings clearly imply that machine learning integrated with IOT can aid in the identification of Cardiac Arrest. It helps in the identification and prediction of abnormalities in Cardiac Function. The ability to detect cardiac arrest at an early stage would allow for early intervention. It also helps the users and doctors to keep track of the data during emergencies.

### Application

1. Prediction of Atrial Fibrillation The chronic, progressive disease known as enlarged heart failure causes your heart muscles to lose their capacity to pump blood (CHF).
2. Atrial fibrillation develops when the heart chambers do not function effectively becauseof improper electrical transmission, which is a precursor to congestive heart failure.

### Future Work

1. The initiative might be deployed in hospitals and regularly assessed and validated using new patient datasets.
2. The project may become more user-friendly by integrating capabilities that the doctor might require in the near future.
3. The project design may even get simpler with advanced sensors by integrating VLSI.

## XIII. References

[1] Image-Based Cardiac Diagnosis with Machine Learning.

[2] ECG-Based Heart Arrhythmia Diagnosis Through Attentional Convolutional Neural Networks.

[3] A Deep Convolutional Neural Network Model to Classify Heartbeats

[4] Cardiologist-level arrhythmia detection and classification in ambulatory electrocardiograms using a deep neural network.

[5] Deep Learning-Based Arrhythmia Detection in Electrocardiograph

[6] Martin-Isla C, Campello VM, Izquierdo C, Raisi-Estabragh Z, Baeßler B, Petersen SE,Lekadir K. Image-based cardiac diagnosis with machine learning: a review. Front Cardiovasc Med 2020;7:1.

[7] Acharya UR, Oh SL, Hagiwara Y, Tan JH, Adam M, Gertych A, Tan R. A deep convolutional neural network model to classify heartbeats. Comput Biol Med 2017;89:389–396

[8] Attia ZI, Kapa S, Lopez-Jimenez F, McKie PM, Ladewig DJ, Satam G, Pellikka PA, Enriquez-Sarano M, Noseworthy PA, Munger TM, Asirvatham SJ, Scott CG, Carter RE, Friedman PA. Screening for cardiac contractile dysfunction using an artificial intelligence- enabled electrocardiogram. Nat Med 2019;25:70–74.

[9] Martin-Isla C, Campello VM, Izquierdo C, Raisi-Estabragh Z, Baeßler B, Petersen SE,Lekadir K. Image-based cardiac diagnosis with machine learning: a review. Front Cardiovasc Med 2020;7:1.

[10] Schla¨pfer J, Wellens HJ. Computer-interpreted electrocardiograms benefits andlimitations. J Am Coll Cardiol 2017;70:1183–1192.

[11] Acharya UR, Oh SL, Hagiwara Y, Tan JH, Adam M, Gertych A, Tan R. A deep convolutional neural network model to classify heartbeats. Comput Biol Med 2017;89:389–396.

[12] Attia ZI, Sugrue A, Asirvatham SJ, Ackerman MJ, Kapa S, Friedman PA, Noseworthy P. Noninvasive assessment of dofetilide plasma concentration using a deep learning (neural network) analysis of the surface electrocardiogram: a proof of concept study. PLoS One 2018;13:e0201059.

[13] William AD, Kanbour M, Callahan T, Bhargava M, Varma N, Rickard J, Saliba W, Wolski K. Assessing the accuracy of anautomated atrial fibrillation detection algorithm using smartphone technology: the iREADstudy. Heart Rhythm 2018;15:1561–1565.