Cairo University

Faculty of Computers and Artificial Intelligence

Advisory-based system for early heart diagnoses

Under Supervision of:

DR: Mona Soliman
Eng. Belal Hassan

Prepared By:

Adel Abdelmonem Arfa	20190280
Marwan Mohamed Abdelmonem	20190513
Rabab Soliman khedr	20190200
Yomna Taher Abdallah	20190624
Youssef Hesham Mohamed	20190648

Artificial Intelligence Department
Academic Year 2022/2023
Jul 8^{th}, 2023

Table of Contents

Table of Figures

List of Tables

Abbreviations list

1. **ECG (Electrocardiography)**
2. **BRFSS (The Behavioral Risk Factor Surveillance System)**
3. **AFIB (Atrial fibrillation)**
4. **XML (Extensible Markup Language)**
5. **CSV (comma-separated values)**
6. **WFDB (Waveform Database Software Package(**
7. **CNN (convolutional neural network)**
8. **ERD (Entity Relationship Diagram)**
9. **API (application programming interface)**
10. **VR (Virtual reality)**
11. **URL (Uniform Resource Locator)**
12. **JSON (JavaScript Object Notation)**
13. **MVC (Model View Controller)**
14. **UI (User Interface)**
15. **KNN (K-Nearest Neighbors)**
16. **ROC (receiver operating characteristic)**
17. **AUC (Area Under the Curve)**
18. **EHRs (Electronic health records)**

Acknowledgment

First and foremost, we would like to thank **ALLAH** who always sustained us during our lives.

This graduation project would not have been accomplished without the help of numerous people. Some contributed directly to the work and some indirectly, and all their contributions have been valuable.

We gratefully acknowledge ***Dr. Mona Soliman*** for his advice, supervision, and crucial contribution, which made her the backbone of this graduation project. Her involvement with her originality has triggered and nourished our intellectual maturity that we will benefit from, for a long time to come.

Special thanks to ***Eng***. ***Belal Hassan*** for his marvelous efforts with us, and for always being there for us, supporting and helping us throughout our project, we won't have accomplished many things without her guidance.

Many thanks to our parents for their support and motivation through our whole lives and their existence, there are no words that can express our thanks for them. We are grateful in every possible way and hope to keep up our collaboration in the future with our supervisors. Finally, we would like to thank everyone who supported the work to be successful.

Abstract

The project "Baymax" aims to address the pressing issue of heart diseases by utilizing AI-based machine learning algorithms and neural network models for accurate prediction and early diagnosis. With the advancement of technology and the emergence of AI, healthcare has been revolutionized, enabling improved healthcare practices, disease prediction, and enhanced patient care. The potential benefits of AI in healthcare are significant, including automation of tasks, analysis of large patient datasets, and improved efficiency in healthcare delivery.

Heart diseases are a major cause of mortality globally, with a significant percentage of deaths resulting from late diagnosis or insufficient attention to health. The objective of the project is threefold: to reduce the number of deaths from heart diseases by detecting and diagnosing them in the early stages, to prevent healthy individuals from developing heart diseases, and to promote advancements in the medical field.

The project's proposed solution involves the advisor of a doctor robot named "Baymax." Baymax acts as a programmed system that provides health guidance to individuals and detailed insights to doctors based on patient records and various features. By leveraging machine learning algorithms, Baymax enables early detection and diagnosis of heart diseases, facilitating timely intervention and treatment. It also educates and guides individuals in maintaining a healthy lifestyle.

To achieve these objectives, extensive data on heart diseases, symptoms, and patient records are collected and analyzed to identify the most relevant features for the predictive models. By harnessing the power of AI, the project aims to provide accurate predictions and valuable insights to both doctors and individuals, improving decision-making and promoting proactive healthcare practices.

Through the implementation of Baymax, this project strives to reduce the mortality rate associated with heart diseases, safeguard the health of individuals, and contribute to the advancement of the medical field. By emphasizing early detection and prevention, the project aims to save lives and enhance overall healthcare outcomes.

Chapter 1: Introduction

Healthcare has been very important since the beginning of civilizations from the Desire of living longer and the capabilities of maintaining a good health For our own sake and for the society as well And with the rapidly advancement in technology and the emergence of AI the limits of achieving a good healthcare way back becomes more adverse From autonomous maintenance to the prediction of future diseases And how to control it , finding cures , Dive deeper in the understanding of our human nature and how we can elevate to other dimensions where this was only from the medical point of view , from the market perspective it can provide a promising opportunities for vast medical society (pharmacies,doctors,surgeons,students,etc..) and it would impact and thrive the medical business AI can add value by either automating or augmenting the work of clinicians and staff. Many repetitive tasks will be fully automated, and AI will help health professionals perform better at their jobs and improve outcomes for patients.

AI in healthcare provides many benefits, including automating tasks and analyzing big patient data sets to deliver better healthcare faster, and at a lower cost. According to Insider Intelligence, 30% of healthcare costs are associated with administrative tasks. Finding more efficient ways to modernize our healthcare ecosystems, AI will have a profound impact in creating more efficiencies and breakthroughs than today we can yet imagine. Many lives every year are lost because of heart disease including late diagnosis or ignorance of their health which can reach up to 31% of deaths around the world and 48% in Egypt .

And so here comes our project Baymax, our Baymax is a doctor robot programed system which informs you of your health and how to maintain it by providing guidance to normal people through professional regime and to doctors by giving insights and detailed information about each patient record regarding different features which enables doctors to enhance decision making regardless missing values or complex data that embeds a hidden value within it. Our main objective is to reduce the number of deaths from heart diseases by diagnosing them in the early stages and trying to help these people. The second one is to protect healthy people from getting infected with any heart diseases. The third one is to highlight the medical field and give it some of our attention. [1]

1.1. Problem

Many lives every year are lost because of heart disease including late diagnosis or don't care for their health which can reach 31% of deaths around the world.[2]

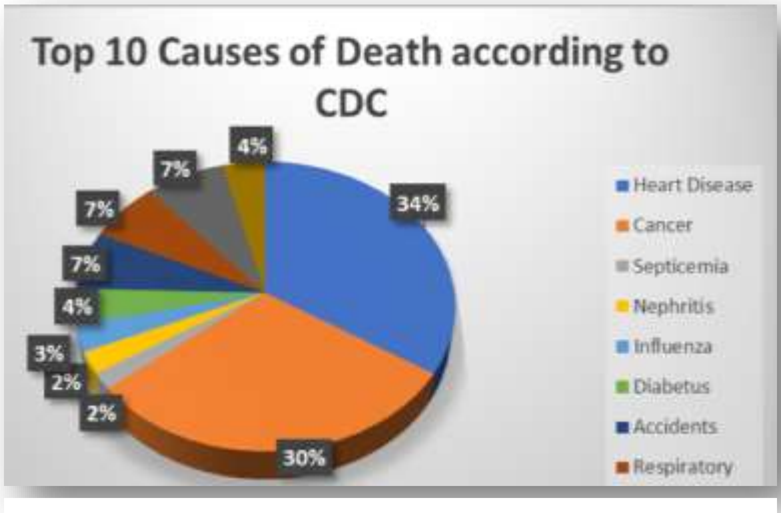


Figure 1: Top 10 Causes of Death

1.2. Scope

We try to collect a lot of data about the heart diseases for healthy people or patients, also focusing on the diseases and symptoms to extract the best features for our model; we try all of these to build our model with high learning.

1.3. Objectives

Our main objective is to reduce the number of deaths from heart diseases by diagnosing them in the early stages and trying to help these people.
The second one is to protect healthy people from getting infected with any heart diseases.
The third one is to highlight the medical field and give it some of our attention.

1.4. Solution

As we see that maybe many people die by heart attack without any warning before, so the solution is to try to diagnose this failure before becoming dangerous throw two ways:

- If the person has some symptoms but he didn't know if he has heart diseases, he can share these symptoms to the model, and it will give him a prediction desiccation if he has heart diseases or not.
- Second one is when the person get check and has ECG signals for his heart, he can give the model these signals, and the model will predict which heart diseases he has and, in this state, he also can give the model some symptoms he has to use them in the predict.

1.5. Methodology definition

For this system, we are going to follow the waterfall approach. It is a rigid linear model which consists of sequential phases: requirement gathering, analysis, design, implementation, deployment, and maintenance as shown in Figure [2]. Each phase must be 100% complete before the next phase can start. Mostly, there is a stage after each phase when the requirements are reviewed and approved by the user. The waterfall method makes it easier to understand and manage the project. It also suits the project with well- 3 defined requirements. On the other hand, its rigid structure, and tight controls, might make the development slow and more costly . In the waterfall model, we start with the requirement gathering phase which includes understanding and defining the problem domain and finding the basic requirements as we did in this chapter. Second the analysis phase, we explain in more detail the requirements of the system, interact with the user gathering as much information on him and understand his mental model to develop what he wants. Third design phase, we design the system architecture and high-level technical details of the project such as class diagram. In addition, plan the technologies we will be using such as programming languages, which database, etc. The Fourth phase is implementation where we will code and implement the system, discussing the models and algorithms we used. In the fifth testing phase, we test the system to verify that it is built as per the requirements and specifications that have been agreed on. The sixth deployment phase is when the system will be launched in the respective environment to be used. Finally, the maintenance phase is when the system is in use, and it is required to change the code.

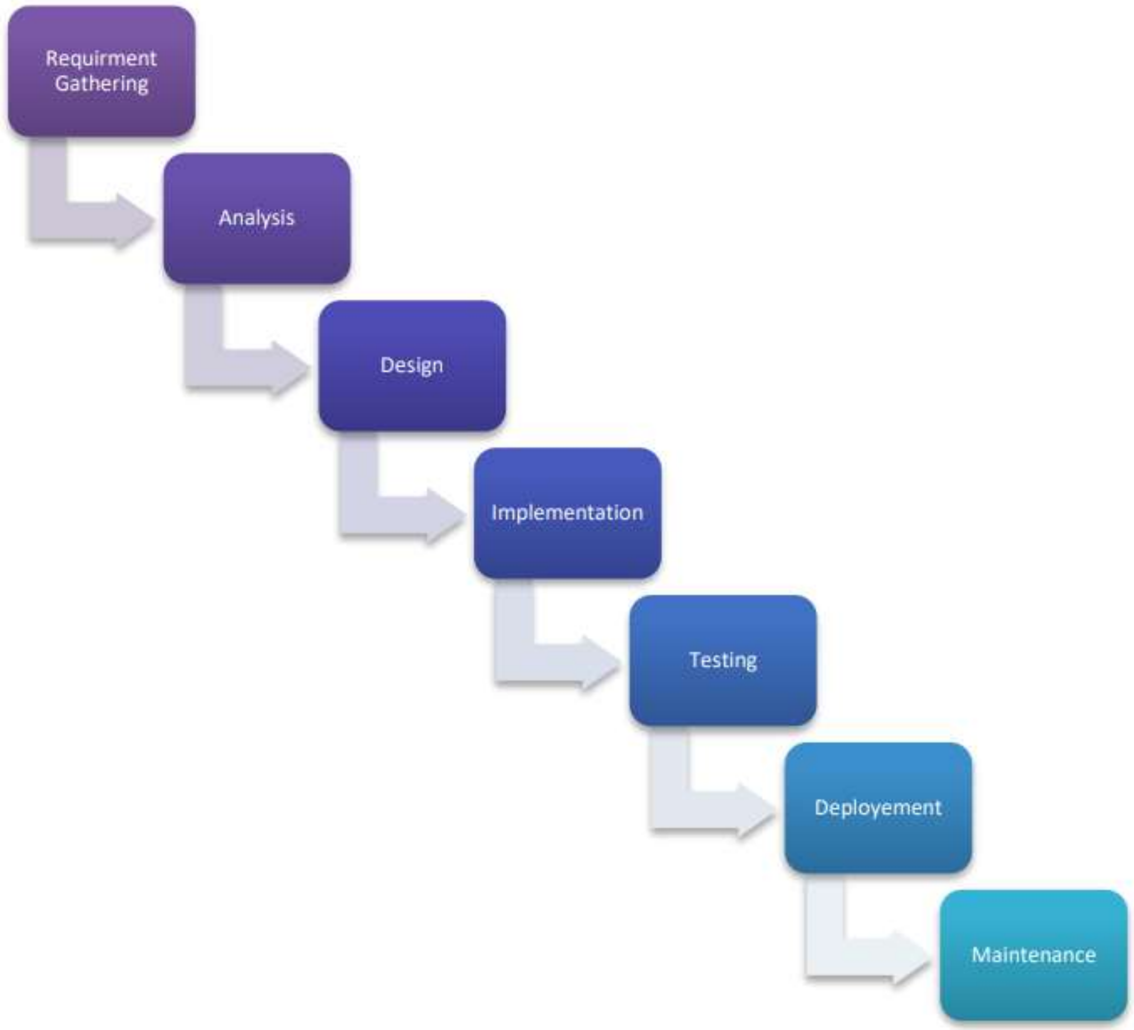


Figure 2: Waterfall Methodology

Chapter 2: Related work

ECG Signal Classification Using Deep Learning Techniques Based on the PTB-XL

The paper focuses on the automatic classification of primary electrocardiogram (ECG) signals using deep neural networks. The study utilizes the PTB-XL database, which contains a large collection of clinical 12-lead ECGs. Three neural network architectures are proposed: a convolutional network, SincNet, and a convolutional network with additional entropy-based features. The dataset is divided into training, validation, and test sets, and experiments are conducted for 2, 5, and 20 classes of disease entities. The results show that the convolutional network with entropy features achieves the best classification performance, while the convolutional network without entropy-based features exhibits higher computational efficiency. The study highlights the importance of machine learning algorithms in accurately and automatically diagnosing cardiovascular diseases using ECG signals.

Information Theory for Non-Stationary Processes with Stationary Increments

We describe how to analyze the wide class of non-stationary processes with stationary centered increments using Shannon information theory. To do so, we use a practical viewpoint and define ersatz quantities from time-averaged probability distributions. These ersatz versions of entropy, mutual information, and entropy rate can be estimated when only a single realization of the process is available. We abundantly illustrate our approach by analyzing Gaussian and non-Gaussian self-similar signals, as well as multi-fractal signals. Using Gaussian signals allows us to check that our approach is robust in the sense that all quantities behave as expected from analytical derivations. Using the stationarity (independence on the integration time) of the ersatz entropy rate, we show that this quantity is not only able to fine probe the self-similarity of the process, but also offers a new way to quantify the multi-fractality.

IBM Clinical Development

IBM Watson Health is a division of IBM that provides data, analytics and AI solutions for the health industry. It aims to help providers, payers, governments and life science companies improve health outcomes, reduce costs and enhance experiences for people around the world. IBM Watson Health was formed in 2015, after IBM acquired several health-related companies, such as Phytel, Explorys and Merge Healthcare. Some of the products and services offered by IBM Watson Health like:

A cloud-based platform that supports clinical trials with electronic data capture, medical coding, and more.

Figure 1 is an excellent app for medical diagnosis. This app is designed for doctors to diagnose an illness. The app has a huge repository of disease images which are extremely effective in positively identifying rare and uncommon diseases. This app is designed to detect and diagnose chronic pain too. Doctors can use Figure 1 to share case studies of early detection and use the video for additional learning. This app is meant for doctors and medical associates and helps healthcare professionals stay connected. [3]

Figure3 : Figure 1 Application

UpToDate is an app that gives you access to a plethora of medical information from its enormous database. You can easily search through more than 8,500 topics, place bookmarks at relevant points, and retrieve recently viewed information quickly. In addition, more than 4,400 experts discuss and update the topics regularly. This huge repository of information is extremely useful to make clinical decisions and correct diagnoses. [4]

Figure4 : UpToDate Application

Chapter 3: Methodology

3.1. The Datasets

The main idea overall is for health care using the ai techniques and tools to diagnosis the disease in the first phases of it, and in this system, we focus on the heart diseases as it is very important to detect them early.

The system has two branches. These two branches use two different data types with two different AI models. The first dataset is numerical data which refers to the medical analysis of the patient, this data is made of merging two datasets from Kagel, this model is friendly for any user because it just asks you some questions that can be answered very easy and can be used by the doctors.

The second branch is ECG signals which we get from reference [11], this type of data cannot be easily to access it, so the main user for this branch is doctor, and anyone with ECG signals data.

3.1.1. Heart Disease Health Indicators Dataset

Our clinical data body is a merger between two authorized datasets. The first is dataset contains 253,680 survey responses from the cleaned BRFSS 2015 dataset. The primary focus of this dataset is the binary classification of heart disease. It's important to note that there is a strong class imbalance in this dataset, with 229,787 respondents not having or having never had heart disease, while 23,893 respondents have had heart disease.[5],[6]

The project aims to explore the extent to which survey responses from the BRFSS can be used to predict the risk of heart disease. Additionally, it investigates whether a subset of questions from the BRFSS can be used for preventative health screening, particularly for diseases like heart disease.

The other one is a subset of the BRFSS 2015 survey data, focusing on the binary classification of heart disease.

The selected features from the BRFSS 2015 dataset are chosen based on research regarding risk factors for heart disease and other chronic health conditions like diabetes. These risk factors include high blood pressure, high cholesterol, smoking, diabetes, obesity, age, sex, race, diet, exercise, alcohol consumption, BMI, household income, marital status, sleep, time since last checkup, education, health care coverage, and mental health.[8]

3.1.2. ECG dataset

There are three main components to an ECG: the P wave, which represents depolarization of the atria; the QRS complex, which represents depolarization of the ventricles; and the T wave, which represents repolarization of the ventricles.

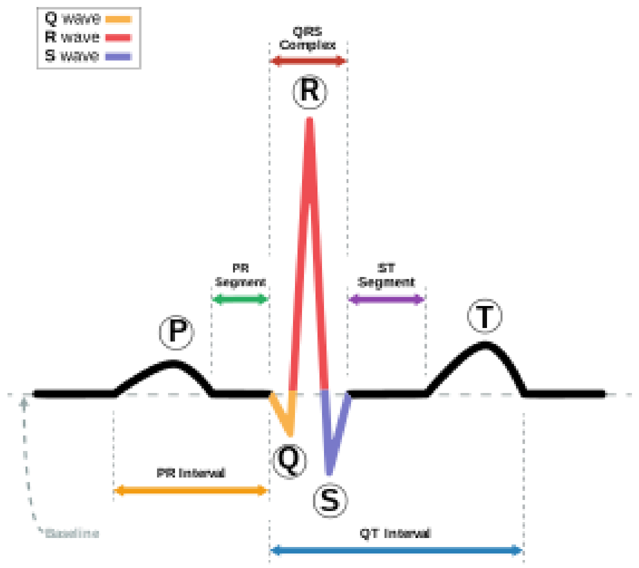
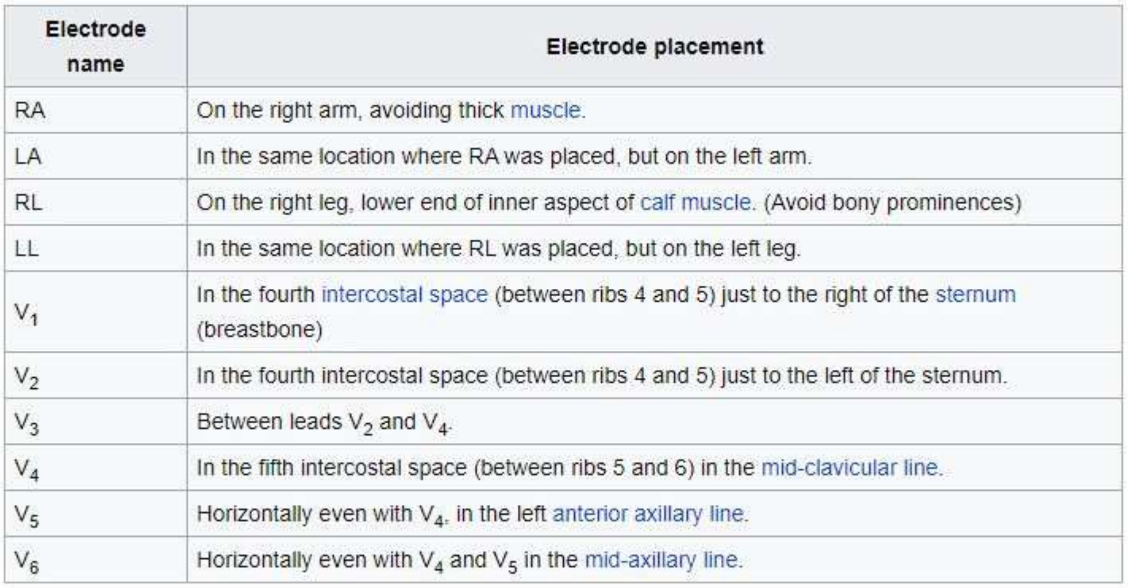


Figure 5: ECG Components

The database consists of 43,777 patient ECGs, each recorded with a 12-lead configuration and a sampling rate of 500 Hz. The ECG graph of a normal beat includes waves such as the P-wave (atrial depolarization), QRS complex (ventricular depolarization), and T-wave (ventricular repolarization), as well as other intervals like PR, ST, and QT. Arrhythmias encompass various cardiac conditions characterized by irregularities in heart rate or rhythm. The most common and severe type of arrhythmia is atrial fibrillation (AFIB), which significantly increases the risk of cardiac dysfunction and stroke.



Electrode name	Electrode placement
RA	On the right arm, avoiding thick muscle.
LA	In the same location where RA was placed, but on the left arm.
RL	On the right leg, lower end of inner aspect of calf muscle. (Avoid bony prominences)
LL	In the same location where RL was placed, but on the left leg.
V_1	In the fourth intercostal space (between ribs 4 and 5) just to the right of the sternum (breastbone)
V_2	In the fourth intercostal space (between ribs 4 and 5) just to the left of the sternum.
V_3	Between leads V_2 and V_4.
V_4	In the fifth intercostal space (between ribs 5 and 6) in the mid-clavicular line.
V_5	Horizontally even with V_4, in the left anterior axillary line.
V_6	Horizontally even with V_4 and V_5 in the mid-axillary line.

Figure 6: Electrode Table

To create the database, each subject underwent a 12-lead resting ECG test, which lasted for 10 seconds. The data was stored in the GE MUSE ECG system. Licensed physicians labeled the rhythm and other cardiac conditions for each subject, and a secondary validation process was performed by another licensed physician. In cases of disagreement, a senior physician made the final decision. The diagnoses, including rhythm and additional conditions, were stored in the MUSE ECG system. The data and diagnostic information were exported from the GE MUSE system to XML files and then converted to CSV format using a specific tool. Finally, the CSV files were converted to the WFDB format.

The database is available in two formats: CSV and WFDB. Initially, a portion of the database (10,646 patients) was shared in CSV format on figshare, including raw and denoised ECG data, diagnoses files, and an attributes dictionary file. Each CSV file contains 5000 rows and 12 columns representing the ECG leads. The WFDB format was later adopted, and the database was moved to the Physionet repository, increasing the number of ECG recordings to 45,152. In the WFDB format, each ECG is represented by a tuple of two files: a mat-file containing the binary raw data and a corresponding header file with annotation information.

The institutional review boards of Shaoxing People's Hospital and Ningbo First Hospital approved the study and granted waivers for obtaining informed consent. After de-identification, the data was allowed to be shared publicly.[11]

Figure 7: ECG Signal

Figure 8: ECG Signal

3.3. Our Models

3.3.1. Gradient Boosting Classifier

Is a popular machine learning algorithm used for classification tasks? It belongs to the ensemble learning methods and is an extension of the Gradient Boosting framework.

The algorithm works by combining multiple weak prediction models, typically decision trees, to create a stronger predictive model. It iteratively builds an ensemble of trees, where each subsequent tree is trained to correct the mistakes made by the previous trees. This process is done by assigning higher weights to the misclassified instances, allowing subsequent trees to focus on those instances and improve their predictions.

Here are some key features and characteristics of the Gradient Boosting Classifier:

- Gradient Boosting: It is a boosting algorithm that minimizes a loss function by optimizing the gradient descent. It sequentially adds models to minimize the overall loss.
- Decision Trees: Gradient Boosting Classifier typically uses decision trees as weak learners. Decision trees are simple binary classifiers that make predictions based on a set of if-else rules learned from the training data.
- Gradient Descent: The algorithm uses gradient descent optimization to minimize the loss function. It calculates the gradients of the loss function with respect to the model's predictions and adjusts the model's parameters in the direction of steepest descent.
- Ensemble Learning: Gradient Boosting Classifier combines multiple weak learners to create a strong ensemble model. By

iteratively adding trees to the ensemble, the model gradually improves its predictive performance.

- Feature Importance: The algorithm can provide information about the importance of each feature in the classification task. This can help identify the most relevant features for the problem at hand.
- Regularization: Gradient Boosting Classifier provides options for regularization to prevent overfitting. Regularization techniques, such as shrinkage and subsampling, can be applied to control the complexity of the model and improve its generalization ability.
- High Performance: Gradient Boosting Classifier is known for its high predictive accuracy and performance. It is widely used in various domains and can handle large-scale datasets efficiently.

Overall, the Gradient Boosting Classifier is a powerful algorithm that can effectively handle complex classification tasks and provide accurate predictions. It is widely used in practice and has been successful in various applications, including healthcare, finance, and natural language processing.[12]

3.3.2. The updated model

Uses a more in-depth approach as an advisory model which utilizes gradient boosting classifier with probalistic output using predict_proba.

The predict_proba method is a common function provided by many machine learning models for classification tasks. It is used to obtain the probability estimates for each class label of the input instances.

Here's a brief explanation of how predict_proba works:

- Classification Models: predict_proba is typically available in classification models such as logistic regression, random forest, gradient boosting, and support vector machines. These models learn from labeled training data and make predictions on unseen data by assigning class labels to instances.

- Probability Estimates: predict_proba returns an array-like structure that contains the probability estimates for each class label.

The probabilities represent the model's confidence or belief in the prediction for each class. Each instance is associated with a set of probabilities, one for each class label.

- Sum of Probabilities: The probabilities returned by predict_proba are normalized, meaning that they sum up to 1. This allows the probabilities to be interpreted as relative likelihoods or proportions.
- Threshold Selection: The probability estimates can be useful for determining an appropriate decision threshold. By default, models typically assign the class label with the highest probability as the predicted label. However, depending on the application, you may want to adjust the threshold to optimize for a specific trade-off between precision and recall or to minimize the number of false positives or false negatives.
- Confidence and Uncertainty: Probability estimates provide additional information about the model's confidence in its predictions. Higher probabilities indicate greater confidence, while lower probabilities suggest higher uncertainty. This information can be valuable for decision-making or for assessing the reliability of predictions.
- Multi-Class Classification: In multi-class classification, where there are more than two possible classes, predict_proba returns a probability estimate for each class label. This allows you to evaluate the model's confidence in assigning an instance to each class.

- Interpretation: Probability estimates can be interpreted as the likelihood of an instance belonging to a specific class. For example, if the predict_proba method returns [0.3, 0.7] for a binary classification problem, it suggests a 30% probability for the first class and a 70% probability for the second class.

By utilizing the predict_proba method, you can gain insights into the model's confidence and uncertainty, and make more informed decisions based on the probability estimates for each class label.[13]

3.3.3. In the context of signal processing

A Convolutional Network with Entropy Features refers to a modified version of a convolutional neural network (CNN) that incorporates entropy calculations as additional features for each channel of an ECG signal.

The ECG (electrocardiogram) signal is a time-series representation of the electrical activity of the heart. Traditional CNNs are commonly used for image analysis, but they can also be applied to sequential data such as ECG signals.

In this modified network, the ECG signal is fed into the convolutional layers, which are responsible for extracting local features from the input data. These convolutional layers apply filters across the signal to capture relevant patterns and information.

In addition to the standard convolutional operations, entropy calculations are performed for each channel of the ECG signal. Entropy is a measure of the uncertainty or randomness in a signal. By computing the entropy for each channel, the network can capture the complexity or irregularity of the signal in a quantitative manner.

The entropy values calculated for each channel are then treated as additional features alongside the raw ECG signal. This extended feature set is used as input to the subsequent layers of the network, which may include fully connected layers for classification or further analysis. By incorporating entropy features, the network can potentially capture more comprehensive information about the ECG signal's complexity and patterns. This can be particularly useful for tasks such as arrhythmia detection, anomaly identification, or other types of ECG signal analysis that benefit from considering both the raw signal data and its entropy characteristics.

3.4. Use case

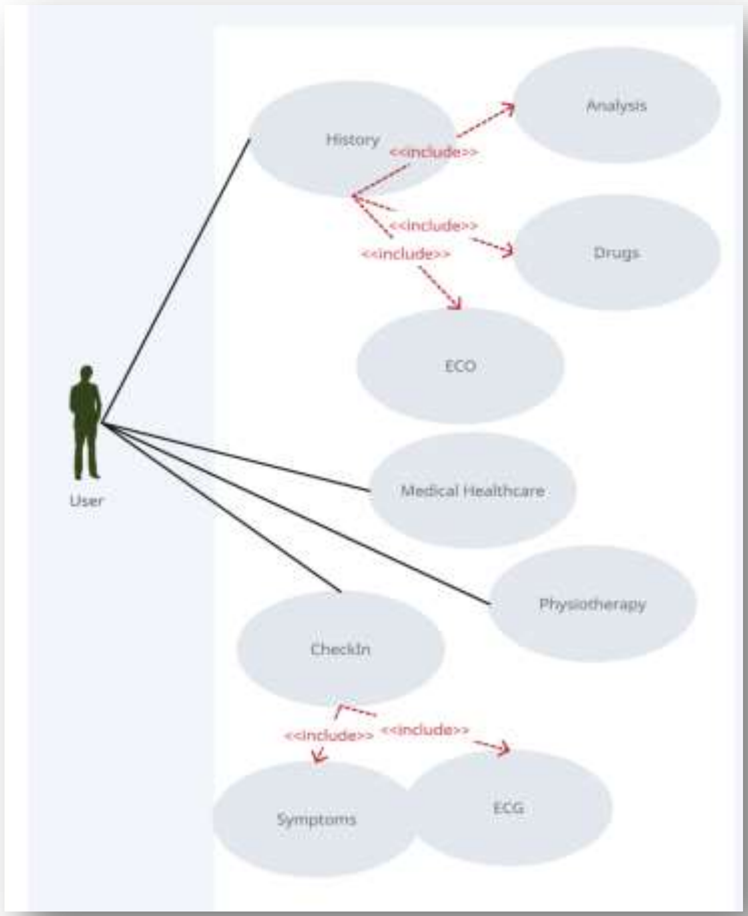


Figure 9: Use Case Diagram

3.5. Sequence Diagram

3.5.1. Symptoms Feature

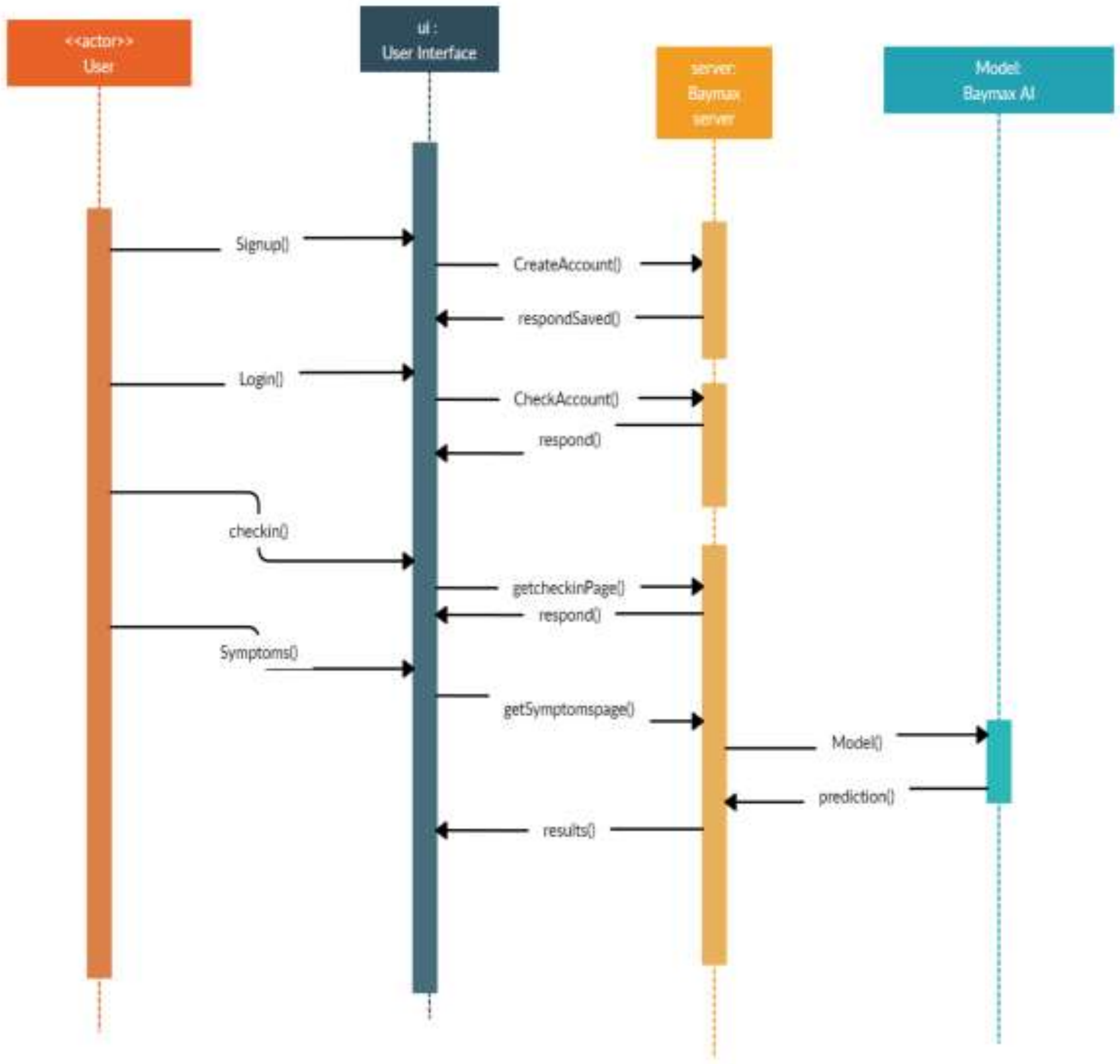


Figure 10: Symptoms Feature Sequence Diagram

3.5.2. ECG Feature

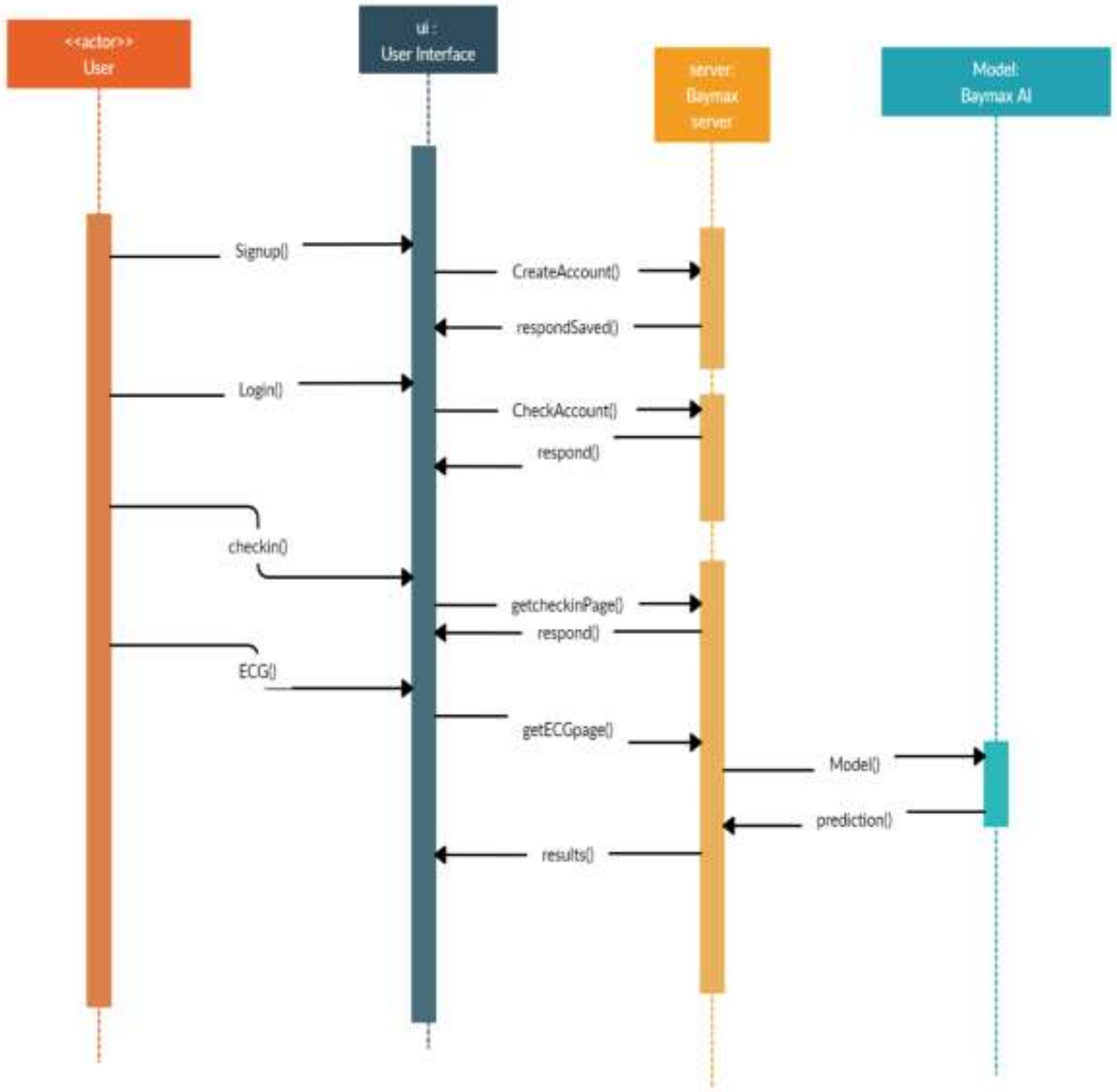


Figure 11: ECG Feature Sequence Diagram

3.5.3. History (analysis) Feature

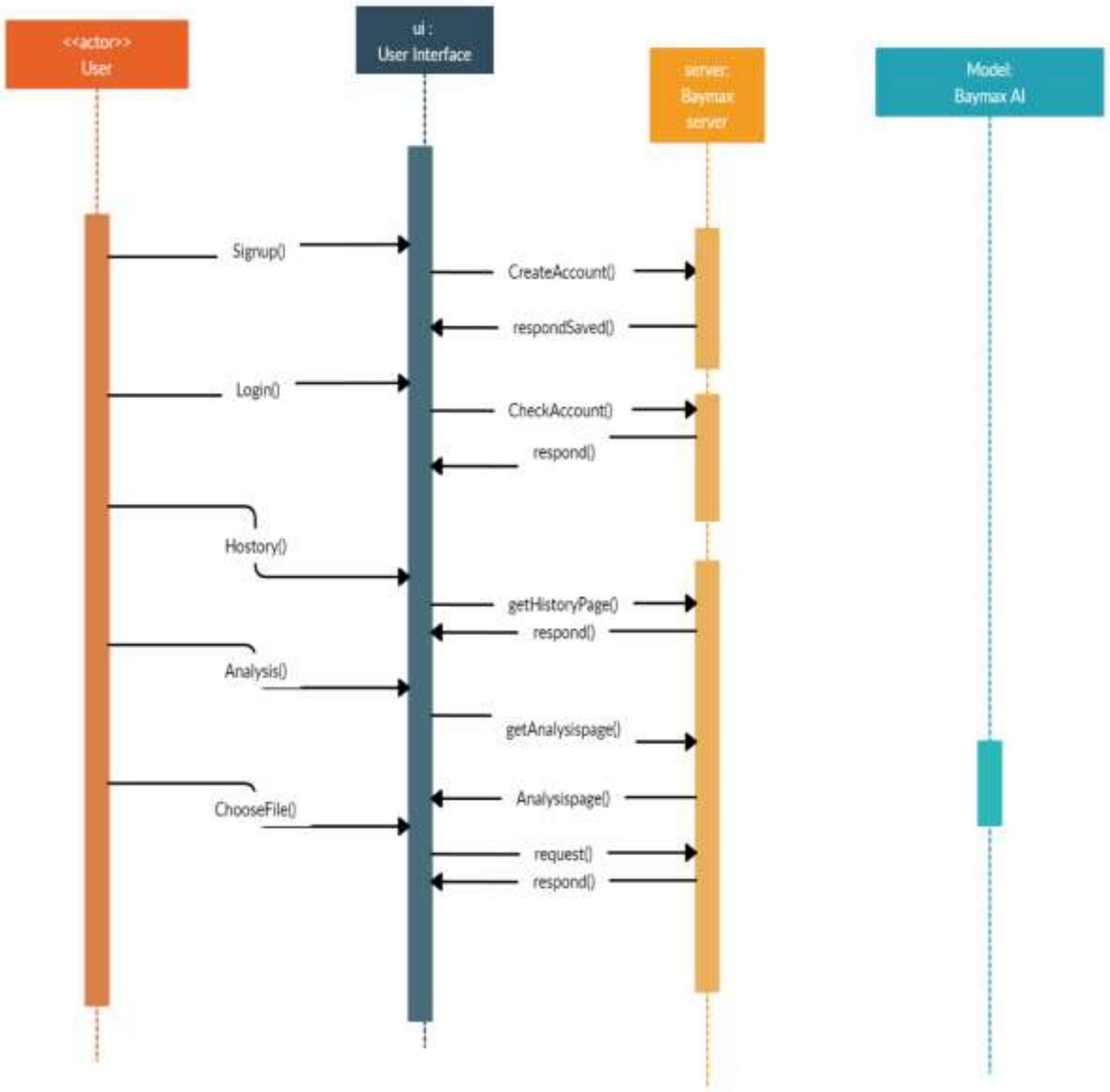


Figure 12: History (analysis) Feature Sequence Diagram

3.5.4. History (medicines) Feature

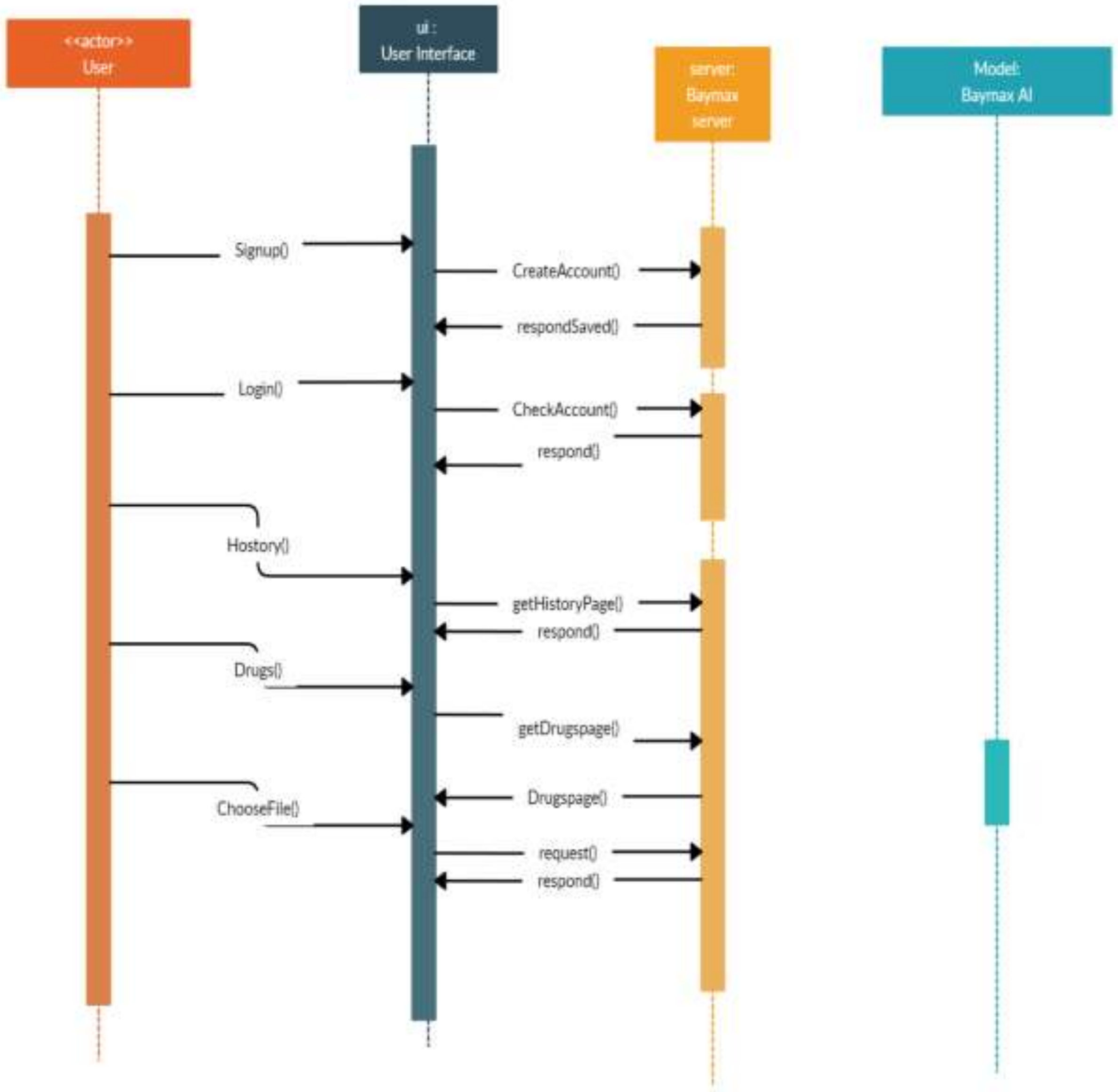


Figure 13: History (medicines) Feature Sequence Diagram

3.5.5. History (ECO) Feature

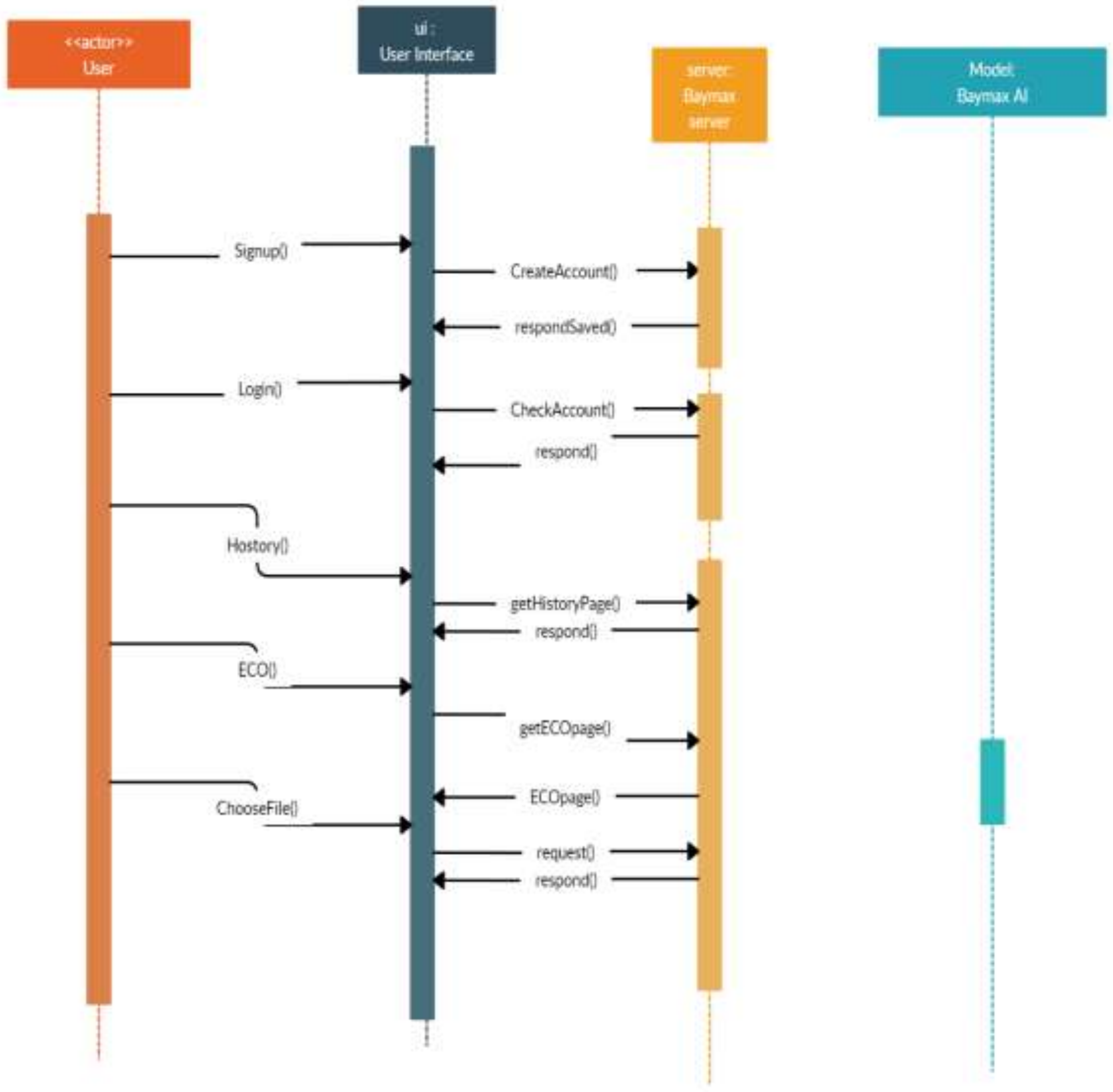


Figure 14: History (ECO) Feature Sequence Diagram

3.5.6. What? Feature

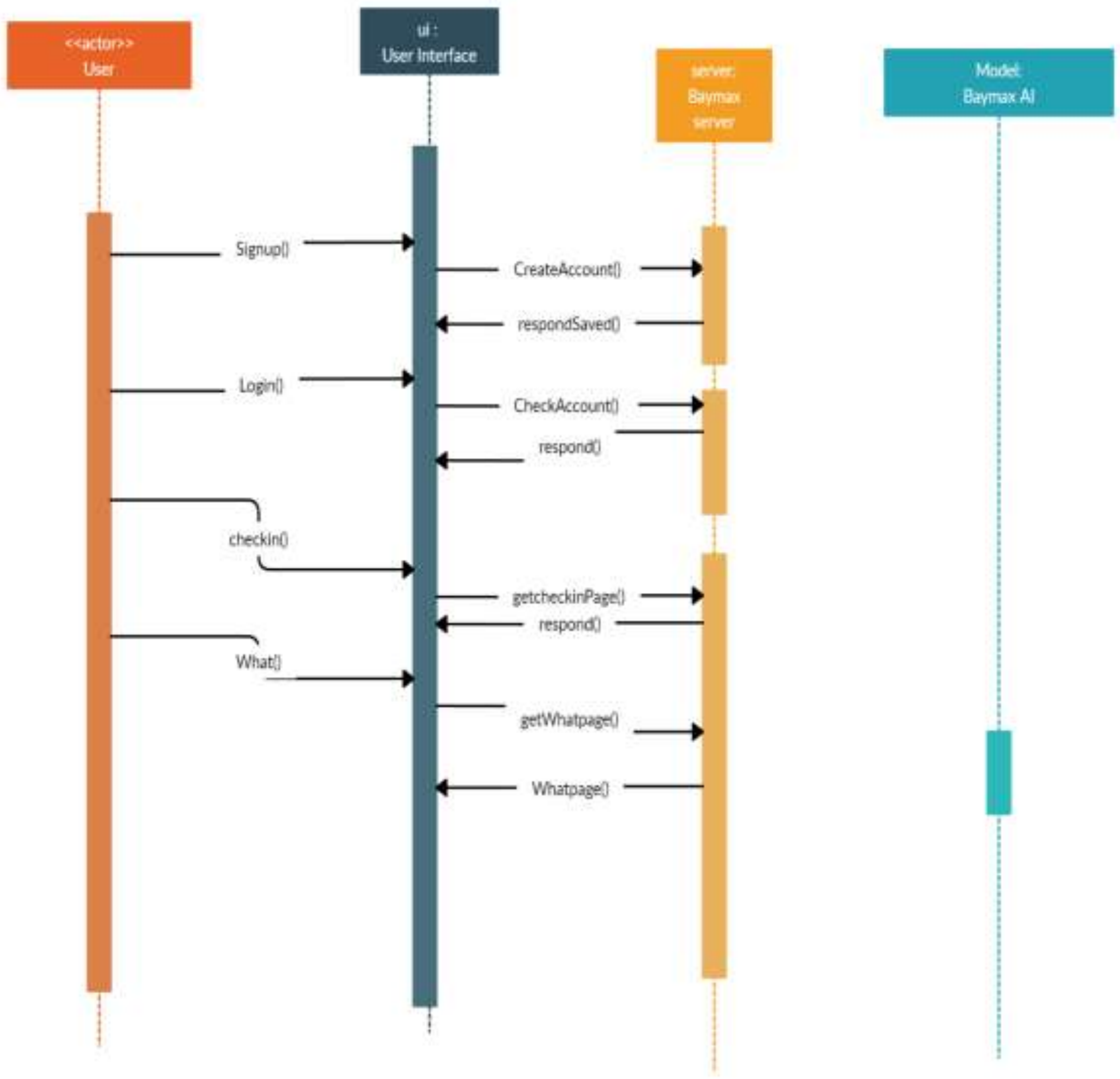


Figure 15: What? Feature Sequence Diagram

3.6. Entity Relationship Diagram (ERD)

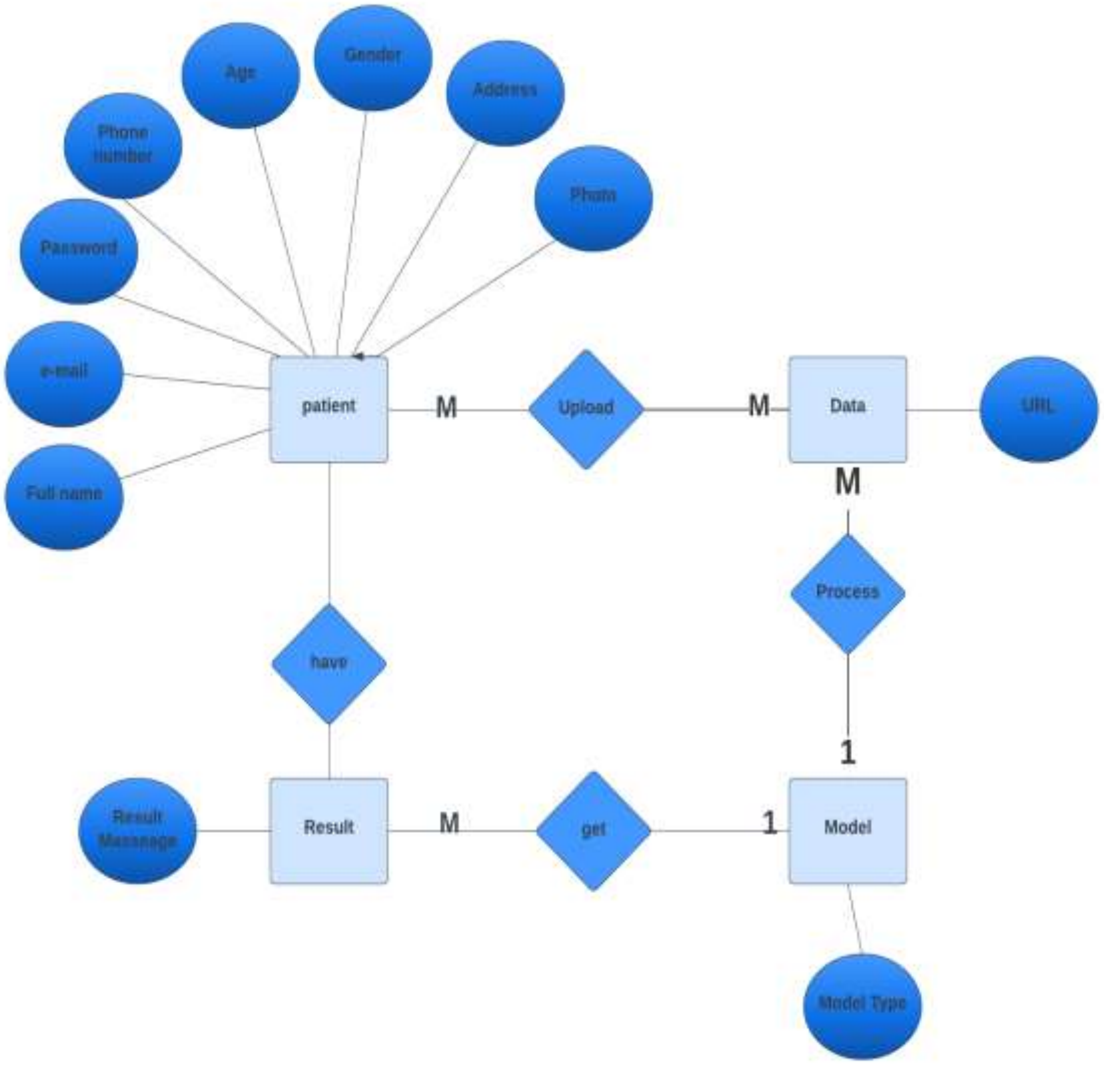


Figure 16: Entity Relationship Diagram (ERD)

3.7. User Interface Design

This is the user interface that we imagine for our design of our system which we will try to implement it.

Service Layer (Backend API):

The backend is following the MVC architectural pattern. Which consist of three parts: Model, View, and Controller. The Model handles data logic. While the View displays the information from the model to the user. And the Controller controls the data flow into a model object and updates the view whenever data changes.

• **View Layer:** Given that the backend is considered a separate API that can have multiple presentation layers, such as web, mobile, or even VR, also The API is encapsulated and only accessed through a façade interface. The view layer isn't much of a visual layer but an interface that can be called and accessed by a specific URL called by whichever front-end. In addition, the feedback or return results are just JSON files that will be received by the front-end to view or manipulate.

• **Controller Layer:** handles the view requests, the data flow from the model to the view and vice versa and returns appropriate responses. Also, it applies most of the business logic and rules

• **Model Layer:** is responsible for handling data and business logic, representing the shape of the data, and handling database related changes.

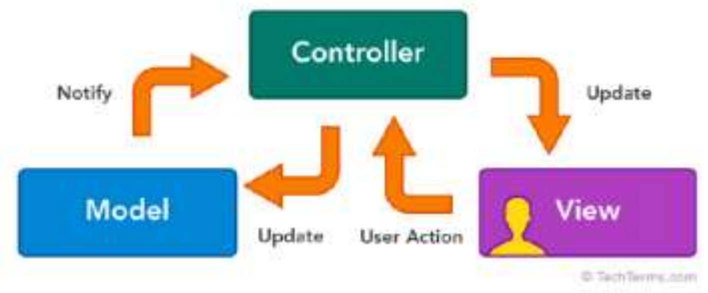
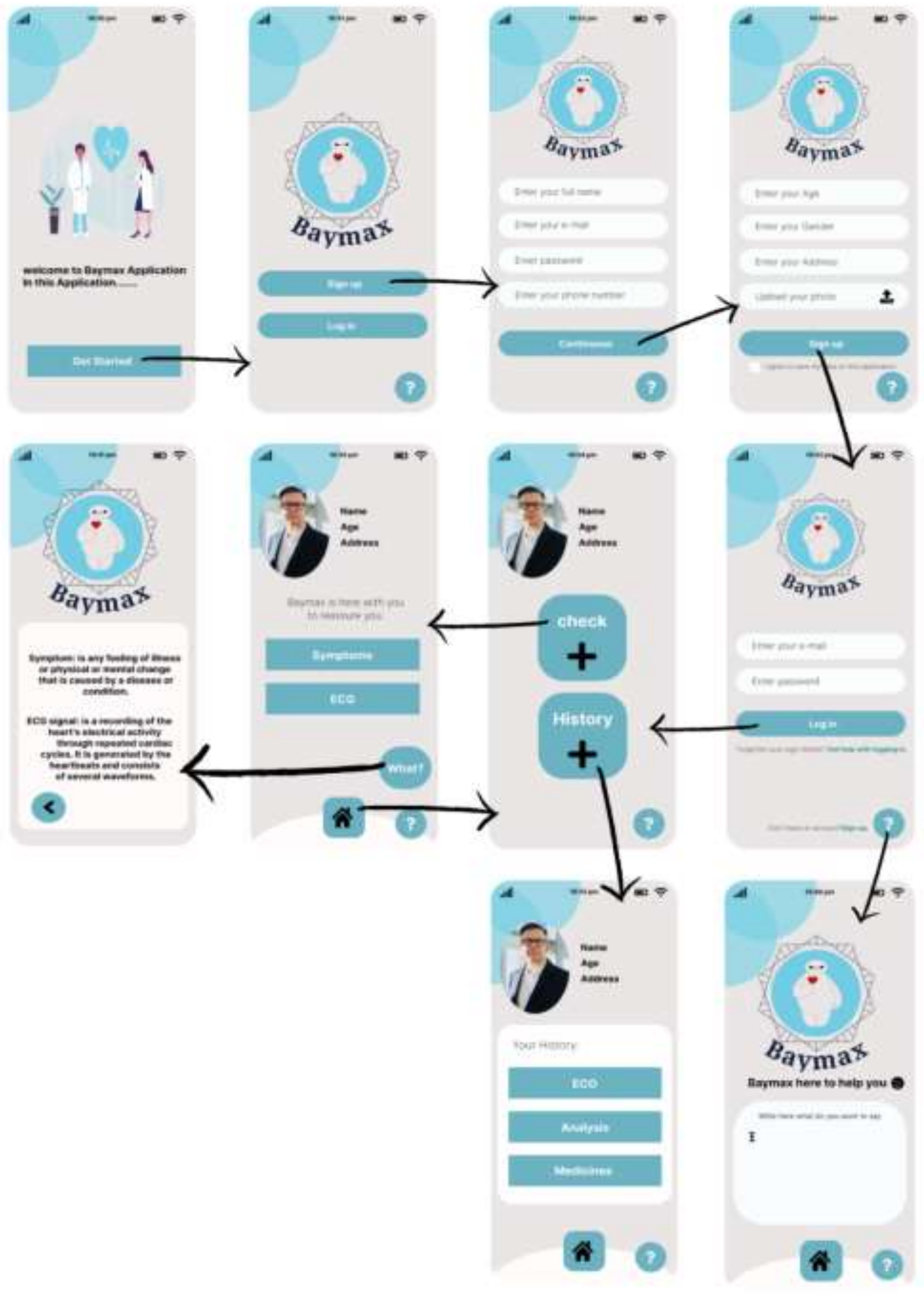


Figure 17: Backend Architecture MVC

3.7.1. Application User Interface (UI)



Symptom: is any feeling of illness or physical or mental change that is caused by a diseases or condition.

ECG signal: is a recording of the heart's electrical activity through repeated cardiac cycles. It is generated by the heartbeats and consists of several waveforms.

Figure 18: User Interface part 1

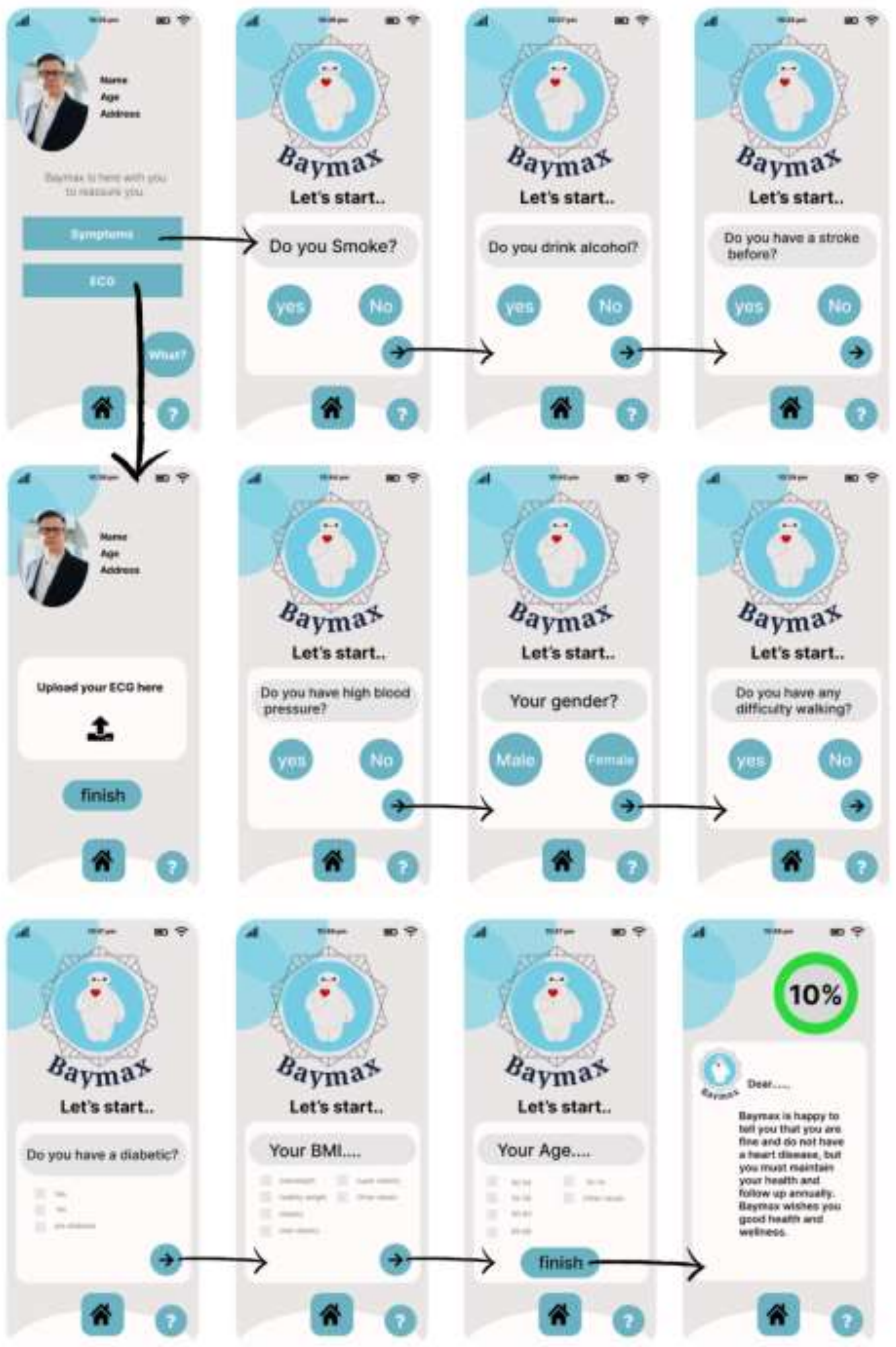

Name
Age
Address
Baymax is here with you to measure you
Symptoms
ECG
What?
Baymax
Let's start..
Do you Smoke?
yes
No
Do you drink alcohol?
Do you have a stroke before?
Upload your ECG here
finish
Do you have high blood pressure?
Your gender?
Male
Female
Do you have any difficulty walking?
Do you have a diabetic?
Your BMI....
Your Age....
10%
Dear.....
Baymax is happy to tell you that you are fine and do not have a heart disease, but you must maintain your health and follow up annually. Baymax wishes you good health and wellness.

Figure 19: User Interface part 2

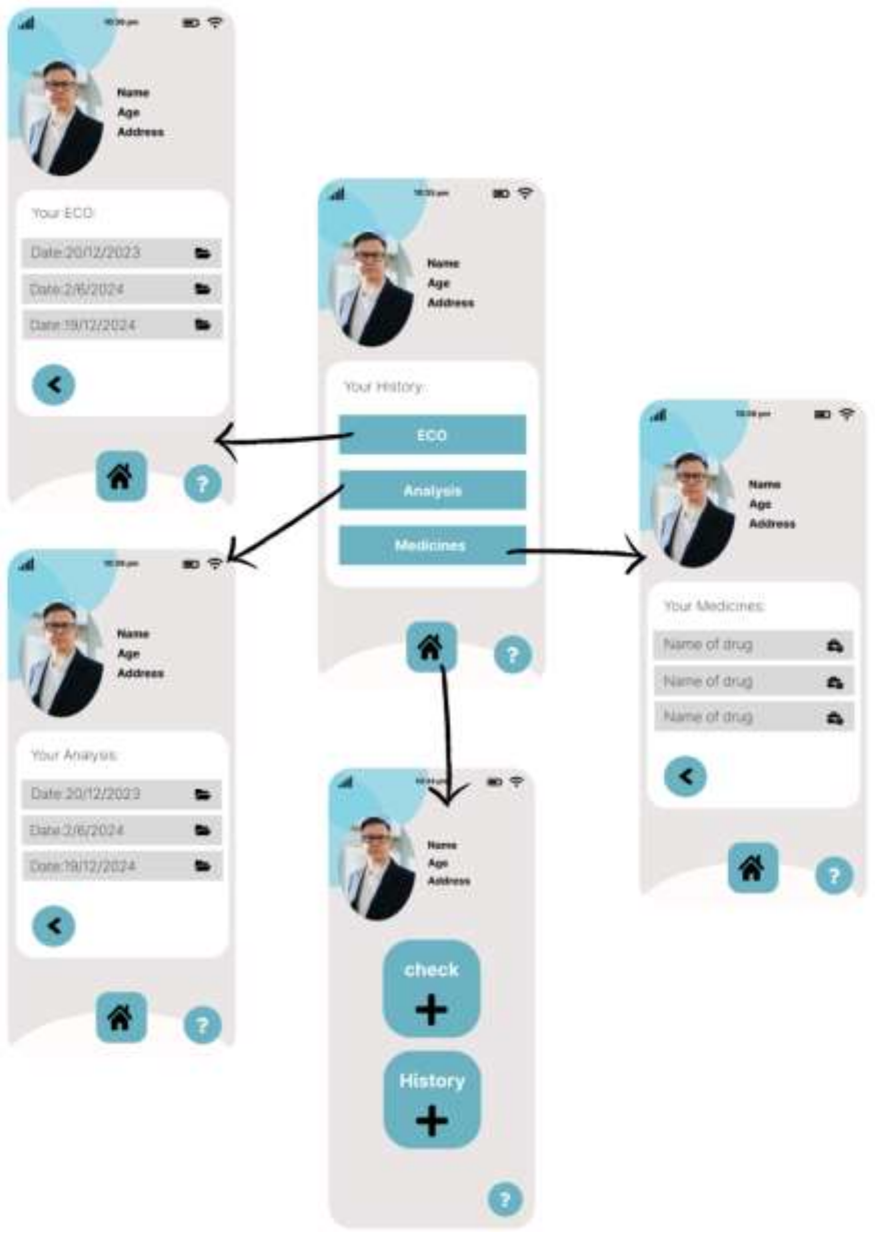


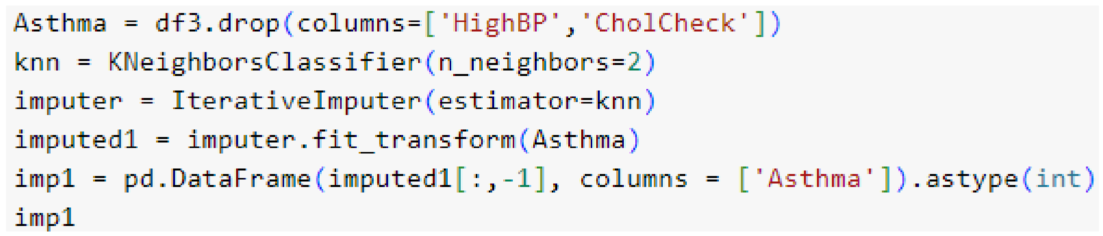
Figure 20: User Interface part 3

3.8 Analysis on Heart Disease Health Indicators Dataset

we know that the medical field need large dataset, so we tried to merge two datasets with some common features, before downing this we dropped some features from the first dataset because they didn't affect very much on our main label like['PhysicalHealth', 'MentalHealth', 'Race', 'GenHealth', 'SkinCancer', 'SleepTime', 'KidneyDisease'], and this after searching about these features and third relations with our main label.
And, after seeing their colorations with it.

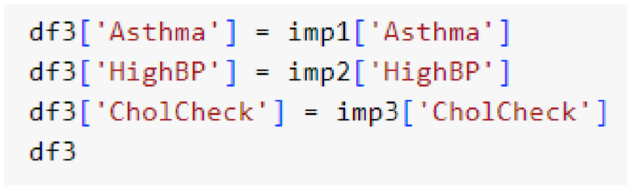
We change the type of label of some features like ['Diabetic'], to be more efficient for our model.
In the second dataset we also drop some features like ['HighChol', 'Fruits', 'Veggies', 'AnyHealthcare', 'NoDocbcCost', 'GenHlth', 'MentHlth', 'PhysHlth', 'Education', 'Income'], because the same ressons of the previous dataset.
Then we change some values of some features like ['HeartDiseaseorAttack', 'Smoker', 'Stroke', 'PhysActivity', 'HvyAlcoholConsump', 'DiffWalk'], to [Yes,No], feature ['Sex'] to ['Male','Female'] and feature ['Age'] to ['18-24', '25-29', '30-34', '35-39', '40-44', '45-49', '50-54', '55-59', '60-64', '65-69', '70-74', '75-79', '80 or older'].
Then do the same analysis from dataset on the feature ['Diabetes'].
Next, we change some name of features to be easy to deal with them.
The next step is merging the two datasets, we change the values of ['BMI'] to [underweight, healthy weight, overweight, obesity, over obesity, super obesity, super super obesity], and this merging make a problem which is some uncommon features between the two datasets have Nan values, which will have bad effects on our results.
So, to solve this problem we need to do imputation for these missing values, this impute is done on three features ['Asthma', 'HighBP', 'CholCheck'].

The KNN is used to impute these values, where dependent on the other features using the definite values of these features ['Asthma', 'HighBP', 'CholCheck'].



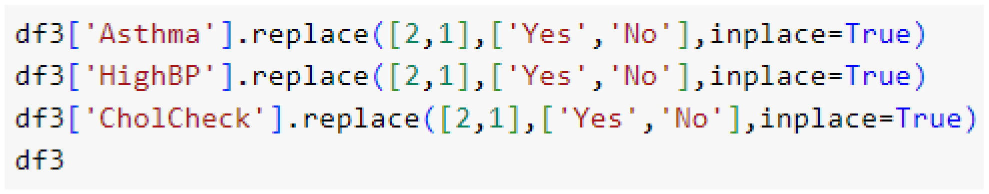
```
Asthma = df3.drop(columns=['HighBP','CholCheck'])
knn = KNeighborsClassifier(n_neighbors=2)
imputer = IterativeImputer(estimator=knn)
imputed1 = imputer.fit_transform(Asthma)
imp1 = pd.DataFrame(imputed1[:,-1], columns = ['Asthma']).astype(int)
imp1
```

Figure 21: Impute 1 (Code)



```
df3['Asthma'] = imp1['Asthma']
df3['HighBP'] = imp2['HighBP']
df3['CholCheck'] = imp3['CholCheck']
df3
```

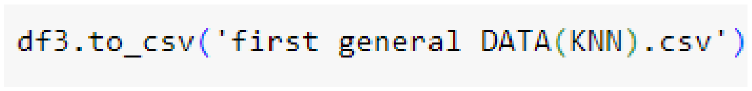
Figure 22: Impute Features (Code)



```
df3['Asthma'].replace([2,1],['Yes','No'],inplace=True)
df3['HighBP'].replace([2,1],['Yes','No'],inplace=True)
df3['CholCheck'].replace([2,1],['Yes','No'],inplace=True)
df3
```

Figure 23: Replace values (Code)

The final step is extracting final dataset to work on it.



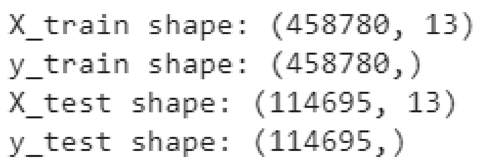
```
df3.to_csv('first general DATA(KNN).csv')
```

Figure 24: New data (Code)

3.9. The first model

Which we used to this model, is starting with loading our new dataset which we generated,

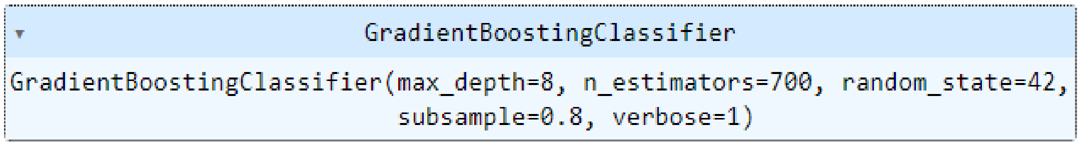
After this split our data to train with ratio 0.8 and testing with 0.2 and random state 42, and this splitting produce these result.



```
X_train shape: (458780, 13)
y_train shape: (458780,)
X_test shape: (114695, 13)
y_test shape: (114695,)
```

Figure 25: Shape of data (Code)

We use the Gradient-Boosting-Classifier to classify our data.



```
▾                    GradientBoostingClassifier
GradientBoostingClassifier(max_depth=8, n_estimators=700, random_state=42,
                           subsample=0.8, verbose=1)
```

Figure 26: Gradient Boosting Classifier

The next step is to find the probability of every sample over all samples.

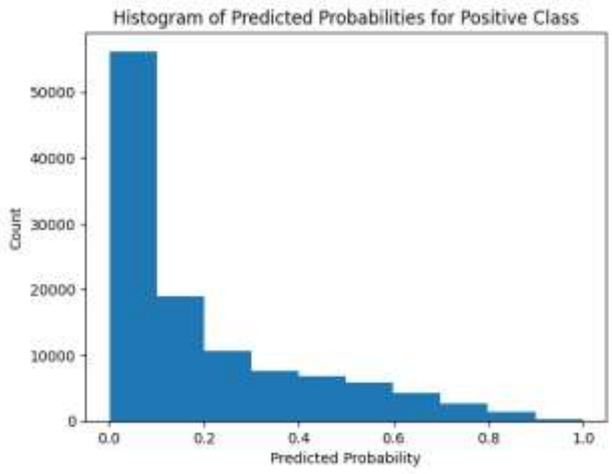
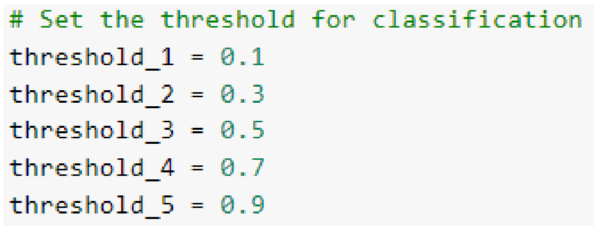


Figure 27: Histogram of Predicted Probabilities for Positive Class

We divided the probability of the samples to five areas under scope of five thresholds.



```
# Set the threshold for classification
threshold_1 = 0.1
threshold_2 = 0.3
threshold_3 = 0.5
threshold_4 = 0.7
threshold_5 = 0.9
```

Figure 28: Thresholds (Code)

Then named these areas with names that refer to their ratio of dangerous and determined a threshold on probability= 0.50 .

The last step is calculating the Precision and Recall for our model to see the result.

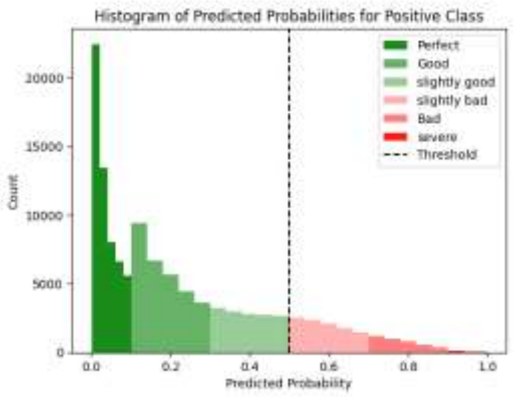


Figure 29: Histogram of Predicted Probabilities for Positive Class

3.10.Analysis on ECG dataset

We divided the dataset into eight batches with random seed so that we generalize our model and do not overfit.

The low_pass_filter function applies a moving

average low-pass filter to a 1D array of voltages. This type of filter is commonly used in signal processing to attenuate high-frequency components and retain low-frequency components. The function takes two parameters: voltages (the input array of voltages) and window_size (the size of the moving average window).

Here's how the function works:
It creates a window array with the length of window_size, where all elements are set to 1/window_size. This window represents the coefficients of the moving average filter.
It applies the filter by convolving the input voltages array with the window array using the np.convolve function with mode='same'. This ensures that the filtered array is the same length as the input array.
The filtered voltages are returned as the output.
The benefits of applying a low-pass filter to ECG data include:
Noise reduction: High-frequency noise, such as electromagnetic interference or muscle artifacts, can interfere with the analysis of ECG signals. Applying a low-pass filter helps attenuate these high-frequency noise components, resulting in a cleaner signal.

Smoothing: The filter can help smooth out variations or irregularities in the ECG waveform, making it easier to analyze and extract important features.

- Scale:

The scale function performs additional preprocessing steps on an input array. It takes one parameter, array, which represents the input data (e.g., ECG signal).

Here's a breakdown of the function:

It first replaces any NaN (Not-a-Number) values in the input array with 0.0 using np.nan_to_num. This step ensures that the array does not contain any missing or invalid values.

It applies the low_pass_filter function to the array, using a window size of 100.

It computes the minimum and maximum values (a_min and a_max) of the filtered array.

If the difference between a_max and a_min is zero (meaning the array has constant values), it returns a zero-filled array of the same shape as the input.

Otherwise, it normalizes the filtered array by subtracting a_min and dividing it by the range (a_max - a_min), resulting in a scaled array where the values range from 0 to 1.

The benefits of the scale function in preprocessing ECG data include:

- Handling missing values: The function replaces NaN values with zeros, ensuring that the data is well-defined and suitable for further processing.
- Further noise reduction: Applying the low-pass filter helps reduce noise in the ECG signal, improving the signal-to-noise ratio and enhancing subsequent analysis steps.
- Data normalization: The scaling step normalizes the filtered data to a common range (0 to 1). This can be useful for models that require input data to be within a specific range or when different features have varying scales.

Overall, the provided preprocessing functions can help improve the quality of ECG data by reducing noise, smoothing the waveform, handling missing values, and normalizing the data for subsequent analysis or model training.

Based on the dataset provided, it is more appropriate to scale the ECG signals with respect to each lead individually rather than across the whole sum of leads. Here are a few reasons to support this approach:

- Lead-specific information: The ECG signals in different leads provide unique information about the electrical activity of the heart from different perspectives. Scaling each lead individually helps preserve the lead-specific characteristics and enables the analysis and classification of each lead independently.
- Lead-specific abnormalities: Different arrhythmias or cardiac conditions may manifest differently in different leads. Scaling each lead individually allows for the identification of lead-specific abnormalities or patterns that may be crucial for accurate diagnosis and classification.
- Feature extraction: Scaling each lead individually can aid in the extraction of lead-specific features that may be relevant for classification. Certain features or characteristics specific to each lead might be significant in distinguishing between different arrhythmias or cardiac conditions.
- Training algorithm diversity: Scaling each lead individually can provide more diversity in the training data, especially if different leads exhibit different scales or ranges of values. This can help improve the generalization and robustness of machine learning algorithms trained on the data.

Considering these factors, it is advisable to scale the ECG signals individually for each lead rather than scaling across the whole sum of leads.

When scaling each lead in the context of ECG analysis, it is common to scale each lead with respect to all samples within that lead. In other words, you would compute the minimum and maximum values for each lead across all samples within that lead, and then scale the values of that lead based on those minimum and maximum values.

Scaling each lead with respect to all samples within that lead allows for normalization of the lead-specific data. It ensures that the values within each lead are scaled consistently and independently of other leads, accounting for any variations or differences in the magnitude of the signals.[9],[10]

3.11. The second model

A detailed explanation of each component and step in the code:

- Function signature: The code defines a function called entropy_cnn_model that takes two input parameters: input_shape (the shape of the input data) and num_classes (the number of classes for classification).
    - Input layer: The function starts by creating an input layer using tf.keras.Input(shape=input_shape). This layer will receive the input data for the model.
    - First branch - Convolutional layers for ECG signal: The code defines a series of convolutional layers for processing the ECG signal. These layers are designed to extract features from the input data. The convolutional layers use the Conv1D class from TensorFlow, with different configurations of kernel sizes, number of filters, and activation functions (in this case, tf.nn.leaky_relu). The output of the final convolutional layer is then flattened using layers.Flatten().
    - Second branch - Entropy computation: The code computes the entropy of the input signal. Entropy is a measure of uncertainty or randomness in data. Here, the entropy is computed by applying a mathematical formula to the input data using TensorFlow operations. The resulting entropy values are then flattened.
    - Concatenation and flattening: The outputs from the convolutional layers and the entropy computation are concatenated using layers.Concatenate(). This combines the extracted features from

both branches into a single vector. The concatenated vector is then flattened using layers.Flatten().

- Fully connected layer: The flattened vector is passed through a fully connected layer (layers.Dense) with 256 units. This layer applies a linear transformation to the input data followed by a leaky ReLU activation function (tf.nn.leaky_relu).
- Output layer: The final layer of the model is another fully connected layer with num_classes units (corresponding to the number of classes for classification). It uses a sigmoid activation function to produce the output probabilities for each class.
- Model creation: Finally, the code creates a Keras model using tf.keras.Model by specifying the input and output layers.
- Model return: The function returns the created model.

Overall, this code defines a CNN model architecture that combines features extracted from convolutional layers processing the ECG signal with entropy values computed from the input data. The model aims to capture both signal-specific features and overall uncertainty information for classification tasks.

- Input shape and number of classes: The code defines the input_shape as (5000, 12), representing the shape of the input ECG signal. It has 5000-time steps and 12 channels (leads). The num_classes is set to 12, indicating that there are 12 output classes for classification.
- Create the model: The entropy_cnn_model function is called with the input_shape and num_classes as arguments. This creates an instance of the model based on the provided architecture.
- Model summary: The model.summary() method is called to print a summary of the model's architecture. The summary provides information about each layer in the model, including the layer type, output shape, and the number of trainable parameters.

compile method. Here's an explanation of the different arguments used:

- Optimizer: The optimizer argument specifies the optimizer to be used for training the model. In this case, Adam optimizer is used with a learning rate of 0.0005. Adam is a popular optimizer that adapts the learning rate during training to achieve better convergence.
- Loss function: The loss argument specifies the loss function to be optimized during training. In this case, 'binary_crossentropy' is used. This loss function is commonly used for binary classification problems.
- Metrics: The metrics argument specifies the evaluation metrics to be computed during training and evaluation of the model. The provided code uses a list of metrics, including:
- tf.keras.metrics.BinaryAccuracy: This metric calculates the binary accuracy of the model's predictions, using a threshold of 0.5 to determine the binary classification.
- tfa.metrics.F1Score: This metric computes the F1 score, which is a measure of the model's performance that combines precision and recall. It is calculated using a micro-average method with a threshold of 0.5 and num_classes set to 12.
- tf.keras.metrics.Recall: This metric computes the recall (sensitivity) of the model's predictions.
- tf.keras.metrics.Precision: This metric calculates the precision of the model's predictions.

The Early Stopping technique provides several benefits during the training of machine learning models:

- Prevents Overfitting: Early Stopping helps prevent overfitting, which occurs when the model performs well on the training data but fails to generalize to new, unseen data. By monitoring the validation loss or other metrics, Early Stopping stops the training process when the model's performance on the validation set starts to deteriorate, preventing overfitting and improving the model's ability to generalize to new data.

- Saves Time and Resources: Training deep learning models can be computationally expensive and time-consuming, especially with many epochs. Early Stopping allows training to be stopped early if the model's performance has plateaued or started to degrade. This helps save time and computational resources by avoiding unnecessary training epochs that do not improve the model's performance significantly.
- Retains Best Model: Early Stopping with the restore_best_weights=True option ensures that the model's weights are restored to the best-performing configuration observed during training. This allows you to retain the model with the best validation performance and use it for further evaluation or inference.
- Enhances Model Generalization: By stopping the training process at an optimal point, Early Stopping helps the model achieve better generalization performance. It prevents the model from memorizing noise or specific patterns present in the training data, encouraging it to learn more robust and representative features.
- Provides Insight into Model Performance: The Early Stopping callback provides feedback on the training progress and performance of the model. It reports information about the stopping criteria, such as the number of epochs before early stopping and the best validation performance achieved. This insight helps you analyze and understand the behavior of the model during training.
- Monitor: The monitor argument specifies the metric to monitor for early stopping. In this case, 'val_loss' is used, indicating that the

validation loss will be monitored. The training process will stop if the validation loss fails to improve.

- Patience: The patience argument defines the number of epochs with no improvement after which training will be stopped. In this case, 6 epochs of no improvement in validation loss will trigger early stopping.
- Restore best weights: The restore_best_weights argument is set to True, indicating that the weights of the model will be restored to the best values found during training. This ensures that the model's performance is based on the best configuration rather than the last epoch.
- Verbose: The verbose argument is set to 1, which means that information about the early stopping process will be printed to the console.

The Early Stopping callback is commonly used during model training to prevent overfitting and to stop training when the model's performance has reached an optimal point. By monitoring the validation loss, it allows for early termination of training if the model starts to overfit or if the validation loss fails to improve for a certain number of epochs.

This demonstrates the usage of the fit method to train a model with Early Stopping. Here's a breakdown of the arguments used:

- X_train: The training data, which is a numpy array or a TensorFlow tensor, containing the input features.
- y_train: The target labels correspond to the training data.
- batch_size: The number of samples per gradient update. It defines how many samples are processed before the model's weights are updated.
- epochs: The number of times the training process will iterate over the entire dataset.

- validation_data: The validation data, which is a tuple (X_val, y_val) containing the input features and target labels for validation. This data is used to evaluate the model's performance during training.
    - callbacks: A list of callbacks to be used during training. In this case, the early_stop callback is included to perform early stopping.

During training, the model will be trained for a maximum of 100 epochs. The training data will be divided into batches of size 32, and after each epoch, the model's performance will be evaluated on the validation data. The early_stop callback will monitor the validation loss and stop the training process if the validation loss fails to improve for 6 consecutive epochs. The training history will be stored in the history_1 object, which can be used to analyze the model's performance and plot training curves.

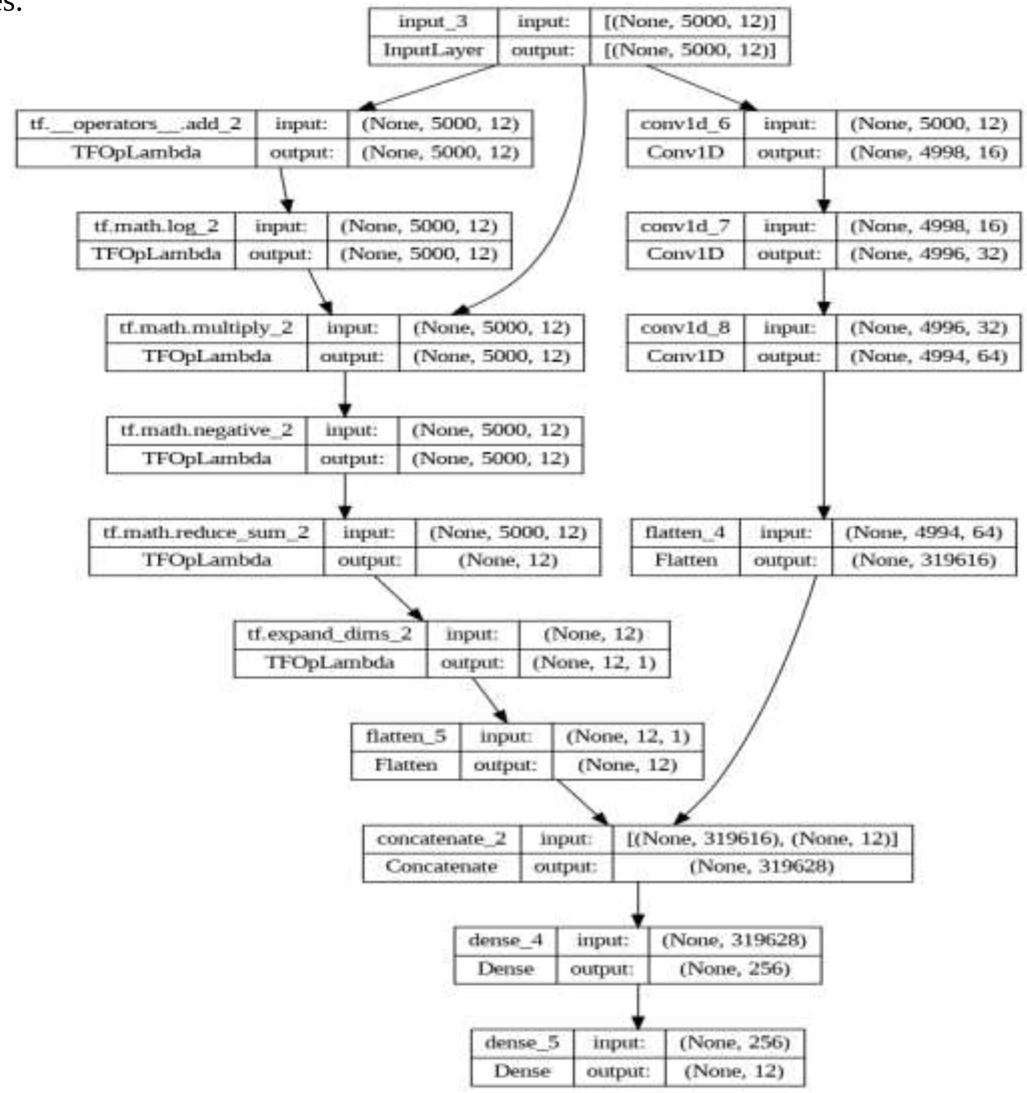

input_3 | input: | [(None, 5000, 12)]
InputLayer | output: | [(None, 5000, 12)]
tf.__operators__.add_2 | input: | (None, 5000, 12)
TFOpLambda | output: | (None, 5000, 12)
conv1d_6 | input: | (None, 5000, 12)
Conv1D | output: | (None, 4998, 16)
tf.math.log_2 | input: | (None, 5000, 12)
TFOpLambda | output: | (None, 5000, 12)
conv1d_7 | input: | (None, 4998, 16)
Conv1D | output: | (None, 4996, 32)
tf.math.multiply_2 | input: | (None, 5000, 12)
TFOpLambda | output: | (None, 5000, 12)
conv1d_8 | input: | (None, 4996, 32)
Conv1D | output: | (None, 4994, 64)
tf.math.negative_2 | input: | (None, 5000, 12)
TFOpLambda | output: | (None, 5000, 12)
tf.math.reduce_sum_2 | input: | (None, 5000, 12)
TFOpLambda | output: | (None, 12)
flatten_4 | input: | (None, 4994, 64)
Flatten | output: | (None, 319616)
tf.expand_dims_2 | input: | (None, 12)
TFOpLambda | output: | (None, 12, 1)
flatten_5 | input: | (None, 12, 1)
Flatten | output: | (None, 12)
concatenate_2 | input: | [(None, 319616), (None, 12)]
Concatenate | output: | (None, 319628)
dense_4 | input: | (None, 319628)
Dense | output: | (None, 256)
dense_5 | input: | (None, 256)
Dense | output: | (None, 12)

Figure 30 Model architecture

Chapter 4: Tools & Technology

4.1. Python

Python	Is an interpreted, object-oriented, high-level programming language.
In All Phases • Used in downloading data with specific attributes needed. • Used in understanding and cleaning data. • Will be used to implement the business logic. • Will be used to test business logic based on predefined test cases.	

Pros	**Cons**
• **Easy to learn.** • ***Scalability***– Python is a highly scalable language and is also much faster than other languages such as R, Stata, and Matlab. • ***Wide choice of libraries***– comes with a host of Data Science and Data Analytics libraries includingPandas, NumPy, SciPy, Scikit- Learn, StatsModels, and many more. • **ActivePython community**– No matter what your issue is, you can always count on the Python ecosystem to help and support you. • **options for visualization**– rich visualization frameworks allow you to make sense of the data at hand and also visualize your findings through. • **Easy to deploy your model**.	• Although Python has some nice visualization libraries. but, compared to R, visualizations are usually more convoluted, and the results are not always so pleasing to the eye.

Table 1: Python

4.2.Flutter

Flutter	It is a free and open-source mobile UI framework. It allows to create a native mobile application with only one codebase.
Phase four: Implementation • Flutter will be used to develop our front end.	

Pros	Cons
• ***Relatively fast development*** – it allows you to use the same code base for building separate iOS and Android apps, so it speeds up the whole development process • ***Flexibility*** – **its automatic reload makes it possible to make changes to the code and see the results immediately** in the app preview, without the need to recompile the code. • ***Full customisation & fast rendering*** – it gives control over every pixel on the screen and let you overlay & animate graphics, video, text and controls without limits.	• ***Flutter apps are quite large and heavy to start with*** – They occupy a lot of space and take longer to download or update.

Table 2: Flutter

4.3Figma

Figma	It is a browser-based UI and UX design application, with excellent design, prototyping, and code-generation tools
Phase three: Analysis	

• Figma was used to create our prototype.	
Pros	**Cons**
• ***Easy to use interface.*** • ***Collaboration*** – you can share the design with your team members, can all logged in at once, simultaneously making changes to it and all changes willbe saved.	• ***Bad Transitions*** – Zooming in and out of designs is always a frustrating experience

Table 3: Figma

4.4.Zoom

Zoom	Videotelephony and online chat services through a cloud-based peer-to-peer software platform
Used in all phases	

• All our online meetings are on zoom.	
Pros	**Cons**
• ***Easy to use interface*** – you can create, accesses, chat, or call with easy and user-friendly manner. • ***Adaptability*** – can use it anywhere and on any device. • ***Screen Sharing*** • ***Clear audio*** • ***Date Consumption*** – compared to other online meeting application it's convenient.	• ***Limited amount time*** – each meeting is limited to 40 minutes. • ***Costly*** – In case you need extra time you must pay.

Table 4: Zoom

4.5.Kaggle Notebook

Kaggle Notebook	It is a cloud computational environment that enables reproducible andcollaborative analysis.

Phase One: Data Gathering • Used to write python code to select attributes needed to download data.	
Pros	**Cons**
• help you run Jupyter Notebook in the cloud with GPU support. • ***Easy to use interface.*** • ***interactive editing platform***– You can execute selected lines of code by highlighting the code in the editor interface. • allows code to be shared in a simple way. • you can easily add data sources from thousands of publicly available Datasets.	• Kaggle will generally autosave yourwork, but if you do not commit it and then reload your page you might find you lost it all.

Table 5: Kaggle Notebook

4.6 Github

Microsoft Excel	Spreadsheet developed by Microsoft for Windows, macOS, Android and IOS.
Phase one: Data Gathering • Excel was used to document our research and was also used to fill in the data that was collected. **Phase two: Data pre-processing** • Excel was used to view the data in an ordered manner (Pivot table).	
Pros	**Cons**
• ***Part of MS Office Suite*** – so almosteveryone has it and there is no additional cost. • ***Shared Drive*** – all data is available locally and can be accessed by anyone with access to a shared drive where the file is stored. • ***Analytical*** – Excel comes with a widerange of reporting tools, including matrices and charts, and it is easy tocreate pivot tables and customize data as required. • ***Simple-to-Use*** – Excel is universallyused, and almost every staff member will know how to use Excel's basic to intermediate functionality.	• ***Inflexible*** – Excel is very clunky when it comes to changing input across multiple tabs or spreadsheets, and there are problems with who will actually be responsible for making these changes and what happens when a mistake is made (as it inevitably will).

ormused
and
ers
rom

Phase two: Data pre-processing
Phase four: Implementation
• GitHub used and will be used to share and work on any code.

Pros	**Cons**
• ***Remote team collaboration*** – It simplifies the process of working with other people and makes it easy to cooperate on projects. • ***Easy to share changes*** – Team members can work on files and easily merge their changes with the master branch of the project.	• ***Not easy at the beginning*** - it needs time to know how to createa repository, make branch, pull, and merge for your work with your team.

Table 6: GitHub

4.7Microsoft Excel

Google Drive	Google Drive is a file storage and synchronization service developed by Google.

4.8Google Drive

Used in all phases • All our documentation, presentations, data or even codes are stored and accessed by all of us with google drive. **Used In the requirement Validation** • Creating the survey form and accessing it.	
Pros	**Cons**
• ***Easy to use interface*** – you can Create a new document, view shared folders between computers, view documents shared outside of your personal drive, see recent, starred, or deleted documents • ***Microsoft office compatible*** – all Microsoft derivatives doc can be uploaded, shared, and can be accessed by anyone with access to the drive where the file is stored.	• ***Limited Storage*** – it only allows upto 15 giga bites. • ***Costly*** – In case you need extra space you must pay.

Table 8: Google Drive

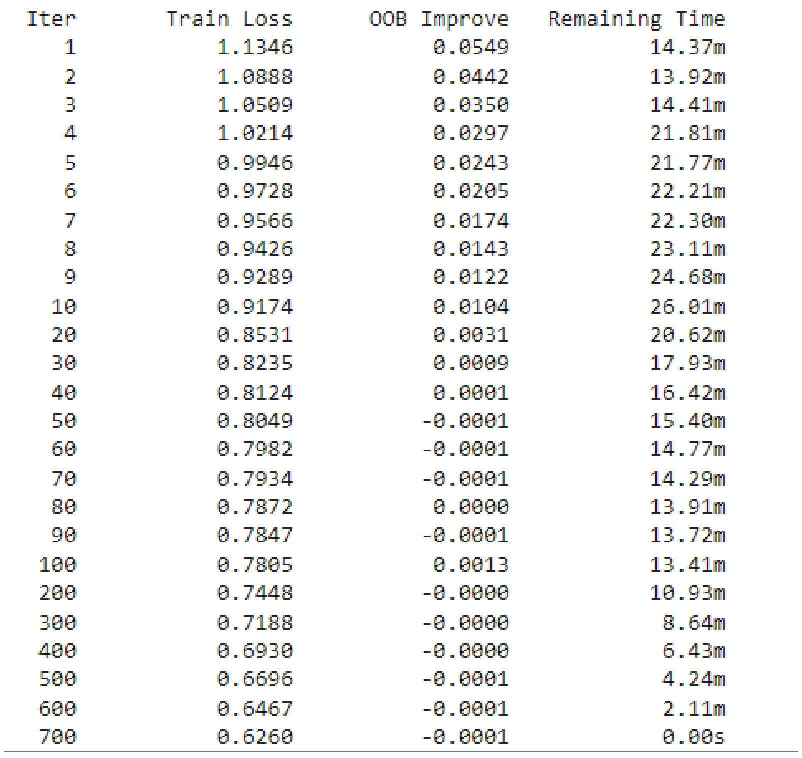
Chapter 5: Results and analysis

5.1The result of the first model

We achieved a Baseline Accuracy score of 0.9105. The baseline accuracy is a simple metric that serves as a reference point for evaluating the performance of a classification model. It assumes that the model predicts the most frequent class in the training data. In this case, the baseline accuracy represents the accuracy achieved by a model that always predicts the majority class (0) in the y_train data.

Then trains the classifier (clf) using the provided training data (X_train and y_train) along with optional weights.

Here is the training result:



```
      Iter       Train Loss      OOB Improve   Remaining Time
         1           1.1346           0.0549           14.37m
         2           1.0888           0.0442           13.92m
         3           1.0509           0.0350           14.41m
         4           1.0214           0.0297           21.81m
         5           0.9946           0.0243           21.77m
         6           0.9728           0.0205           22.21m
         7           0.9566           0.0174           22.30m
         8           0.9426           0.0143           23.11m
         9           0.9289           0.0122           24.68m
        10           0.9174           0.0104           26.01m
        20           0.8531           0.0031           20.62m
        30           0.8235           0.0009           17.93m
        40           0.8124           0.0001           16.42m
        50           0.8049          -0.0001           15.40m
        60           0.7982          -0.0001           14.77m
        70           0.7934          -0.0001           14.29m
        80           0.7872           0.0000           13.91m
        90           0.7847          -0.0001           13.72m
       100           0.7805           0.0013           13.41m
       200           0.7448          -0.0000           10.93m
       300           0.7188          -0.0000            8.64m
       400           0.6930          -0.0000            6.43m
       500           0.6696          -0.0001            4.24m
       600           0.6467          -0.0001            2.11m
       700           0.6260          -0.0001            0.00s
```

Figure 31: Training Results

The classification report provides various performance metrics for each class in the classification problem, as well as overall metrics. Let's break down the output of the classification report:

- Precision: Precision is the ratio of true positives to the sum of true positives and false positives. It measures how many of the positive predictions made by the classifier are actually correct. In the report, it is shown for each class (0 and 1) separately.

- Recall: Recall, also known as sensitivity or true positive rate, is the ratio of true positives to the sum of true positives and false negatives. It measures the proportion of actual positives that are correctly identified by the classifier. Like precision, it is shown for each class separately.
- F1-score: The F1-score is the harmonic mean of precision and recall. It provides a single metric that balances both precision and recall. It is calculated for each class separately.
- Support: Support represents the number of occurrences of each class in the test dataset.
- Accuracy: Accuracy is the overall percentage of correct predictions made by the classifier on the test dataset.
- Macro average: The macro average is the average of precision, recall, and F1-score across all classes, regardless of class imbalance. It gives equal weight to each class.
- Weighted average: The weighted average is the average of precision, recall, and F1-score across all classes, weighted by the number of occurrences of each class. It accounts for class imbalance in the dataset.

In the provided classification report, precision, recall, and F1-score are shown for two classes: 0 and 1. The accuracy of the classifier on the test dataset is 0.88. The macro average and weighted average of precision, recall, and F1-score are also provided. These metrics help assess the performance of the classifier in terms of its ability to correctly classify instances from different classes.

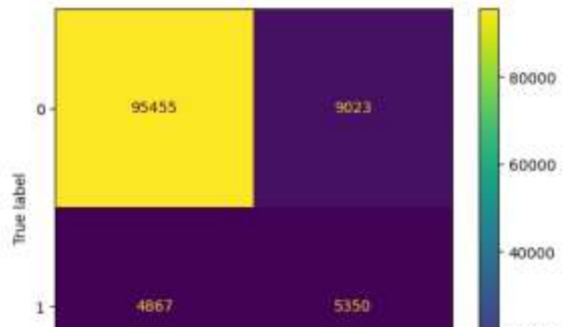
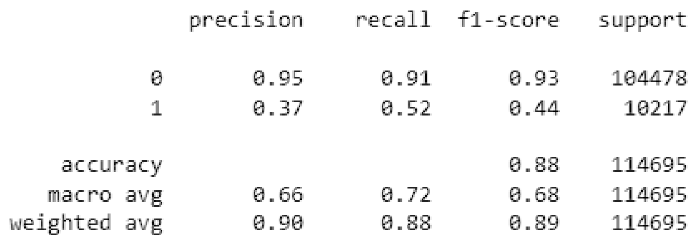


Figure 32: Binary Accuracy



```
              precision    recall  f1-score   support

           0       0.95      0.91      0.93    104478
           1       0.37      0.52      0.44     10217

    accuracy                           0.88    114695
   macro avg       0.66      0.72      0.68    114695
weighted avg       0.90      0.88      0.89    114695
```

Figure 33: The result of the first model

Here is a histogram to visualize the predicted probabilities for the positive class of a classification model.

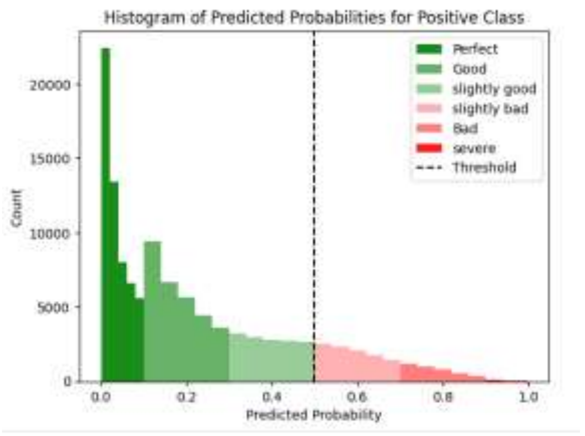


Figure 34: Histogram of Predicted Probabilities for Positive Class

The histograms are plotted with X-axis as the probability of having disease and Y-axis as counts of samples with different colors and alpha values to represent different levels of prediction quality. The "Perfect" group is plotted in green and has the lowest risk factor, while the "severe" group is plotted in red and has the highest risk factor. It helps to visualize the distribution of predicted probabilities for the positive class, and it categorizes the predictions into different quality levels based on defined thresholds. The plot helps in understanding the model's confidence in predicting the positive class and provides insights into the distribution of probabilities.

And as we can see the minor count is presented after 0.5 thresholds as the number of sick patients in the provided dataset is scarce. So, we transform the nature of our problem from an imbalance dataset to probabilistic outputs of an imbalanced data set to reduce but the impact of the unbalancing.

Here we visualize the precision-recall curve for a binary classification model:

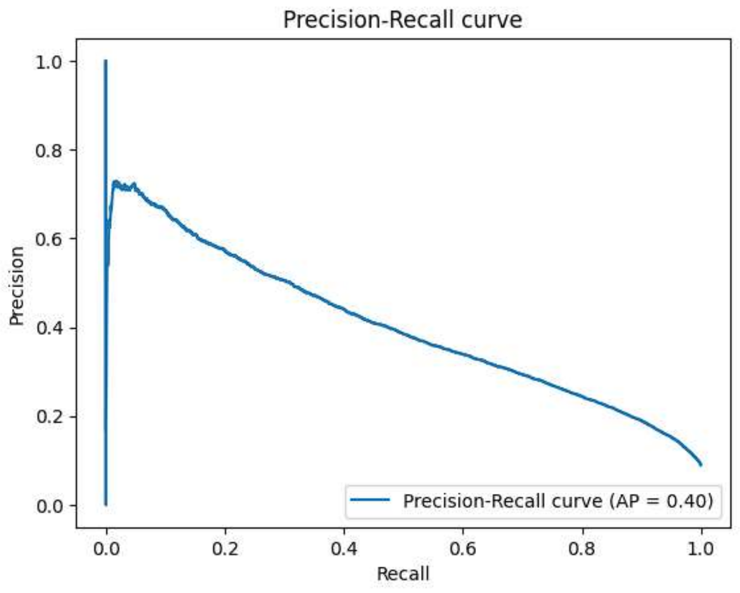


Figure 35: Precision-Recall curve

The recall values are plotted on the x-axis, and precision values are plotted on the y-axis. The label of the curve includes the average precision score. Here we can see the trade-off between precision and recall values we start off at 1 on the Y-axis and the and zero on X-axis as we have imbalance problem, we see a non-normal deflection in the slope of this curve and then as the model trains it gets back to normal to achieve a recall of 1, with precision recall curve of average precision 0.40.

Here is the visualisation of the Roc curve:

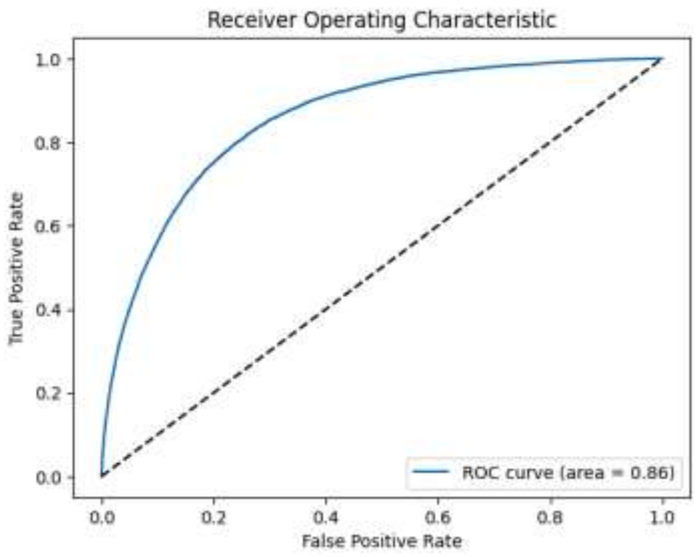


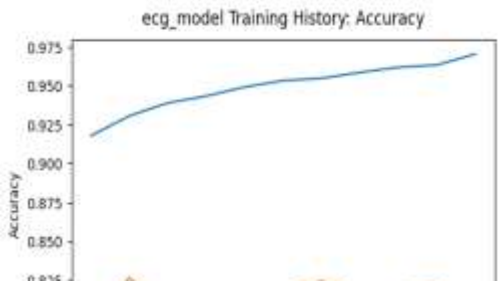
Figure 36: Receiver Operating Characteristic

This shows the trade-off between the true positive rate and false positive rate for different classification thresholds. The ROC AUC score provides a summary of the curve's performance, with higher values indicating better discrimination between the positive and negative classes.

Here in our case this curve tells us that the model was able to discriminate between the true positive rate and false positive rate with accuracy above the predicted accuracy given by the statistical analysis until it converges with AUC of 0.86.

5.2. The result of the second model

5.2.1. ECG_Model Training Accuracy and Loss



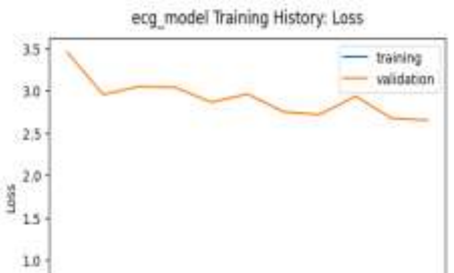
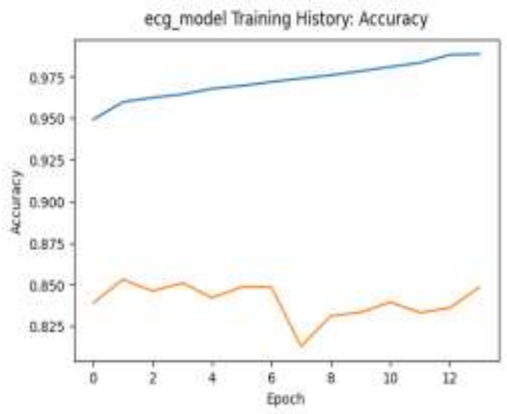


Figure 37: ECG_Model Training History (Accuracy) Figure 38: ECG_Model Training History (Loss)



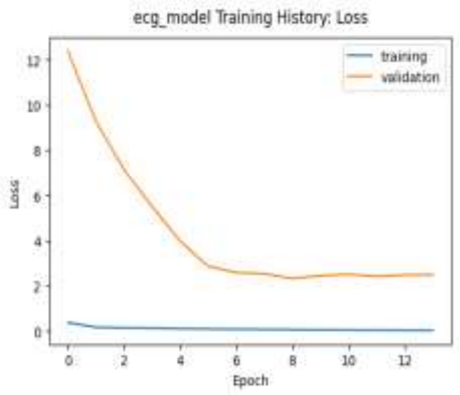
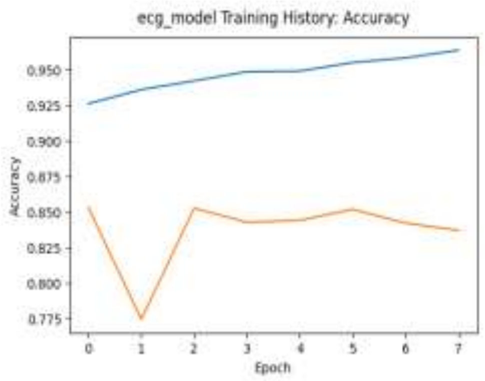


Figure 39: ECG_Model Training History (Accuracy) Figure 40: ECG_Model Training History (Loss)



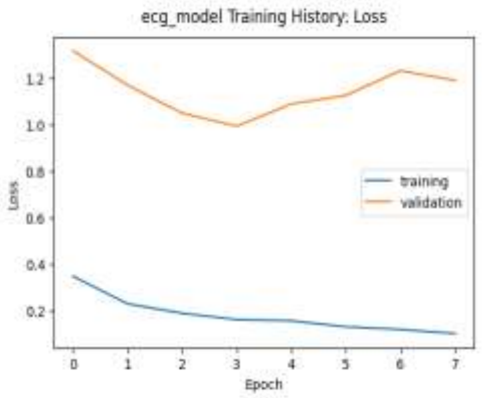
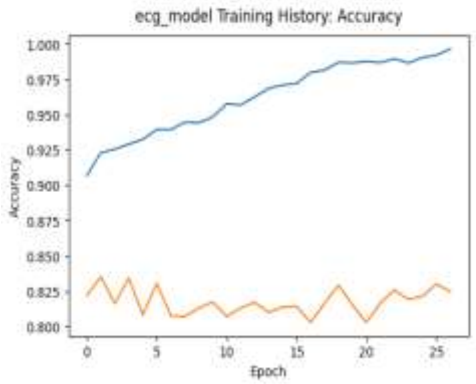


Figure 41: ECG_Model Training History (Accuracy) Figure 42: ECG_Model Training History (Loss)



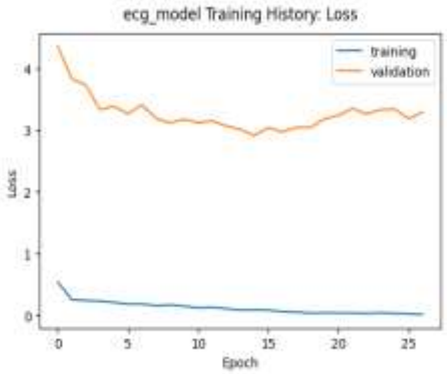
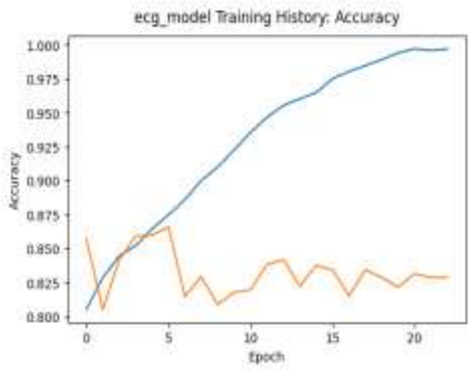


Figure 43 : ECG_Model Training History (Accuracy) **Figure 44: ECG_Model Training History (Loss)**



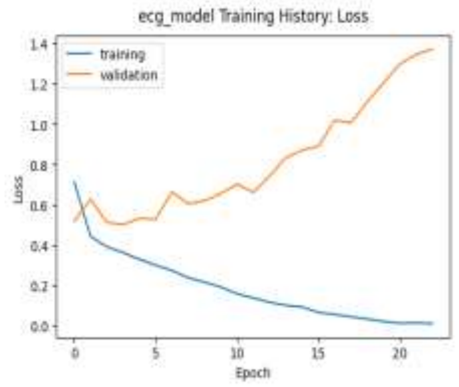
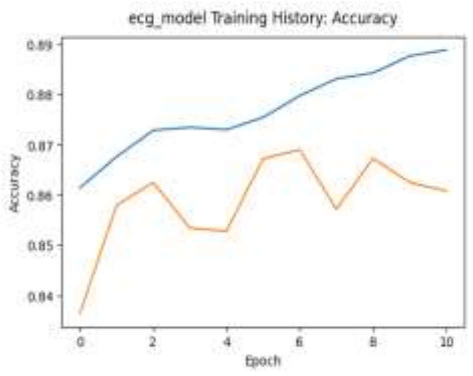


Figure 45: ECG_Model Training History (Accuracy) **Figure 46: ECG_Model Training History (Loss)**



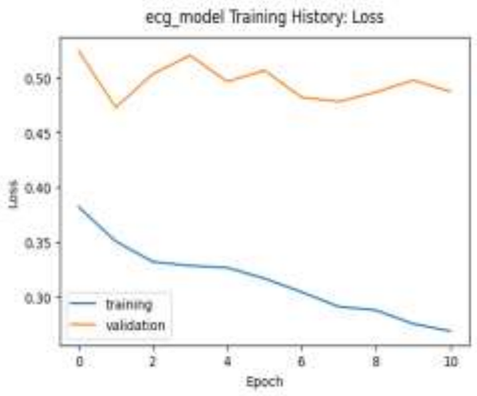
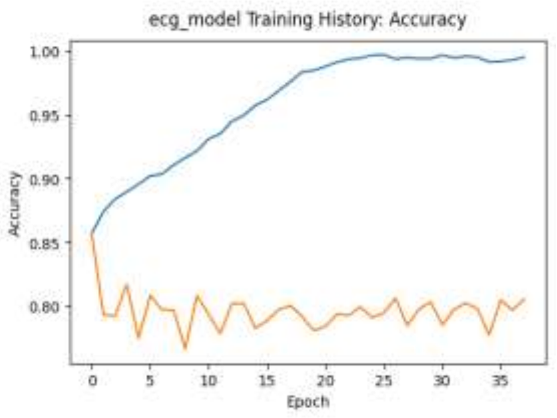


Figure 47: ECG_Model Training History (Accuracy) **Figure 48: ECG_Model Training History (Loss)**



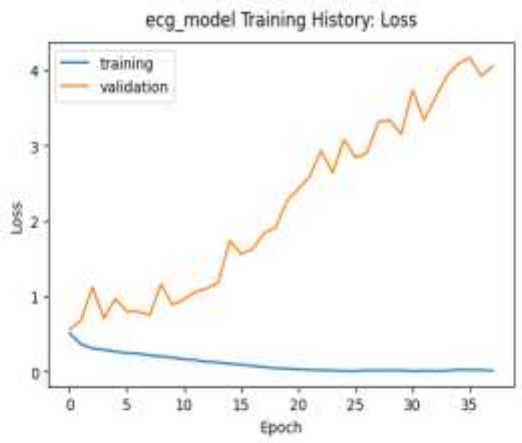
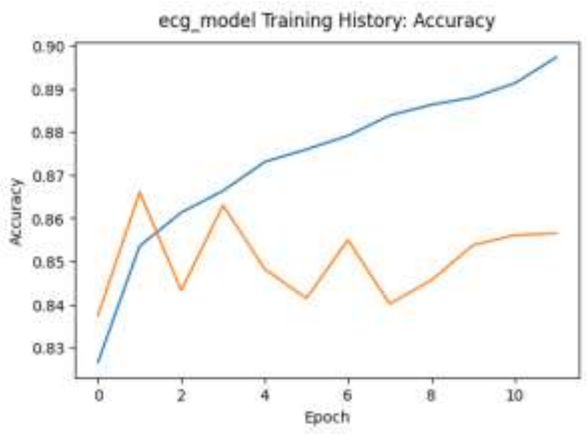


Figure 49: ECG_Model Training History (Accuracy) Figure 50: ECG_Model Training History (Loss)



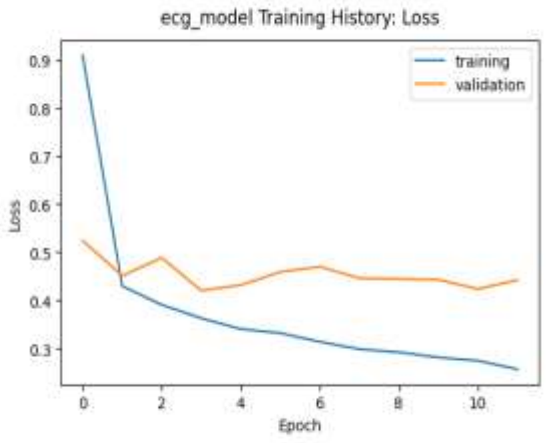


Figure 51: ECG_Model Training History (Accuracy) Figure 52: ECG_Model Training History (Loss)

5.2.2. Binary accuracy for batches

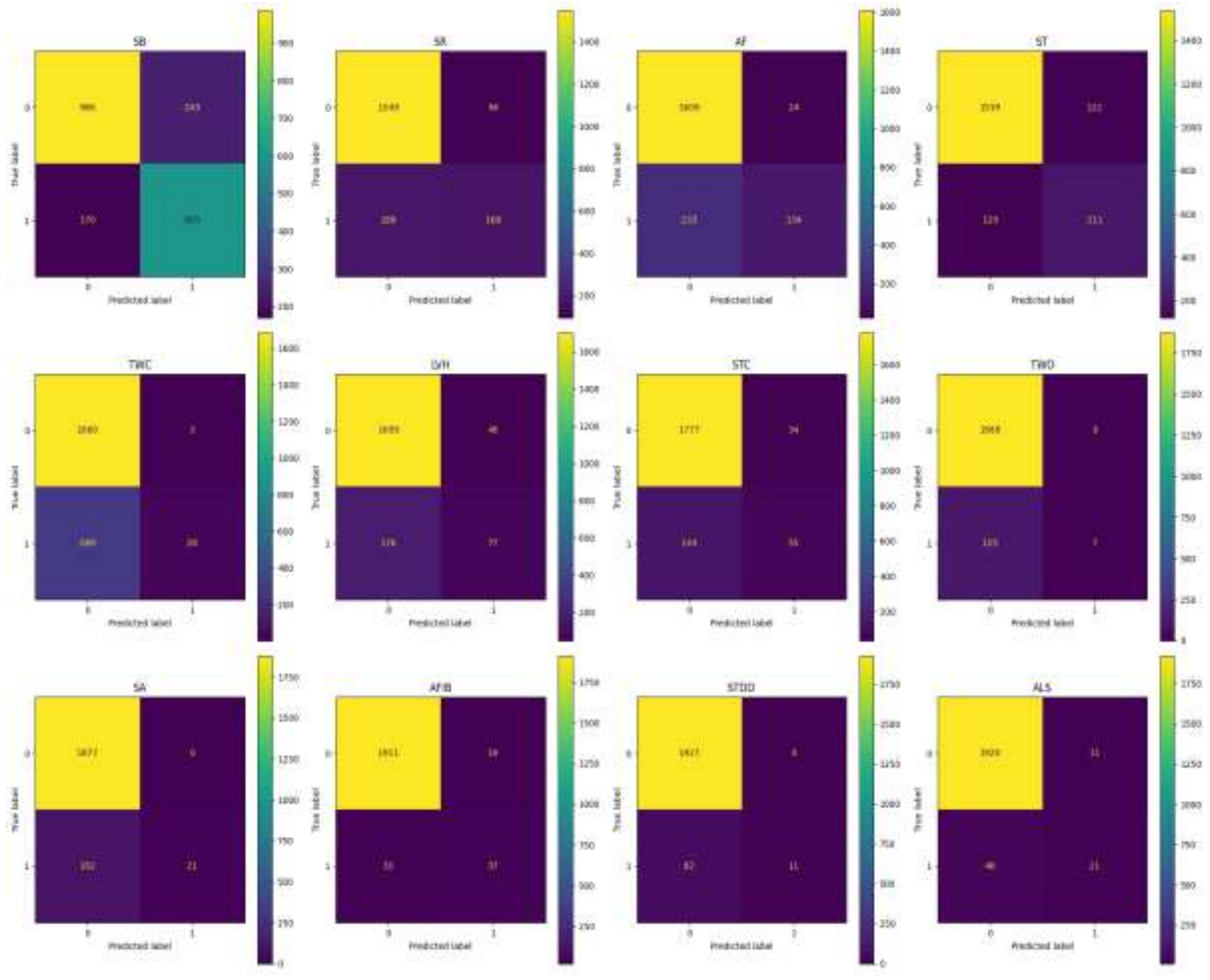


Figure 53: Binary Accuracy for Batches

These results are from 8 batched data represent:

Binary accuracy which is a performance metric used to evaluate the accuracy of binary classification models. In binary classification, the goal is to classify instances into one of two classes, often labeled as positive and negative.

Binary accuracy is calculated by dividing the number of correctly classified instances by the total number of instances in the dataset. It represents the proportion of correctly classified instances out of all instances.

Mathematically, binary accuracy can be expressed as:

- Binary Accuracy = (Number of correctly classified instances) / (Total number of instances)

Loss affiliated with the training and validation data is displayed to present another form of evaluating the model. It fared good with loss of 0.4965. As we can see here with a more batches it trains on the less the loss become on both training and validation sets which is a good indicator of the learning model

This is the confusion matrix of our data sets representing 12 class of 12 heart disease, these numbers show the number of true positive, true negative, false positive and false negative samples that the model have been trained on and the actual sample labels. Each one of these 12 represent how the model distinguished these samples.

The model achieved a total precision across the data set of 61 and total recall of 30. Here unlike our first model these numbers according to the nature of our problem are high results compared to other research papers That worked on this problem have given that our problem is a multiclass binary problem of 12 class so statistically a threshold of 8.3% represent the random number of getting a classifier to this class so achieving a higher number means a greater contribution.

To produce our project with easy way for using. We made a mobile application that you can download it for free and benefit from our system.[14]

Figure 54: Qr of the Application

Chapter 6: Conclusion and recommendation

6.1. Conclusion

This project, named Baymax, focuses on using AI models and machine learning algorithms to predict heart failures with a high percentage of accuracy. The research aims to address the significant impact of heart diseases on human life and healthcare systems. By utilizing diverse datasets and conducting statistical preprocessing, the project aims to minimize bias and enhance the flexibility of the predictive model.

The chapter highlights the importance of healthcare throughout history and the potential advancements brought about by AI. The integration of AI in healthcare can automate tasks, predict diseases, and contribute to better patient outcomes. The project's specific focus is on heart diseases, which are a leading cause of death worldwide, accounting for a significant percentage of mortalities. The objective of the project is threefold: to reduce deaths from heart diseases by early diagnosis, to protect healthy individuals from heart diseases, and to emphasize the significance of the medical field.

The problem statement identifies the high number of deaths attributed to heart diseases, often resulting from late diagnosis or individuals neglecting their health.

The scope of the project encompasses the collection of extensive data on heart diseases, including information about both healthy individuals and patients. The focus is on diseases, symptoms, and extracting the most relevant features to build a highly accurate predictive model.
The main objectives of the project are:
To reduce the number of deaths from heart diseases by enabling early-stage diagnosis and providing assistance to affected individuals.
To protect healthy individuals from developing heart diseases through preventive measures and awareness.
To raise awareness and emphasize the importance of the medical field.

The proposed solution involves diagnosing heart failures before they become dangerous through two approaches:
For individuals experiencing symptoms but uncertain about the presence of heart diseases, they can input their symptoms into the model, which will provide a prediction of whether they have heart diseases or not.
For individuals who have undergone heart examinations and have ECG signals available, they can provide the model with these signals. The model will then predict the specific heart diseases and can incorporate additional symptoms for a more accurate prediction.
Overall, the project aims to leverage AI models and machine learning techniques to improve early detection and prevention of heart diseases, ultimately saving lives and enhancing healthcare outcomes.

6.2. Recommendation

6.2.1. Digital history

A medical history is a record of a patient's health information, including symptoms, diagnoses, procedures, and medications. It is important to keep track of your medical history as it can help diagnose possible illnesses, understand hereditary and likely diseases in your family, as well as allergies, so we want to save the patient’s history as digital data, to make it easy to save and easy to access.

In this case it may be called “Electronic health records” (EHRs) have many benefits for patients. They can contain a patient’s medical history, diagnoses, medications, treatment plans, immunization dates, allergies, radiology images, and laboratory and test results. EHRs improve all aspects of patient care, including safety, effectiveness, patient-centeredness, communication, education, timeliness, efficiency, and equity.

Some benefits of using electronic medical records and electronic health records are comprehensive patient-history records; makes patient data shareable; improved quality of care; convenience and efficiency.

The doctor can see the patient’s history at any time for argent situations, like an accident or emergency surgery, all of this can be saved and accessed with many ways like the fingerprint and blood imprint.

The patient can access his history before taking any medicine that has any side effects on him.

6.2.2. Smart Watch

One of the primary functions of smartwatches in heart health monitoring is heart rate tracking. Built-in optical sensors on the underside of the watch continuously monitor your heart rate throughout the day. This feature allows you to keep an eye on your resting heart rate, exercise heart rate, and overall heart rate variability. By analyzing these data points, smartwatches can provide indications of your cardiovascular fitness and alert you to any irregularities or significant changes in heart rate patterns.

Additionally, smartwatches often incorporate electrocardiogram (ECG) capabilities. ECG is a diagnostic tool that measures the electrical activity of the heart and can help identify irregular heart rhythms such as atrial fibrillation. With the touch of a finger on specific sensors or electrodes on the watch, users can obtain a single-lead ECG reading. These readings can be stored on the device or shared with healthcare professionals for further analysis and evaluation.

So we want to connect our app with the patient’s watch and if there any changeable in heart it will Warne the person to open the app and check it, also we can use it if the patient get a heart attack, it can send for the nearest hospital to rescue him.

Figure 55: Smart Watch

Figure 56: Smart Watch

6.2.3. Diagnosis diseases

The field of medicine often encounters challenges in diagnosing certain diseases due to various factors such as overlapping symptoms, rarity of the condition, or limited understanding of the disease mechanism. In such cases, artificial intelligence (AI) models have shown promise in assisting healthcare professionals by providing insights, analyzing large amounts of data, and potentially improving diagnostic accuracy.

AI models can be trained to analyze complex medical data, including medical images, genetic information, patient records, and clinical data. By leveraging machine learning algorithms, these models can identify patterns and associations that may be difficult for human practitioners to detect, leading to more accurate and timely diagnoses.

One notable application of AI in disease diagnosis is in the field of radiology. AI models trained on vast databases of medical images can help radiologists detect abnormalities or early signs of diseases in X-rays, MRIs, CT scans, and other imaging modalities. These models can highlight suspicious areas, assist in prioritizing cases, and provide a second opinion to aid in diagnosis.

Furthermore, AI can be used to analyze genomic data and identify genetic markers or mutations associated with certain diseases. This can facilitate early detection, personalized treatment plans, and better understanding of the underlying mechanisms of various conditions.

It's important to note that while AI has shown promise in assisting with disease diagnosis, it should be considered as a supportive tool rather than a replacement for human expertise. The final diagnosis and treatment decisions should always involve the collaboration between AI systems and healthcare professionals.

So, in the future we will try to provide more datasets and trained models to many diseases, that will try to cover all our medical health to help for save and protect our lives.

6.2.4. Medical prescription and analysis

Many of people always has problems to understand their medical prescription and analysis, because they don't have an experience in this type of knowledge, and for this reason many people may take some wrong medicine and get sicker, so, including models that help the people to read and understand their medical prescription and analysis will be very helpful for them and save more lives.

6.2.5. ECG device and healthcare

The system can be connected directly with the ECG signal device and received the patient signal and after processing it, it will give the doctor its result which will help him to detect the issues.

After this process the results can be sent to some healthcare organization with the improvement of the patient to follow his health and provide him with suitable food and exercise that can help him to get better.

References

1. مصطفى, خالد "رقم مرعب.. وزير الصحة يكشف معدل الوفيات في مصر بسبب أمراض القلب." المصري اليوم, المصري اليوم, 17 Dec. 2022, https://www.almasryalyoum.com/news/details/2770454.

2. "Heart Disease and Stroke Statistics - 2023 Update." Professional.heart.org, https://professional.heart.org/en/science-news/heart-disease-and-stroke-statistics-2023-update.

3. Figure 1. (2023, January 26). Medical Education with Real Medical Cases | Figure 1. https://www.figure1.com/

4. Evidence-Based Clinical Decision Support at the Point of Care | UpToDate. (n.d.). https://store.uptodate.com/

5. *Kamilpytlak*, https://kamilpytlak-heart-condition-checker-app-2r42q4.streamlit.app/.

6. Teboul, Alex. "Heart Disease Health Indicators Dataset." Kaggle, 10 Mar. 2022, https://www.kaggle.com/datasets/alexteboul/heart-disease-health-indicators-dataset.

7. Alexteboul. "Heart Disease Health Indicators Dataset Notebook." Kaggle, Kaggle, 10 Mar. 2022, https://www.kaggle.com/code/alexteboul/heart-disease-health-indicators-dataset-notebook/notebook.

8. Pytlak, Kamil. "Personal Key Indicators of Heart Disease." Kaggle, 16 Feb. 2022, https://www.kaggle.com/datasets/kamilpytlak/personal-key-indicators-of-heart-disease.

9. Wagner, P., Strodthoff, N., Bousseljot, R., Samek, W., & Schaeffter, T. (2022). PTB-XL, a large publicly available electrocardiography dataset (version 1.0.3). PhysioNet. https://doi.org/10.13026/kfzx-aw45.

10. Wagner, P., Strodthoff, N., Bousseljot, R.-D., Kreiseler, D., Lunze, F.I., Samek, W., Schaeffter, T. (2020), PTB-XL: A Large Publicly Available ECG Dataset. Scientific Data. https://doi.org/10.1038/s41597-020-0495-6

11. https://www.findacode.com/snomed/#:~:text=SNOMED%20CT%20is%20an%20organized,documentation%2C%20claims%20billing%20and%20reporting

12. Shannon C. A mathematical theory of communication. Bell Syst. Tech. J. 1948;27:379–423. doi: 10.1002/j.1538-7305.1948.tb01338.x. [CrossRef] [Google Scholar]

13. Śmigiel S, Pałczyński K, Ledziński D. ECG Signal Classification Using Deep Learning Techniques Based on the PTB-XL Dataset. Entropy (Basel). 2021 Aug 28;23(9):1121. doi: 10.3390/e23091121. PMID: 34573746; PMCID: PMC8469424.

14. https://drive.google.com/drive/folders/1k81LCCHrwLwlmGWdY14r7qO5RqQz2RfQ?usp=sharing