Helping Doctors Visualize and Predict Cardiovascular Diseases Using Machine Learning

Rafael Kollyfas

Student Number: 170404745

Supervisor: Dr Stephen Riddle

Word Count: 14,740

Abstract

Cardiovascular Disease (CVD) is the number one cause of death worldwide, killing thousands of people every day. Recent advances in Machine Learning (ML) have brought new ways to diagnose CVD. The purpose of this project is to create a web platform that uses an ML model and a visualization library to enable cardiologists to visualize and predict their patient's condition in seconds. The project experimented with five ML classification algorithms, with Random Forest performing the best and therefore being deployed on a web platform developed. The model proved to be more reliable than a similar model published in a recent relevant paper. The completed platform was compared with HeartScore, a well-established CVD prediction platform. It was found that the developed platform is a useful tool to assist the cardiologist's thought process and provides richer graphical representation, while HeartScore is a more autonomous and complete system that provides more textual guidelines.

Declaration

I declare that this dissertation represents my own work except where otherwise stated.

Acknowledgments

First and foremost, I would like to thank my parents for allowing me to study at this amazing city and university. Furthermore, I would like to express my gratitude to my supervisor Dr Steve Riddle, for answering every single one of my questions and for his excellent guidance throughout the process of writing this dissertation. Moreover, I want to thank Dr V, who helped me greatly with his expertise and constructive feedback. Finally, I would like to thank Justyna Orłowska for being supportive throughout our whole university journey.

Table of Contents

Terminology

Below is a table defining used terminology in this document:

Term	Definition
Feature	A column/variable in the dataset (Eg. Cholesterol) [28].
scikit-learn	A python popular open-source machine learning platform [25].
Underfitting	When a model fails to predict classes correctly because the model hasn't captured the complexity of the training data [25].
Overfitting	When a model learns the classes for the training data far too accurately and therefore is unable to generalize to new testing data [25].
Bias	An error that occurs due to erroneous assumptions in the learning algorithm [27].
Variance	The model's sensitivity to small changes in training data. Lower variance is preferable [27].
parsleyJS	A JavaScript client-side form validation library [26].
GitHub	GitHub is a platform for hosting code that allows for version control and collaboration [33].
Euclidean Algorithm	Euclidean distance is calculated as the square root of the sum of the squared differences between the two vectors [35].
User Interface (UI)	The means in which a person controls a software application or hardware device [59].
Harmonic Mean	The harmonic mean is calculated by dividing the number of observations by the reciprocal of each number in a series [85].

1. Introduction

1.1 Purpose

Cardiovascular disease (CVD) is a general term for conditions that affect the heart or blood vessels. There are several types of CVDs such as coronary heart disease, stroke, and aortic disease [18]. Cardiovascular diseases are the number one cause of death worldwide, claiming an estimated 17.9 million lives each year [17]. Early detection of CVDs is vital as it can tremendously increase the survival rate of patients. However, detection of CVDs can sometimes be inaccurate due to limitations in common methods used by doctors. Recent advances in machine learning (ML) have brought about revolutionary new ways to detect cardiovascular diseases.

This project will first experiment with different Machine Learning classification algorithms to identify which algorithm has the highest accuracy in predicting whether a patient is at risk of suffering from cardiovascular disease. Once this is achieved, the ML model will be deployed in a web application that will be developed. The main purpose of the web application will be to allow doctors to accurately predict whether a patient is suffering from CVD with minimal cost and effort. It will allow doctors to perform tests much faster which will hopefully lead to more lives being saved. Additionally, the platform will give doctors meaningful insights by automatically visualizing the patient's data.

1.2 Aim and Objectives

Aim: Create a web platform that cardiologists will be able to use to predict, visualize and treat cardiovascular diseases.

To achieve this aim seven measurable objectives were set:

1. Gain a better understanding of cardiovascular diseases, their causes and how they can be prevented.
2. Research current challenges cardiologists face when diagnosing patients.
3. Experiment with at least 5 Machine Learning classification algorithms. Perform hyperparameter tuning on the 3 best models. Identify the most accurate model.
4. Use visualizations to identify 3 health data attributes that correlate highly with CVDs.
5. Deploy a reliable Machine Learning model with more than 90% accuracy and more than 85% recall (sensitivity).
6. Research and critically evaluate two different visualization libraries. Select the best one. Visualize patient data.
7. Improve the platform based on feedback received from 2 cardiologists.

The aim and objectives remained consistent during the project's development. However, the first interview with a cardiologist yielded several additional smaller development-oriented objectives which helped guide the development of the platform. These are enumerated in section 3.1.

1.3 Outline

The following is an outline of the dissertation's structure, as well as a brief description of each chapter.

Background

Gives more details about CVDs and ML. A detailed explanation of the algorithms that are used is also made. Furthermore, comparison between competing technologies is made where appropriate.

- Cardiovascular Diseases, an epidemic
- Machine Learning
- Developing the web application
- Similar platform
- Summary of background review

Interviews

Briefly describes the two interviews that were conducted with a cardiologist. The specific requirements for the platform and the graphical representation that were derived from the first interview are set.

- First interview
- Second interview

Methodology & Development

Describes the workflow and explains what was done and why. It addresses the sequence of these steps, explaining the implementation from beginning to end.

- Data overview
- Data exploration
- Developing the ML model
- Developing the platform
- Deployment

Testing

Tests and evaluates the model and the web platform to ensure that they both function properly and are resilient.

- Testing the model
- Testing the web platform

Results and evaluation

It discusses the software engineering process, confidentiality, results derived from this project and the attainment of the project's aim and objectives. Finally compares to a similar study and platform.

- Software engineering process
- Confidentiality
- Improvements lead by interview
- ML model evaluation
- Aim evaluation
- Objectives evaluation
- Similar study comparison
- Similar platform comparison

Conclusion

A reflection on what I learned while working on this dissertation, what could have gone better, setbacks, and intended future improvements.

- What I learned
- What could have been done better
- Setbacks
- Future improvements

2. Background

This chapter presents a summary of cardiovascular diseases, how they are currently treated and how ML can aid this process. Section 2.2 introduces various ML algorithms that are intended to be used, and a discussion about competing technologies that are relevant to the development of a high-quality medical platform is made in 2.3. Finally, a similar platform is introduced.

2.1 Cardiovascular Diseases, an Epidemic

In recent years, cardiovascular diseases showed epidemiological behaviour very similar to those of the great epidemics and were responsible for high mortality rates worldwide. In 2017, there were 2,353 deaths from CVD each day in the USA [51]. Similarly, in 2020, at the peak of the COVID-19 pandemic, there were on average 2,500 deaths per day.

The mortality rates from CVD are much lower in developed countries, and higher in lower-income countries. In 2008 approximately 80.1 % of CVD deaths occurred in low- and middle-income countries, and only 19.9 % occurred in high-income countries [34]. This is an important epidemiological fact that needs to be considered, the platform should be as accessible to lower-income countries and should operate with minimal costs.

The mortality rates of CVDs demonstrate the importance of the topic and the necessity to find modern ways to prevent, diagnose and treat them.

2.1.1 Risks and Prevention

The primary risk factors of CVD are high blood pressure, high cholesterol, high blood sugar (diabetes), obesity, and smoking [18]. The more risk factors an individual has, the more likely they are to develop CVD.

There are ways to prevent CVD, which mainly involve changing habits and living a healthy lifestyle. For example, it is recommended to exercise at least 150 minutes a week, eat a healthy diet low in sugar and saturated fats, and quit smoking [18].

2.1.2 Current challenges cardiologists face when diagnosing CVD

The most common way of diagnosing CVD is by using Electrocardiogram (ECG) which measures the patient’s heart electrical activity using electrical wires attached to their body. The most significant limitation of ECG is that it reveals the heart rate and rhythm only during the few seconds it takes to perform the test. If a heart rhythm irregularity does not occur while having the test, then the ECG will not pick up the disease [19]. This can be a very serious problem as people with potential cardiovascular problems may not be detected.

The platform aims to alleviate this problem as it will predict a patient's condition based on their health parameters. It will not rely on a brief test and thus it might give a more appropriate representation of a patient's current condition.

2.2 Machine Learning

Machine learning is a subfield of artificial intelligence (AI) that enables self-learning from data and then applies that learning without the need for human intervention. Its primary aim is to identify patterns in complex data and then being able to utilize them to make accurate predictions on new unseen data [77].

2.2.1 Supervised vs Unsupervised Learning

The concept of learning is a very wide domain. Subsequently, the field of ML has diverged into several subfields, including the supervised vs unsupervised learning domain.

Supervised learning can be compared to the learning achieved when guided by a teacher. The training data contains labels that classify the category each item in the dataset belongs to. The learner/algorithm aims to gain expertise on the patterns that distinguish one category from another. This acquired expertise is aimed to predict missing information for new unseen test data which does not contain a label that identifies their category [23].

In unsupervised learning, there is no distinction between training and test data. Its goal is to extrapolate the natural structure present within a set of data points, and it is used for exploratory analysis because it can automatically identify structure in data. Unsupervised learning can provide initial insights in situations where it is either impossible or impractical for a human to propose trends in the data. These insights can then be used to test individual hypotheses [23, 24].

In this dissertation, supervised learning will be used as the model will be trained on a dataset that classifies patients between healthy and suffering from CVD. Once the model is successfully trained, its accuracy will be tested by predicting the condition of new unseen patients.

2.2.2 Supervised Learning Algorithms

This subsection introduces the five supervised learning algorithms that will be used during the ML model development.

2.2.2.1 Random Forest Classifier

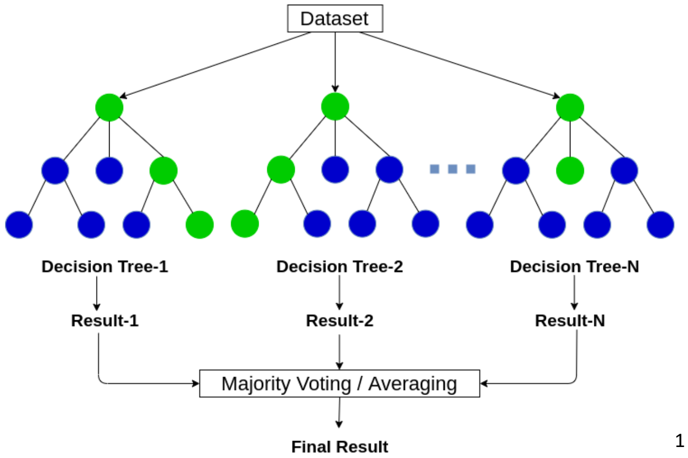


Figure 1: *Many decision trees providing their results for majority voting/average to occur.*

As shown in Figure 1, Random Forest (RF) is an ensemble classifier consisting of many decision trees (DT) [8]. A DT can be thought as a series of yes/no questions asked about our data which eventually lead to a predicted class. RF has the word "random" in its name due to two key concepts.

1. Random sampling of training data points when building trees

   Each tree learns from a random sample of the dataset. The samples are drawn with replacement which means that some samples will be used multiple times in a single tree. By training each tree on different samples, each might have high variance concerning a particular set of training data but overall, the entire forest will have a lower variance [8, 9].

2. Random Subsets of features for splitting nodes

   The number of features that are used is normally the square root of the total number of features in the training data. For example, if there are 16 features in the dataset, the algorithm would randomly select 4 features to use [8, 9].

These two concepts are used to ensure that any noise that exists in the training dataset that might influence the ability of the algorithm to correctly classify samples, gets reduced.

RF makes predictions by averaging the predictions of each decision tree. This procedure of aggregating multiple decision trees to come out with one result is known as bagging.

2.2.2.2 CatBoost Classifier

Catboost is a new ML technique for gradient boosting on decision trees [10]. Gradient boosting (GB) is the ensemble of weak predictive models, typically decision trees. It suggests that the best possible next model, when combined with previous models, minimizes the overall prediction error.

[1] https://cdn.analyticsvidhya.com/wp-content/uploads/2020/02/rfc_vs_dt1.png

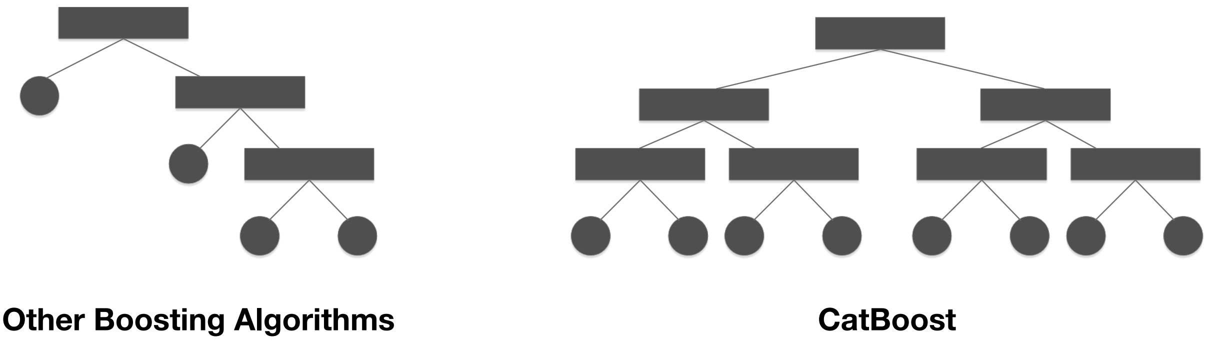


Figure 2: *Other boosting algorithms asymmetric trees vs Catboost symmetric trees.*

As evident from Figure 2, CatBoost in contrast with other boosting algorithms implements symmetric trees, which decreases prediction times.

Instead of training a different model for each data point in a dataset, Catboost trains only log(number_of_datapoints) models. If a model has been trained on ***n*** data points, then that model is used to calculate the next ***n*** data points. Hence, a model that has been trained on the first two data points is used for calculating residuals of the third and fourth data points. This is known as **ordered boosting** and reduces the computational expensiveness of the algorithm [10, 11].

CatBoost combines multiple categorical features, which is known as **categorical feature combination**.

country	hair color	class_label
India	black	1
India	black	1
India	black	1
india	black	1
russia	white	0
russia	white	0
russia	white	0
russia	white	0

Figure 3: *Catboost categorial feature combination.*

In Figure 3, CatBoost recognizes that whenever the country is “India”, hair colour is “black” and therefore considers this combination as a single feature. When the level of the tree increases, the number of categorical feature combinations increases proportionally [10, 11].

2.2.2.3 Logistic Regression

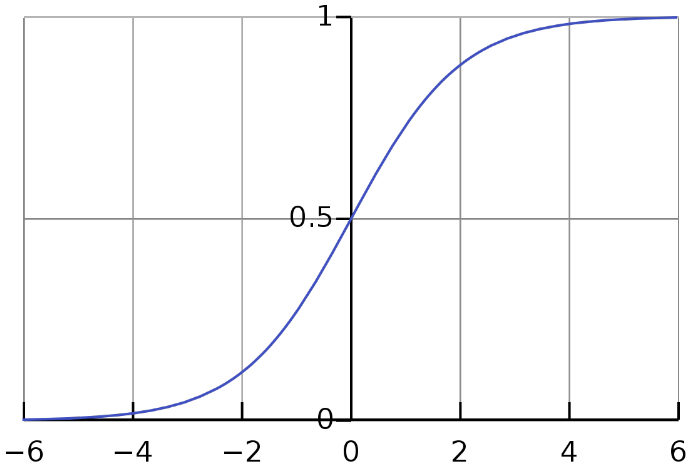
At the core of logistic regression, there is the Sigmoid Function shown below.

$$f(x) = \frac{1}{1+e^{-(x)}}$$ [2]

Equation 1: *Sigmoid Function equation.*

[2] https://miro.medium.com/max/674/0*pvMD0iSS8Mb2zy6W.png

The Sigmoid Function resembles an S-shaped curve that can take any real-valued number and map it into a value of 0 or 1 [13].



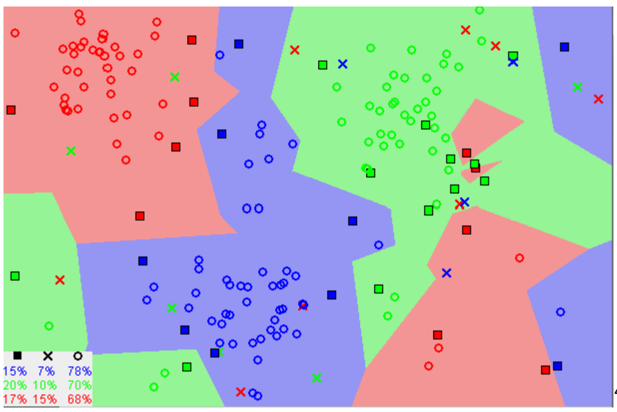
[3]

Figure 4: Sigmoid function

When using logistic regression, a threshold is usually specified to indicate at which value the input will be classified into one class vs the other class. In most cases, this threshold has a value of 0.5 [12].

2.2.2.4 K-Nearest Neighbours

The K-Nearest Neighbours (KNN) hinges on the assumption that similar things exist close to each other. It captures the idea of data points similarity by using the Euclidean algorithm to calculate the distance between points [14].

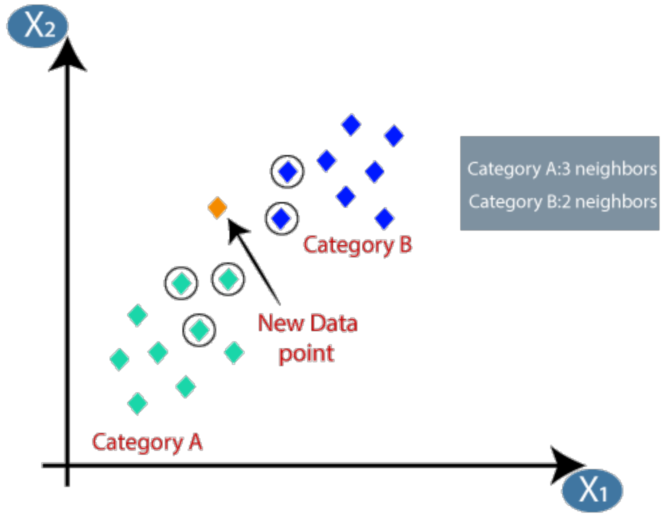


[4]

Figure 5: KNN - Similar data points are close to each other.

[3] https://upload.wikimedia.org/wikipedia/commons/thumb/8/88/Logistic-curve.svg/1200px-Logistic-curve.svg.png
[4] https://miro.medium.com/max/1222/1*wW8O-0xVQUFhBGexx2B6hg.png

KNN does not learn from the training set immediately and is known as a lazy learner algorithm. KNN at the training phase stores the dataset and when it gets new data, it classifies it into a category based on the classification of most of its K neighbours.



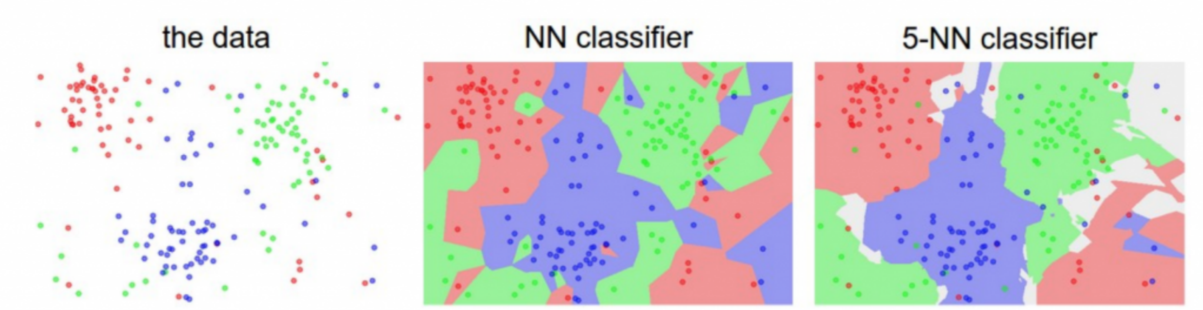
[5]

Figure 6: *New Data Point.*

The new data point shown in Figure 6 will be classified as A as it has 3 neighbours from category A as opposed to 2 neighbours from category B.

While using KNN, the value of K (number of neighbours) needs to be selected. There is no specific way to determine the best value for "K" and hence trial and error are required. We choose the K that reduces errors encountered while maintaining the ability to make precise predictions when given new data.

As we increase the value of K, predictions become more stable and precise due to majority voting. However, if K = N (N being the number of observations), there is a risk of overfitting the model as its limits are chaotic and irregular (Figure 7) [14, 15, 16, 20].



[6]

Figure 7: *Boundaries of different values of K.*

[5] https://static.javatpoint.com/tutorial/machine-learning/images/k-nearest-neighbor-algorithm-for-machine-learning3.png

[6] https://miro.medium.com/max/1400/1*QeHKidWVRNV27VeRt0CKEg.png

2.2.2.5 XGBoost

XGBoost is a decision-tree-based ensemble ML algorithm using a gradient boosting framework. It improves the foundation of GB through system optimization and algorithmic improvements [44].

System Optimisation:
XGBoost approaches the process of sequential tree building through a parallelized implementation. The order of loops is swapped at initialization by a global scan of all instances and sorting by parallel threads. This switch improves algorithmic performance by offsetting the parallelization overhead in computation. Also, XGBoost uses the 'max_depth' parameter as a stopping criterion and prunes the trees backwards [45]. This 'depth-first' approach significantly improves its computational performance[44]. Finally, the algorithm uses hardware resources efficiently, which is achieved through cache awareness by allocating internal buffers in each thread to store gradient statistics.

Algorithmic Optimisation:
More complex models are penalised by XGBoost regularization, this prevents overfitting [43, 44]. XGBoost is sparsity aware as it naturally maps sparse features to inputs by automatically "learning" the best missing value based on training loss and handling different types of sparsity patterns in the data more efficiently. Furthermore, the algorithm effectively finds the optimal split points between weighted datasets using the quantile sketch algorithm (Figure 8) [45]. XGBoost has a built-in cross-validation method at each iteration, eliminating the need to explicitly specify the exact number of boosting iterations required in a single run [44].

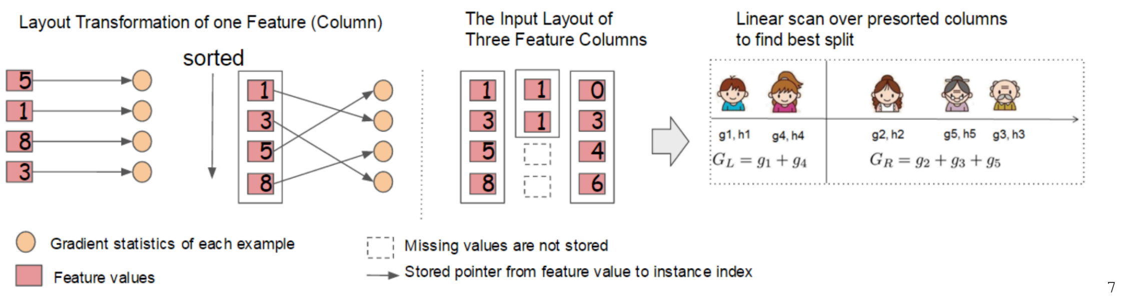


Figure 8: *Each column in a block is sorted by the value of the corresponding feature. To enumerate all split points, a linear search across a column in the block is necessary.*

2.2.2.6 Summary of Supervised Learning Algorithms

The research undertaken provided an understanding of the five algorithms that will be used. Random Forest reduces noise in the training dataset and uses ensemble decision trees to make a prediction, Catboost uses categorical feature combination and ordered boosting to increase its accuracy. Furthermore, the mathematical notion of Logistic Regression and sigmoid function were explored. K-NN was found to be a lazy learner as it does not train on the dataset itself but instead makes predictions based on data point neighbours. Finally, it was found that XGBoost improves gradient boosting by new methods such as system and algorithmic optimisations which can be a significant advantage.

[7] Taken from the paper: https://dl.acm.org/doi/pdf/10.1145/2939672.2939785

2.3 Developing the web application

The use of the right technologies is crucial in creating a high-quality medical web platform which is the goal of this project. In this section, we compare the SQL and NoSQL database types, contrast two Python web frameworks, and finally discuss the backend services of the platform.

2.3.1 SQL vs NoSQL

In terms of the nature of the database to be used, there is a choice between structured and unstructured data. For this project, the basis for comparison between these two structures is 1) the ease of development and flexibility, 2) speed, and 3) scalability. These factors that will have the greatest impact on the platform's performance and quality.

SQL (Structured Query Language) is a programming language used for accessing and managing databases. SQL can manipulate and retrieve data from a structured data format in the form of tables and holds relationships between those tables making it have a relational nature [1]. NoSQL offers a mechanism for the storage and retrieval of unstructured data. Data is stored in the format of collections and documents [1].

Table 1 compares SQL and NoSQL based on the platform's needs.
Rating *- is based on the needs of my platform.

The Basis Of Comparison	SQL	NoSQL
Ease of development & Flexibility for changes	Fixed schema for the tables - Changing schema is not possible. SQL has a huge community of developers which might help with development. Rating: 3/5	Changes in a document's schema can be done very easily. NoSQL has a big community of developers but smaller than SQL. Rating: 4/5
	Comment: For this project, a plan for the schema has been made but it is not certain if it requires an update later on. **NoSQL** is preferred as it gives the ability to make changes if necessary. Larger community of **SQL** is an advantage as it makes finding resources easier.	
Scalability	Vertical scalability - one server that can increase its scalability by increasing RAM, SSD, or CPU	Horizontal scalability - increased scalability by adding more servers to the database.

	Being relational in nature can be tough to scale the data. Rating: 3/5	Rating: 5/5
	Comment: **NoSQL** databases can scale easier than SQL and as such, they are preferred.	
Speed	SQL is relatively fast. Its performance gets gradually better as more reads and writes are performed. Rating: 3/5	NoSQL databases are faster than SQL, particularly regarding key-value storage, which means less waiting time in database transactions. Rating: 5/5
	Comment: Speed is vital for this platform to ensure that doctors can diagnose patients as fast as possible. **NoSQL** is faster and therefore it is preferred.	
Total Rating	9/15	14/15

Table 1: *Comparison between SQL and NoSQL. Information from [75,76]*

As discussed in table 1, NoSQL gives more flexibility in making changes than SQL. Both have very large communities; however, SQL has a slight advantage. Furthermore, NoSQL offers greater scalability and speed than SQL. As reflected by the rating, NoSQL better suits the needs of this platform, and as such it will be used.

2.3.2 Deciding on Python Web Framework

To utilize the ML model on the platform, a python web framework must be used. The options considered are Django and Flask - the most well-established python web frameworks.

Django is a free, open-source Python-based full-stack web framework. It focuses on automating the web development process and adheres to the "Don't-Repeat-Yourself" principle. Django follows the "Batteries-included" philosophy meaning it is self-sufficient and comes with everything included [2]. The framework provides support for communicating with databases and includes a built-in object-relational mapping layer (ORM) that can be used to interact with relational databases such as MySQL. Django is a heavy framework with various features and hence is better suited for bigger, more complex projects as otherwise, it is wasteful [3].

Flask is a Python-based microframework used for developing small or medium-size web applications. It is classified as a microframework as it does not explicitly require any tools or libraries [3,79]. Flask is simple, yet extensible giving developers the power to choose their own configuration. It does not support database access, user authentication, or any other high-level utility out of the box, but it provides support for extensions enabling these features. By only installing the extensions required, the application is more concise and secure as there are fewer vulnerabilities due to fewer dependencies. Finally, applications developed using Flask are usually faster than their Django equivalents.

Flask's simplicity and extensible nature were decisive factors in choosing it over Django.

2.3.3 Backend services

For the backend services of the web platform, Firebase and AWS Amplify were considered as they both offer NoSQL solutions and are very popular. The basis for comparison is the ease of use and the features offered.

AWS Amplify helps build scalable full-stack applications/websites powered by Amazon Web Services. Amplify offers numerous features, including a real-time NoSQL database, authentication, analytics, storage, and interactions such as creating conversational bots [38]. The main criticism of Amplify is its steep learning curve and lack of clear documentation.

Firebase is based on Google's infrastructure and offers a variety of tools to help developers build high-quality applications/websites quickly and scalably [5, 6]. Firebase services include analytics, real-time NoSQL databases, authentication, storage, and cloud functions. It is better suited for small- to medium-sized projects when compared with Amplify that is targeting larger projects and companies [39]. Firebase is easy to get started and has a bigger community of developers than Amplify which can help with the smooth development of the platform.

Both backend services cover my feature needs as only a real-time database, authentication, and storage are required. Considering my prior experience with Firebase and Amplify having a steep learning curve, Firebase was chosen for the platform's backend services.

2.4 Similar platform

HeartScore is one of the most popular CVD web platforms created by the Joint European Societies' Task Force and aims to help clinicians optimize individual cardiovascular risk reduction [31]. HeartScore offers two versions, a quick no-login CVD calculator and a full version that can track patient history and display progression graphs.

Their test requires the patient's name, age, gender, cholesterol, HDL cholesterol, and whether they smoke [32]. Upon diagnosis, it provides a 10-year CVD risk and the factors that lead to this risk. Furthermore, HeartScore provides detailed prevention recommendations and generates a pie chart showing the contribution of risk factors and a bar chart comparing the patient's absolute risk with the target value.

Once my platform is developed, it will be compared against HeartScore.

2.5 Summary of background review

The chapter provided an understanding of CVDs' epidemiological characteristics, causes, preventions, and current challenges cardiologists face. ML, unsupervised/supervised learning and five classification algorithms of the latter that will be used were then introduced. Moreover, NoSQL was found more suitable for this particular project and will help in the development of a fast, flexible and scalable web platform. Flask was chosen as the platform's Python web framework as it has a secure and uncluttered nature and offers more extensibility than Django. Finally, Firebase was chosen for the backend services and a similar platform was introduced. The next chapter discusses the two interviews conducted with a cardiologist.

3. Interviews

The project was driven by two interviews with a cardiologist in Cyprus, Dr V. The interviews were conducted in a semi-structured format, with some questions prepared in advance and some others appearing as follow-ups from the discussion. The questions were consciously designed to be open-ended, encouraging the interviewee to speak freely and express his opinions without external input. Section 3.1 describes the first interview and the requirements it derived and 3.2, the content of the second interview.

3.1 First Interview (*Appendix 1*)

The first interview took place during the development of the ML model and aimed to obtain guidelines and requirements for the platform. It began with a general discussion about the current challenges' cardiologists face and whether Dr V believes AI can help improve healthcare. This was followed by a discussion about design prototypes sent to him before our interview, with him providing useful feedback and guidelines. Finally, the generation of a graphical report after diagnosis was discussed.

3.1.1 and 3.1.2 include the requirements generated from Dr V's answers.

3.1.1 Platform Requirements

1) Simple, not monotonous, and easy to use.
2) Ability to view and add patients on the same page.
3) Automatically generating a graphical report after diagnosis, within 30 seconds.
4) Showing the factors influencing the decision of the ML model.

3.1.2 Graphical Requirements

1) Distribution of healthy and CVD patients in a scatter chart as a function of age against various parameters.
2) Clear statistical (percentage) comparison between patients who have a higher or lower value of a feature than the current patient.
3) Interactivity with graphs e.g. display only healthy/CVD patients or hover over specific patients.

3.2 Second Interview (*Appendix 2*)

The second interview took place after developing the platform and was designed to review the system. The selected dataset was first examined, where he found some limitations. Dr V then explored the pages of the medical website and especially the "Diagnosis" input form where he made corrections. The report page was then shown, the functionality of which Dr V found to be good, but still gave constructive feedback. Finally, Dr V evaluated the correctness of the factors that influenced the model's predictions, this is analysed in section 5.1.

Dr V's overall feedback led to several improvements being made which are addressed in section 6.3.

3.3 Summary

The platform and graphical requirements generated from the first interview were enumerated and the second interview provided constructive feedback on the functionality of the platform. The next chapter discusses the methodology followed and the development of the platform.

4. Methodology & Development

This chapter outlines the methodology & development followed for this project. Section 4.1 describes the workflow and tools used, an overview of the dataset is presented in 4.2 and the development process is discussed in the following section. In 4.4 data exploration is carried out and the development of the ML model is addressed in 4.5. The development of the medical web platform is described in 4.6 and finally the deployment of the platform in 4.7. A detailed insight for each tool and decision made is given, along with a comparison to alternatives where appropriate.

4.1 Workflow & Tools used

At the beginning of this project, a detailed diagrammatic work plan (*Appendix 4*) was prepared that divided the tasks into three phases, planning & analysis, design & implementation, and evaluation/testing.

This project followed the Agile methodology, which is a product development approach that adheres to the values and principles described in the Agile Manifesto for software development [65]. It aims to deliver the right product by delivering small pieces of functionality incrementally, with frequent customer feedback (in this case from my supervisor and Dr V). Furthermore, Agile development encourages frequent revisiting of previous sections to improve them based on new insights or feedback received. This was followed throughout the development of this dissertation.

Table 2 briefly lists and describes the tools used.

Tools Used	
Tool	**Purpose**
Python and its libraries (sklearn, matplotlib, pandas, lime, pickle)	Development of the ML model.
Google Colab	Run python code. It has the benefit of allocating a GPU to the user – faster than running your files locally.
Visual Studio Code	Development of the web platform.
Github	Backup of the code which has version history in case something goes wrong.
Flask	Python web microframework. The backbone of the web platform.
ChartJS	Web graphical library. Used to generate a graphical medical report.
Firebase	Database for the web platform, authentication, and storage.
Adobe XD	Creation of platform's design prototypes.
Heroku	Deployment of the platform.

Table 2: *Tools that are used for the development of this project.*

4.2 Data overview

The chosen dataset combines the commonalities of several datasets, specifically combining the Cleveland (UCI), Hungarian, Switzerland, Long Beach VA, and Stalog (Heart) datasets [78]. The inclusion of data from many organizations from different countries makes the dataset broad-based and robust.

Table 3 describes the 12 features that are used in this project.

Feature	Description
Age	Age of patient in years (Numerical)
Sex	Gender of patient [Male - 1, Female - 0]
Chest Pain Type	Type of chest pain experienced by patient - [1 atypical, 2 typical angina, 3 non- anginal pain, 4 asymptomatic]
Resting bp s	Level of blood pressure at resting mode in mm/HG (Numerical)
Cholesterol	Serum cholesterol in mg/dl (Numerical)
Fasting blood sugar	Blood sugar levels on fasting - [> 120 mg/dl is 1, less or equal than 120 mg/dl is 0.]
Resting ECG	Result of electrocardiogram while at rest - [0: Normal 1: Abnormality in ST-T wave 2: Left ventricular hypertrophy]
Max heart rate	The maximum heart rate achieved (Numerical)
Exercise Angina	Angina that was induced by exercise - [0 is NO, 1 is Yes]
Oldpeak	Exercise induced ST-depression in comparison with the state of rest (Numerical)
ST slope	Sinus Tachycardia (ST) segment measured in terms of slope during peak exercise 0: Normal 1: Upsloping 2: Flat 3: Downsloping (Nominal)
Target	1 - patient suffering from CVD, 0 - patient is not at a risk of suffering from a CVD.

Table 3: *Dataset features explanation.*

4.3 Development Process

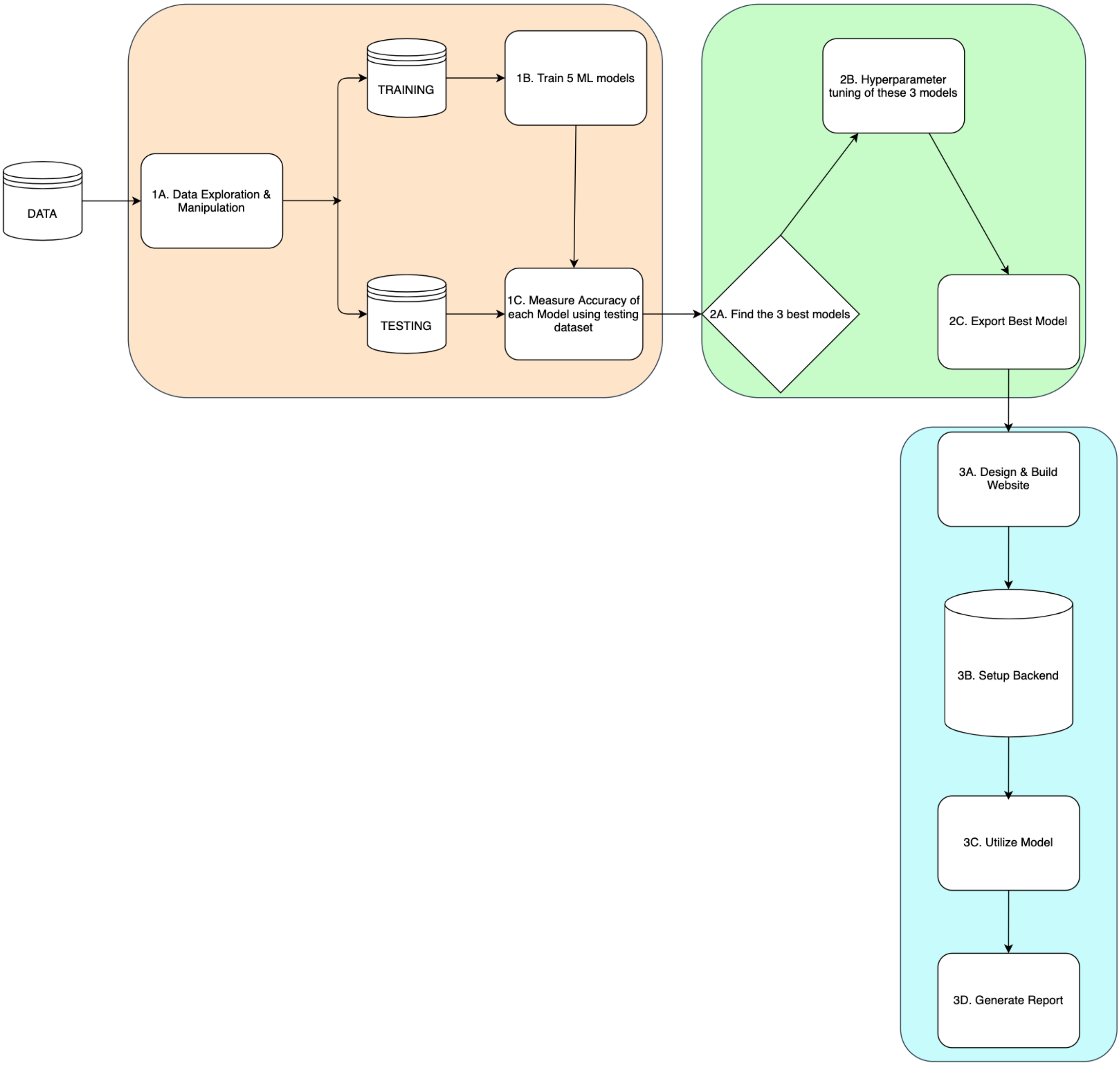


Figure 9: *A high-level flowchart description of the implementation process followed.*

Figure 9 depicts the development process followed at a high level. The process was divided into 3 major parts:

1) The training and testing of different models.
2) Tuning of the three best models, followed by the extraction of the best model.
3) The development of a web platform and utilization of the best model.

Each process in this flowchart is referenced and explained in depth in the following sections. If a section refers to a process, it is stated in that section's title e.g., section 4.4 below, explains process 1A.

4.4 Data Exploration – Process 1A

To fully understand the chosen dataset, graphical data exploration was performed.

4.4.1 Checking the dataset

The dataset was initially checked for any missing values. Code 1 shows that the dataset does not contain any missing values, and consequently, there is no need to impute or remove any data.

```
# Are there any missing values?
df.isna().sum()
```

```
age                    0
sex                    0
chest pain type        0
resting bp s           0
cholesterol            0
fasting blood sugar    0
resting ecg            0
max heart rate         0
exercise angina        0
oldpeak                0
ST slope               0
target                 0
dtype: int64
```

Code 1: *Missing values counter check.*

Because only numeric values can be used by ML models, the categorical values of the features in the dataset must be converted to numeric values. To check if the dataset requires this, the type of each variable was checked. Code 2 shows that all data types are numeric and therefore suitable.

```
df.dtypes
```

```
age                      int64
sex                      int64
chest pain type          int64
resting bp s             int64
cholesterol              int64
fasting blood sugar      int64
resting ecg              int64
max heart rate           int64
exercise angina          int64
oldpeak                float64
ST slope                 int64
target                   int64
dtype: object
```

Code 2: *The data types of variables.*

4.4.2 Visualizing the dataset

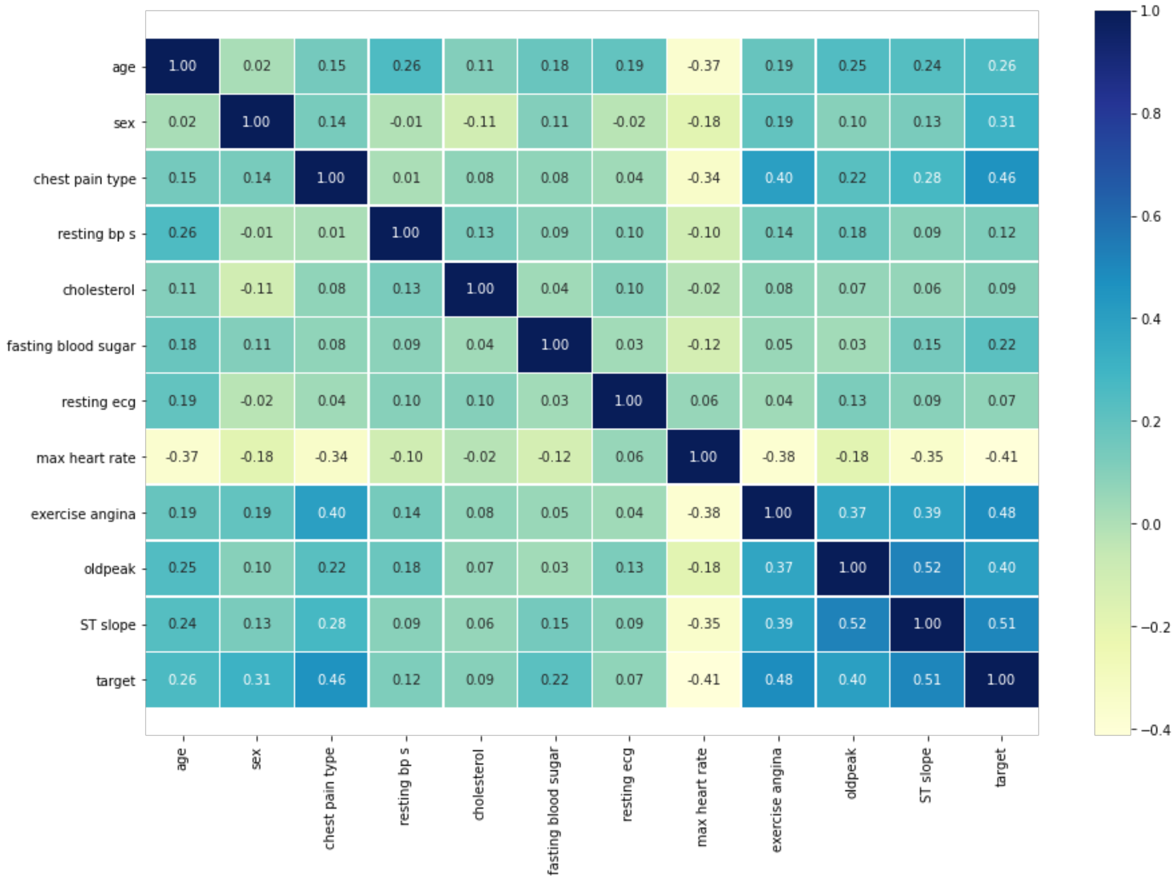


Figure 10: *Correlation diagram.*

Each cell in Figure 10 shows the correlation between two variables. ST slope has the most positive correlation with the presence of CVD. Furthermore, exercise angina, chest pain type, and oldpeak also have a significant positive correlation. The attribute that has a negative correlation with CVD is max-heart rate.

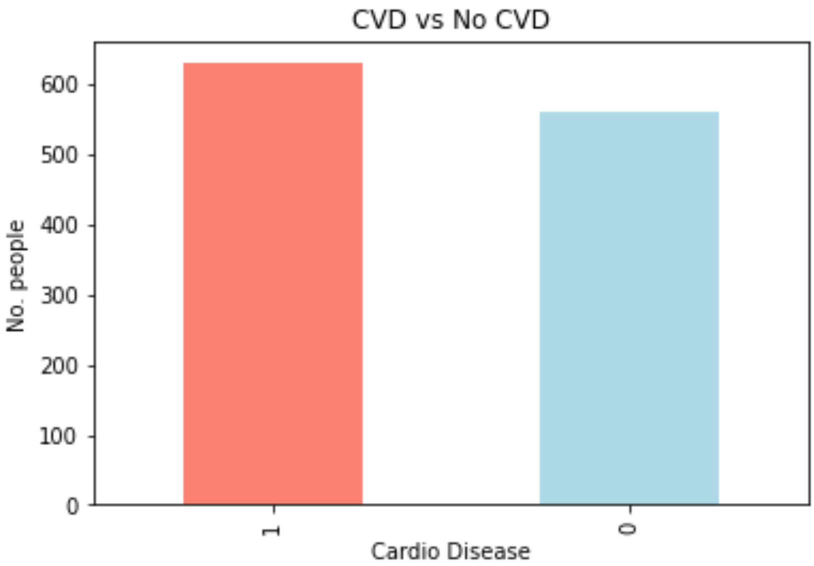


__Figure 11:__ Number of people suffering from CVD VS number of healthy patients in the dataset.

As we can see from Figure 11, the dataset is relatively balanced in terms of the “target” attribute, as it contains almost as many healthy patients as CVD patients.

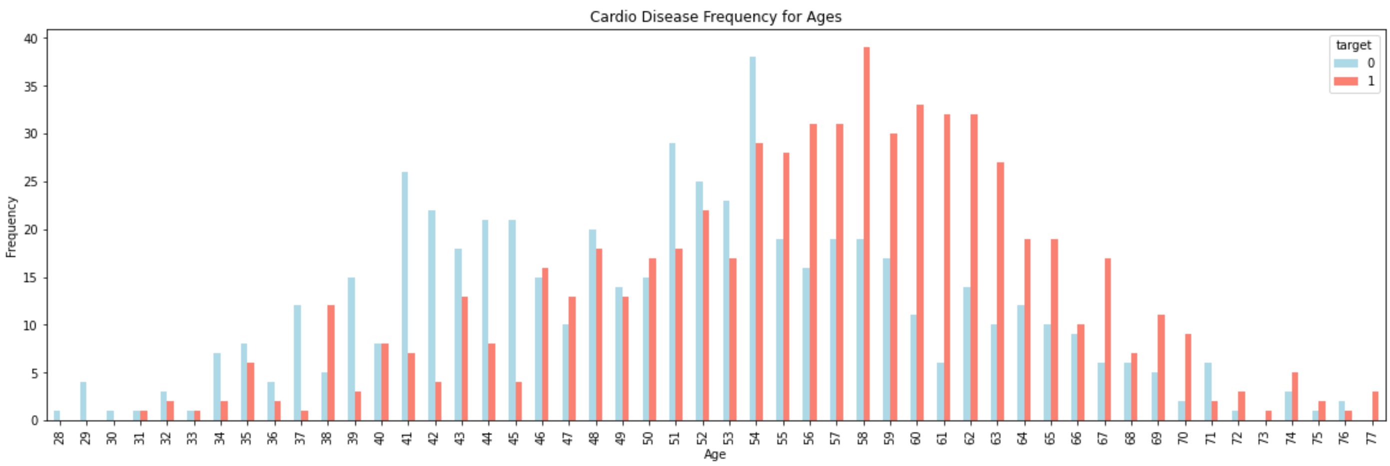


__Figure 12:__ Shows the frequency of cardio disease against age.

Figure 12 shows there is a positive correlation between age and CVD. The older the patient, the more likely they will suffer from CVD. This is also confirmed by Figure 10 where age was found to be correlated with CVD.

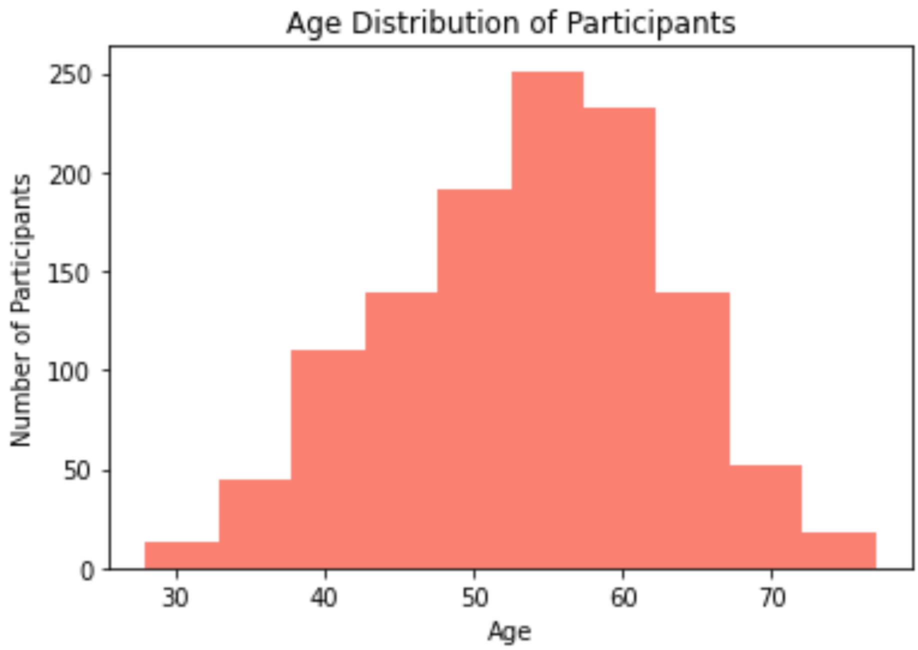


Figure 13: *Age distribution of patients.*

Figure 13 shows most people in the dataset are between 55-65 years old. The dataset does not contain many young (under 35) and elderly people (over 65), which can be a potential limitation because as seen in Figure 12, older age affects the likelihood of CVD.

Figure 14: *Gender distribution. 1 - Males, 0 - Females.*

Figure 14 shows the dataset contains more men although it is considered best practise to have an equal amount of people from both genders. Furthermore, being male has a positive correlation with having cardiovascular disease.

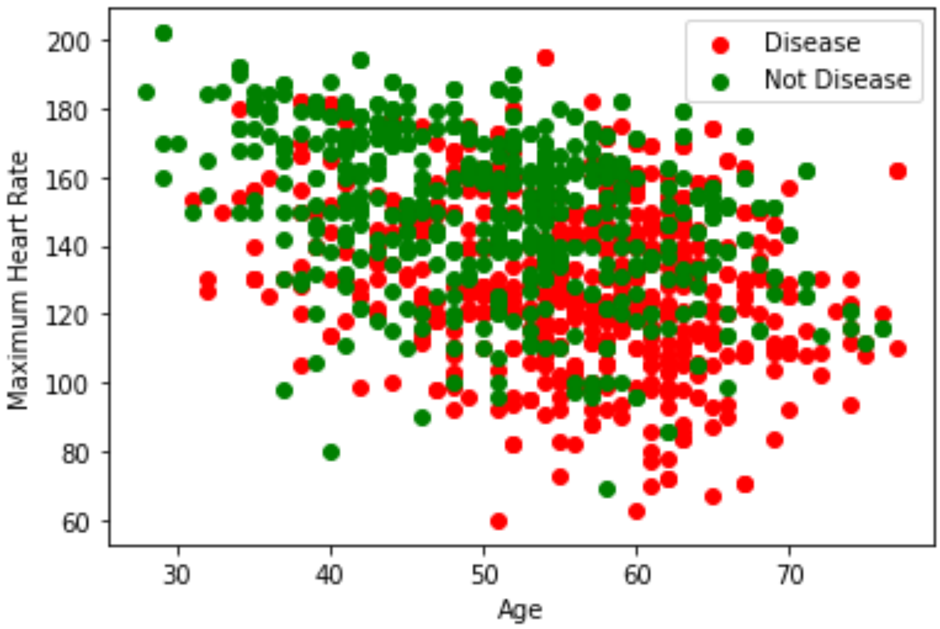


__Figure 15:__ Maximum heart rate against age.

CVD patients are usually older and have lower maximum heart rate when compared to healthy patients, who tend to be younger and have relatively high maximum heart rate as shown in Figure 15.

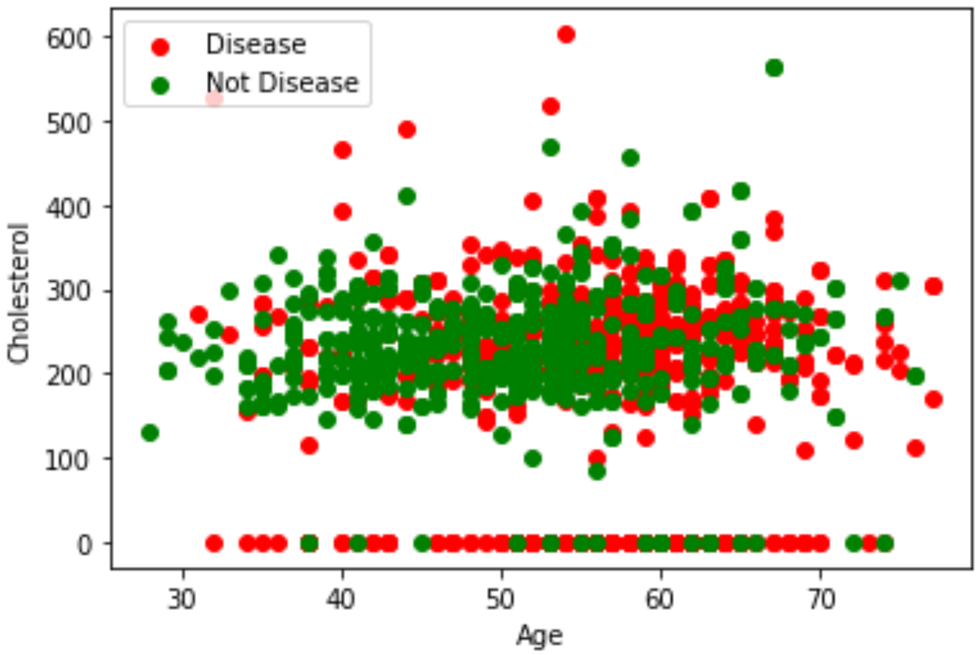


__Figure 16:__ Cholesterol against age.

Figure 16 indicates that there is no clear relationship between cholesterol and the presence or absence of CVD. Also, there are a few patients that have a cholesterol value of 0; this is a dataset error that should be corrected.

4.4.3 Summary of Data Exploration

In 4.4.1 it was found that there were no missing values in the dataset and feature coding was not necessary.

The dataset is reasonably balanced, with almost as many healthy patients as CVD patients. The dataset's age distribution inequality was also expressed, revealing that the majority of patients are middle-aged. Furthermore, as shown in Figure 18, the dataset contains more men than women.

This dataset visualization revealed a clear positive association between age and CVD, as well as the fact that CVD patients typically have a lower maximum heart rate. Furthermore, a mistake in the dataset was identified as there are a few patients who have a cholesterol value of 0, which is addressed in 4.5.1.1.

4.5 Developing the ML model

This section describes the process of developing an ML model that classifies a patient's condition.

4.5.1 Getting the dataset ready

The dataset was split into two, with one dataset containing all the features used to train the models (X), and the other containing only the target column (y), the presence or absence of CVD.

4.5.1.1 Fixing the cholesterol error – Process 1A

As shown in Figure 16 in 4.4.2, the dataset contained an error in which several (172) cholesterol values were 0. It is critical to correct them since missing data will distort the validity of the model and lead to incorrect conclusions [61].

There are 3 options to fix the problem:

1) Drop the missing data
2) Fill the missing data with the average or median of the column
3) Use KNN algorithm to predict the values of missing data

The first choice would require the deletion of approximately 15% of the data from the dataset. As a result, less training and testing data will be available, resulting in a less reliable model [62]. Option 2) is a fast solution, but it reduces the variance of the data, which could harm the model's learning [62]. In traditional ML fashion, choice 3) was selected.

KNN is used to predict and replace the value of the missing elements using other features in the dataset. To do this, it employs a method known as imputing, which is the process of replacing missing data with substituted values [74]. When imputing, we must be careful not to use the target feature, as this can cause leakage and reduce the efficacy of future models [62]. Therefore, the X dataset was used (without target).

Code 3 and 4 show the imputation process followed in Python.

```
# Replace 0's with NaN
X['cholesterol']=X['cholesterol'].replace(0,np.nan)
```

Code 3: *0s of cholesterol were replaced with NaN (to be able to execute imputation).*

```
# Impute missing values.
from sklearn.impute import KNNImputer
imputer = KNNImputer(n_neighbors=2)
X =  pd.DataFrame(imputer.fit_transform(X),columns = X.columns)
```

Code 4: *KNN imputation of missing values, using the X dataset.*

4.5.1.2 Splitting the datasets

The two datasets (X & y) are split again so that there are four datasets in total, two for training and two for testing (Code 5). The training datasets are used to train the model, and the testing datasets are used to evaluate how well the model performs on unseen data. The percentage of data that goes into the testing and training datasets is subject to our control. Using little of the available data for training may result in underfitting because the model is not complex enough and using too much data may result in overfitting [64].

To achieve this, different test size percentages were manually tried, with the accuracy score obtained by the Random Forest algorithm recorded (Figure 17).

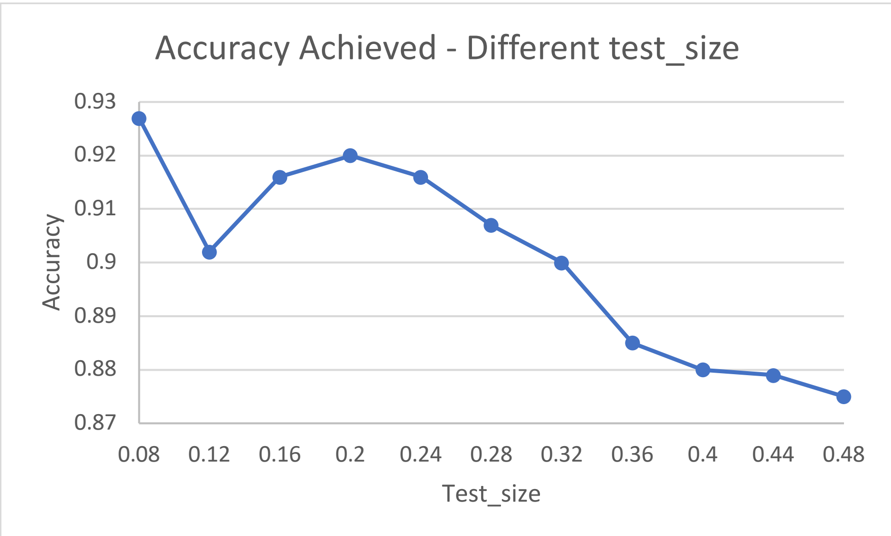


Figure 17: *Accuracy achieved against a range of test_size values using RF.*

The highest accuracy value was 0.93 when test_size was 0.08. However, using only 8% of the dataset for testing is not reliable, as there is a high probability that the high accuracy value is due to chance. It was therefore decided to choose 0.2 for test_size as it gives the second-best accuracy value and is more reliable because it has more data to test itself.

```python
#Split the data between train and test group
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y)
```

Code 5: *Splitting data into training and testing datasets.*

4.5.2 Building the models – Processes 1B, 1C, 2A

To increase code reusability, a function that trains ML models passed in and evaluates their accuracy was created, Code 6. The function takes a dictionary containing the models, as well as the training and testing data as parameters. This method was used to train and test the accuracy of 5 different ML algorithms: Random Forest, Catboost, KNN, Logistic Regression, and XGBoost.

```python
# Function to fit and score models.
def fit_score_models(models, X_train, X_test, y_train, y_test):
    """
    Fits and evaluates machine learning models passed in.
    """
    # Set random seed.
    np.random.seed(20)
    # Keep model scores in a dictionary.
    scores = {}
    # Loop through passed in models.
    for name, model in models.items():
        # Fit the model to the data.
        model.fit(X_train, y_train)
        # Evaluate the model and append its score to scores dictionary.
        scores[name] = model.score(X_test, y_test)
    return scores
```

Code 6: *Function that fits and scores the various models that are passed in.*

Upon call of the function, a dictionary mapping each model's name with their achieved accuracy is returned.

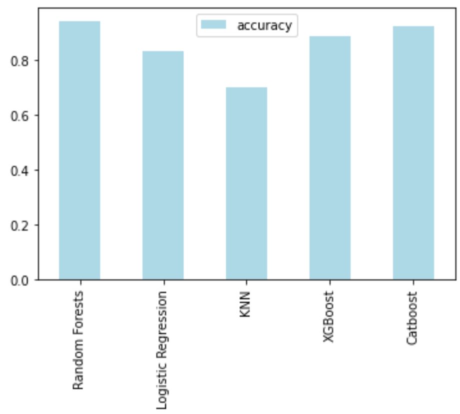


Figure 18: *Accuracy comparison between the 5 different models.*

Figure 18 shows that the 3 best-performing models are Random Forest, Catboost, and XGBoost.

4.5.3 Hyperparameter Tuning of 3 best models – Process 2B

Hyperparameters are the model's parameters that cannot be estimated by the model in the learning process from the given data, and as such, they are under the control of the programmer. Hyperparameter tuning is the process of determining the right combination of hyperparameters that will maximise the performance of the models [36]. This process was conducted on the 3 best-performing models.

4.5.3.1 Types of Hyperparameter Search - Grid Search Vs Random Search

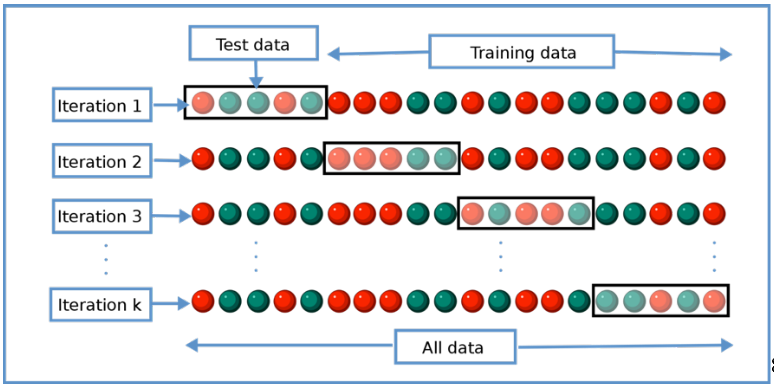
Grid Search is an exhaustive search through a grid containing different hyperparameter values. It produces a model and scores on the test data for each combination. In this process, each combination of hyperparameter values is measured, which can be inefficient. However, the optimal possible results are guaranteed. The runtime of Grid Search also depends on the speed of the algorithm being tuned [46].

Random Search picks random combinations from the grid to train the model and score. It allows the programmer to restrict the number of iterations-combinations that will be attempted. It runs much faster than Grid Search, but the optimal possible results are not guaranteed, and the global minimum is unlikely to be found [46].

For a relatively small number of possible combinations, Grid Search is a great tool that gives accurate results. Otherwise, it can be extremely computationally intensive and time-consuming. Randomized Search is not as effective but is much faster. Both methods were used depending on the number of parameters to be tried and the speed of the algorithm.

4.5.3.2 K Fold Cross-validation

K Fold cross-validation divides the data into K subsets. The holdout process is repeated K times, with one of the K subsets acting as the test/validation set each time, and the other K-1 subsets representing the training set. Thus, each data point occurs exactly once in a validation set and K-1 times in a training set (Figure 19). This reduces bias and variance since most of the data is used for both fitting and validation. The overall effectiveness of the model is determined by combining the error estimation in all k trials. [41]



[8]

Figure 19: *Visualisation of K-Fold Cross Validation.*

While conducting hyperparameter tuning of the models, Cross-Validation was assigned to be equal to five, which means that the dataset will be trained and validated with five different dataset subsets.

[8] https://miro.medium.com/max/3840/1*fW_qNKmvmg8duWoQnp_PoQ.png

4.5.3.3 Evaluating Accuracy

To evaluate the overall effectiveness of the classification models, different model performance measures besides accuracy, such as recall, precision, and F1 score were used.

Accuracy is the baseline evaluation for all classification models, it measures the number of correct predictions divided by the total number of predictions. Recall is the proportion of actual positives that are correctly identified. Precision is the proportion of true positives to total predicted positives. [47,48]

The F1 score combines the precision and recall of a classifier in a single metric by taking their harmonic mean. Suppose classifier A has higher recall, while classifier B has higher precision. In this case, the F1 values of the two classifiers can be used to decide which one performs better [63].

When all these performance metrics are used together, we get a representative picture of the true performance of a model. A function, that returns all of these accuracy metrics for a given model was created (Code 7).

```
#Evaluate a model's accuracy.
def evaluate_accuracy(model, name):
  y_preds = model.predict(X_test)
  print("Evaluating different criteria for "+name)
  print(f"Accuracy: {accuracy_score(y_test, y_preds)*100:.2f}%")
  print(f"Precision: {precision_score(y_test, y_preds)}")
  print(f"Recall: {recall_score(y_test, y_preds)}")
  print(f"F1: {f1_score(y_test, y_preds)}")
```

Code 7*: A method I created which evaluates various accuracy metrics of a model.*

4.5.3.4 Tuning of the algorithms

The tuning of the algorithms began with determining the best hyperparameters to experiment with. An overview of the hyperparameters tested, values tried, and results obtained are shown below.

Random Forest

The random forest (RF) algorithm has several hyperparameters that can be changed by the developer, below are the ones experimented with.

Hyperparameter	Description	Range of values allowed	Default Value
n_estimators	Number of trees. More trees lead to more generalized results. However, also increases the time complexity of the model.	>=0	100
max_features	Number of maximum features available to each tree.	“auto“, “sqrt“, “log2” and None.	*'auto'*

max_depth	The height of the trees. As height increases, the model accuracy increases up to a certain point before steadily decreasing due to overfitting in the model.	>=0	None
min_samples_split	Observations needed in any given node to split it. Increasing its value reduces overfitting of the model. However, it should not be so high as underfitting may occur.	>=0	2
min_samples_leaf	The minimum number of samples required in the leaf node after splitting a node.	>=0	1
bootstrap	Whether bootstrap samples are used when building trees.	True, False	True

Table 4: *Random Forest Hyperparameters [22,66].*

```
# Parameters to try.
params ={
"n_estimators" : [int(x) for x in np.linspace(start = 200, stop = 2000, num = 10)],
"max_features" : ['auto', 'sqrt'],
"max_depth" : [int(x) for x in np.linspace(10, 110, num = 11)],
"min_samples_split" : [2, 5, 10],
"min_samples_leaf" : [1, 2, 4],
"bootstrap" : [True, False]
}
```

Code 8: *The values chosen for each hyperparameter.*

RandomizedSearchCV was used as Random Forest needs significant time to run and there were thousands of possible combinations (Code 8).

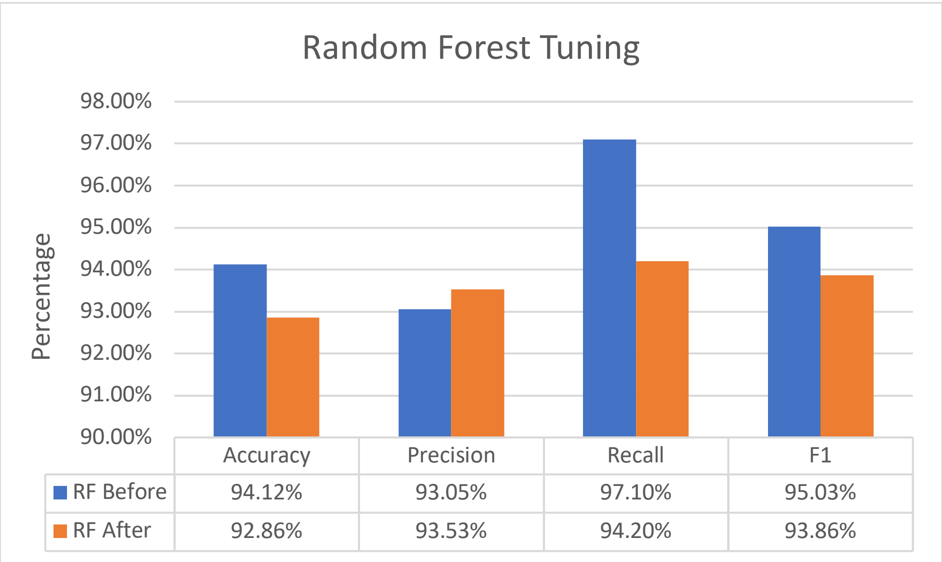


Figure 20: *Random Forest performance difference after hyperparameter tuning.*

Figure 20 shows that all the metrics except precision decreased, indicating that the RF tuning was unsuccessful. Therefore, the tuned model was discarded, and the original RF model was restored.

Catboost

The Catboost algorithm also has many hyperparameters that can be tweaked to increase its performance even more. These are the five hyperparameters that were tuned.

Hyperparameter	Description	Range of values allowed	Default Values
depth	The height/depth of the trees.	0 to 16	6
iterations	The maximum number of trees that can be formed.	>= 0	1000
learning_rate	Used to reduce the gradient step. It affects the total time of training. The smaller the value, the more iterations are required for the training.	0 to 1	0.010
L2_leaf_reg	Tries different values for the regularizer to find the best possible.	>= 0	3
border_count	The number of splits for numerical features.	1 to 65535	254

Table 5: *Catboost Hyperparameters [29].*

```
# Parameters to try.
parameters = {'depth':[3,1,2,6,4,5,7,8,9,10],
            'iterations':[250,100,500,1000],
            'learning_rate':[0.03,0.001,0.01,0.1,0.2,0.3],
            'l2_leaf_reg':[3,1,5,10,100],
            'border_count':[32,5,10,20,50,100,200]}
```

Code 9: *The values chosen for each hyperparameter.*

While Catboost is a relatively fast algorithm, an exhaustive search of these parameters (Code 9) would be very computationally and time-consuming. Therefore, RandomizedSearchCV was used.

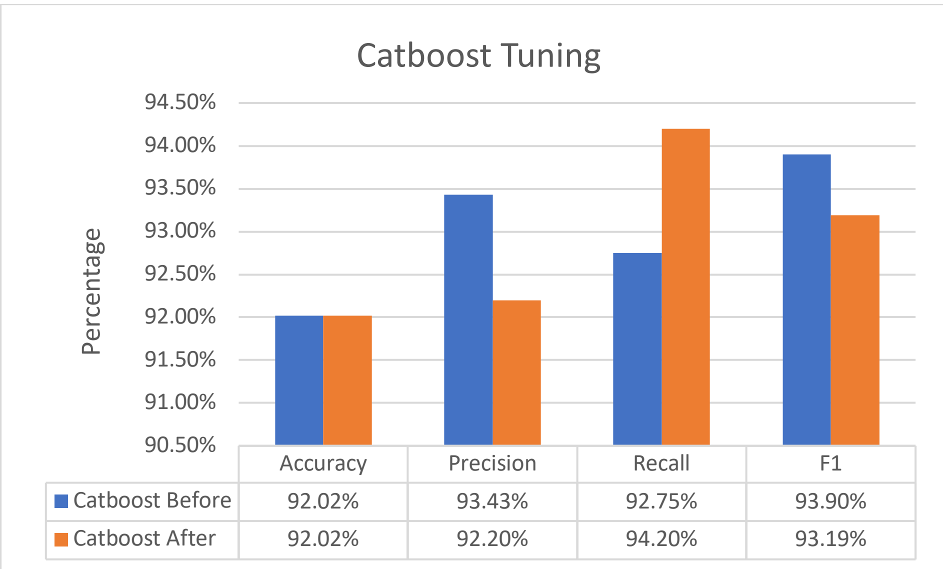


Figure 21: *Catboost performance difference after hyperparameter tuning.*

Figure 21 indicates that the hyperparameter tuning was beneficial to Catboost. The minor drops in precision and F1 score were considered, but the gains in recall are more important for a healthcare ML model (see 6.4).

XGBoost

XGBoost has several hyperparameters that can be adjusted to optimize its performance. These are the five hyperparameters that were tuned.

Hyperparameter	Description	Range of values allowed	Default Values
learning_rate	Reduces the weight of new features in each boosting stage, preventing overfitting or a local minimum.	0 to 1	0.3
max_depth	Maximum depth/height of a tree.	>= 0	6
Min_child_weight	Minimum number of instances needed in each node. The larger the value of min_child_weight the more conservative the algorithm.	>= 0	1
gamma	The minimum loss necessary for a leaf to split.	>= 0	0
colsample_bytree	The fraction of columns to be subsampled. Relates with the speed of the algorithm and prevents overfitting.	0 to 1	1

Table 6: *XGBoost hyperparameters. [21, 42]*

GridSearchCV was used to iterate though values in Code 10 as XGBoost runs much faster than the other algorithms due to its system and algorithmic optimisations.

```
# Parameters to try.
params={
 "learning_rate"    : [0.05, 0.10, 0.15, 0.20, 0.25, 0.30 ] ,
 "max_depth"        : [ 3, 4, 5, 6, 8, 10, 12, 15],
 "min_child_weight" : [ 1, 3, 5, 7 ],
 "gamma"            : [ 0.0, 0.1, 0.2 , 0.3, 0.4 ],
 "colsample_bytree" : [ 0.3, 0.4, 0.5 , 0.7 ]
}
```

Code 10: *The values chosen for each hyperparameter.*

	Accuracy	Precision	Recall	F1
XGBoost Before	88.66%	92.37%	87.69%	89.96%
XGBoost After	92.44%	94.78%	92.03%	93.38%

Figure 22: *XGBoost performance difference after hyperparameter tuning.*

XGBoost's hyperparameter tuning was successful, as all accuracy metrics scores improved (Figure 22). The 3 most significant improvements were recall, accuracy, and F1.

4.5.4 Exporting the best model – Process 2C

Random Forest classifier was the best performing model and was exported from Colab using **pickle**, a module for converting a Python object (list, dictionary, model, etc.) into a character stream. The idea is that this character stream contains all of the information needed to rebuild the object in a subsequent Python script [55] that will be used on the web platform.

4.5.5 Summary

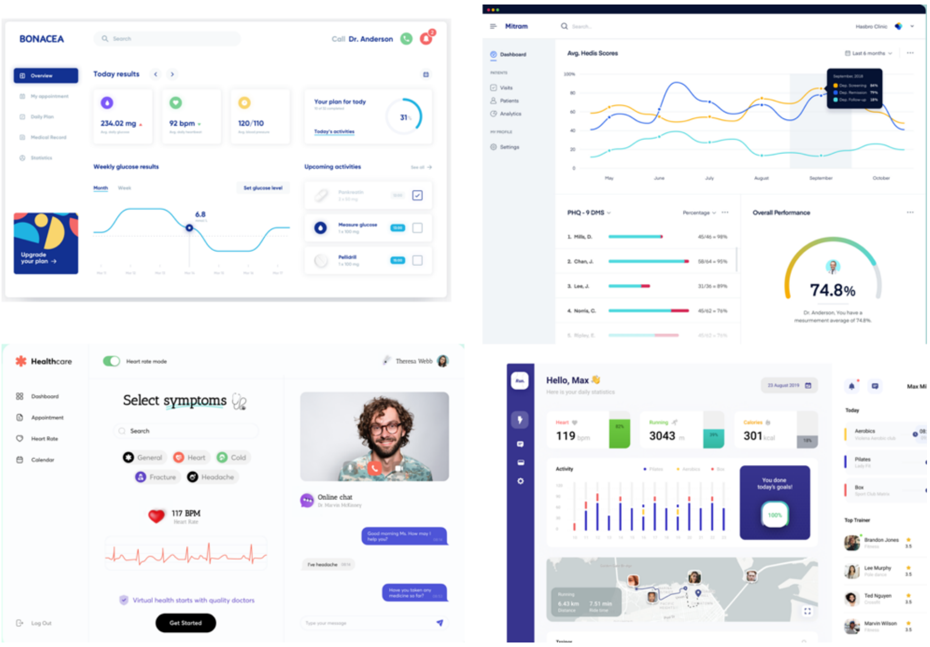
The dataset was first split into training and testing datasets, and the five models were trained for the first time. Hyperparameter tuning was then performed to improve the output/accuracy of the three best-performing models. Finally, Random Forest was deployed.

4.6 Developing the platform

The development of a web platform that meets the objectives of this project took place in several steps. This section describes the sequence of this process and how it was carried out.

4.6.1 Designing – Process 3A

To design an effective, appealing web platform that doctors can use with ease, inspiration was taken from dribbble [40], an online platform where talented designers share their creations.



UIs 1: UIs for medical platforms shared on Dribbble.

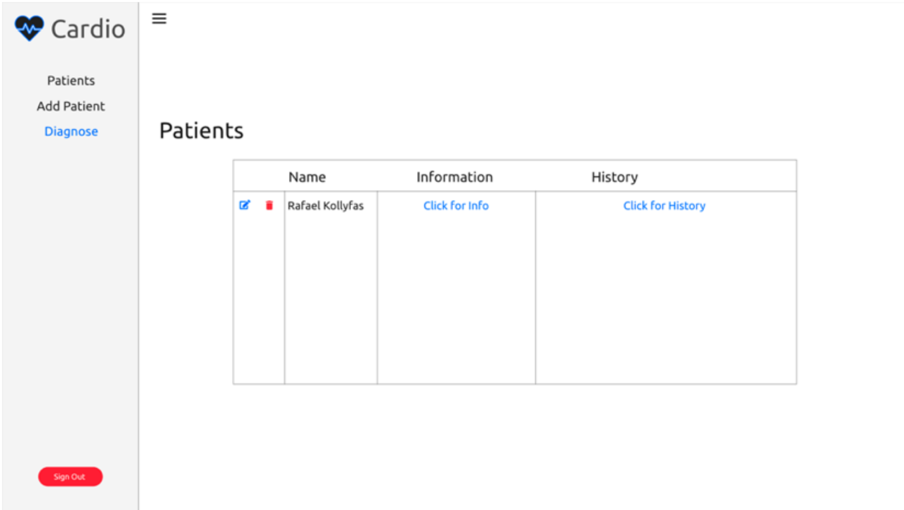
UIs 1 shows the UI of different medical/health-focused websites which inspired the design of the platform.

As we can see, the designs share a few similarities/patterns:

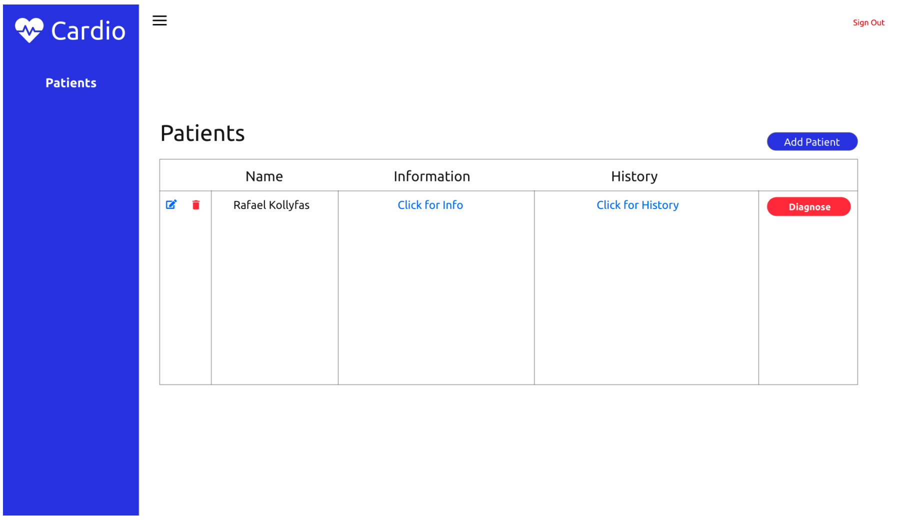
1. All of them have a sidebar that shows the different pages of the website.
2. Graphs are a vital part of showing medical information.
3. Deep Blue is a very common colour on medicinal UI.
4. All of them have a light background.

Taking these observations into consideration, Adobe XD was used to develop a prototype for the platform. The system's design evolved gradually, and the prototype's first iteration was minimalistic. Considering Dr V requirements 1) and 2) in 3.1.1, the design was changed by adding a deep blue colour to the sidebar and allowing patients to be added from the “Patients” page. The final system resembles the final prototype, but it has minor improvements in UI such as breadcrumbs and icons.

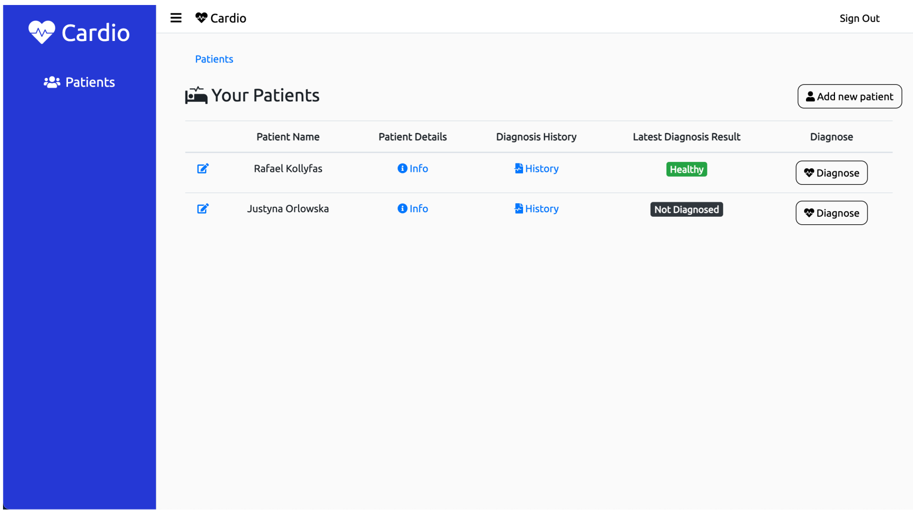
Platforms 1-3 below show the evolution of the “Patients” page, from the initial prototype to the final system.



Platform 1: *Initial Design*



Platform 2: *Improvements in the design based on Feedback.*



Platform 3: *Final design of the system.*

4.6.2 Creating the website – Process 3A

This subsection describes the page structure of the platform, how Flask works and was used, and finally the backend of the platform.

4.6.2.1 Website pages structure

Figure 23 shows a navigation diagram of the different pages of the web platform.

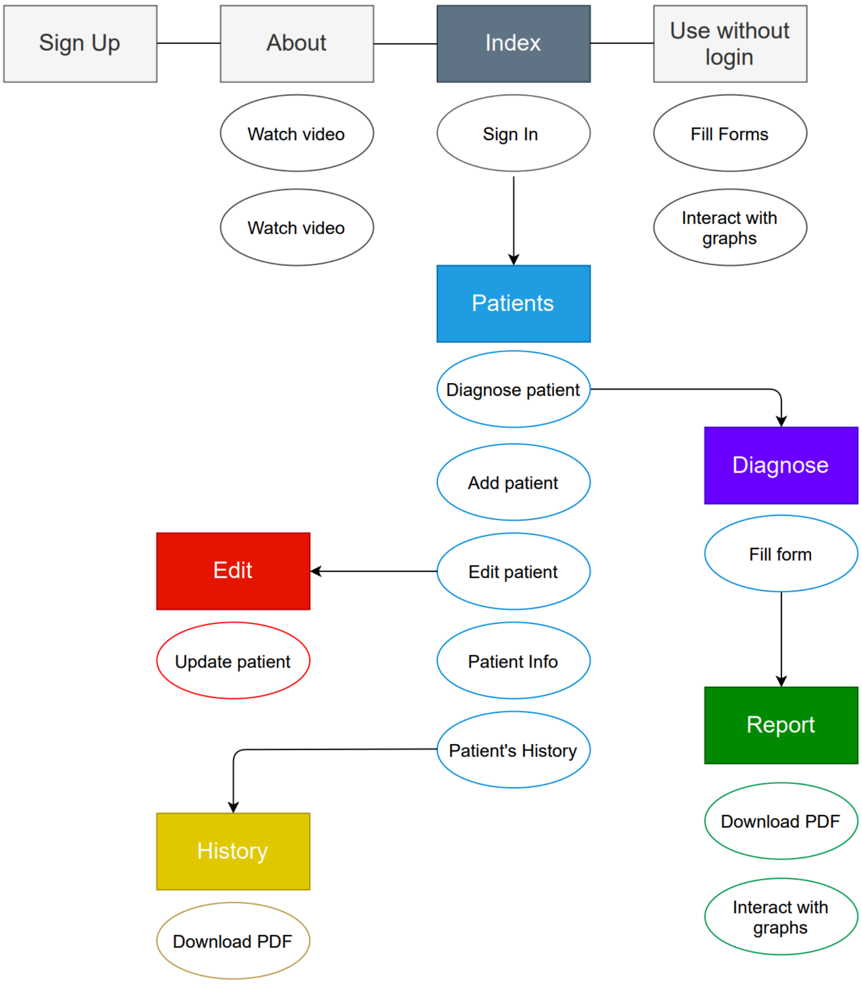


__Figure 23:__ Navigation diagram of all the pages of the platform. Rectangles are pages and the circular objects are functionalities.

Table 7 describes the functionality of each page.

Pages Functionality	
Page	**Description**
Index	Doctors can sign in with their details.
About	Tutorial video and a user manual.
SignUp	Doctors can sign up using their email.
No_login	Fill form to diagnose a patient without the necessity to login. The data does not get stored in the database.
Patients	Doctors can view their patients. They can also: 1) Diagnose a patient 2) Add a new patient

	3) Open edit page for a patient's information 4) Open history page of a patient
Diagnose	A doctor can fill a medical form to diagnose a patient's condition.
Report	An automated report gets created upon diagnosis, stating the model's prediction along with several other graphs.
History	View a patient's diagnosis history. Download a PDF showing the various charts generated for that diagnosis.
Edit Patient	Update patient's information.

Table 7: *The functionality of each page in the platform.*

4.6.2.2 Flask

Flask consists of two major components Werkzeug which handles routing, debugging, and the Web Server Gateway Interface (WSGI) and Jinja2 which serves as Flask's template engine [4].

Flask's concept of routes and view functions allow the client to send requests and the webserver to handle them. For example, when a user clicks the submit button on the "add patient" form, the client sends a request to the WSGI server by appending the text /addpatient at the end of the URL. The WSGI server then calls the function specified for that path, which executes the back-end code to process that specific request (Figure 24). The same thing happens when a user navigates to a page, a request is sent for that page, and the server renders and displays it [52].

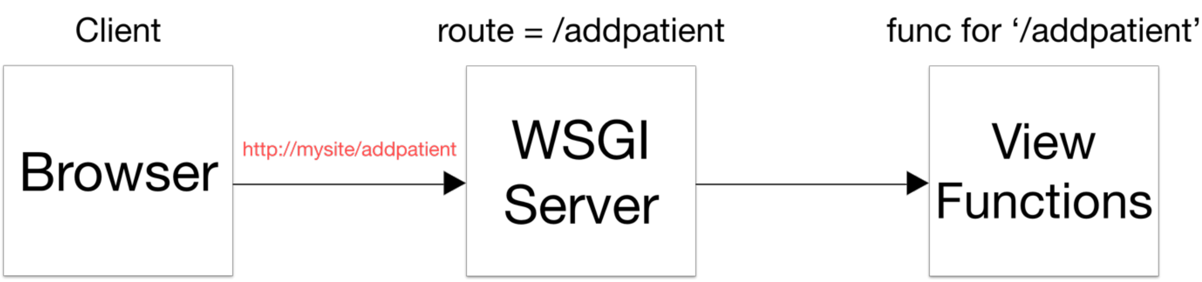


Figure 24: *The process for a request.*

Jinja2 replaces variables in a template by values passed when the template is rendered making it dynamic. The most powerful feature of Jinja2 is template inheritance which enables the creation of a base skeleton template that contains all of the application's common elements as well as blocks that child templates can override. This feature was used by creating a template called layout.html, which was inherited by the majority of HTML pages of the platform. It includes the application's basic HTML skeleton as well as its dependencies, stylesheets, and scripts. Additionally, Jinja2 supports the creation of reusable components [53]. When these features are combined, an otherwise complex page appears to be clean and concise as shown in Code 11.

templates > report.html > ...

```
{% extends 'layout.html' %}
{% block scripts %}{% endblock %}
{% block body %}

    {# Download PDF. #}
    <div class="button-right">
        <button class="btn btn-primary mt-3" id="download">
            <i class="fas fa-download"></i>
            PDF
        </button>
    </div>

    {# Prediction. #}
    <h2 class="mt-5 mb-5 text-center">{{pred}}</h2>

    {# Explainable AI. #}
    <div class="mb-5">{{exp|safe}}</div>

    {# Indlude Graphs #}
    {% include 'includes/_graphs.html' %}

    {# Include message box #}
    {% include 'includes/_add_message.html' %}

    {# Add download PDF functionality #}
    {% include 'includes/_pdf.html' %}

{% endblock %}
```

Code 11: *Separation of concerns on a complicated page.*

4.6.2.3 Firebase – Process 3B

Firebase Authentication was used to enable users to sign up with their email. Once a user enters their details in the sign-up form, the data gets sent in the “signUp” view function on Flask server which evaluates the data and then calls a firebase script for signing up the user.

For database, Firebase currently offers two cloud-based solutions that support real-time data synchronization: Cloud-Firestore (CF) and the Real-Time Database (RT). Even though both databases are excellent, CF is not fully supported with Python and if chosen, all operations would have to be performed on the client-side of the website, which would be neither ideal nor safe. Therefore, RT was chosen.

RT is Firebase's original cloud-based database, and it is a fast, low-latency solution that allows real-time synchronization of client states [7]. It uses NoSQL and stores data as a JSON tree making it easy to store and read simple data but harder to perform complex queries. Therefore, we need to be careful when structuring the data to avoid complex queries as much as possible. RT can scale to around 200,000 concurrent connections and 1,000 writes/second in a single database making manual scaling of the platform redundant if there is a surge in demand.

Knowing the benefits and limitations of RT database, the data was structured accordingly.

Firebase 1: *A doctor’s document.*

The doctors are identified by a unique identifier (uid) that is generated when they sign up. They are stored in the database with that specific uid and their patients are stored as separate children under their “Patients” document as shown in Firebase 1. That means, once we have a doctor's uid, we can access their patients from the same document.

Firebase 2: *The structure of a patient’s document.*

The document shown in Firebase 2 contains all of the patient's data in one place. This eliminates the need for looking at another table or making complex queries. For example, despite their very different purposes, both the “Patients” and “Diagnosis History” pages retrieve data from the same document.

Upon a patient’s diagnosis, a PDF is generated containing various graphs (see 4.6.4.2). To store this PDF, the Cloud Storage capabilities of Firebase were used. Once stored, its path is also saved on the database as shown in Firebase 2 on the “pdf” child, this allows the easy download of that PDF. The same path logic is used for storage (*Doctor → Patient → Diagnosis)*.

Firebase 3 below shows the rules written on the Firebase server to ensure that only the current user can access their data. Any read or write request will only be executed if the rules are followed. This prevents unauthorized users from accessing and manipulating data that does not belong to them.

```
{
  "rules": {
    "$userId": {
      // Read only the user's data.
      ".read": "auth.uid == $userId",
      // Write only to the user's data.
      ".write": "auth.uid == $userId"
    }
  }
}
```

Firebase 3: *Firebase server rules.*

4.6.3 ML on web platform – Process 3C

The model was imported in the Flask web server with pickle and was utilized to make predictions on patients' data.

Here is a flowchart showing the diagnosis process on the web platform

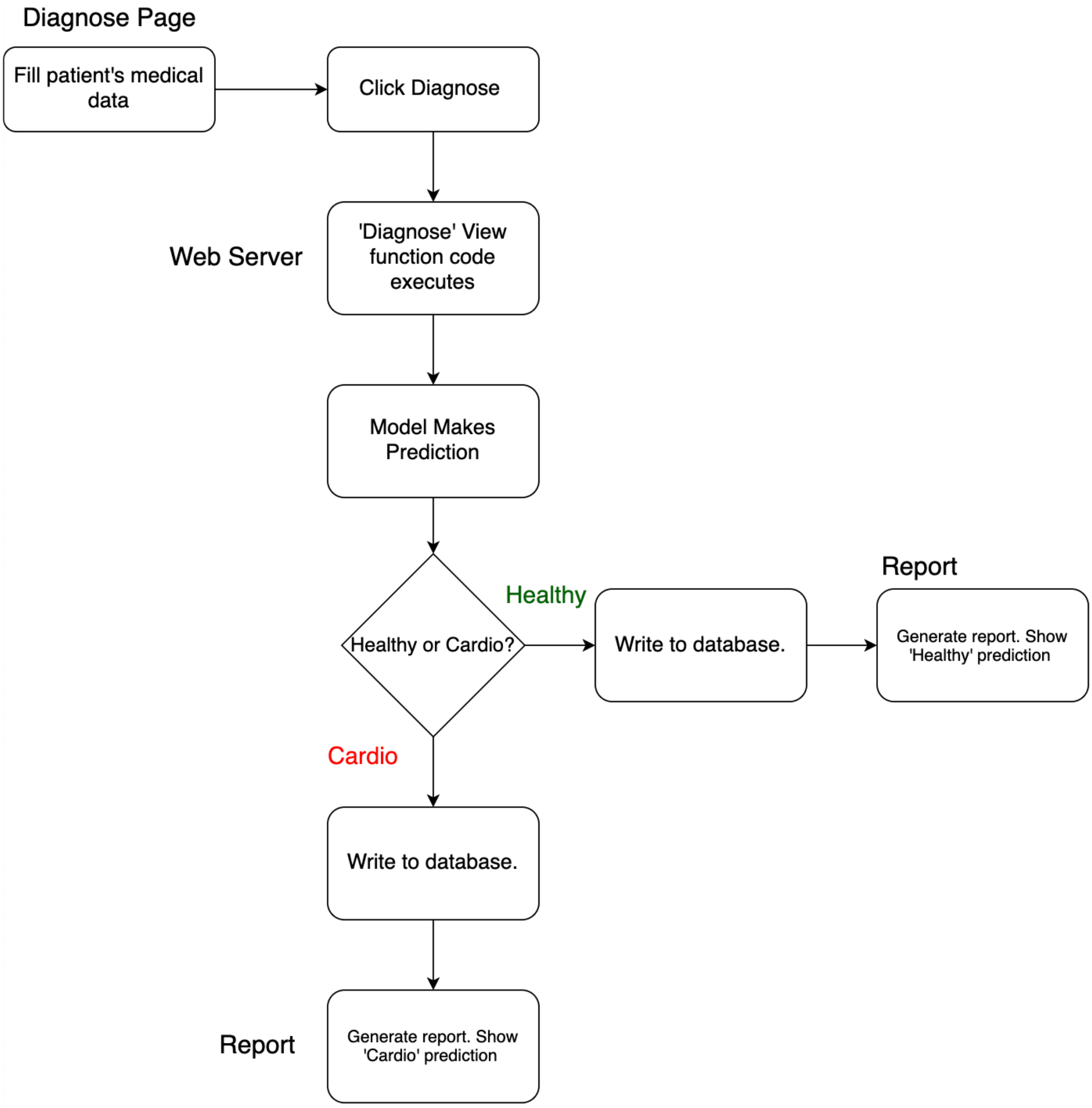
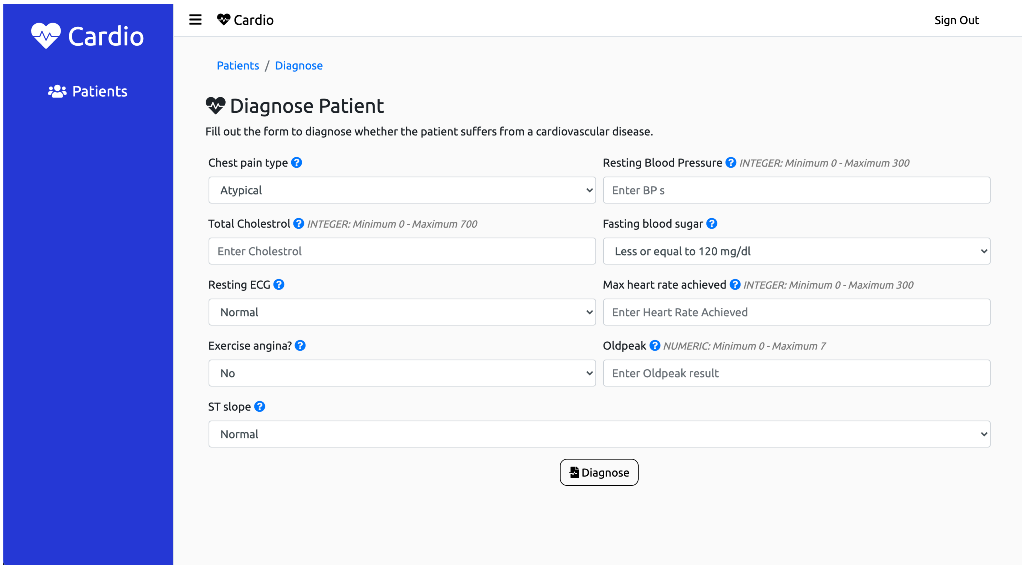


Figure 25: The diagnosis process on the website.

The same process applies for the "no_login" page but without writing to the database



Platform 4: *The Diagnose page.*

When a doctor fills in all of the patient's medical parameters shown on Platform 4 and clicks the "Diagnose" button, parsley is used to validate the input on the client-side. If all of the inputs pass parsley's validation, they are sent to the diagnose view function as a POST request. The server performs further error checking and then passes these inputs to the ML model to make a prediction.

```
# Write to the database.
if prediction==1:
    # Save to database.
    save_to_db(ct, userID, pid, chest, bps, chol, fbs, ecg, maxheart, exang, oldpeak, stslope, 1)
    # Redirect to the report page.
    return redirect(url_for('report', pred = "Suffers from a CVD", neg = prob_neg, pos = prob_pos, pid = pid, ct = ct ))
else:
    # Save to database.
    save_to_db(ct, userID, pid, chest, bps, chol, fbs, ecg, maxheart, exang, oldpeak, stslope, 0)
    # Redirect to the report page.
    return redirect(url_for('report', pred= "Most Likely Healthy", neg = prob_neg, pos = prob_pos, pid = pid, ct = ct ))
```

Code 12: *ML model decision and action.*

Code 12 shows that depending on the prediction of the model, different values for variables are passed in the "Report" page and written to the database.

4.6.4 The medical report – Process 3D

When the system redirects to the "report" page, all of the data to be displayed using graphs are prepared. The meaningful graphical representation of data is fundamental in providing the doctor with a clear understanding of the patient's condition.

4.6.4.1 Choosing a graphical library

To satisfy the graphical requirements set by Dr V in 3.1.2, a graphical library was required. D3.js and Chart.js are two popular and powerful JavaScript libraries for data visualization [30], however, they take very different approaches.

Chart.js allows developers to use a variety of pre-built charts. It requires minimal programming and the charts have a legend, hover pop-ups, serial logging, and are completely interactive. Chart.js includes customizable components and supports basic animations [54]. These features make Chart.js a good choice for smaller projects that require simple visualisations.

D3 can produce powerful visualizations by allowing CRUD operations on HTML/SVG elements, animations, interactions, and data-structure-based arrangement. D3 is not responsive by default but it can be, and custom styles are supported. D3 offers a lot more options than Chart.js, however, it has a steep learning curve, necessitates a lot more code and time, and is thus inappropriate for this specific project mainly due to time constraints [54,56].

The demanding schedule of the project, and the fact that Chart.js covers all the requirements set by Dr V, led to the decision to use it.

4.6.4.2 Dynamically generating the report

The report page compares the user's current diagnosis to other patients in the dataset on various criteria. To accomplish this, a new Python file called "dataset.py" was created in which various functions to collect data from the dataset were written e.g., collecting all healthy/CVD patients or determining the number of healthy/CVD patients who have more/less value for a particular feature. These functions were called from the main webserver and the information was then passed to the Jinja template engine as arguments which are used to generate the graphs shown in Figure 26 using Chart.JS in the “graphs” component of the application.

***Figure 26**: Graphs generated on the report section in a test performed.*

4.6.5 Explainable AI

To satisfy requirement 4) in 3.1.1 stating that an explanation of the model’s prediction is needed, an Explainable AI library had to be used. Explainable AI displays textual or visual artifacts that provide a qualitative understanding of the relationship between the components of the instance (e.g., levels of cholesterol, glucose) and the prediction of the model [60]. If the explanations are credible and understandable, the model is more reliable. The two libraries that were considered are SHAP and LIME.

Local Interpretable Model-Agnostic Explanations (LIME) is a novel explanation technique that learns an interpretable model locally around the prediction to describe any classifier's predictions in an interpretable and faithful manner [60]. It integrates interpretability into both the optimization and the concept of interpretable representation, allowing it to accommodate domain and task-specific interpretability requirements [58]. LIME faithfully explains the predictions of any model in an interpretable way as shown in Figure 27.

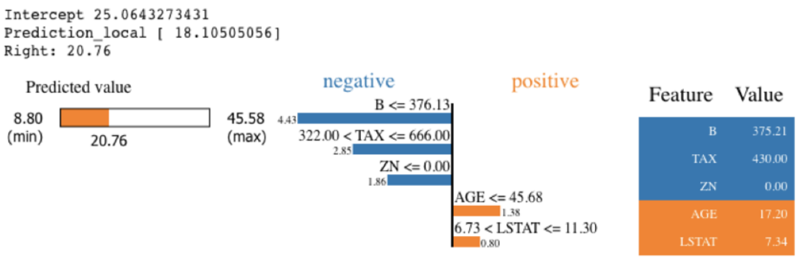


Figure 27: *LIME interpretability.*

SHapley Additive exPlanations (SHAP) applies a game theory approach (Shapley Values) to ML to "fairly assign contributions" to model features for a given output and make understandable interpretations (Figure 28).

Figure 28: *SHAP interpretability.*

Shapley values take into account all possible predictions for an instance based on all possible input combinations. SHAP can guarantee properties such as continuity and local accuracy as a result of this thorough approach, although comes at the cost of longer running times [57].

Given requirement 3) in 3.1.1 for a report generation within 30 seconds and the fact that LIME can faithfully interpret any model, it was chosen to interpret my model's predictions.

4.6.6 Summary

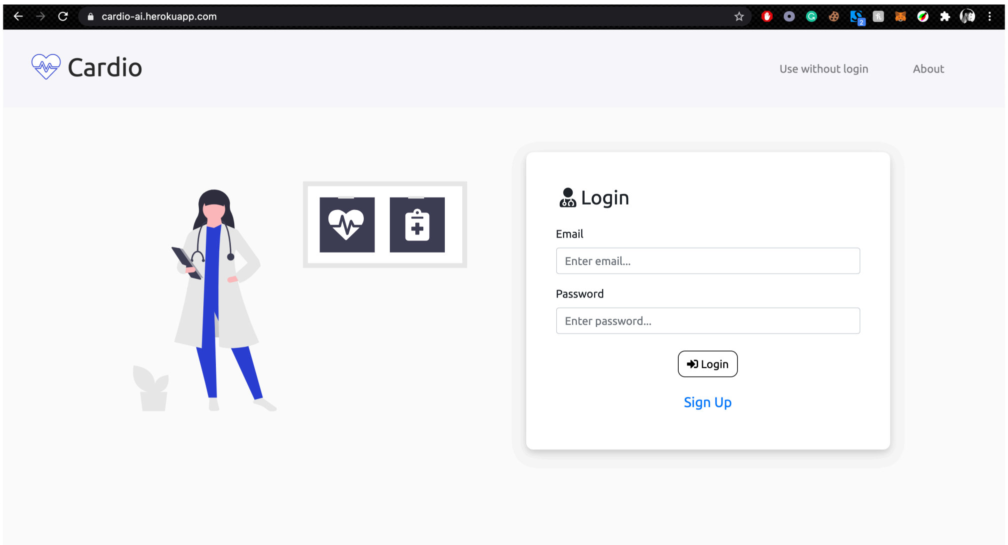
Medicinal UI images and feedback from Dr V helped the evolution of the platform's design. The page structure of the application was described and then the app's backend services were presented where RT was found a more suitable database than CF as it is fully supported with Python. Finally, LIME was chosen as the Explainable AI library because it is faster than SHAP and can correctly interpret a given model.

4.7 Deployment & Supplementary

Once the platform's implementation was completed, Heroku was used for hosting. It uses Git to create a server similarly to a localhost, but with the difference that it is visible to everyone [37]. For deployment on Heroku, a requirements.txt file listing all the dependencies of my application was needed, this was done as shown in Code 13. The web application's GitHub repository was then integrated into Heroku, and the deployment has been done automatically ever since as soon as changes are committed to the master branch.

```
(flask) (base) Rafaels-MBP:Web-diss rafaelkoll$ pip freeze > requirements.txt
```

Code 13: *Creation of requirements.txt based on application dependencies.*



Platform 5: *Deployed platform.*

Platform 5 shows the deployed platform, available at: https://cardio-ai.herokuapp.com/.

The supplementary material includes all of the written code for this project, a user manual (also available in *Appendix 7*) and a video tutorial on how doctors can use the platform.

4.8 Summary of Methodology & Development

This chapter first introduced the workflow and tools used for the development of this project. Then the dataset was explained and visually explored where an error was found and fixed. The development of the model was then described with RF being selected as the best model. The web development of the platform was described from the design phase to deployment. The next chapter tests whether the developed model and platform are reliable.

5. Testing

This chapter determines the correctness of the model and the resilience of the platform. Section 5.1 uses the expertise of Dr V to evaluate the factors influencing the model's prediction according to LIME. Section 5.2 ensures that the platform is reliable and that errors are appropriately handled and monitored.

5.1 ML model

Since we are using an ML model in a real medical scenario, it is important to ensure the factors leading to the model's predictions are appropriate. To accomplish this, Dr V completed the input form on the "Diagnosis" page and evaluated the returned LIME explanation.

Below is a detailed table evaluating the model's decision:

Variable	Value Tried	LIME Prediction	Correctness
Age	77	CVD	YES
Chest Pain Type	Asymptomatic	CVD	NO
Resting Blood Pressure	138	Healthy	YES
Cholesterol	308	CVD	YES
Fasting Blood Sugar	Less or equal than 120 mg/dl	Healthy	YES
Resting ECG	Abnormality in ST – T wave	CVD	YES
Max Heart Rate	60	CVD	YES
Ex. Angina	No	Healthy	YES
Oldpeak	2	CVD	YES
ST slope	Downsloping	CVD	YES
FINAL DIAGNOSIS		CVD	YES

Table 8: *Dr V's evaluation of LIME interpretation.*

Table 8 shows that the model correctly predicted the relationships for the majority of the features. Dr V disagreed with the model at Chest Pain Type, arguing that the attempted value "Asymptomatic" is not a symptom of CVD. Besides, all other returned explanations appear to be correct, and Dr V agreed with the final diagnosis of the system.

5.2 Web Platform

On the server side, JUnit tests and error handlers were written to make the platform more resilient to errors and an error-monitoring system was integrated.

5.2.1 Junit Tests (*Appendix 3*)

Unit testing entails testing specific units of source code to determine if they execute correctly [73]. Following the completion of the system's implementation, three separate unit test classes containing seventeen different tests were created to ensure that the app runs correctly. The testing revealed a few errors that were fixed.

Table 9 describes the purpose of each test class:

Class	Purpose
1. test_load	Tests that the pages of the app load without errors.
2. test_forms	Tests that the forms of the app execute without errors.
3. test _content	Tests the content of each page when correct and incorrect parameters are given.

Table 9: *Purpose of each test class. Test classes are included in Appendix 3.*

Since the testing occurred at the end of the development of the system, there was no additional functionality that needed to be tested and subsequently, no additional tests were written. However, when refactoring the code, regression testing was done after each update/enhancement. Regression testing is a re-run of all the tests to ensure that the previously developed and tested software still works [83]. All regression tests were successful, indicating that the functionality of the system was not affected by the changes.

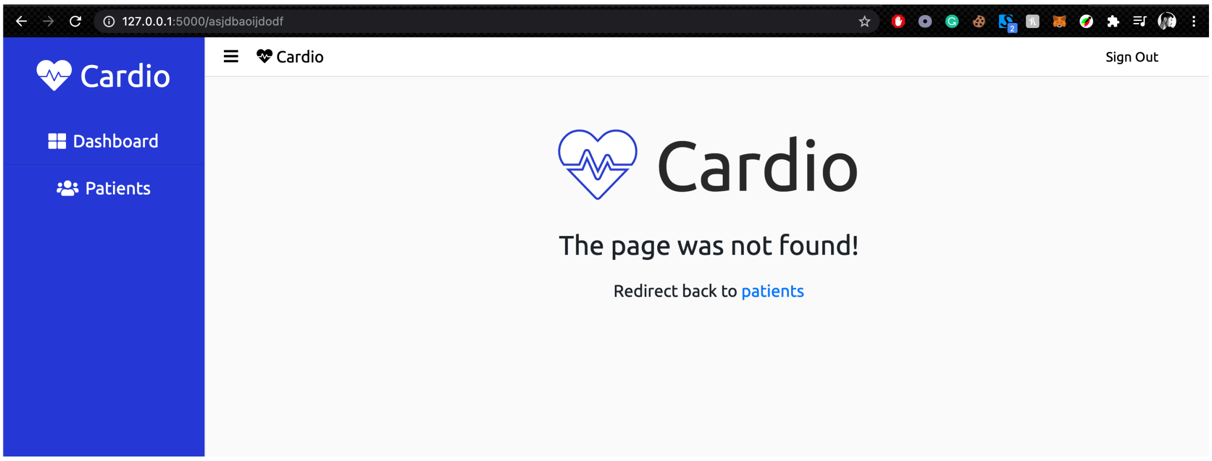
5.2.2 Error handlers

An error-handler is a function that returns a response when a certain type of error occurs, similar to how a view is a function that returns a response when a URL is requested [82].

The most common web error is the "not found-404 error". It is usually triggered when the URL entered in the browser is invalid. Unhandled 404 errors result in a poor user experience because the style of the page displaying the error is incompatible with the other pages [72]. Therefore, the function in Code 14 was created to handle 404 errors and redirect them to a custom page shown in Platform 6.

```
# Not found error.
@app.errorhandler(404)
def not_found(e):
    return render_template("404.html")
```

Code 14: *404 server-side error handler.*



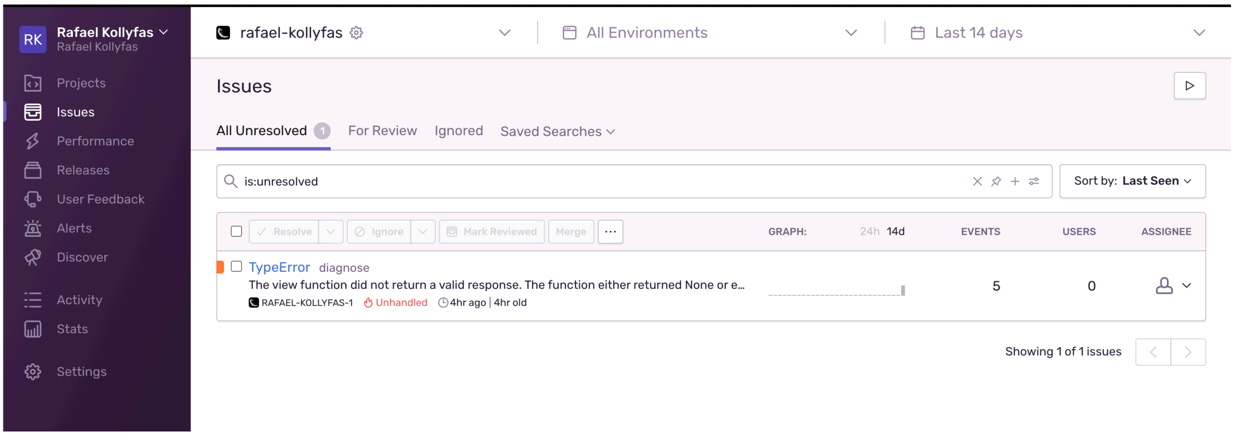
Platform 6: *Custom 404 page.*

The 403 and 500 errors are also handled similarly.

5.2.3 Error monitoring and tracking

Error monitoring and tracking tools are used to proactively detect, diagnose, and fix errors that were not known to exist [80]. The web platform uses Sentry.io, an open-source full-stack error tracking system that provides all the information necessary to identify and respond to errors in real-time [81]. Its explanation includes where and when the error occurred, an easy-to-understand definition of the error, and the body and query string of the request that caused it.

When a new error is found, it is immediately logged in the "Issues" tab of Sentry.io as shown in Sentry 1, and a notification is sent to my phone. These features ensure that unforeseen errors are detected and fixed promptly.



Sentry 1: *Sentry Issues tab.*

5.3 Summary of Testing

This chapter first showed that the model behaves well overall as it correctly defined the relationships for the majority of features with CVDs except with Chest Pain Type. The following section showed the platform's unit tests, error handlers, and error-monitoring system. The next chapter critically evaluates the project and the results it produced.

6. Results and Evaluation

This chapter first evaluates the effectiveness of the software engineering process followed and then discusses the confidentiality measures taken. Section 6.3 describes the improvements that were derived from the second interview. The ML models' performance is then evaluated in 6.4, followed by a critical evaluation of the fulfilment of the project's aim and objectives in sections 6.5 & 6.6 respectively. Finally, 6.7 presents results comparison with a similar project, and 6.8 compares the platform with HeartScore.

6.1 Software Engineering Process

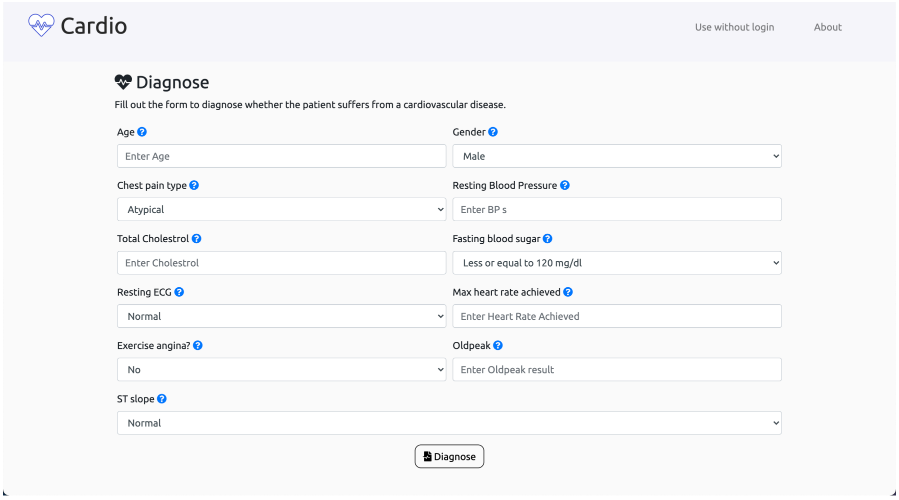
The initial work plan (*Appendix 4*) was followed as much as possible, however, the demanding obligations from other modules lead to a slight push in the completion date of the web platform. Furthermore, there were unexpected new tasks that were implemented very close to the deadline as a result of constructive feedback received.

The Agile development methodology was effective as frequent feedback allowed for continuous improvement of the platform. It led to the creation of new measurable requirements and features that were not originally considered, e.g., adding diagnosis comments (see 6.3) and diagnosing without login. The testing of the system was successful in identifying unknown errors, and the inclusion of an error-monitoring system also helped in ensuring that the platform will continue to be improved and new errors to be fixed. Finally, the deployment of the platform was conducted without facing any issues and continues to be updated automatically whenever new commits are made.

6.2 Confidentiality

Healthcare professionals have an ethical obligation to protect patient confidentiality, including their personal information and medical records [84]. Consequently, the platform respects and follows these obligations, requiring only the patient's age and gender from their personal data as these are needed by the model. Any other personal data, such as the patient's name or email, are unnecessary for this system and could also pose a risk of revealing the patient's identity and medical history in the event of a cyberattack.

A "use without login" page shown in Platform 7 helps doctors use the system simply and confidentially as it allows the rapid diagnosis of a patient without storing its results in the database.



Platform 7: *Use without login page.*

6.3 Improvements led by the 2nd interview

The free-flowing conversation style of the interviews helped uncover topics that were not thought of before and led to improvements in the final system. This section describes the improvements that resulted from the platform's evaluation done on the second interview.

These are the main points raised by Dr V and how they were addressed.

1) "You can't have a cardiological platform without having 'smoking' as a feature. It is extremely correlated."

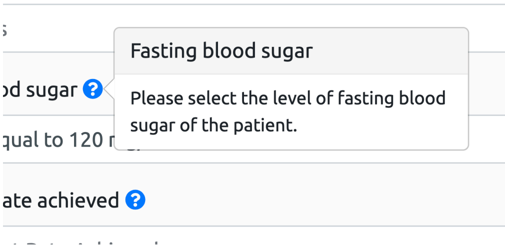
The dataset has a limitation as it lacks a "smoking" feature. However, as explained to Dr V, the chosen dataset was the right choice among the available datasets, since most of the alternatives had a lot of missing data or were predicting death (not disease). Furthermore, the chosen dataset combines data from five different institutions which makes its data more generalisable.

Although it is possible to combine the current dataset with another one which includes smoking and then impute the missing values, this would require a lot of time and research to get it right on such a large scale (1190 patients). Unfortunately, as the deadline for this project was approaching, this was impossible as all the models would need to be re-trained and the platform adjusted, which could become a major risk to the successful completion of the project on time.

2) "If you want to build a web platform for doctors, you have to speak our language"

He noted that it was obvious that this platform was developed by someone without medical expertise. One example is the "Diagnosis" page with the input screen having the variable "Cholesterol". He explained that there are different versions of cholesterol, so it is unknown which one is expected by the system. Upon further examination of the dataset, Dr V concluded that this is the "total cholesterol".

He also found some other minor inconsistencies, e.g., on the instructions of Resting Blood Pressure, the unit of measurement was written as mm/HG where it should have been mm/Hg. It is a minor error, he admits, but "these mistakes add up and make all the difference in the professionalism of the platform".



Platform 8: *Fasting Blood Sugar original description.*

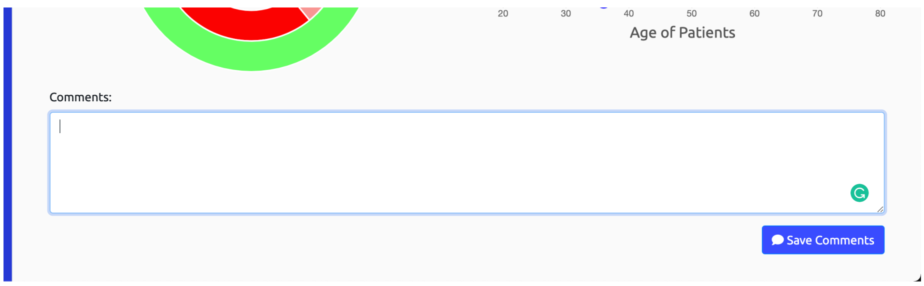
The description of "Fasting Blood Sugar" shown in Platform 8 was also wrong. According to Dr V, it should have been "the blood sugar of a person in a fasting state," meaning that they have not eaten for 8 hours.

Another minor adjustment was to change the "Flat" ST slope option to "Horizontal" as it is more accurate.

All of these changes were made in the final system.

3) "Give us the ability to add comments after each diagnosis-report."

Dr V felt that one crucial feature was missing from the generated report: the ability to add comments to each diagnosis. He noted that if doctors were given the ability to write comments on the same page as the graphical report, they will be able to immediately note any correlations and findings they see after a diagnosis.



Platform 9: *Add comment feature.*

This feature has since been implemented at the bottom of the report as shown on Platform 9. The comments are stored in the database together with the diagnosis and are visible on the "History Diagnosis" tab.

6.4 ML Model Performance Evaluation

This section introduces a new evaluation metric in 6.4.1, which shows the diagnostic accuracy of a model. Subsection 6.4.2 includes a discussion about reliability in a medical ML model and finally, the overall performance of the three best models is compared in 6.4.3.

6.4.1 Area Under the Receiver Operating Characteristics (AUROC) - (*Appendix 5*)

Each prediction of an ML model falls into one category of four outcomes shown below:

Outcome	Explanation
False Positive (FP)	Actual class is NO, predicted class is YES.
True Positive (TP)	Actual class is YES, predicted class is YES.
False Negative (FN)	Actual class is YES, predicted class is NO.
True Negative (TN)	Actual class is NO, predicted class is NO.

Table 10: *The possible predictions of a model.*

An ROC curve is a graphical representation of a binary classifier's diagnostic ability [71] and is constructed by plotting the true positive rate (TP) versus the false positive rate (FP) at various threshold settings.

The area under the ROC curve, called AUC, is effective in summarizing the overall diagnostic accuracy of a model. One very important advantage of AUC is that it is immune to class imbalance, meaning that having more CVD patients than healthy in the dataset does not affect the AUC score. The trapezoidal rule is used to calculate AUC and its values range from 0 to 1, with 0 indicating a perfectly inaccurate test and 1 indicating a perfectly accurate test [49, 67, 68]. The higher the AUC, the better the model is at distinguishing between CVD patients and healthy.

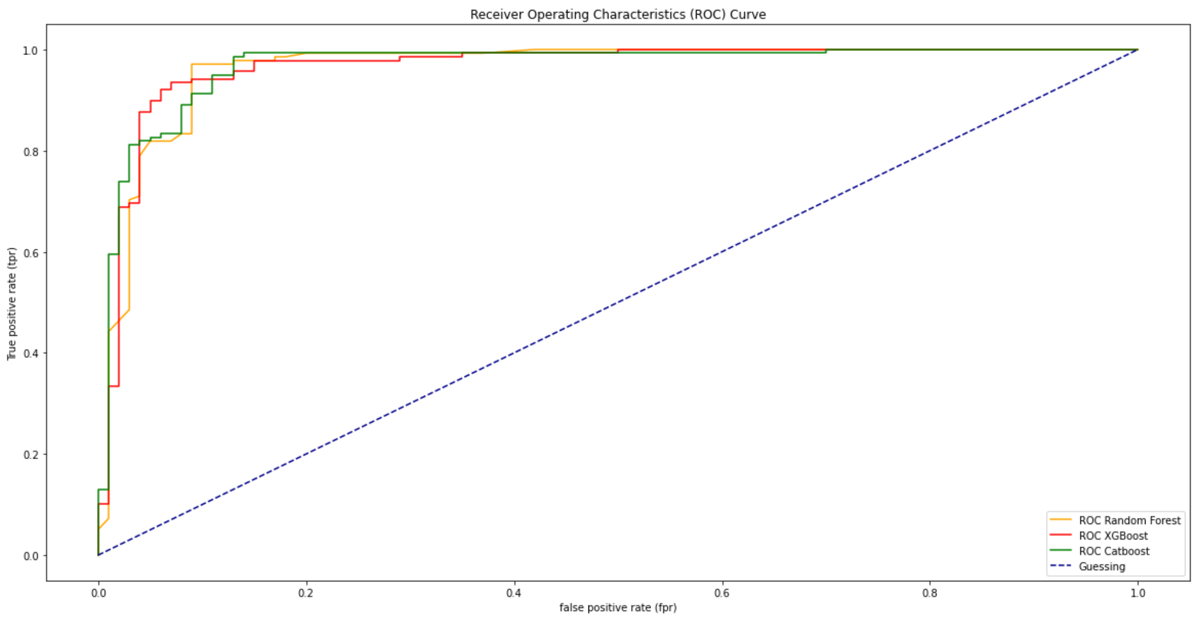


Figure 29: *ROC curve for the 3 models.*

In ROC the closer the curves are on the border, the better. Figure 29 shows the ROC curves of our 3 best models and as evident, all the models perform well. To gain a full understanding of their performance, their AUC must be calculated.

Model	AUC
Random Forest	96.26%

XGBoost	95.21%
Catboost	95.63%

Table 11: *AUC performance of the models.*

Any score of AUC that is above 90% is considered outstanding and therefore, all of our models' diagnostic abilities can be considered highly accurate. Random Forest has the highest AUC score by a slight difference from the other models.

6.4.2 Reliability in a medical scenario (*Appendix 6*)

In a medical scenario, recall (sensitivity) is the most important metric as it measures the percentage of all individuals who got a positive test result (TP) out of all individuals who actually have the disease (TP + FN). Therefore, for a medical ML model to be considered reliable, it must minimise its false negatives, i.e., when a patient with cardiovascular disease is not detected [50]. To calculate and visualise these misclassifications, confusion matrixes were used.

A confusion matrix categorizes the model's correct and incorrect classifications and assigns them to the appropriate label [50]. It gives us further insight into how well the model performs when predicting the two classes.

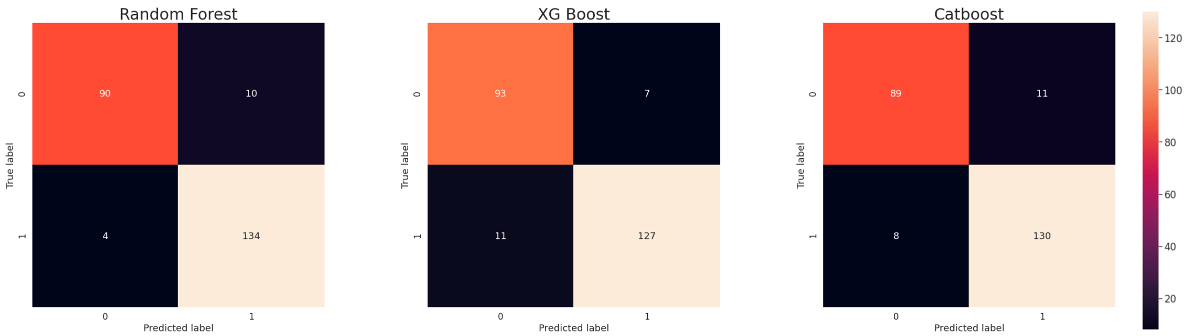


Figure 30: *Confusion matrix comparison between the three models.*

A perfect model has all of its predictions on the top left (true negative) and bottom right (true positive). False positives, however, are on the top right and false negatives - on the bottom left. Figure 30 shows that Random Forest performs best in minimising false-negative predictions as it only made 4 errors compared to 11 by XGBoost and 8 by Catboost. XGBoost on the other hand, made 7 errors compared to 10 by Random Forest and 11 by Catboost which makes it the best performing model at minimizing false positives. It can be concluded that Random Forest is the most reliable model as it performs exceptionally well in the metric that matters the most (recall).

6.4.3 Overall Performance

Figure 31 compares the three best performing models on a variety of classification metrics such as Accuracy, Precision, Recall (sensitivity), F1, and AUC score.

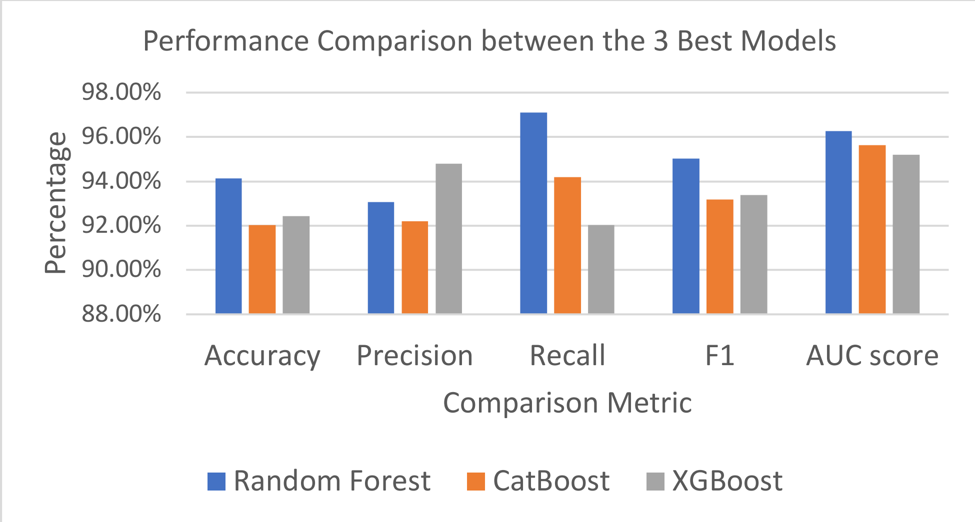


Figure 31: *Visualisation of how the algorithms performed on different metrics.*

The best performing model is Random Forest as all of its metrics, except for precision are better than the other models. Between XGBoost and CatBoost, the performance is very similar as XGBoost is better in three metrics and CatBoost in the other two. If one of them has to be chosen for second place, that would be Catboost as it has better recall which is crucial for healthcare ML as addressed in 6.4.2.

6.4.4 Summary

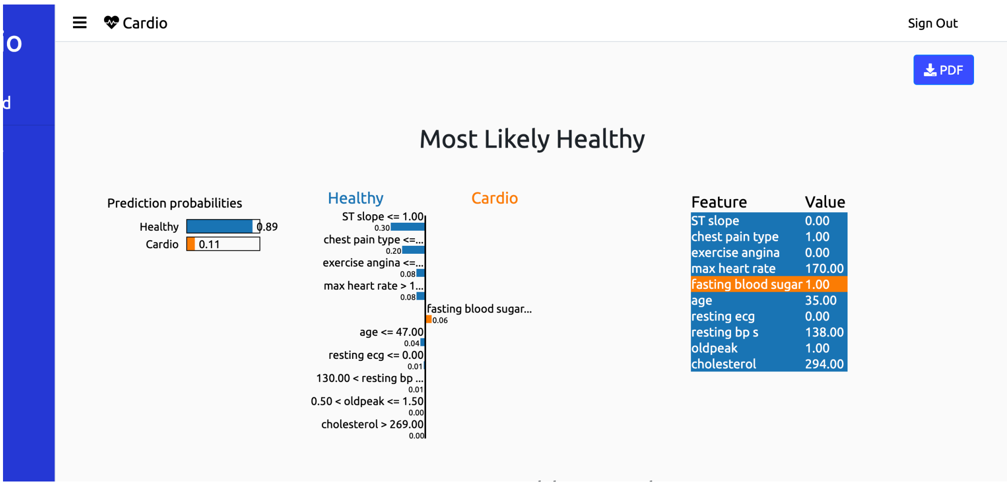
The findings of this section confirm that RF was correctly deployed on the platform as it has the best diagnostic accuracy, is the most reliable and performs the best overall.

6.5 Aim Evaluation

The main aim of this project was to create a web platform that allows cardiologists to 1) predict, 2) visualize, and 3) treat cardiovascular disease. This section evaluates the developed platform with its aim.

6.5.1 Predict

Once a doctor fills the patient's health data attributes in the diagnosis form, the data is passed on the ML model which returns a prediction for the patient's current condition.



Platform 10: *The diagnosis/prediction of the platform on the generated report.*

As shown from Platform 10, a prediction appears at the top of the page, along with a detailed insight on why this decision was taken (LIME). The platform can successfully and with a high percentage of accuracy predict a patient's condition.

6.5.2 Visualize

Upon diagnosis, fourteen fully interactive graphs are produced giving insight into the patient's condition. The decision on the type of graphs that are produced was derived from the first interview (see 3.1.2).

There are 3 main types of graphs generated:

1) **A radar graph -** Showing the variation between the **average** healthy and CVD values against patient's values.

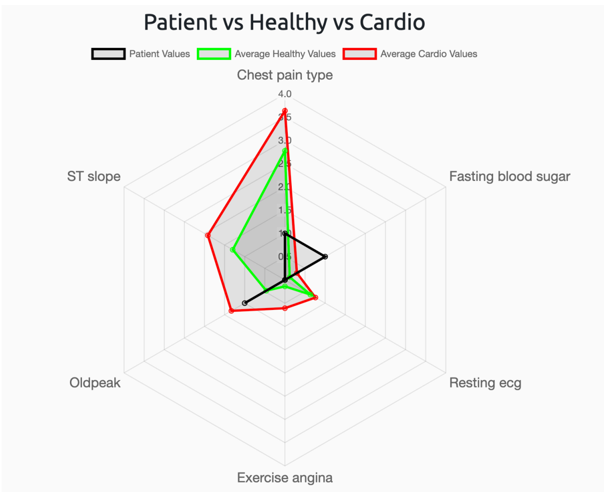


Figure 32: *One of two radar graphs used in the platform.*

Figure 32 allows the doctor to quickly determine whether one of the patient's values is closer to the CVD average than to the healthy average (or vice versa).

2) **Scatter Age graph** – Showing the distribution of healthy and CVD patients in a scatter chart as a function of age against various parameters

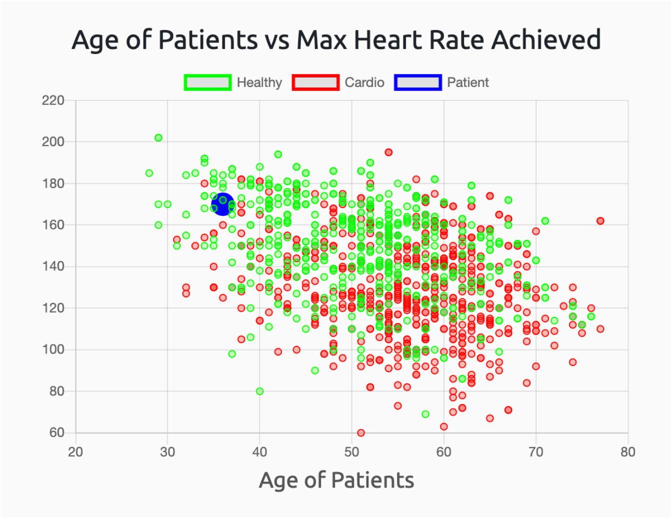


__Figure 33:__ Scatter diagram, age against max heart rate.

The doctor can see the distribution of patients and compare how healthy patients differ from CVD patients. They can also see where the patient stands in comparison with other patients.

For example, from Figure 33 three things are observable:
1) Older patients are more likely to have CVD
2) CVD patients have a lower maximum heart rate
3) The current patient is very likely to be healthy because a) they are young, b) have a high maximum heart rate, and c) they are surrounded by mostly healthy patients.

The same approach can also be used to draw conclusions from the other graphs of the same style.

3) **Doughnut –** Showing the number of patients who have a higher or lower value for a health feature than the current patient.

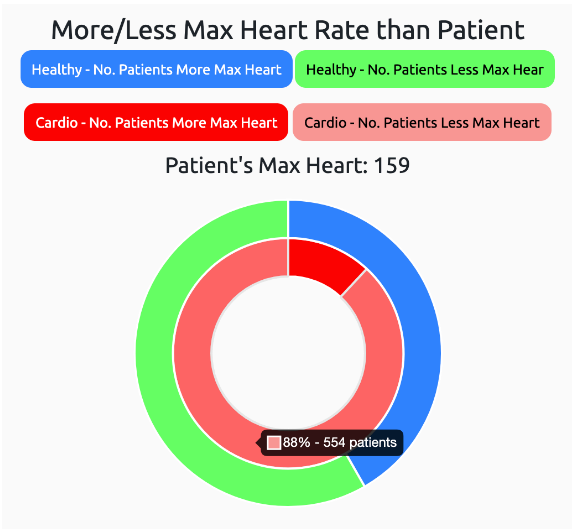


__Figure 34:__ Doughnut graph – Max Heart.

These graphs allow the doctor to get a statistical analysis of the number of healthy and CVD patients who have less and more of the current patient's feature value.

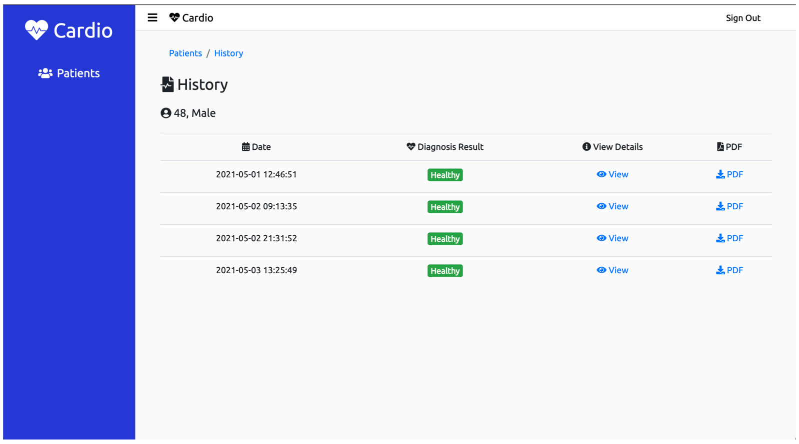
From Figure 34, a doctor can see that the vast majority (88%) of CVD patients have a lower Max Heart Rate than the current patient. They can also see that the current patient is almost in the center of the Healthy Distribution. This could lead them to conclude that the patient's heart rate is normal.

The graphs generated successfully meet the requirements of Dr V.

6.5.3 Treat

Upon diagnosis, the platform enables the download of a PDF file that contains all the generated graphs from the report page. The doctor can then send this PDF to the patient which might help them better understand the doctor's diagnosis as their condition is visualised.

A patient's diagnosis history is also stored on the platform (Platform 11) and therefore, the doctor can compare the patient's diagnoses (performed at different times) to see if their condition is improving or deteriorating.



Platform 11: *A patient's diagnosis history.*

Finally, the speed and accuracy of the platform allow rapid diagnosis for a large number of patients. We can imagine how useful this can be if a medical centre can run a test on all of their patients within minutes and then contact the patients who were predicted to have CVD (or whose prediction was unclear) for additional testing and further diagnosis by a cardiologist. This has the potential to save lives, which is the ultimate goal of this project.

6.5.4 Summary

This section showed that the project achieved its aim as a platform that can predict, visualise and treat CVD was developed.

6.6 Objectives Evaluation

This section evaluates the outcomes of this project against its seven initial objectives.

1. Gain better understanding of cardiovascular diseases, their causes and how they can be prevented.

Subsection 2.1.1 showed that CVDs are caused by several factors such as high blood pressure, high cholesterol, smoking, diabetes, and obesity. Numerous ways of preventing CVDs were also identified, e.g., regular exercise, a balanced diet, and quitting smoking.

The research also showed that CVDs kill thousands of people daily and there is a great disparity in cases between high-income and low/middle-income countries. This finding prompted me to ensure the platform runs at zero costs and can be offered free of charge to doctors/hospitals everywhere.

2. Research current challenges cardiologists face when diagnosing patients.

Two very important challenges were identified. In the background section, it was found that if a heart rhythm irregularity does not occur while conducting an ECG test, the disease will not be picked up, which can be a major setback in the accuracy of patients' diagnosis.

Another challenge that cardiologists face according to Dr V is the inappropriate delay of referrals from primary-care physicians (*Appendix 1*). This leads to the "deterioration of patient's health to a non-reversible state" and as such cardiologists receive these patients too late to impact their health outcome.

3. Experiment with at least 5 Machine Learning classification algorithms. Perform hyperparameter tuning on the 3 best models.

This objective was achieved as Random Forest, Catboost, XGBoost, KNN, and Logistic Regression were selected for experimentation which was discussed in 4.5.2. It was found that Catboost and XGBoost performed better than the other models and were therefore selected for hyperparameter tuning.

Hyperparameter tuning (discussed in 4.5.3.4) was successful except on the RF algorithm where worse results were produced.

4. Use visualizations to identify 3 health data attributes that correlate highly with CVDs

This objective was achieved in the data exploration part of the development, section 4.4 where a correlation matrix showed the 3 health data attributes from the dataset that correlate the most with CVDs are ST slope, Exercise angina and chest pain.

5. Deploy a reliable Machine Learning model with more than 90% accuracy and more than 85% recall (sensitivity).

Random Forest associations were evaluated in 5.1 where Dr V found the model logical. Subsection 6.4.2 showed that RF was the most reliable model as it performs best in minimising false-negative predictions and finally, 6.4.3 showed that RF performed the best on almost all metrics.

RF achieved a 94% accuracy and 97% recall which both exceeded this objective's target.

6. Research and critically evaluate at least 2 different visualization libraries. Select the best one. Visualize patient data.

A critical comparison between two visualization libraries, Chart.js and D3, was performed in 4.6.4.1. For this particular project and its requirements, the simple nature and features of Chart.js were better suited than D3.

As shown in previous sections, various charts are generated automatically according to the diagnosis of the patient.

7. Improve the platform based on feedback received from 2 cardiologists.

Due to the COVID-19 situation, it was challenging to organise an interview with a second cardiologist as medical facilities were overwhelmed. However, two interviews were conducted with Dr V which benefitted the project as there was a continuation for the vision of the platform.

This objective was not fully achieved as only one cardiologist was consulted. However, after a thorough review of the platform with Dr V, the project was successfully improved as explained in 6.1. Further improvements could have been made if more time was available.

6.6.1 Summary

Six of the seven objectives were successfully met, while the seventh was only partially met.

6.7 Similar study comparison

This section compares the performance of this study's ML model with that of [69], a relevant paper published recently (Feb. 2021).

Models	Accuracy	Precision	Sensitivity	Specificity	F-Measure
Naïve Bayes	75.8	90.5	79.8	60.0	84.5
Linear Model	85.1	88.8	94.9	20.0	91.6
Logistic Regression	82.9	89.6	91.1	25.0	90.2
Decision Tree	85	86	98.8	0.0	91.8
Random Forest	86.1	87.1	98.8	10.0	92.4
Support Vector Machines	86.1	86.1	100	0.0	92.5
HRFLM	88.4	90.1	92.8	82.6	90

[9]

Figure 35: *Performance metrics of the algorithms used by Kondababu et al.*

Figure 35 shows the results obtained in [69]. Their study used seven models, which is two more than this study used. Both studies use similar datasets but theirs contain two additional features (*ca* and *thal*). A graph comparing the best results for each metric between the studies is shown below.

[9] Taken from the paper: https://doi.org/10.1016/j.matpr.2021.01.475

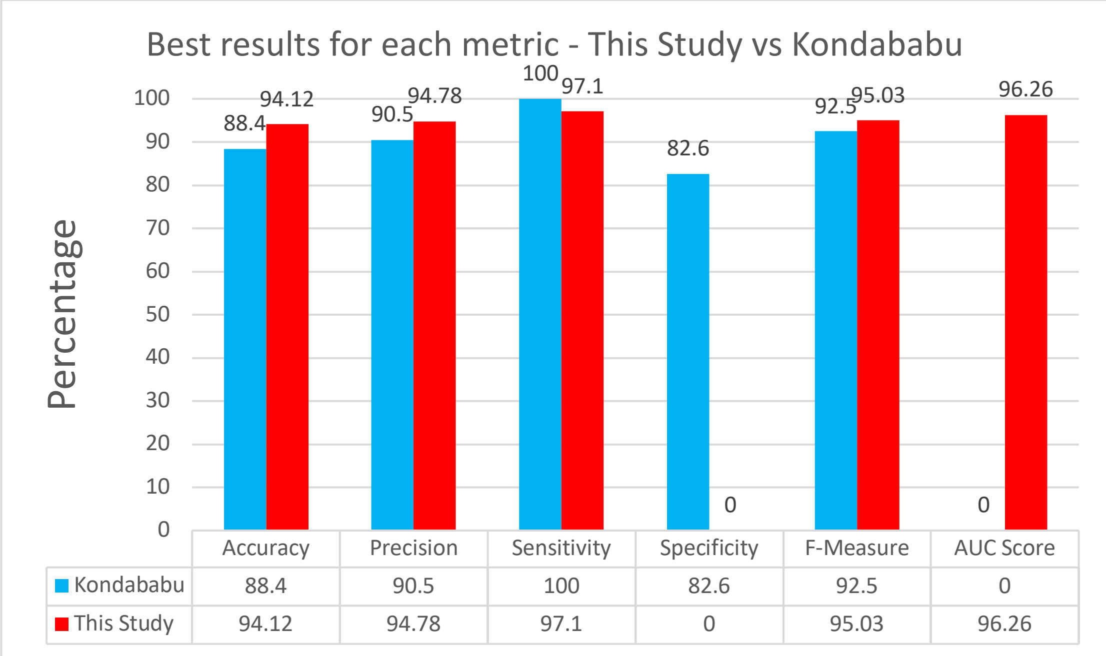


Figure 36: *Best result of each metric. This study vs [69].*

This study achieved the best accuracy as RF has an accuracy of 94.12% compared to 88.4% of HRFLM from [69]. Our XGBoost model also achieved the highest precision with 94.78% compared to 90.5%. Kondababu et al. achieved a sensitivity of 100% with their SVM model compared to 97.1% of our RF. The F-measure this study achieved was 2.53% higher than theirs. Finally, a comparison of specificity and AUC score is not possible because they were not measured in both studies.

Overall, better results were produced from this study. This is especially true when you compare the results of their best model (HRFLM) to our best model (RF).

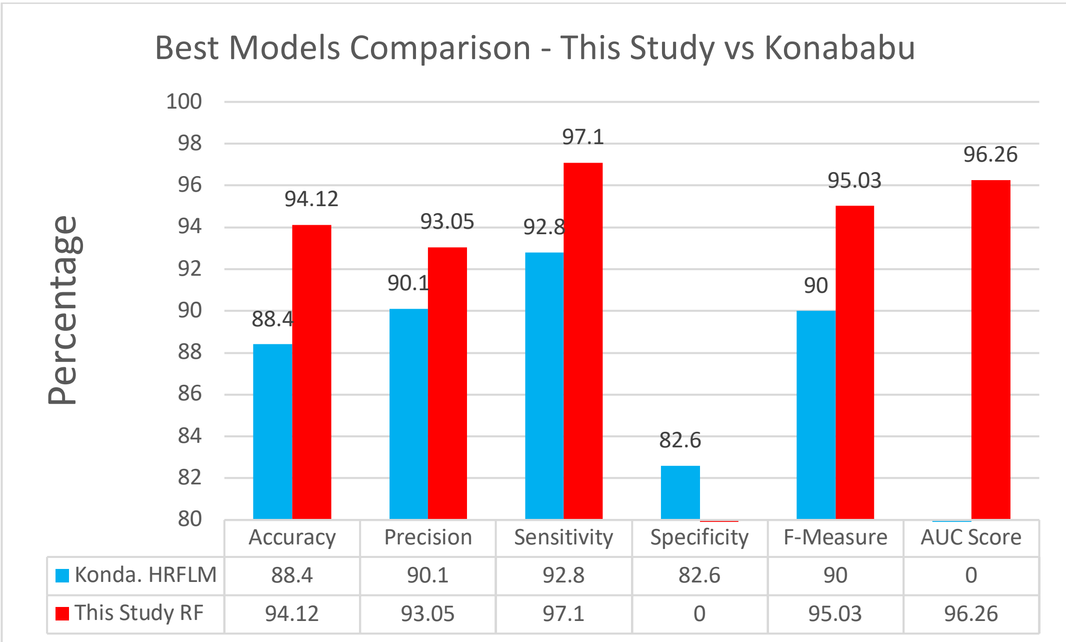


Figure 37: *Best model comparison between the two studies.*

All common metrics are higher for our best model than for theirs (Figure 37). Of course, we have to take into account the fact that the other study's dataset contained two additional features and that this might have had an impact on performance. Therefore, absolutely firm conclusions cannot be drawn, but it is safe to say that the accuracy metrics are excellent on both studies.

6.8 Similar Platform Comparison

This section compares the platform created with the one identified in 2.4, HeartScore.

This platform requires 11 medical features to perform a diagnosis, where HeartScore only requires 6 (see 2.4). This means that our model predicts using a more representative picture of the condition of the patient.

Upon diagnosis, HeartScore provides rich textual recommendations and guidelines, whilst mine provides a plethora of graphs that help the cardiologist draw conclusions on their own. It can be argued that HeartScore is more autonomous in its decision making and mine is more of a tool complimenting the doctor's thought process.

Both platforms can be used without login and both store patient diagnostic history in their full versions. HeartScore can produce progression graphs of patient data over time, something not yet supported on my platform. In contrast with HeartScore, my platform does not require a patient's name to perform a test, possibly making it more confidential. Additionally, both platforms provide a rationale for what factors lead to their decisions.

HeartScore is offered in a variety of languages and is a more comprehensive system than mine overall, having been developed over a long time by the European Association of Preventive Cardiology. The potential for further improvements in my platform is fully acknowledged and addressed in section 7.4.

6.9 Results and Evaluation Summary

This chapter began by evaluating the software engineering process followed by the confidentiality of the model. Dr V's suggestions for improvements and how they were handled were discussed, and RF emerged as the best model in all comparisons in 6.4. Moreover, the achievement of the project’s aim and objectives was discussed followed by comparisons with a similar study and a platform. The next chapter reflects on the whole experience of the project and discusses further possible improvements.

7. Conclusion

This project began by identifying the epidemic behaviour of CVDs and set as its purpose to develop a medical web platform that allows cardiologists gain better insights into a diagnosis through visualisations and ML fast and free. This l was achieved as a fully functional web platform that can predict, visualise and treat CVDs was successfully developed. The evaluation conducted showed that the platform has potential, but to fully understand its true usefulness, it needs to be tested in a medical facility over an extended period.

Most of the initial objectives were met, except objective 7), as only one cardiologist was interviewed instead of two. Moreover, the requirements set by Dr V were all successfully met.

7.1 What I learned

Before this project, I had some basic knowledge about CVDs as unfortunately, I lost my grandfather due to a heart attack some years ago. By undertaking this project, I learned more about CVDs, their causes, preventions, and current challenges that doctors face. The thing that surprised me the most was the significant difference in CVD cases between low- and middle-income countries.

My knowledge of Machine Learning was limited before this project as despite having completed an online course over the summer, I had never done a major project to put my skills to practise. Although there is still much to learn, I believe I have now gained a solid understanding of various ML concepts, algorithms, and techniques. My web development, data science, and programming skills have been greatly improved whilst I also learned Flask which is widely used in the industry, Flask.

One of the most important lessons I have learned is how to conduct effective research under the guidance of a supervisor. I believe this is something that will help me for my further studies and open up new opportunities. Finally, I appreciated the impact that planning and consistent daily work can have on the result of a project as without planning I wouldn't have completed it on time.

7.2 What could have been done better

The development of this project was not flawless, and certain aspects could have been done differently. One of them was to show Dr V the dataset before the final review of the system. That way, he could alert me early on about the absence of important features (e.g., smoker feature), and thus I could have merged it with other datasets before it was too late.

If I had the chance to go back, I would also integrate ReactJS with Flask as this could have made my platform more sustainable and future-focused.

The issue of confidentiality was also addressed very late, which meant that I had to make significant changes near the end of the deadline. Even though I believe that the changes significantly improved the project, it would have been better if I had considered them from the start as I had to implement some things twice.

7.3 Setbacks

The first setback that occurred was the change of my supervisor in the middle of the academic year. This led to one month of uncertainty where I didn't know who my next supervisor would be and was therefore not progressing the project as much as it was probably needed to.

Another major setback that cost me time was the strict lockdown imposed by the UK government which led me to return to my home country where I had to quarantine for 10 days in complete isolation. Despite having reasonably reliable Wi-Fi, my top priority was my mental health, as being alone for so long had taken its toll on me.

7.4 Future improvements

A desktop version of the platform that can run without internet access could be a potential improvement that aligns with my overall vision of allowing cardiologists, especially in developing countries to benefit from using the platform. Furthermore, I aim to include more highly correlated features in the dataset such as smoking and weight. The generation of progression graphs of patient's data could also be very beneficial as the cardiologist will have the ability to visualize the change in a patient's condition over time.

It is necessary to consider system maintenance and error-fixing while the system is deployed. To accomplish this, a new GitHub branch has to be created that will be used to fix the errors; once the errors have been fixed, the new branch will be merged with the main branch. That way, I'll be able to ensure that the problem has been resolved and the regression tests succeed before disrupting the main program.

A final future enhancement would be the development and integration of a Convolutional Neural Network model that will be able to diagnose computed tomography (CT) scans which provide clear images of a patient's arteries, aorta, heart, and valves [70]. The most challenging aspect of developing this would be to learn how to use Deep Learning techniques as it is something I never practised before.

I intend to explore these enhancements on my own initiative soon, as I love this project and I truly enjoyed the learning odyssey it has taken me.

References:

[1] Kaur, M., 2018. *SQL vs NoSQL : MySQL vs MongoDB — The Difference*. [online] Medium. Available at: https://medium.com/@mandeepkaur1/sql-vs-nosql-mysql-vs-mongodb-the-difference-6145e437cd40

[2] Wiki.python.org. n.d. *WebFrameworks - Python Wiki*. [online] Available at: https://wiki.python.org/moin/WebFrameworks

[3] Semik, W. and Stempniak, A., n.d. *Flask vs. Django: Which Python Framework Is Better for Your Web Development?*. [online] Stxnext.com. Available at: https://www.stxnext.com/blog/flask-vs-django-comparison/

[4] Relan, K., 2019. *Building REST APIs with Flask*. Berkeley, CA: Apress.

[5] Educative: Interactive Courses for Software Developers. n.d. *What is Firebase?*. [online] Available at: https://www.educative.io/edpresso/what-is-firebase

[6] Kody Techno Lab. 2020. *Why and When Should You Opt For Firebase? Features, Advantages, and Disadvantages Discussed*. [online] Available at: https://kodytechnolab.com/features-benefits-firebase

[7] Firebase. n.d. *Choose a database: Cloud Firestore or Realtime Database | Firebase*. [online] Available at: https://firebase.google.com/docs/firestore/rtdb-vs-firestore

[8] Breiman, L., 2001. *Random Forests. Machine Learning* 45, 5–32. https://doi.org/10.1023/A:1010933404324

[9] Koehrsen, W., 2018. *An Implementation and Explanation of the Random Forest in Python*. [online] Medium. Available at: https://towardsdatascience.com/an-implementation-and-explanation-of-the-random-forest-in-python-77bf308a9b76

[10] Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., & Gulin, A. (2017). CatBoost: unbiased boosting with categorical features. Available at: https://papers.nips.cc/paper/2018/hash/14491b756b3a51daac41c24863285549-Abstract.html

[11] Pallapothu, H., 2019. *What's so special about CatBoost?*. [online] Medium. Available at: https://hanishrohit.medium.com/whats-so-special-about-catboost-335d64d754ae

[12] Kambria. 2021. Logistic Regression For Machine Learning and Classification - Kambria. [online] Available at: https://kambria.io/blog/logistic-regression-for-machine-learning/

[13] Brownlee, J., 2016. Logistic Regression for Machine Learning. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/logistic-regression-for-machine-learning/

[14] Kramer O., 2013. K-Nearest Neighbors. In: Dimensionality Reduction with Unsupervised Nearest Neighbors. Intelligent Systems Reference Library, vol 51. Springer, Berlin, Heidelberg. Available at: https://doi.org/10.1007/978-3-642-38652-7_2

[15] Harrison, O., 2018. *Machine Learning Basics with the K-Nearest Neighbors Algorithm*. [online] Medium. Available at: https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761/

[16] www.javatpoint.com. n.d. *K-Nearest Neighbor(KNN) Algorithm for Machine Learning - Javatpoint*. [online] Available at: https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning

[17] Who.int. 2020. *Cardiovascular Diseases*. [online] Available at: https://www.who.int/health-topics/cardiovascular-diseases/#tab=tab_1

[18] nhs.uk. 2021. *Cardiovascular Disease*. [online] Available at: https://www.nhs.uk/conditions/cardiovascular-disease

[19] Fogoros, R., 2021.*What Is An EKG?*. [online] Verywell Health. Available at: https://www.verywellhealth.com/the-electrocardiogram-ecg-1745304

[20] M'Haimdat, O., 2020. *Understand the Fundamentals of the K-Nearest Neighbors (KNN) Algorithm*. [online] Medium. Available at: https://heartbeat.fritz.ai/understand-the-fundamentals-of-the-k-nearest-neighbors-knn-algorithm-533dc0c2f45a

[21] Xgboost.readthedocs.io. n.d. *XGBoost Parameters — xgboost 1.5.0-SNAPSHOT documentation*. [online] Available at: https://xgboost.readthedocs.io/en/latest/parameter.html

[22] Scikit-learn.org. n.d. *sklearn.ensemble.RandomForestClassifier — scikit-learn 0.24.2 documentation*. [online] Available at: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[23] Shalev-Shwartz, S., & Ben-David, S., 2014. Introduction. In *Understanding Machine Learning: From Theory to Algorithms* (pp. 1-10). Cambridge: Cambridge University Press. Available at: doi:10.1017/CBO9781107298019.002

[24] Soni, D., 2018. *Supervised vs. Unsupervised Learning*. [online] Medium. Available at: https://towardsdatascience.com/supervised-vs-unsupervised-learning-14f68e32ea8d

[25] Google Developers. n.d. *Machine Learning Glossary | Google Developers*. [online] Available at: https://developers.google.com/machine-learning/glossary

[26] Potier, G. and Lafortune, M., 2017. *Parsley - The ultimate JavaScript form validation library*. [online] Parsleyjs.org. Available at: https://parsleyjs.org/

[27] En.wikipedia.org. n.d. *Bias–variance tradeoff - Wikipedia*. [online] Available at: https://en.wikipedia.org/wiki/Bias%E2%80%93variance_tradeoff

[28] DataRobot. n.d. *What are Feature Variables in Machine Learning | DataRobot AI Wiki*. [online] Available at: https://www.datarobot.com/wiki/feature/

[29] Catboost.ai. n.d. *Parameter tuning - CatBoost. Documentation*. [online] Available at: https://catboost.ai/docs/concepts/parameter-tuning.html

[30] Majorek, J., 2020. *14 JavaScript Data Visualization Libraries in 2021*. [online] Monterail.com. Available at: https://www.monterail.com/blog/javascript-libraries-data-visualization

[31] Heartscore.org. n.d. *Access HeartScore - full version*. [online] Available at: https://www.heartscore.org/en_GB/access

[32] Heartscore.escardio.org. n.d. *Heart Score*. [online] Available at: https://heartscore.escardio.org/2016/quickcalculator.aspx?model=EuropeLow

[33] Guides.github.com. n.d. *Hello World · GitHub Guides*. [online] Available at: https://guides.github.com/activities/hello-world/

[34] Andrade J.P., Andrade M.D., Mattos L.A. (2015) Cardiovascular Disease Worldwide: A Global Challenge. In: Andrade J., Pinto F., Arnett D. (eds) Prevention of Cardiovascular Diseases. Springer, Cham. Available at: https://doi.org/10.1007/978-3-319-22357-5_2

[35] Brownlee, J., 2020. *4 Distance Measures for Machine Learning*. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/distance-measures-for-machine-learning/

[36] ES, S., 2021. *Hyperparameter Tuning in Python: a Complete Guide 2021 - neptune.ai*. [online] neptune.ai. Available at: https://neptune.ai/blog/hyperparameter-tuning-in-python-a-complete-guide-2020

[37] Devcenter.heroku.com. n.d. *How Heroku Works | Heroku Dev Center*. [online] Available at: https://devcenter.heroku.com/articles/how-heroku-works

[38] Amazon Web Services, Inc. n.d. *Build Mobile & Web Apps Fast | AWS Amplify | Amazon Web Services*. [online] Available at: https://aws.amazon.com/amplify/

[39] Sitanggang, H., 2020. *AWS Amplify VS Google Firebase, Which is Better? | Mitrais Blog*. [online] Mitrais.com. Available at: https://www.mitrais.com/news-updates/aws-amplify-vs-google-firebase-which-is-better/

[40] Dribbble.com. n.d. *Dribbble - Discover the World's Top Designers & Creative Professionals*. [online] Available at: https://dribbble.com/

[41] Gupta, P., 2017. *Cross-Validation in Machine Learning*. [online] Medium. Available at: https://towardsdatascience.com/cross-validation-in-machine-learning-72924a69872f

[42] Miranda, S., 2019. *Ensemble Methods: Tuning a XGBoost model with Scikit-Learn*. [online] Medium. Available at: https://medium.com/@juniormiranda_23768/ensemble-methods-tuning-a-xgboost-model-with-scikit-learn-54ff669f988a

[43] Brownlee, J., 2019. *Overfitting and Underfitting With Machine Learning Algorithms*. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/

[44] Morde, V., 2019. *XGBoost Algorithm: Long May She Reign!*. [online] Medium. Available at: https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d

[45] Chen, T. , Guestrin, C, 2016. XGBoost: A Scalable Tree Boosting System. Available at: https://doi.org/10.1145/2939672.2939785

[46] Worcester, P., 2019. *A Comparison of Grid Search and Randomized Search Using Scikit Learn*. [online] Medium. Available at: https://blog.usejournal.com/a-comparison-of-grid-search-and-randomized-search-using-scikit-learn-29823179bc85

[47] K2 Analytics. 2020. *Classification Accuracy & AUC ROC Curve | K2 Analytics*. [online] Available at: https://www.k2analytics.co.in/classification-accuracy-auc-roc-curve/

[48] Brownlee, J., 2014. *Classification Accuracy is Not Enough: More Performance Measures You Can Use*. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/classification-accuracy-is-not-enough-more-performance-measures-you-can-use/

[49] Ling C.X., Huang J., Zhang H. (2003) *AUC: A Better Measure than Accuracy in Comparing Learning Algorithms*. Advances in Artificial Intelligence. Canadian AI 2003. Available at: https://doi-org.libproxy.ncl.ac.uk/10.1007/3-540-44886-1_25

[50] Panesar, Arjun. *Machine Learning and AI for Healthcare*. 1st ed. Berkeley, CA: Apress L. P, 2019. Available at: https://doi-org.libproxy.ncl.ac.uk/10.1007/978-1-4842-3799-1

[51] professional.heart.org. 2020. *Heart Disease and Stroke Statistics - 2020 Update*. [online] Available at: https://professional.heart.org/en/science-news/heart-disease-and-stroke-statistics-2020-update

[52] Pythonist, 2020. *A brilliant introduction to Flask*. [video] Available at: https://www.youtube.com/watch?v=F7AK-WzpYdY&t=185s&ab_channel=Pythonist

[53] Flask.palletsprojects.com. n.d. *Template Inheritance — Flask Documentation (1.1.x)*. [online] Available at: https://flask.palletsprojects.com/en/1.1.x/patterns/templateinheritance/

[54] Cook, P., 2019. *D3 or Chart.js for Data Visualisation?*. [online] Createwithdata.com. Available at: https://www.createwithdata.com/d3js-or-chartjs/

[55] GeeksforGeeks. 2018. *Understanding Python Pickling with example - GeeksforGeeks*. [online] Available at: https://www.geeksforgeeks.org/understanding-python-pickling-example/

[56] Pettis, B., 2020. [online] Rokkincat.com. Available at: https://rokkincat.com/blog/2020-2-14-d3-vs-chartjs/

[57] Poduska, J., 2018. *SHAP and LIME Python Libraries: Part 1 – Great Explainers, with Pros and Cons to Both*. [online] Data Science Blog by Domino. Available at: https://blog.dominodatalab.com/shap-lime-python-libraries-part-1-great-explainers-pros-cons/

[58] Sagar, R., Sagar, R. and Sagar, R., 2019. *8 Explainable AI Frameworks Driving A New Paradigm For Transparency*. [online] Analytics India Magazine. Available at: https://analyticsindiamag.com/8-explainable-ai-frameworks-driving-a-new-paradigm-for-transparency-in-ai/

[59] Techterms.com. n.d. User Interface Definition. [online] Available at: https://techterms.com/definition/user_interface

[60] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. Available at: https://doi.org/10.1145/2939672.2939778

[61] Htoon, K., 2020. *A Guide To KNN Imputation*. [online] Medium. Available at: https://medium.com/@kyawsawhtoon/a-guide-to-knn-imputation-95e2dc496e

[62] Berdikulov, D., 2019. *Dealing with Missing Data*. [online] Medium. Available at: https://medium.com/@danberdov/dealing-with-missing-data-8b71cd819501

[63] Educative: Interactive Courses for Software Developers. n.d. *What is the F1-score?*. [online] Available at: https://www.educative.io/edpresso/what-is-the-f1-score

[64] En.wikipedia.org. n.d. *Training, validation, and test sets - Wikipedia*. [online] Available at: https://en.wikipedia.org/wiki/Training,_validation,_and_test_sets

[65] Agilemanifesto.org. 2001. *Manifesto for Agile Software Development*. [online] Available at: https://agilemanifesto.org/

[66] Saxena, S., 2020. *Random Forest Hyperparameter Tuning in Python | Machine learning*. [online] Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2020/03/beginners-guide-random-forest-hyperparameter-tuning/

[67] Jayawant N. Mandrekar, 2010. *Receiver Operating Characteristic Curve in Diagnostic Test Assessment.* Available at: https://doi.org/10.1097/JTO.0b013e3181ec173d

[68] Andrew P. Bradley, 1997. *The use of the area under the ROC curve in the evaluation of machine.* Available at: https://doi.org/10.1016/S0031-3203(96)00142-2

[69] Kondababu, A, Siddhartha, V, Kumar, BHK. Bhagath, and Penumutchi, Bujjibabu 2021. *A Comparative Study on Machine Learning Based Heart Disease Prediction.* Materials Today : Proceedings 2021-02. Available at: https://doi.org/10.1016/j.matpr.2021.01.475

[70] Preventative Diagnostic Center. n.d. *What Can a Heart CT Scan Detect? | Preventative Diagnostic Center*. [online] Available at: https://www.pdcenterlv.com/blog/what-can-a-heart-ct-scan-detect/

[71] En.wikipedia.org. n.d. *Receiver operating characteristic - Wikipedia*. [online] Available at: https://en.wikipedia.org/wiki/Receiver_operating_characteristic

[72] Smith, C., 2020. *The Importance of a Custom 404 Page | HIPBlog*. [online] HIPB2B. Available at: https://www.hipb2b.com/blog/the-importance-of-a-custom-404-page

[73] Sharma, A., 2020. *Unit Testing in Python*. [online] Datacamp. Available at: https://www.datacamp.com/community/tutorials/unit-testing-python

[74] En.wikipedia.org. n.d. *Imputation (statistics) - Wikipedia*. [online] Available at: https://en.wikipedia.org/wiki/Imputation_(statistics)

[75] Shiff, L. and Rowe, W., 2018. *SQL vs NoSQL Databases: What's The Difference?*. [online] BMC Blogs. Available at: https://www.bmc.com/blogs/sql-vs-nosql/

[76] Wang, R. and Yang, Z., n.d. *SQL vs NoSQL: A Performance Comparison*. [ebook] Available at: https://www.cs.rochester.edu/courses/261/fall2017/termpaper/submissions/06/Paper.pdf

[77] Expert.ai. 2020. *What is Machine Learning? A definition - Expert System*. [online] Available at: https://www.expert.ai/blog/machine-learning-definition/

[78] Siddhartha, M., 2020. *Heart Disease Dataset (Comprehensive)*. [online] Kaggle.com. Available at: https://www.kaggle.com/sid321axn/heart-statlog-cleveland-hungary-final

[79] Ronacher, A., 2013. *Foreword — Flask Documentation (0.10)*. [online] Web.archive.org. Available at: https://web.archive.org/web/20171117015927/http://flask.pocoo.org/docs/0.10/foreword

[80] Buckler, C., 2018. *Getting Started with Sentry.io Error Tracking - SitePoint*. [online] Sitepoint.com. Available at: https://www.sitepoint.com/getting-started-with-sentry-io-error-tracking/

[81] Stepnov, E., 2021. *10 Best Error Monitoring and Error Tracking Tools - Flatlogic Blog*. [online] Flatlogic Blog. Available at: https://flatlogic.com/blog/10-best-error-monitoring-and-error-tracking-tools/

[82] Flask.palletsprojects.com. n.d. *Custom Error Pages — Flask Documentation (1.1.x)*. [online] Available at: https://flask.palletsprojects.com/en/1.1.x/patterns/errorpages/

[83] Precursorsecurity.com. n.d. *Vulnerability Remediation – Do not forget Regression Testing – Precursor Security*. [online] Available at: https://www.precursorsecurity.com/vulnerability-remediation-do-not-forget-regression-testing/

[84] Contacttracing.ashm.org.au. n.d. *Why are Privacy and Confidentiality Important? - Contact Tracing Guidelines*. [online] Available at: http://contacttracing.ashm.org.au/why-are-privacy-and-confidentiality-important

[85] Hayes, A., 2020. *Understanding the Harmonic Mean*. [online] Investopedia. Available at: https://www.investopedia.com/terms/h/harmonicaverage.asp

Appendix

1) First Interview with Dr V – Conducted by a telephone call on 04/02/2021

R: Thank you for agreeing to do this interview. I have some questions that I would like to ask you as I am slowly starting to develop the medical platform. As I told you via text a few days ago, the aim of the web platform is to enable cardiologists to predict Cardiovascular Diseases. The development of the platform is still at early stages. I am currently working on the ML model.

V: Yeah, you can ask me whatever questions you have.

R: What are some current challenges that cardiologists face?

V: I believe one significant challenge is the inappropriate delay of referrals from primary-care physicians (GPs), at least in Cyprus. This often leads to the deterioration of patient's health to a non-reversible state and as such we receive these patients too late to impact their health outcome.

R: Ok I understand. What do you believe about Healthcare AI?

V: Hmm, I believe it can have a really positive impact, but it has to be done right. Such a system is only useful if it can give useful insights that would otherwise be hard to obtain.

R: Would you be ever willing to try one then?

V: I am open to the idea, but I am very demanding. I will only use one if it undoubtedly improves my work.

R: Ok, I understand. Do you use any systems to help you make a diagnose?

V: No, I do not use any systems or tools. I just use my knowledge and experience.

R: I sent you the other day the first prototypes of the platform. What are your thoughts about them?

V: Yeah, I saw them. I like that they are simple, I do not like overcomplicated and congested systems especially when I am going to use them for work. However, the prototypes you sent me are a bit monotonous, they would be better if they had colour in them. Also, from what I saw you have to go to another page to add a patient, I would prefer if I can add a patient using a simple modal in the same page. That way I will not have to go back and forth constantly.

R: That's very useful feedback. What do you think about automatically generating a report containing graphs after a diagnosis which will visualise the patient's condition?

V: That's a very good idea. I think it can be extremely helpful, not only for us but also for the patient. If done right, it can help me explain to them where they stand and what we can improve. If you can, do that. Also, try to make it relatively fast as I don't want to wait minutes before I have a report I can see or show to a patient.

R: I will definitely try. What are some graphs/data visualisation features you would like to see in the platform?

V: OK first of all I want to identify why the ML model made that prediction. Tell me exactly which are the factors that affected its decision. I cannot trust a black box - something that I do not know how it derives a diagnosis. In terms of graphs, I would like to display all of the patients both healthy and CVD in a scatter graph for various health attributes and have the ability to compare the current patient with them. I would also like to have a statistical comparison between patients who have a higher or lower value of a variable than the current patient. It can be very useful if you can add some interactivity on the graphs.

R: I am researching graphical libraries now, I believe I can make what you asked, very useful suggestions, thank you. You said to have a scatter graph, I understand that the one axis will be a health data attribute, what about the other axis?

V: It has to be numerical, so I would make it the age variable. Each patient has an age, and it can also help us show relationships with age.

R: OK, thank you very much for your time. I will try to implement these and get back to you. I was wondering if it would be possible to arrange a physical meeting after I finish the implementation of the final system? To review it and give me feedback on how I can improve it?

V: Yeah, I think we can arrange it. I am very busy but if you tell me a few days in advance then I will try to arrange it.

R: OK thank you very much, I appreciate it a lot.

V: You are welcome. Good luck.

2) Second Interview with Dr V – Physical Meeting on 21/03/2021

R: Thank you for agreeing to meet in person and for inviting me to your house. Today I will show you the web platform I developed, and I would like to get your constructive feedback on how we can improve this platform.

Dr V: You are welcome. I am excited to see it.

R: I want to start by showing you the dataset. By taking a look at this, do you understand all of its features?

Dr V: Of course. It has some very important features that indeed affect CVDs. However, I notice that it is missing some features such as smoking. You can't have a cardiological platform without having 'smoking' as a feature. It is extremely correlated.

R: I understand what you are saying. Unfortunately, the chosen dataset doesn't include the smoking feature. I can reassure you that this dataset was the best out of all the available ones online. The other ones contained a lot of missing data and were predicting death instead of disease. This dataset also contains data from five different institutions so it's more generalisable.

Dr V: The chosen dataset is acceptable but in a real-world scenario you would want to get your dataset from a medical facility.

R: Yeah, I agree with this. Should I show you the web platform?

Dr V: Yes, let's see it.

Showing Dr V the medical platform pages (Except report)…

Dr V: I believe this is excellent work in terms of usability and interface. It looks much better than the prototypes you showed me initially. Also, the sign up process is very easy and gives a lot of different options which is nice.

R: Thank you. Yeah, I listened to your feedback and requirements from the first interview and adjusted its style accordingly.

Dr V: That is nice to hear, well done.

R: I want you to please give me constructive feedback on two pages specifically. The "diagnosis" page and then the "report" page.

Showing diagnosis page and its input in detail…

R: Do you think the input form is easy to fill and understand?

Dr V: I think it's easy to fill. However, to me it's obvious that this platform was made from someone without medical expertise, if you want to build a web platform for doctors, you have to speak our language**.**

R: How can it be improved?

Dr V: OK. First of all you write Cholesterol, which Cholesterol is it? There are various types of Cholesterol. Can you please show me the dataset again?

R: I am not sure which one it is. Yes.

Dr V: From looking at this, I believe it's the Total Cholesterol. You should change this. Another small mistake can be seen on the Resting Blood Pressure instructions. You have the unit of measurement as mm/HG when it should be mm/Hg. I also see another mistake at the Fasting Blood Sugar instructions. It says, "Please select the level of fasting blood sugar of the patient". This is wrong, it's not blood sugar of a person who is fasting, for religious reasons for example, it's the blood sugar in a fasting state, which is when a person doesn't eat or drink for 8 hours. These mistakes add up and make all the difference in the professionalism of the platform.

R: Thank you very much this is extremely useful feedback. I will change everything. Do you see any other mistakes?

Dr V: Hmm just one more. On ST slope you have Flat. Even though I understand what it means, the common way to describe this is Horizontal, this is what a cardiologist would say.

R: Ok thank you. Now let's fill the form and click on "Diagnose" for the system to make a prediction and generate the report. Can you please fill the form with whatever values you want?

Dr V: Yes, let's see what it will predict.

Dr V filling in the form and pressing the "Diagnose" button. The graphical report is generated…

R: Here it shows it's prediction and the factors that affected it's prediction. Does this give you an understanding of why the project made this prediction like you asked me on the first interview?

Dr V: This is useful. I want to evaluate it first. I will evaluate each reasoning and tell you whether I find it accurate.

Dr V evaluating the predictions…

Dr V: It's overall predictions are correct. The only mistake in terms of how it affects the diagnosis is Chest Pain Type as "Asymptomatic" is not a symptom of CVD patients and here it predicts it as CVD. I find that the reasoning it gives, for example above 60 years old then CVD, is very specific and it feels like its binary.

R: Thank you. These are the associations the model made based on the hundreds of examples it saw when it was trained. The model makes its predictions using a combination of features and as it shows the testing I performed showed, most times it's correct.

Dr V: Yeah I understand that. In reality however it's a bit more complicated.

R: Yeah of course, this tool is to be used by a doctor as a complementary tool. It can never replace a doctor of course.

Dr V: Yeah exactly. Overall, you did a good work with ML model especially considering that you didn't have an incredibly detailed dataset.

R: Thank you very much. Now can you take a look at the graphs generated? How do you find them?

Dr V looking at the graphs…

Dr V: They make sense to me. It's amazing to be able to place and compare the current patient between healthy and CVD patients. I believe you did all the graphs that I asked you during our phone call.

R: Yeah I tried to do them all. Do you think the report page can be improved in any way?

Dr V: Yes. Currently you do not give us the ability to add any comments, please do that. It will allow us to write our comments/diagnosis as soon as we see a correlation/something interesting from the graphs.

R: Thank you very much for the feedback. I will implement that as well. This is all I wanted to show you.

Dr V: Very good, I believe the platform has potential.

R: Would you use it in your work?

Dr V: Even though I think it's really useful, especially in enabling the patient to gain insight to the diagnosis, I believe I would not currently use it. If you make the changes we said, include more features in your dataset and add some extra functionalities such as diagnosis of CT scans, then I would definitely try it. Did you ever think about implementing CT scans diagnose?

R: Ok I understand. No, I never thought about it. I would need to create a deep learning model to achieve that but I think it's a really good idea!

Dr V: I don't know what technical stuff you have to do to implement it, but think about it, it will make the platform even more useful.

R: Thank you very much. I think these are all the questions I had. I am very grateful for your help.

Dr V: I am happy I helped.

3) Unit Testing

test_load.py

```python
from app import app, sign_in_test
import unittest

# Patient id used for testing purposes.
pid = "-MZxdolOqXRO79q_7VwB"
userID = sign_in_test("demo-cardio21@gmail.com", "Cardiodemo100!")

class FlaskTest(unittest.TestCase):
    """
    Test that the pages of the app load without errors.
    """

    # Set up config.
    def setUp(self):
        # Initialize test client.
        app.secret_key = "OMONOIALAOSPROTATHLIMA"
        self.app = app.test_client()


    # Check that "Index" page loads.
    def test_index(self):
        with app.test_client() as c:
            with c.session_transaction() as sess:
                sess['usr'] = userID

            response = c.get('/', follow_redirects=True)
            self.assertTrue(response.status_code, 200)

    # Check that "Diagnose" page loads.
    def test_diagnose(self):
        with app.test_client() as c:
            with c.session_transaction() as sess:
                sess['usr'] = userID

            response = c.get('/diagnose', follow_redirects=True)
            self.assertTrue(response.status_code, 200)

    # Check that "Edit" page loads.
    def test_edit(self):
        with app.test_client() as c:
            with c.session_transaction() as sess:
                sess['usr'] = userID

            response = c.get("/edit?pid="+pid)
            self.assertEqual(response.status_code, 200)

    # Check that "History" page loads.
```

```python
    def test_history(self):
        with app.test_client() as c:
            with c.session_transaction() as sess:
                sess['usr'] = userID

            response = c.get("/history?pid="+pid)
            self.assertEqual(response.status_code, 200)

    # Check that "Patients" page loads.
    def test_patients(self):
        with app.test_client() as c:
            with c.session_transaction() as sess:
                sess['usr'] = userID

            response = c.get("/patients/")
            self.assertEqual(response.status_code, 200)


if __name__ == "__main__":
    unittest.main()
```

test_forms.py

```python
from app import app, sign_in_test
import os
import unittest

# Patient id used for testing purposes.
pid = "-MZxdolOqXRO79q_7VwB"
userID = sign_in_test("demo-cardio21@gmail.com", "Cardiodemo100!")

class FormsTest(unittest.TestCase):
    """
    Test that the forms of the app execute without errors.
    """

    # Set up config.
    def setUp(self):
        self.app = app.test_client()
        app.secret_key = "OMONOIALAOSPROTATHLIMA"

    def test_add_patient(self):
        with app.test_client() as c:
            with c.session_transaction() as sess:
                sess['usr'] = userID

            response = c.post(
                '/patients/',
                data=dict(age="24", gender="0"),
                follow_redirects=True
            )
```

```python
            self.assertEqual(response.status_code, 200)

    def test_edit_patient(self):
        with app.test_client() as c:
            with c.session_transaction() as sess:
                sess['usr'] = userID

            response = c.post(
                '/edit?pid='+pid,
                data=dict(age=31, gender=0),
                follow_redirects=True
            )
            self.assertEqual(response.status_code, 200)

    def test_diagnose_patient(self):
        with app.test_client() as c:
            with c.session_transaction() as sess:
                sess['usr'] = userID
            response = c.post(
                '/diagnose?pid='+pid,
                data=dict(chest=2,
                        bps=170,
                        chol=300,
                        fbs=0,
                        ecg=1,
                        maxheart=120,
                        exang=1,
                        oldpeak=1.5,
                        stslope=2),
                follow_redirects=True
            )
            self.assertEqual(response.status_code, 200)


if __name__ == "__main__":
    unittest.main()
    app.run()
```

test_content.py

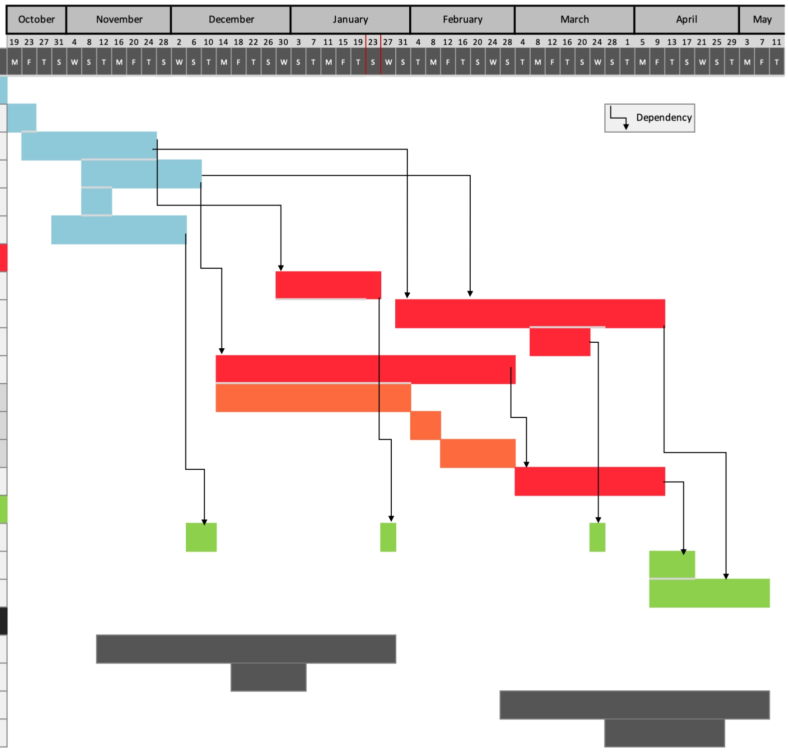
```python
from app import app, sign_in_test
import unittest

# Patient id used for testing purposes.
pid = "-MZxdolOqXRO79q_7VwB"
userID = sign_in_test("demo-cardio21@gmail.com", "Cardiodemo100!")

class ContentTest(unittest.TestCase):
    """
```

```
    Test the content of the pages when correct and incorrect parameters are passed
in.
    """

    # Set up config.
    def setUp(self):
        # Initialize test client.
        self.app = app.test_client()
        app.secret_key = "OMONOIALAOSPROTATHLIMA"
        # Initialise USERID to be used for test purposes.


    # Test that "/" - index, loads the correct data.
    def test_index(self):
         with app.test_client() as c:
            with c.session_transaction() as sess:
                sess['usr'] = userID

            # Since user is logged in, it will take them to the patients page.
            response = c.get('/', follow_redirects=True)
            self.assertTrue(b'Patients' in response.data)

    # Test that "diagnose" page, loads the correct data.
    def test_diagnose(self):
         with app.test_client() as c:
            with c.session_transaction() as sess:
                sess['usr'] = userID

            response = c.get('/diagnose?pid='+pid, follow_redirects=True)
            self.assertTrue(b"Fill out the form to diagnose whether the patient
suffers from a cardiovascular disease." in response.data)

    # Test that if pid is not passed in the "diagnose" page, the "diagnose" page is
not loaded.
    def test_diagnose_without_pid(self):
        with app.test_client() as c:
            with c.session_transaction() as sess:
                sess['usr'] = userID

            response = c.get('/diagnose', follow_redirects=True)
            self.assertFalse(b"Fill out the form to diagnose whether the patient
suffers from a cardiovascular disease." in response.data)

    # Test that "edit" page, loads the correct data.
    def test_edit(self):
          with app.test_client() as c:
            with c.session_transaction() as sess:
                sess['usr'] = userID

            response = c.get(
                'edit?pid='+pid, follow_redirects=True)
```

```
            self.assertTrue(b"Edit Patient" in response.data)

    # Test that if pid is not passed in the "edit" page, it's not loaded.
    def test_edit_without_pid(self):
         with app.test_client() as c:
            with c.session_transaction() as sess:
                sess['usr'] = userID

            response = c.get('/edit', follow_redirects=True)
            self.assertFalse(b"Edit Patient" in response.data)

    # Test that "history" page, loads the correct data.
    def test_history(self):
        with app.test_client() as c:
            with c.session_transaction() as sess:
                sess['usr'] = userID

            response = c.get(
            '/history?pid='+pid, follow_redirects=True)
            self.assertTrue(b"View Details" in response.data)

    # Test that if pid is not passed in the "history" page, it's not loaded.
    def test_history_without_pid(self):
        with app.test_client() as c:
            with c.session_transaction() as sess:
                sess['usr'] = userID

            response = c.get('/history', follow_redirects=True)
            self.assertFalse(b"View Details" in response.data)

    # Test that "report" page loads the correct data.
    def test_report(self):
        with app.test_client() as c:
            with c.session_transaction() as sess:
                sess['usr'] = userID

            response =
c.get('/report?pred=Most+Likely+Healthy&neg=0.82&pos=0.18&pid='+pid+'&ct=1619438610
', follow_redirects=True)
            self.assertTrue(b"Most Likely Healthy" in response.data)

    # Test that if pid is not passed in the "report" page, it's not loaded.
    def test_report_without_vars(self):
         with app.test_client() as c:
            with c.session_transaction() as sess:
                sess['usr'] = userID

            response = c.get('/report', follow_redirects=True)
            self.assertFalse(b"Most Likely Healthy" in response.data)
```

```python
if __name__ == "__main__":
    unittest.main()
```

4) Work Plan



TASK			
Planning & Analysis Phase	**START**	**END**	**DAYS**
Reviewing Project Idea	19/10/2020	24/10/2020	6
Researching relevant papers and ideas	26/10/2020	28/11/2020	31
Researching different A.I. classification models	08/11/2020	08/12/2020	31
Ethics Form	11/11/2020	12/11/2020	2
Working on presentation	02/11/2020	05/12/2020	34
Design & Implementation Phase	**START**	**END**	**DAYS**
Project Proposal	02/01/2021	25/01/2021	24
Writing Dissertation	30/01/2021	02/05/2021	93
Poster	10/03/2021	26/03/2021	17
Development of A.I. model	11/12/2020	02/03/2021	82
-> Experimenting with various algorithms	11/12/2020	30/01/2021	51
-> Hyperparameter tuning	31/01/2021	08/02/2021	9
->Improve and deploy model	09/02/2021	02/03/2021	22
Website Construction, implement with model	04/03/2021	10/04/2021	38
Evaluation/Testing Stage	**START**	**END**	**DAYS**
Evaluation of Deliverables	05/12/2020	24/03/2021	110
Evaluate & Test Programming	10/04/2021	20/04/2021	11
Evaluate & Improve Dissertation	10/04/2021	07/05/2021	28
Other Deadlines			
Semester 1 Coursework	12/11/2020	29/01/2021	79
Christmas Holiday	18/12/2020	04/01/2021	18
Semester 2 Coursework	28/02/2021	07/05/2021	69
Easter Holiday	26/03/2021	26/04/2021	32

5) ROC Curves

```python
# RANDOM FOREST
#Make predictions with probabilities
y_probs = rf_in.predict_proba(X_test)
y_probs_positive = y_probs[:,1]

#Calculate fpr, tpr and thresholds
fpr, tpr, thresholds = roc_curve(y_test, y_probs_positive)

#Check the false positive rates
fpr


# XGBOOST
#Make predictions with probabilities
y_probs_xg = xg_in.predict_proba(X_test)
y_probs_positive_xg = y_probs_xg[:,1]

#Calculate fpr, tpr and thresholds
fpr1, tpr1, thresholds1 = roc_curve(y_test, y_probs_positive_xg)

#Check the false positive rates
```

```
fprl


# CATBOOST
#Make predictions with probabilities
y_probs_cat = cat_in.predict_proba(X_test)
y_probs_positive_cat = y_probs_cat[:,1]

#Calculate fpr, tpr and thresholds
fpr2, tpr2, thresholds2 = roc_curve(y_test, y_probs_positive_cat)

#Check the false positive rates
fpr2

# Function to plot ROC curve
def plot_roc_curve(fpr, tpr, fpr1, tpr1, fpr2, tpr2):

  #Plot roc curve
  plt.figure(figsize=(20,10))
  plt.plot(fpr, tpr, color="orange", label="ROC Random Forest")
  plt.plot(fpr1, tpr1, color="red", label="ROC XGBoost")
  plt.plot(fpr2, tpr2, color="green", label="ROC Catboost")
  #Plot line with no predictive power (baseline)
  plt.plot([0,1],[0,1],color="darkblue",linestyle="--
",label="Guessing")

  #Customize the plot
  plt.xlabel("false positive rate (fpr)")
  plt.ylabel("True positive rate (tpr)")
  plt.title("Receiver Operating Characteristics (ROC) Curve")
  plt.legend()
  plt.show()



#Call function
plot_roc_curve(fpr,tpr, fpr1, tpr1,fpr2,tpr2)
```

6) Confusion Matrix

```
from sklearn.metrics import confusion_matrix

#Make predictions.
y_preds_rf= rf_in.predict(X_test)
y_preds_xg = xg_in.predict(X_test)
y_preds_cat = cat_in.predict(X_test)

#Create a confusion matrix
```

```python
conf_mat_rf = confusion_matrix(y_test, y_preds_rf)
conf_mat_xg = confusion_matrix(y_test, y_preds_xg)
conf_mat_cat = confusion_matrix(y_test, y_preds_cat)

# Function that plots a confusion matrix
def plot_conf_mat(conf_mat,name):

  fig,ax = plt.subplots(figsize=(10,10))
  ax = sns.heatmap(conf_mat, annot=True, cbar=True, fmt="d")
  plt.title(name , fontsize=30)
  plt.xlabel("Predicted label")
  plt.ylabel("True label");


#Call the function
plot_conf_mat(conf_mat_rf, "Random Forest")
plot_conf_mat(conf_mat_xg, "XG Boost")
plot_conf_mat(conf_mat_cat, "Catboost")
```

7) User Manual

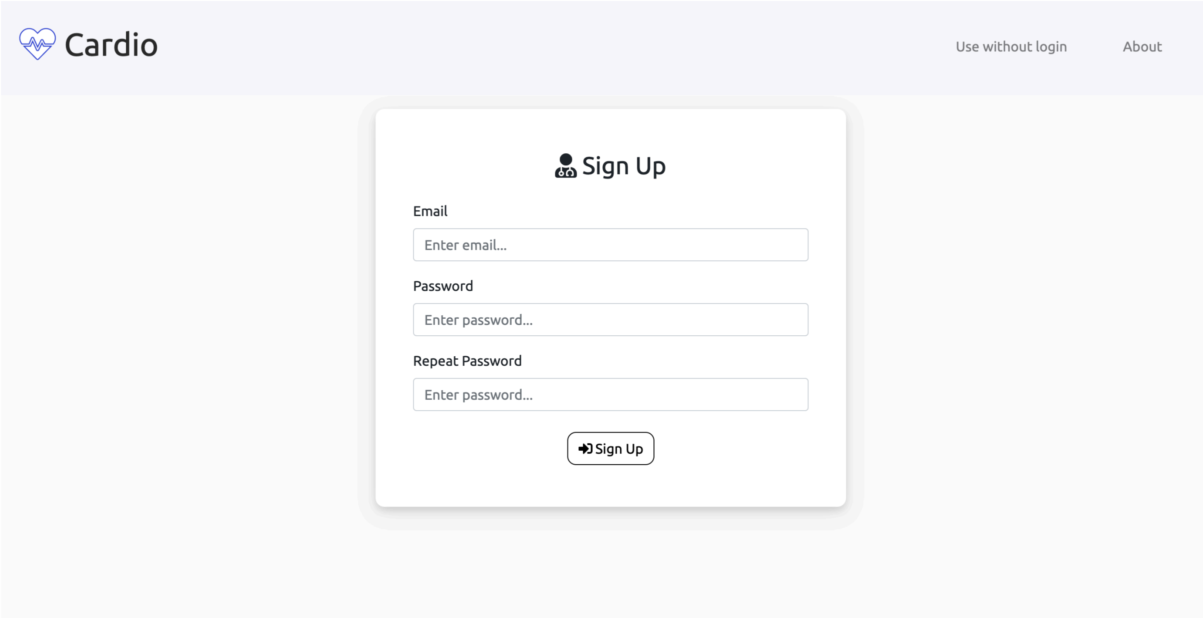
Cardio

Cardio is a medical web platform that was created to help doctors visualize and predict cardiovascular diseases. This is a brief user manual to explain how you can use the system.

Table of Contents

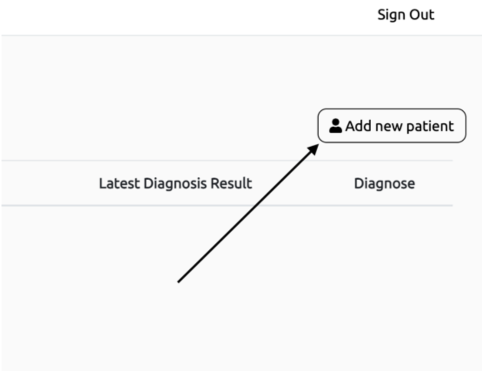
Sign up

To sign up you have to enter a valid email address and two matching passwords. If this is satisfied, your account will be created, and you will be redirected to the “Patients” page.

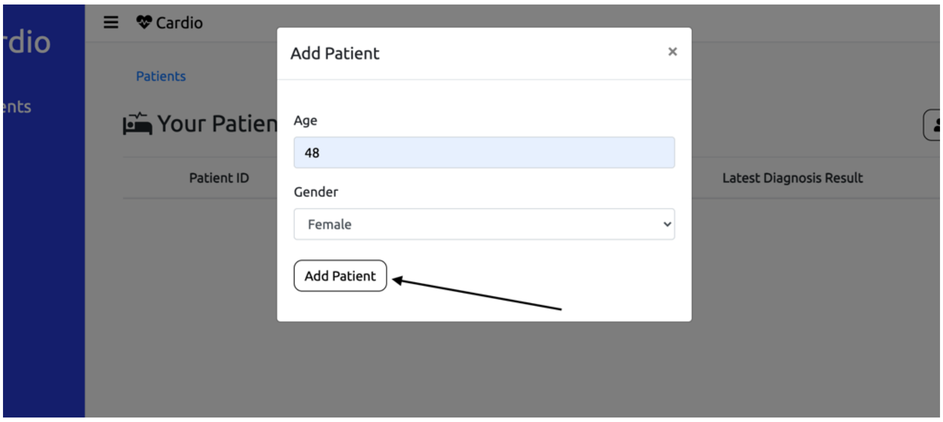


Add new patient

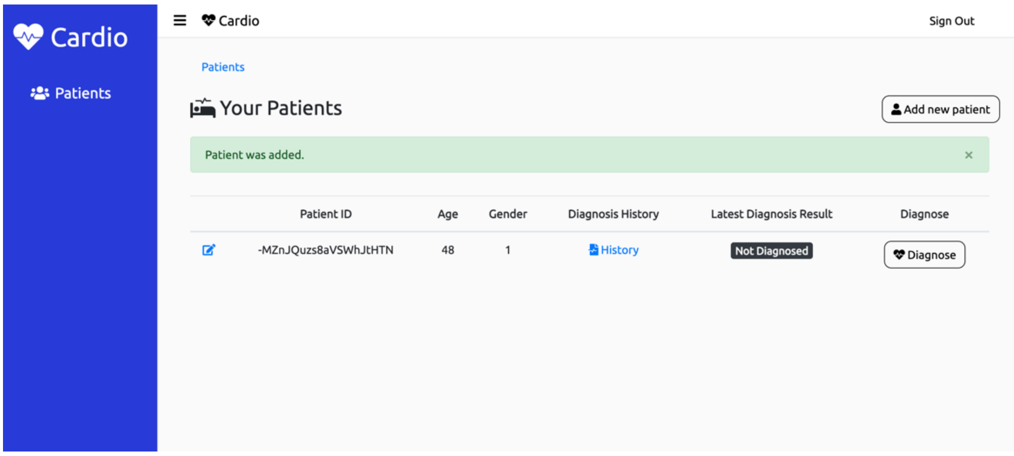
Once you are in the patients page you can add new patients and edit and diagnose existing patients. To add a new patient click on the “add patient” button.



A modal will open requiring the age of the patient and their gender. Once successfully entered, a new patient will be created in the database with a unique patient id that you will use to identify the patient.

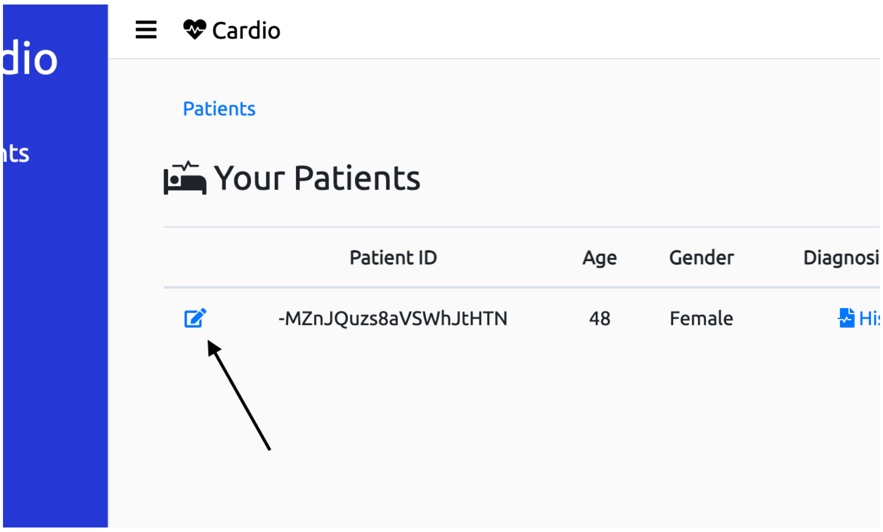


The new patient will appear on your “Patients” page.

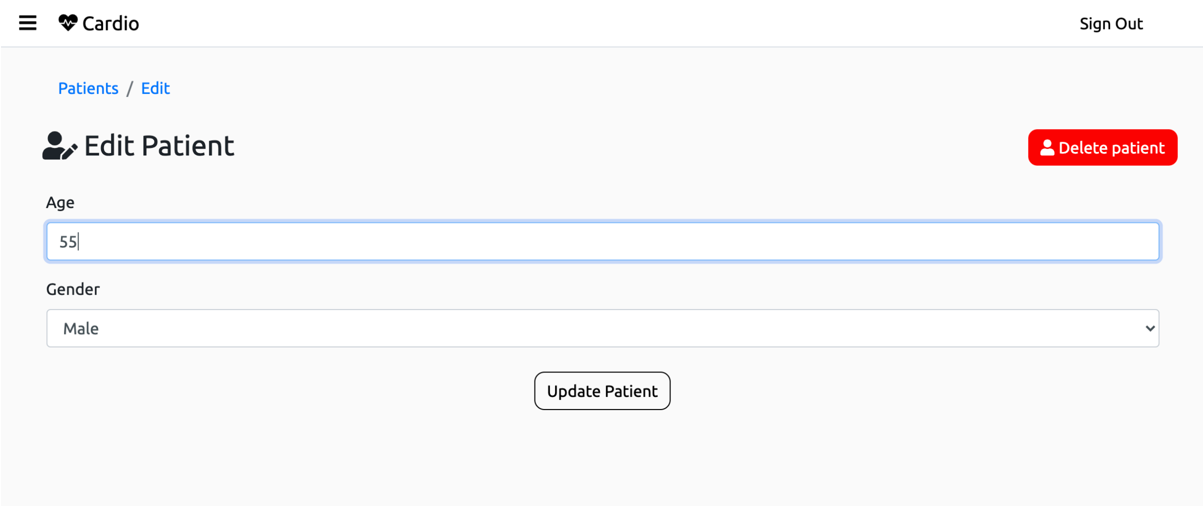


Edit patient

You can edit the patient information by clicking on the “edit” button on their row.

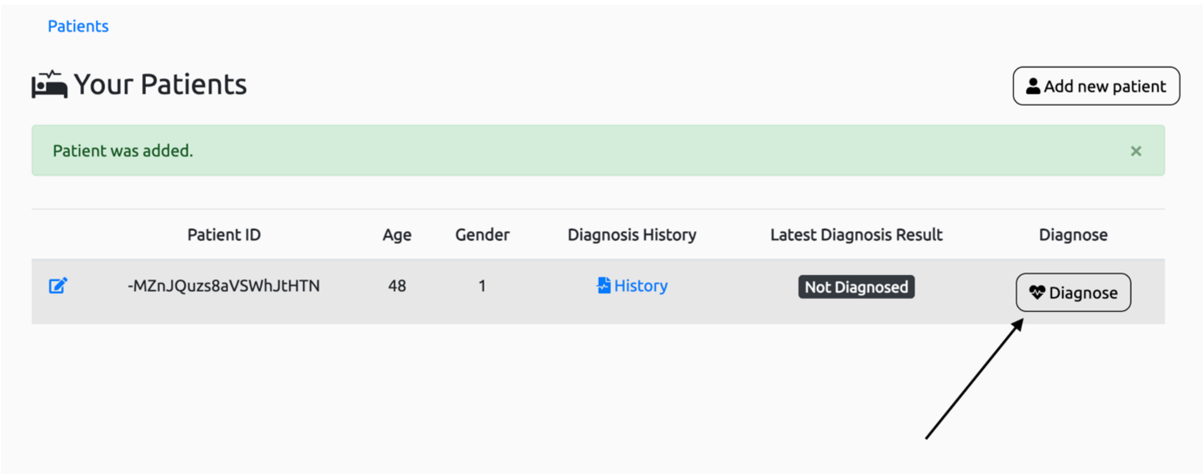


In the edit page you can update a patient’s age and gender.

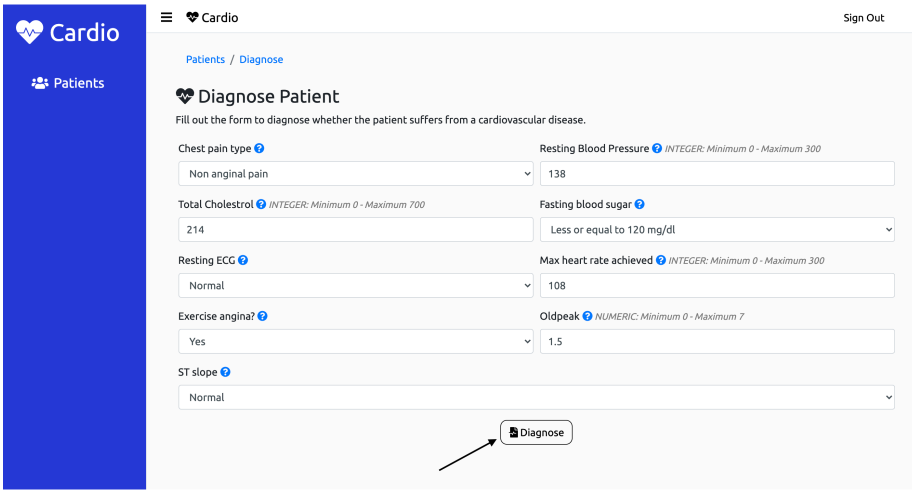


Diagnose

To diagnose a patient, click on the diagnose button on the patient's row.



An input form will appear where you have to fill the values for all the inputs. Once successfully filled click on the “diagnose” button.



This will automatically generate a medical report displaying all of the patient’s data in a meaningful graphical way.

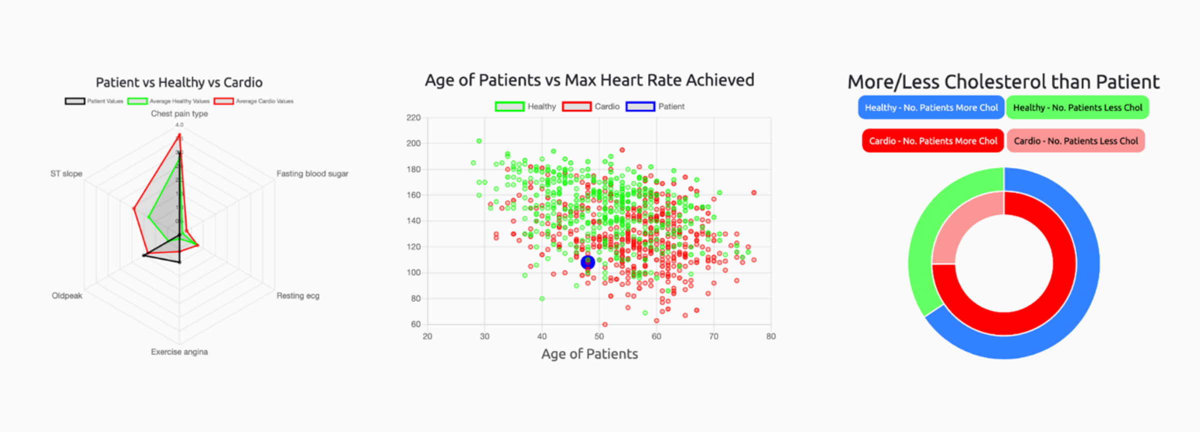
Treat

Three types of graphs are generated in the visual report:

1) A radar graph - Showing the variation between the **average** healthy and CVD values against patient’s values.

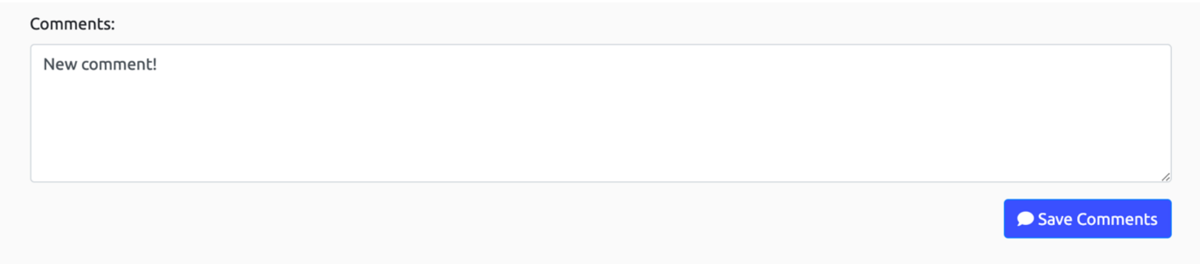
2) Scatter Age graph – Showing the distribution of healthy and CVD patients in a scatter chart as a function of age against various parameters

3) Doughnut – Showing the number of patients who have a higher or lower value for a health feature than the current patient.



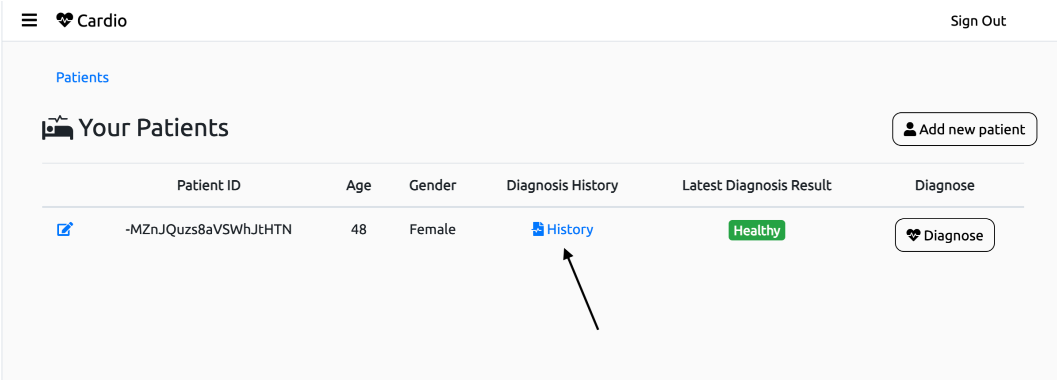
You can use the graphs to further understand the condition of the patient and compare them against other healthy and CVD patients. The graphs generated in the medical report can also be downloaded as a PDF. If you feel appropriate you can show these graphs to the patient to help them understand better their condition and your diagnosis.

If you want to add any comments on the report, you can do that at the bottom of the report page. This comment will be saved in the database along with the diagnosis values and generated pdf.

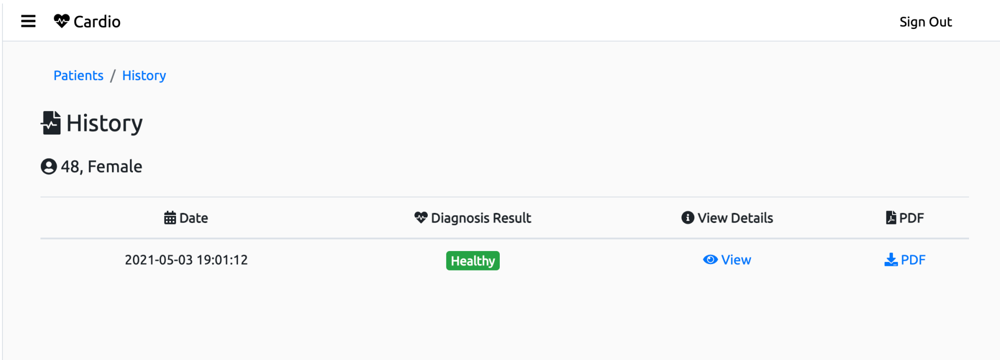


History

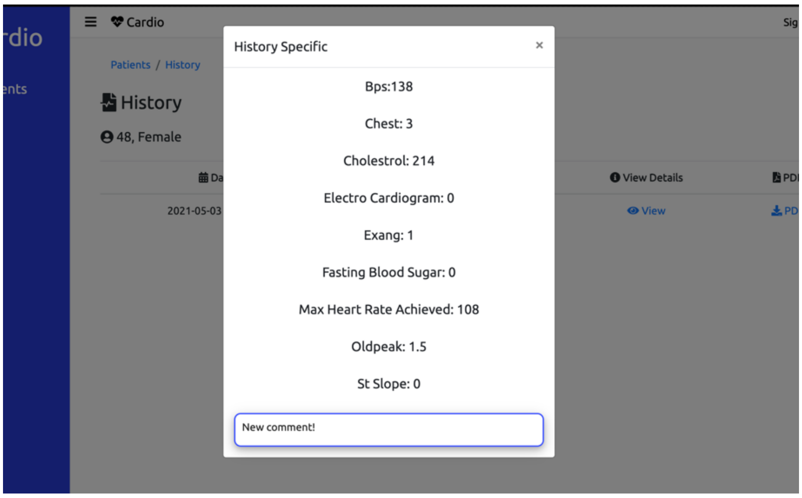
Once you are done with your examination and return to the “Patients” page you will see that the “Latest Diagnosis Result” of the patient changed accordingly to the prediction of the diagnosis.
You can view the diagnosis history of the patient by clicking on the “history” button on the patient’s row.



Every diagnosis that you perform on the patient will appear here. A diagnosis history row includes the date, result, details and pdf of the graphs of the diagnosis.

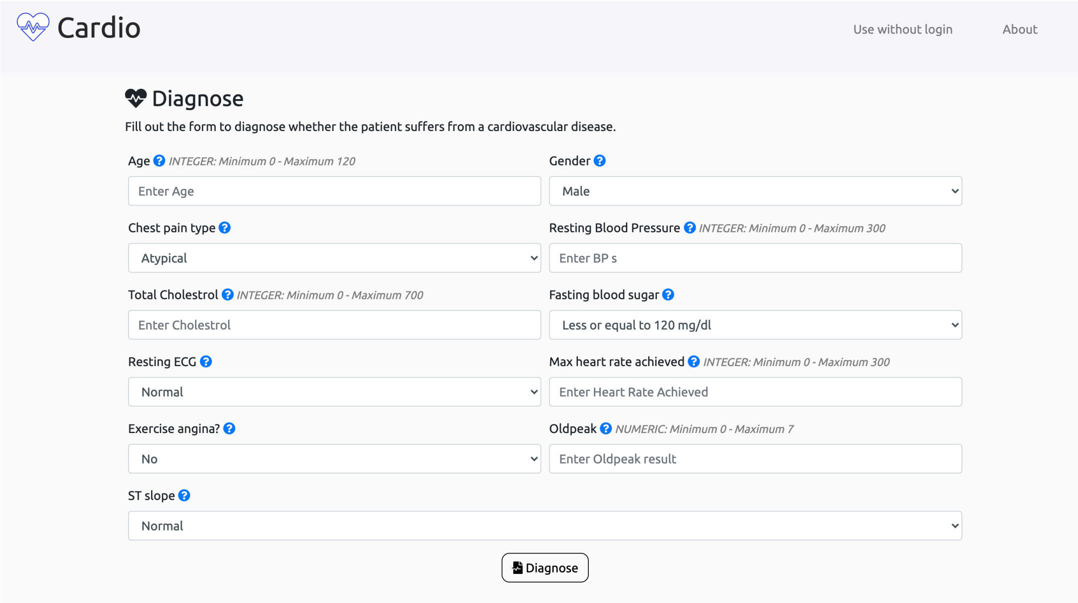


If you click on view details you can see all the values entered and the comment for that diagnosis.



Use without login

The “Use without login” feature enables you to quickly diagnose a patient without having to login. The result of the diagnosis will not be stored in the database.



Once the form has been filled successfully and the “Diagnose” button was clicked, a graphical report will be generated. The report does not include the download PDF functionality, or the ability to add a comment.

For any enquiries please contact:

r.kollyfas2@newcastle.ac.uk