THE SOCIALIST REPUBLIC OF VIETNAM
Independence – Freedom– Happiness
****

ADVISOR'S EVALUATION SHEET

Student name: **PHAM CONG THANH** Student ID:18146213

Major: Mechatronics

Project title: **IoT Device with AI System for ECG monitor and classification**

Advisor: **PhD. NGUYEN VAN THAI**

EVALUATION

1. Contents of project:

.....
.....
.....
.....

2. Strengths:

.....

3. Weaknesses:

.....

4. Approval for oral defense? *(Approved or denied)*

.....

5. Overall evaluation: *(Excellent, Good, Fair, Poor)*

.....

6. Mark: ..... (in words: .....)

Ho Chi Minh City, Feb 1, 2023

ADVISOR
(Sign with full name)

THE SOCIALIST REPUBLIC OF VIETNAM
Independence – Freedom– Happiness
****

PRE-DEFENSE EVALUATION SHEET

Student name: **PHAM CONG THANH** Student ID:18146213

Major: Mechatronics

Project title: **IoT Device with AI System for ECG monitor and classification**

Name of Reviewer:

.....

EVALUATION

1. Contents of project:

.....
.....
.....
.....

2. Strengths:

.....

3. Weaknesses:

.....

4. Approval for oral defense? *(Approved or denied)*

.....

5. Overall evaluation: *(Excellent, Good, Fair, Poor)*

.....

6. Mark: ..... (in words: .....)

Ho Chi Minh City, month day, year

REVIEWER
(Sign with full name)

THE SOCIALIST REPUBLIC OF VIETNAM
Independence – Freedom– Happiness
****

EVALUATION SHEET OF DEFENSE COMMITTEE MEMBER

Student name: **PHAM CONG THANH** Student ID:18146213

Major: Mechatronics

Project title: **IoT Device with AI System for ECG monitor and classification**

Name of Defense Committee Member:

.....

EVALUATION

1. Contents of project:

.....
.....
.....
.....

2. Strengths:

.....

3. Weaknesses:

.....

4. Approval for oral defense? *(Approved or denied)*

.....

5. Overall evaluation: *(Excellent, Good, Fair, Poor)*

.....

6. Mark: ..... (in words: .....)

Ho Chi Minh City, Feb 1, 2023

COMMITTEE MEMBER
(Sign with full name)

THE SOCIALIST REPUBLIC OF VIETNAM
Independence – Freedom– Happiness
****

PROJECT PLANNING RECORD

Student name: **PHAM CONG THANH** Student ID:18146213

Major: Mechatronics

Project title: **IoT device with AI system for ECG Monitoring and Classification**

To-do list:

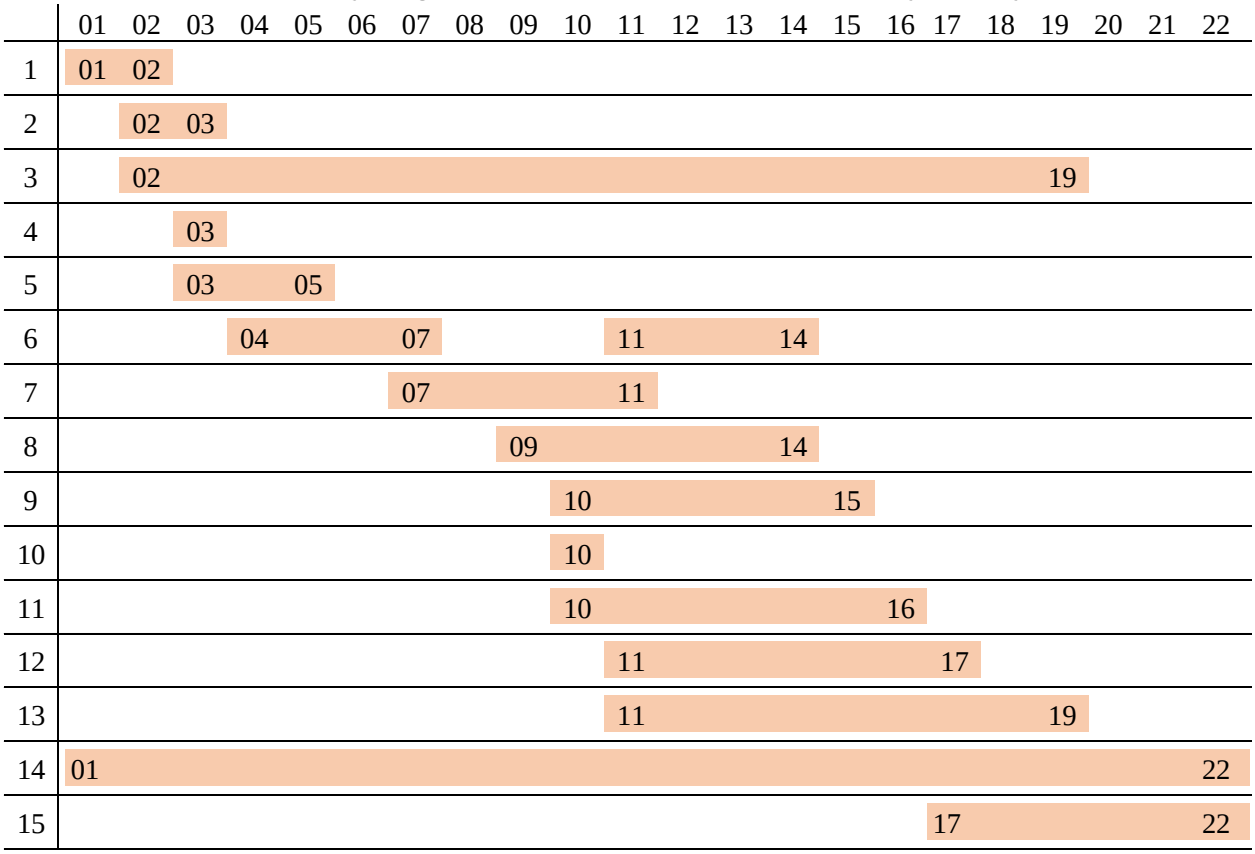
1. Problem analyst and find solutions for ECG classification.
2. Study about cardiovascular system.
3. Reading books and thinking about all aspects of an AI system in industry.
4. Find dataset for training.
5. Find models that are suitable for this case and choose one of them.
6. Feature engineer (filtering, scaling, feature extraction).
7. Resampling dataset.
8. Training and tuning hyperparameters with wandb.
9. Model validation.
10. Read data from device and feed it into model for classification.
11. Storing data and result into databases.
12. Create Docker image.
13. Deploy Machine Learning model in single-board computers (Raspberry Pi, Jetson Nano DevKit, Orange Pi).
14. Write thesis.
15. Design poster and video.

Review of advisor:

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

Week

Start date: Wednesday, August 31, 2022 End date: Tuesday, January 31, 2023



Ho Chi Minh City, Feb 1, 2023

ADVISOR

(Sign with full name)

ACKNOWLEDGEMENTS

After nearly 6 months working on my graduation project, I want to express my gratitude to those who have spent time and effort on me during this project.

First, I sincerely thank Dr. Nguyen Van Thai, who directly guided me to create the conditions for me to carry out the subject, kindly helped me, and suggested ideas during the difficulties I encountered. Then, MS. Nguyen Phuong Nam in the Economic Department and MS. Nguyen Thi Nhat Quynh - Deputy Director of R&D, Head of the Institute of Physics and Applied Sciences, who have rich experience, have given me many opportunities to learn useful knowledge.

In addition, Mr. Duc, Mr. Khoa, Mr. Uy, Ms. Mai, and Mr. Thao, who are members of the 3D Vision Lab, have directly supported me with issues related to IoT and AI systems. As well as other members of the lab have created a favorable and lively environment, which helped me have beautiful memories.

I also want to express my gratitude to MS. Vo Lam Chuong and my graduation project group including Mr. Quy, Mr. Hiep and Mr. Long, who have given me the opportunity to participate in designing, developing and testing the project with them. What I have learned is much more than my contribution to this project.

Ho Chi Minh City, Feb 1, 2023

STUDENT

(Sign with full name)

PHAM CONG THANH

LỜI CẢM ƠN

Sau gần 6 tháng làm đồ án tốt nghiệp, em mượn trang giấy này để bày tỏ lòng biết ơn của mình đối với những người đã dành thời gian và công sức cho em trong quá trình làm đồ án này.

Đầu tiên, em chân thành cám ơn thầy TS. Nguyễn Văn Thái, người đã trực tiếp hướng dẫn tạo điều kiện cho em thực hiện đề tài, tận tình giúp đỡ, đề xuất các ý tưởng trong quá trình em gặp khó khăn. Kế đến là thầy Nguyễn Phương Nam khoa Kinh tế và chị Nguyễn Thị Nhật Quỳnh – Phó phòng R&D, Trưởng phòng Viện Vật lý và khoa học ứng dụng, hai người dày dặn kinh nghiệm đã cho em nhiều cơ hội để học hỏi những kiến thức hữu ích.

Ngoài ra, anh Đức, anh Khoa, bạn Uy, bạn Mai, bạn Thảo, là những thành viên của 3D Vision Lab, đã trực tiếp hỗ trợ em những vấn đề về hệ thống IoT và AI. Cũng như là những thành viên còn lại của lab đã tạo điều kiện thuận lợi và môi trường tươi vui, năng động, giúp em trải qua những ngày tháng cuối cùng trên ghế giảng đường với những kỷ niệm đẹp.

Em cũng muốn bày tỏ lòng biết ơn đến thầy Võ Lâm Chương và nhóm đồ án tốt nghiệp của ba bạn bao gồm bạn Quý, bạn Hiệp và bạn Long, đã cho em tham gia thiết kế robot Delta. Những gì em nhận được và học được trong suốt quá trình làm nhiều hơn so với những gì em đã đóng góp cho nhóm.

Thành phố Hồ Chí Minh, ngày 01 tháng 02 năm 2023

SINH VIÊN

(Ký và ghi đủ họ tên)

PHẠM CÔNG THÀNH

ABSTRACT

Chapter 1 Introduction

An overview of the subject is presented. Introduce the reasons for choosing the subject, objectives, research content and limitations within the scope of research.

Chapter 2 Cardiovascular and ECG

The basic theory of cardiovascular, the operation of the heart and ECG machines.

Chapter 3 Machine learning systems

The basic theory of machine learning.

Chapter 3 explains the differences between the preparation and processing of data, data extraction, labeling, evaluation, ... between research and deployment stage of machine learning systems. These are the necessary knowledge to be able to carry out the project as well as to consolidate knowledge before going to work.

Chapter 4 Data collection, analysis and storage

Presentation of the data sets used to train the model, including the MIT-BIH Arrhythmia dataset and data collected from the device.

In addition, techniques for data processing such as noise filtering, sample extraction, handling imbalanced data, ... and the database used to store data collected from the device.

Chapter 5 Model development

An overview and concept of AI algorithms used in this project are presented. Specifically, XGboost is a well-known model. Until the expected result is achieved, we can deploy it on single-board computers.

Chapter 6 Model deployment

In Chapter 6, the presentation will be on how to initialize the initial parameters, package, and create a Docker Image for convenience in running the algorithm on different platforms. And finally, the design of the dashboard.

Chapter 7 Results

Chapter 8 Conclusion and future development

TÓM TẮT

Chương 1 Giới thiệu

Giới thiệu tổng quan về đề tài. Dẫn nhập lý do chọn đề tài, mục tiêu, nội dung nghiên cứu và những giới hạn trong phạm vi nghiên cứu.

Chương 2 Hệ tim mạch và điện tâm đồ

Lý thuyết cơ bản về tim mạch, cách hoạt động của tim và các máy đo ECG.

Chương 3 Hệ thống Machine learning

Lý thuyết cơ bản về machine learning.

Chương 3 nêu ra sự khác biệt giữa các khâu chuẩn bị và xử lý dữ liệu, trích xuất dữ liệu, gán nhãn, đánh giá,…giữa nghiên cứu và triển khai hệ thống machine learning. Đây là những kiến thức cần thiết để có thể thực hiện đồ án cũng như củng cố kiến thức trước khi đi làm.

Chương 4 Thu thập, phân tích và lưu trữ dữ liệu

Trình bày về các bộ dữ liệu được dùng để huấn luyện trong model bao gồm tập dữ liệu MIT – BIH Arrhythmia và dữ liệu được lấy từ thiết bị.

Kèm theo đó là các kỹ thuật để xử lý dữ liệu như lọc nhiễu, trích xuất mẫu, xử lý tập dữ liệu không cân bằng, … và cơ sở dữ liệu dùng để lưu trữ dữ liệu được lấy từ thiết bị.

Chương 5 Phát triển mô hình

Trình bày tổng quan và khái niệm về thuật toán AI được sử dụng trong đồ án này. Cụ thể ở đây là XGboost là một mô hình nổi tiếng và mạnh mẽ trong Machine Learning.

Cuối cùng, khi đạt kết quả mong muốn trong quá trình huấn luyện, chúng ta có thể triển khai nó trên hệ thống Machine Learning.

Chương 6 Triển khai mô hình

Trong chương 6 sẽ trình bày về cách khởi tạo các thông số ban đầu, đóng gói và tạo Docker Image để thuận tiện trong việc chạy thuật toán trên nhiền nền tảng khác nhau. Và cuối cùng là thiết kế dashboard.

Chương 7 Kết quả đạt được

Chương 8 Kết luận và hướng phát triển

DECLARATION OF AUTHORSHIP

I hereby declare, with the help of PhD. Nguyen Van Thai, I do this thesis on my own. I am the sole author and composer of my thesis and that no other sources or learning aids, other than those listed, have been used. Furthermore, I declare that I have acknowledged the work of others by providing detailed references of said work.

Ho Chi Minh City, Feb 1, 2023

STUDENT
(Sign with full name)

PHAM CONG THANH

TABLE OF CONTENTS

LIST OF ACRONYMS

ECG	Electrocardiogram
TPR	True positive rate
FPR	False positive rate
TNR	True negative rate
FNR	False negative rate
CV	Cross validation
OPi	Orange Pi
OPi z2	Orange Pi zero 2
RPi	Raspberry Pi
DevKit	Developer Kit

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

1.1. Introduction

Nowadays, cardiovascular disease has become the leading cause of death in most countries worldwide and in Vietnam. Before 1900, infectious diseases and malnutrition were the main causes of death, while deaths due to cardiovascular disease accounted for less than 10% of causes of death. However, by the early 21st century, deaths from cardiovascular disease are estimated to be 17.9 million worldwide and account for 33% of causes of death. Alarmingly, the total number of deaths from cardiovascular disease is still rapidly increasing and a high proportion in developing countries with medium-low incomes. [21]

There are many methods of electrocardiogram in medicine for diagnosing cardiovascular diseases. In medicine, to fully express the state of the heart's operation, the patient is attached with 12 electrocardiogram electrodes to the skin and measures the ECG in a sitting or lying position. The disadvantage of this measurement method is the use of fixed, cumbersome, and costly equipment, which is only available in hospitals, making it difficult to monitor the patient's condition for a long time.

The traditional Holter monitor overcomes the weakness of mobility and allows long-term monitoring, suitable for patients who need to monitor their heart and pulse regularly without interrupting their daily activities, however, the machine still needs to be equipped with electrodes and wires, which is still inconvenient for patients. However, there are still some limitations that patients, their relatives, and doctors are unable to monitor the patient's condition in real-time.

1.2. Reason for writing

In order to improve overall health and prevent, reduce deaths from cardiovascular disease specifically in real-time, I have chosen the topic of **IOT DEVICE WITH AI SYSTEM FOR ECG CLASSIFICATION AND MONITORING** to provide timely warnings of potential hidden dangers in cardiovascular system.

1.3. Research Objectives

The purpose of this thesis is to use data collected from iot wearable device, and using machine learning algorithms written in Python language, to identify abnormal signs and symptoms of cardiovascular diseases.

1.4. Scope of study

In this graduation project, the algorithm that will be used is XGBoost to classify and warn signals of electrocardiogram that are at risk or already have cardiovascular disease, specifically heart arrhythmia.

Then I will propose two solutions for storing data in databases, one solution for continual learning, solution for optimize gateway and one dashboard for end user.

IoT device is invented by Mr. Khoa [20] so I do not mention how to create it in this project.

1.5. Methodology

Data analysis and data visualization skill are used to understand hidden patterns in ECG signal.

Data version management is needed to control specific version of data.

Experiment tracking skill is used to find the good parameters for model during training process.

Researching documents, thesis, and reports both domestically and internationally.

CHAPTER 2: CARDIOVASCULAR AND ECG

2.1. The heart

2.1.1. How the heart works

The human heart is a four-chambered muscular organ that is roughly the size and shape of a man's closed fist, with two-thirds of its bulk located on the left side of the midline.

The heart is housed in a sac known as the pericardial sac, which is surrounded by a serous membrane and lined with a layer known as the parietal layer. The inner layer of the serous membrane is known as the epicardium.

Layers of the Heart Wall

The heart wall is composed of three layers of tissue. The epicardium is the outer layer, the myocardium is the middle layer, and the endocardium is the inner layer.

Chambers of the Heart

The heart has four internal chambers:

- Right atrium
- Right ventricle
- Left atrium
- Left ventricle

The two atria and the two ventricles. The atria are chambers with thin walls that receive blood from the veins, whereas the ventricles are chambers with thick walls that pump blood out of the heart with force. The thickness of the walls of the chambers varies due to the amount of myocardium present, which determines the amount of force each chamber needs to generate. The right atrium receives blood without oxygen from the body's veins, and the left atrium receives oxygenated blood from the lungs' veins.

Valves of the Heart

The heart has two types of valves to ensure the blood flows in the correct direction. Atrioventricular valves, also known as cuspid valves, are located between the atria and ventricles. Semilunar valves are located at the base of the large vessels that carry blood out of the ventricles.

The right atrioventricular valve is called the tricuspid valve, the left atrioventricular valve is called the bicuspid or mitral valve. The valve between the right ventricle and the pulmonary trunk is known as the pulmonary semilunar valve and the valve between the left ventricle and the aorta is known as the aortic semilunar valve.

The atrioventricular valves close to prevent blood from flowing back into the atria when the ventricles contract and the semilunar valves close to prevent blood from flowing back into the ventricles when the ventricles relax.

Pathway of Blood through the Heart

It's easy to discuss the flow of blood through the heart by separating it into the right and left side, but it's essential to note that both the atria and ventricles contract simultaneously. The heart functions as two pumps that work together. Blood flows from the right atrium to the right ventricle, where it is then pumped to the lungs to receive oxygen. After receiving oxygen in the lungs, the blood flows to the left atrium, then to the left ventricle, and finally, it is pumped out to the rest of the body.

Blood Supply to the Myocardium

The myocardium of the heart wall is a functioning muscle that requires a constant supply of oxygen and nutrients in order to operate properly. As a result, heart muscle has a dense network of blood arteries to deliver oxygen to contracting cells and eliminate waste products.

The right and left coronary arteries, which are branches of the ascending aorta, deliver blood to the myocardium's walls. After passing through the capillaries of the myocardium, blood enters a network of cardiac (coronary) veins. The coronary sinus, which opens into the right atrium, drains the majority of the cardiac veins.

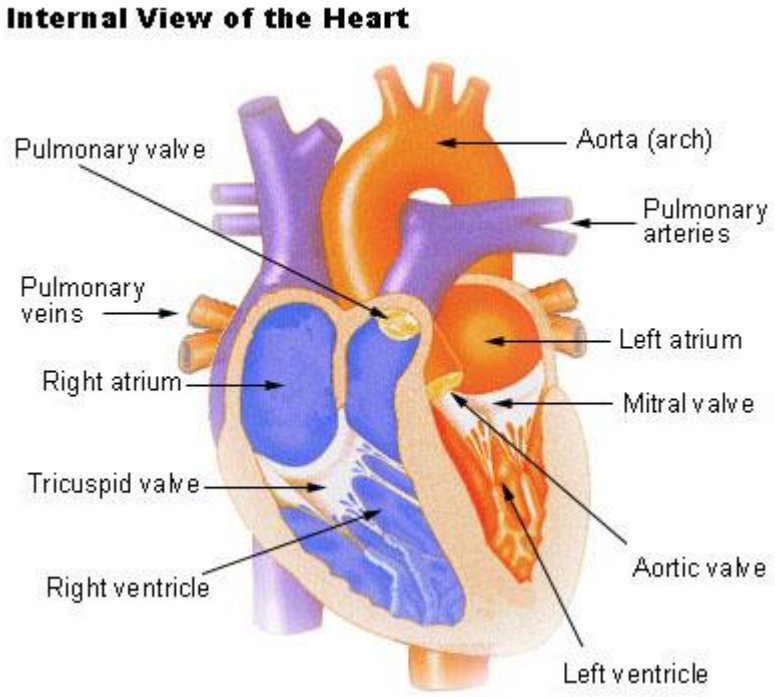


Figure 2-1. Internal view of the Heart

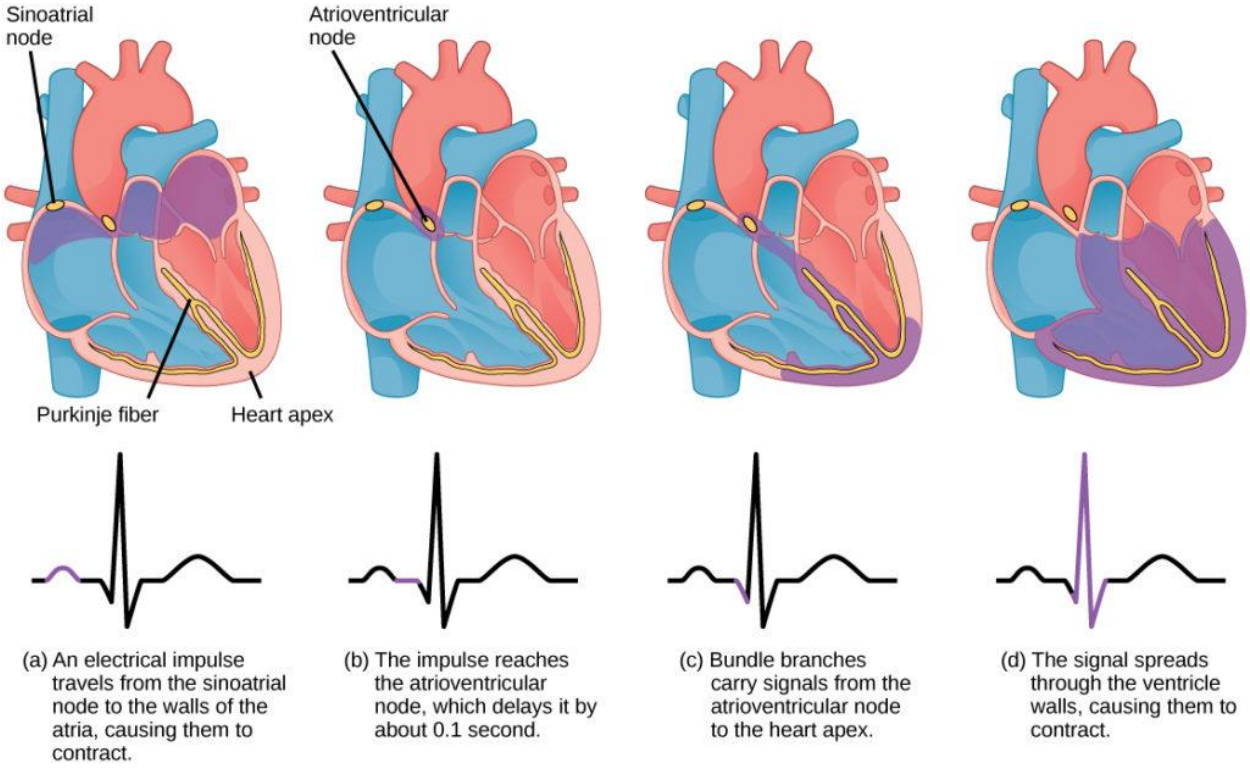


Figure 2-2. Phase of cardiac cycle

The heart's internal pacemaker, which uses electrical signals to time the heart's pumping, regulates the independent beating of cardiac muscle cells. The electrical signals and mechanical operations depicted in Figure 2-2 are inextricably linked. The internal pacemaker begins in the sinoatrial (SA) node, which is positioned near the right atrial wall. Electrical charges from the SA node spontaneously pulse, causing the two atria to constrict in tandem.

The pulse travels to a second node, the atrioventricular (AV) node, located between the right atrium and right ventricle, where it stops for about 0.1 second before spreading to the ventricle walls.

The electrical impulse travels from the AV node to the His bundle, then to the left and right bundle branches that extend through the interventricular septum. Finally, the impulse is carried up the ventricular myocardium by the Purkinje fibers, and the ventricles contract. This gap permits the atria to completely empty into the ventricles before the ventricles pump blood out. Electrical currents are produced by electrical impulses in the heart and can be detected on the skin using electrodes. This information is available.

2.2. ECG monitors

There are several ECG monitors over the world, they can be categorized by 2 groups:

- 12-lead ECG monitor
- Holter

2.2.1. 12-lead ECG monitor

A 12-lead electrocardiogram (ECG) is a standard medical machine that use leads, or nodes, attached to the body to capture the electrical activity of the heart and transfer it to graphed paper. The results can then be analysed by medical professionals, such as cardiologists, cardiac nurses and technicians.

It was firstly invented by Willem Eithoven in 1924, who was achieved Nobel Prize in medicine for this invention.

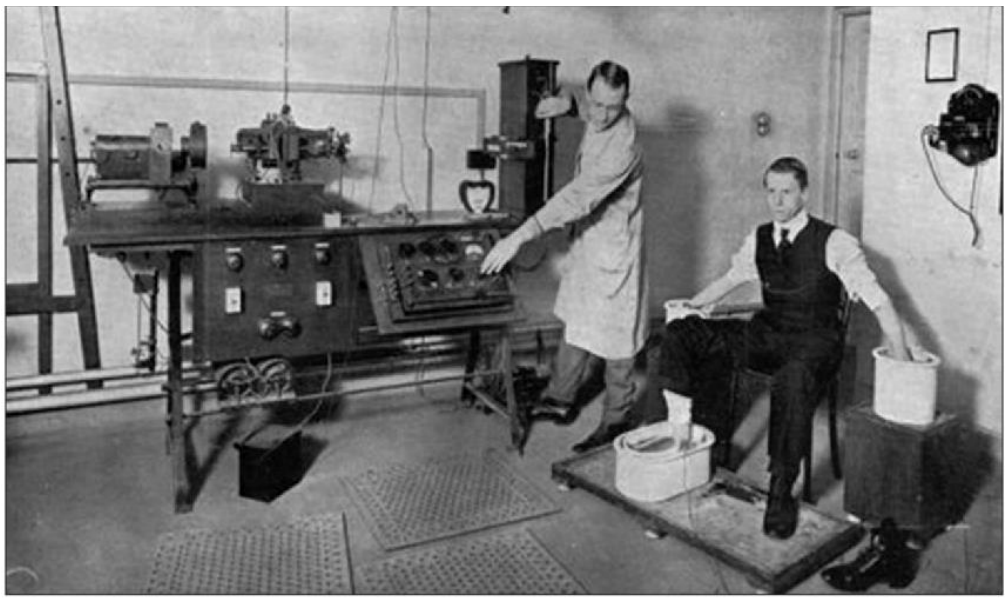


Figure 2-3. The first ECG machine

Recent machine is much more portable and easier to use than the first ECG monitor machine.

2.2.2. Holter

The Holter monitor has been widely used by cardiologists since its initial release in 1962. With the prevalence of life-threatening cardiac disorders on the rise, more cardiologists are in need of greater knowledge into their patients' heart health.

The Holter monitor is a form of portable electrocardiogram (ECG). It constantly monitors the electrical activity of the heart for 24 hours or more while you are gone from the doctor's office.

Electrodes (small, plastic patches that attach to the skin) are implanted on the chest and belly at various sites. Wires connect the electrodes to a holter. The heart's electrical activity may then be monitored, recorded, and printed. No electricity is sent to the body.

Natural electrical impulses coordinate contractions of the various heart chambers. This maintains the blood flowing normally. The machine captures these impulses to determine the heart's rate, rhythm (steady or irregular), and the intensity and timing of the electrical impulses. Changes in an ECG can indicate a variety of cardiac problems.

However, despite its current incarnation, the Holter monitor continues to be a poor answer to the issues of cardiology diagnostics and remote patient monitoring (RPM).

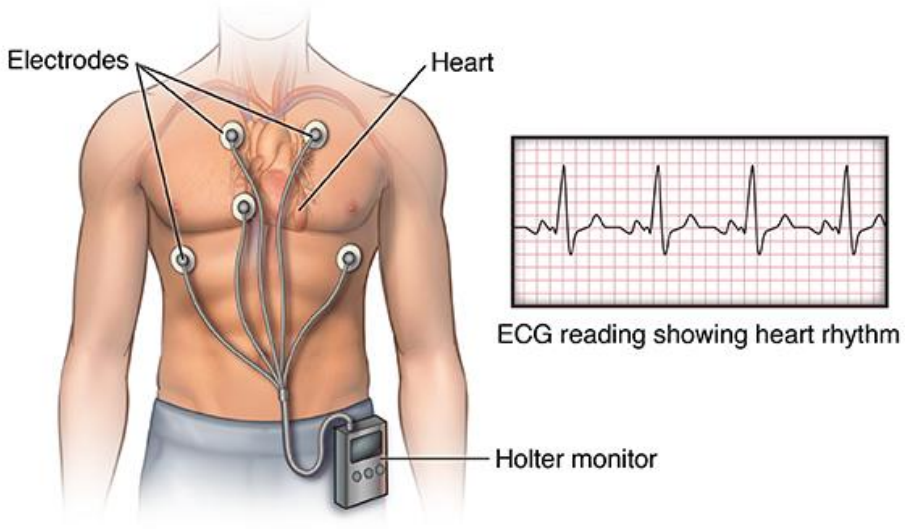


Figure 2-4. Holter monitor

CHAPTER 3: MACHINE LEARNING SYSTEM

3.1. An introduction to Machine Learning

One of the most frequently discussed subjects in the world of technology is machine learning (ML). You must be aware of it by this point. Machine learning is a type of technique that allows software programs to predict outputs more accurately without being explicitly programmed. The core idea behind machine learning is to create algorithms that can take in input data and apply statistical analysis to predict an output, while updating results as new data becomes available.

If you are already familiar with the fundamentals of machine learning, its applications, and how it works, this chapter will further your understanding. Moreover, the following information is needed to understand the work I have done.

Up to now, there are many machine learning algorithms that have been and are being developed, we can classify them in two ways, grouping based on methods or way of learning. There are:

- Supervised Machine Learning
- Unsupervised Machine Learning
- Semi-Supervised Machine Learning
- Reinforcement Learning

3.1.1. Supervised Learning

Supervised machine learning, as the name implies, is based on supervision. It implies that in the supervised learning approach, we train the machines using the "labelled" dataset, and the machine predicts the output based on the training. The labeled data in this case indicates that some of the inputs have already been mapped to the output. We may be more precious; first, we train the computer with the input and associated output, and then we ask the machine to predict the outcome using the test dataset.

Supervised machine learning issues may be divided into two categories, which are classification and regression.

Classification

Classification algorithms are used to handle classification issues using categorical output variables, such as "Yes" or "No," Male or Female, Red or Blue, and so on. The categories in the dataset are predicted by the categorization algorithms. Spam detection, email filtering, and other real-world applications are examples of classification algorithms.

Regression

Regression algorithms are used to address regression issues when the input and output variables have a linear relationship. These are used to forecast continuous output variables such as market movements, weather forecasting, and so on.

3.1.2. Unsupervised Learning

Unsupervised learning differs from supervised learning in that there is no requirement for supervision, as the term implies. It indicates that in unsupervised machine learning, the machine is trained with an unlabeled dataset and predicts the output without supervision.

Unsupervised learning involves training models with data that is neither categorized nor labeled, and then allowing the model to operate on that data without supervision.

The unsupervised learning algorithm's main goal is to group or categorize the unsorted information based on similarities, patterns, and differences. Machines are told to discover hidden patterns in the input dataset.

Clustering

When we want to locate the inherent groupings in data, we apply the clustering approach. It is a method of clustering things so that the objects with the highest similarities remain in one group and have few or no similarities with the objects in other groups. A clustering algorithm example is categorizing clients based on their purchase behavior.

Association

Association rule learning is an unsupervised learning approach used to discover interesting relationships between variables in a big dataset. The primary goal of this learning method is to determine the dependency of one data item on another data item and map those variables accordingly in order to maximize profit. This method is mostly used in Market Basket analysis, Web usage mining, continuous production, and other similar applications.

3.1.3. Semi-Supervised Learning

Semi-supervised learning is a sort of machine learning algorithm that falls between between supervised and unsupervised learning. It bridges the gap between Supervised (with labeled training data) and Unsupervised (with no labeled training data) learning methods by combining labelled and unlabeled datasets throughout the training phase.

Although semi-supervised learning acts on data with a few labels and is the middle ground between supervised and unsupervised learning, it is largely unlabeled data. Labels are expensive, yet for corporate purposes, companies may only need a few labels. It differs from supervised and unsupervised learning in that they are dependent on the presence or lack of labels.

To address the shortcomings of both supervised and unsupervised learning algorithms, the notion of semi-supervised learning is presented. The basic goal of semi-supervised learning is to use all available data efficiently, rather than only labelled data as in supervised learning. Initially, related data is grouped using an unsupervised learning technique, and it then aids in labeling the unlabeled data. It is because labeled data is more expensive to acquire than unlabeled data.

With an example, we may visualize these algorithms. Supervised learning occurs when a student is supervised by an instructor both at home and at college. Furthermore, unsupervised learning occurs when a learner self-analyzes the same subject without assistance from the teacher. In semi-supervised learning, the user must modify oneself after evaluating the same subject under the supervision of a college teacher.

3.1.4. Reinforcement Learning

Reinforcement learning is a feedback-based method in which an AI agent (a software component) explores its surroundings autonomously by striking and trailing, taking action, learning from experiences, and improving its performance. Because the agent is rewarded for every good activity and penalized for every negative action, the purpose of the reinforcement learning agent is to maximize the rewards.

There is no labelled data in reinforcement learning, as there is in supervised learning, and agents learn only from their experiences.

The reinforcement learning process is comparable to that of a human being; for example, a youngster learns new things via his daily encounters. Playing a game is an example of reinforcement learning, where the Game is the environment, an agent's movements at each step create states, and the agent's aim is to acquire a high score. The agent is given feedback in the form of punishments and incentives.

Because of how it works, reinforcement learning is used in a variety of domains including game theory, operations research, information theory, and multi-agent systems.

The Markov Decision Process can be used to formalize a reinforcement learning problem (MDP). In MDP, the agent continually interacts with the environment and performs actions; the environment replies and develops a new state with each action.

3.2. Machine Learning system difference between Research and Deployment

Section from 3.2 to 3.4 are mostly based on [11].

The application of Machine Learning in reality is relatively new, because most people know about Machine Learning through classroom, personal projects or reading scientific articles/papers. However, in reality, a Machine Learning system is very different.

As shown in **Figure 3-1**, the code for the Machine Learning algorithm takes up a very small proportion in real-world applications, as when creating a product, many other factors such as data, hardware, management, analysis, and product tracking need to be

considered. In addition, particularly paying attention to and satisfying the needs of customers.

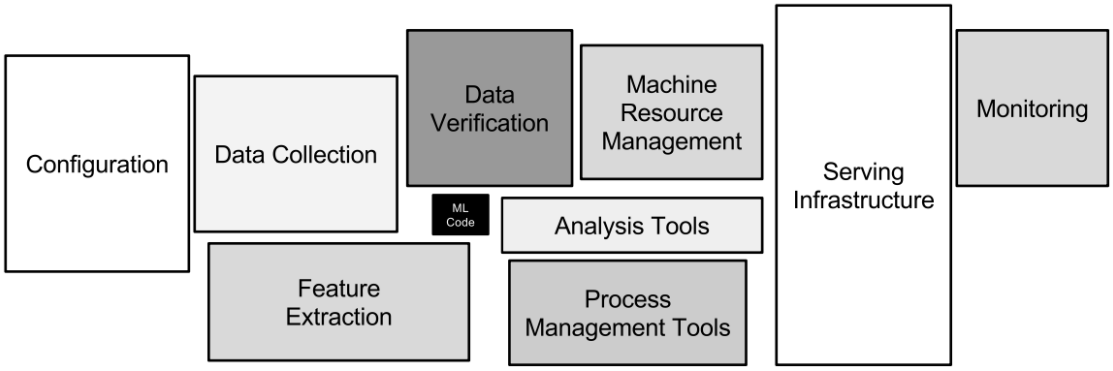


Figure 3-1. Machine learning system in real-world

According to [11], **Table 3-1** shows the comparison between machine learning in research stage and a machine learning product.

Table 3-1. Key differences between ML in research and product

	Research	Product
Requirement	The best and up to date algorithms on a benchmark data set	Many algorithms are based on customer requirements
Computational priorities	Fast training, high throughput	Fast training, high throughput
Data	Static	Constantly shifting
Fairness	Often not a focus	Must be considered
Interpretability	Often not a focus	Must be considered

Requirement

In research and academy, researchers tend to create more complex Machine Learning models than industrial products, with the goal of achieving the best results. However, for a real-world product, implementing a Machine Learning system requires many other components. As mentioned and shown in **Figure 3-1**, it is necessary to consider the hardware system, whether the hardware can meet the demands of a complex Machine Learning algorithm. This is entirely based on the requirements and customer's facility.

Computational priorities

When starting to design a Machine Learning model for practical use, many people often make a common mistake of focusing too much on the algorithm and not considering other factors such as deployment, operation, and maintenance.

During research, the research team may train many different machine learning models and get different results. In the research and academic environment, training time

and throughput are two important factors to consider. However, when bringing a product to the consumer, what they care about is the time it takes to see results.

Data

In research and academy environments, data used is often standardized, cleaned, filtered, and noise-free. It is a static source of data that has been recorded previously, making it easier to process for machine learning models.

However, in real-world applications, data sources are dynamic and constantly changing over time, with noise and lack of clear structure. Each real-world project has different data sources with noise and different structures.

As shown in [11], **Figure 3-2** illustrates the difference in the impact of data on machine learning systems between research and real-world environments.

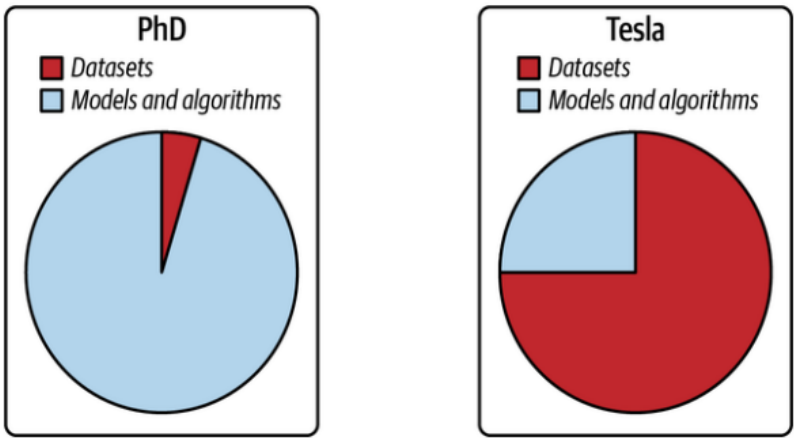


Figure 3-2. The impact of data on machine learning systems

Fairness

It's possible that you or someone you know may have been affected by biased mathematical algorithms without realizing it. For example, a loan application may be denied based on the applicant's zip code, which can reflect biases related to their socioeconomic background. Similarly, a resume may be ranked lower due to the spelling of the applicant's name, and a mortgage may have a higher interest rate because it takes credit scores into account, which can favor wealthy individuals and disadvantage those who are less well-off. Other examples of biases in machine learning can be found in predictive policing algorithms, personality tests used by employers, and college rankings.

Interpretability

In early 2020, Professor Geoffrey Hinton, who won the Turing Award, brought up a controversial topic about the significance of interpretability in machine learning systems. He asked, " Suppose you have cancer and you have to choose between a black box AI surgeon that cannot explain how it works but has a 90% cure rate and a human surgeon with an 80% cure rate. Do you want the AI surgeon to be illegal?".[12]

When this question was posed to a group of 30 technology executives [11], half of them chose the human surgeon while the other half preferred the highly effective but unexplainable AI surgeon.

Many people are comfortable using technology like a microwave without understanding how it works, but they don't feel the same way about AI, especially when it makes important decisions that affect their lives.

Currently, most machine learning research is evaluated based on performance alone, so there is not much incentive for researchers to focus on interpretability. However, interpretability is crucial for users to trust and detect potential biases in a model, as well as for developers to improve and debug it. Despite this, as of 2019, only 19% of large companies were working on making their algorithms more understandable. [13]

3.3. Requirements for a machine learning system

It is important to understand the specific requirements that a Machine Learning (ML) system needs to meet. These requirements can vary depending on the specific application or use case, but generally speaking, an ML system should have certain key characteristics such as reliability, scalability, maintainability, and adaptability. In this discussion, we will delve into each of these concepts in more detail.

Reliability

It is essential that the system maintains its ability to carry out its intended function at the desired level of performance, even in challenging circumstances such as hardware or software failures, or human errors.

In contrast to traditional software systems, ML systems may fail silently, meaning that users may not be aware of the failure and may continue to use the system as if it were functioning correctly. One example of this is using Google Translate to translate a sentence into an unknown language, it may be difficult for the user to determine if the translation is inaccurate.

Scalability

An ML system can increase in complexity over time. For example, last year a logistic regression model that only required 1 GB of RAM on Amazon Web Services (AWS) was used, but this year a neural network with 100 million parameters that requires 16 GB of RAM to make predictions was implemented.

Your ML system can grow in terms of the amount of traffic it receives. When you first deployed your ML system, it only received 10,000 prediction requests per day. However, as your company's user base expands, the number of prediction requests your ML system receives varies between 1 million and 10 million per day.

An ML system may increase in the number of models it uses. Initially, it may only have one model for a specific task, such as identifying trending hashtags on social media.

However, as time goes on, more features may be added, leading to the addition of additional models for tasks such as filtering out offensive content or identifying automated tweets. This is particularly common in ML systems that serve enterprise clients. At first, a startup may only have one model for one customer, but as they acquire more clients, they may have a unique model for each customer.

No matter how your ML system expands, there should be effective methods for managing that expansion. When considering scalability, many people focus on adjusting the resources available, such as increasing or decreasing the resources as needed.[11]

Maintainability

Many individuals will be working on an ML system. They are machine learning (ML) engineers, DevOps engineers, and subject matter experts (SMEs). They may have very diverse backgrounds, use quite different programming languages and tools, and control distinct aspects of the process.

It is important to arrange the tasks and set up the infrastructure in a way that allows individuals from different backgrounds and expertise to work efficiently using the tools they are familiar with, rather than one group imposing their tools on others. The work should be well-documented, versioned and models should be easily reproducible, so that even when the original contributors are not present, others can understand and continue their work. Collaboration should be encouraged to solve problems and avoid blame-shifting when issues arise.

Adaptability

The system should have the ability to adapt to changes in data and business needs by identifying areas for improvement and making updates without disrupting service. Since ML systems involve both code and data, which can change quickly, they must be able to evolve rapidly. This is closely related to maintainability, and will be discussed further in section 3.6.4.

3.4. Building Machine learning system workflow

An ML system requires ongoing maintenance and improvement, as it is a process that is ongoing and never truly finished. Once the system is implemented, it needs to be constantly monitored and updated.

The diagram in Figure 2-2 presents a simplified view of the repetitive process of creating and maintaining ML systems from the point of view of a data scientist or an ML engineer.

Later chapters will dive deeper into what each of these steps requires in practice. Here, let’s take a brief look at what they mean.

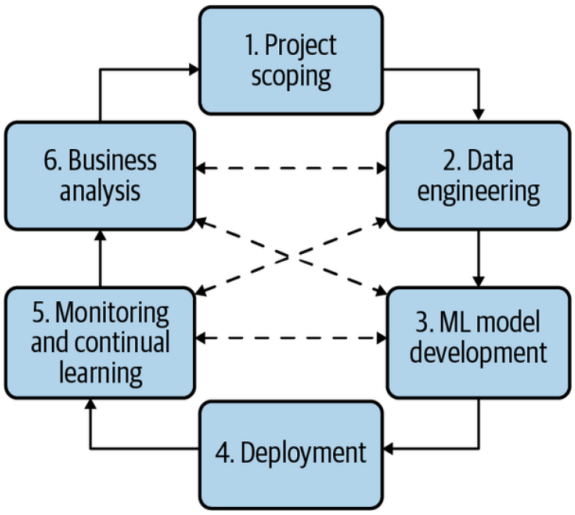


Figure 3-3. Workflow

- Step 1: The initial phase of the project involves defining the scope, setting goals and objectives, and identifying any limitations or constraints. It's important to involve relevant stakeholders and to properly allocate resources.

- Step 2: A vast majority of ML models today learn from data, so developing ML models starts with engineering data. We will discuss the fundamentals of data engineering, which covers handling data from different sources and formats. With access to raw data, we'll want to curate training data out of it by sampling and generating labels.

- Step 3: The next step use the initial training data to identify relevant features and create initial models based on those features. This step requires a deep understanding of ML and is typically covered in ML courses.

- Step 4: : Once a model is created, it needs to be made available to those who will use it. Developing an ML system is similar to writing, it is never truly finished, but there comes a point when it needs to be released.

- Step 5: Once the model is deployed and being used, it needs to be regularly monitored for any decline in performance and updated to adapt to new conditions and evolving requirements.

- Step 6: The performance of the model must be measured against the objectives and constraints set out at the beginning of the project, and the results should be used to identify new opportunities or discontinue unproductive efforts. This process is closely linked to the initial planning phase.

3.5. Machine Learning in Research

3.5.1. Prepare dataset

Preparing data is a crucial process in the research of machine learning algorithms. It takes a lot of time and effort to prepare a large amount of data, but an inappropriate system configuration will waste human resources, money, and time. Or the data prepared

is not good will also drag the learning results of the machine to not achieve the desired results.

The data obtained in reality is very chaotic, complex, and tends to be unpredictable as to whether there are any errors in the data collection process. Therefore, we need to know how to prepare data well before training the machine learning model.

Sampling, also known as taking a sample or subdividing, is the most commonly overlooked process in classroom instruction. Sampling is widely applied in the development cycle of a machine learning system, such as taking a small portion of real data to create a training dataset, dividing a dataset into smaller subsets for training, evaluating, and testing models, etc. However, in this section, we will focus on the process of sampling to create data for training.

In most cases, accessing and collecting real data faces many obstacles such as complying with information security conditions and the scarcity of data. Therefore, to test whether a new machine learning model can meet the desired results, we need a solution and that is to extract a certain amount of data from the closest-to-reality dataset. This will help us have a general view of a machine learning algorithm quickly and with minimal resource waste of the business.

Sampling methods are divided into two groups, non-probability sampling and probability sampling.

3.5.1.1. Nonprobability Sampling

Convenience sampling

A convenience sample is a type of non-probability sampling method where the sample is taken from a group of people easy to contact or to reach; for example, standing at a mall or a grocery store and asking people to answer questions. This type of sampling is also known as grab sampling or availability sampling. There are no other criteria to the sampling method except that people be available and willing to participate. In addition, this type of sampling method does not require that a simple random sample is generated since the only criterion is whether the participants agree to participate. [14]

Snowball sampling

This is a sampling approach in which existing individuals refer new subjects to be recruited for a research project.

For example, if you are researching the level of customer satisfaction among members of an elite country club, collecting primary data sources will be extremely difficult unless a member of the club agrees to have a direct conversation with you and provides contact information for the other members of the club.

This sampling strategy entails a main data source designating other possible data sources who will be allowed to participate in the research investigations. The snowball

sampling approach is entirely reliant on referrals, which is how a researcher generates a sample. As a result, this approach is also known as the chain-referral sampling method.

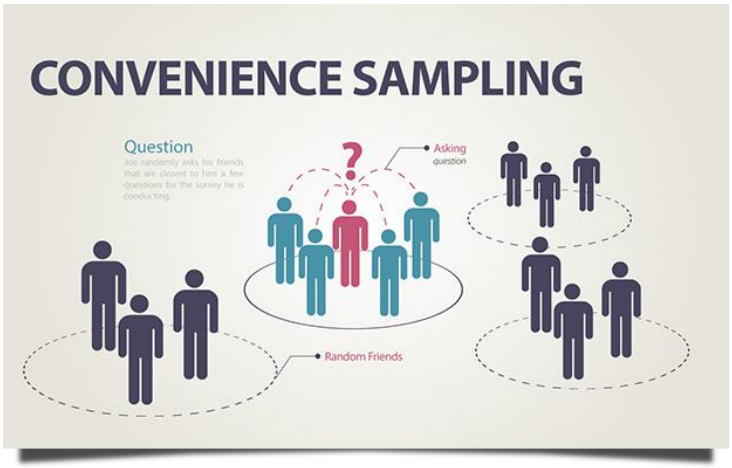


Figure 3-4. Convenience sampling

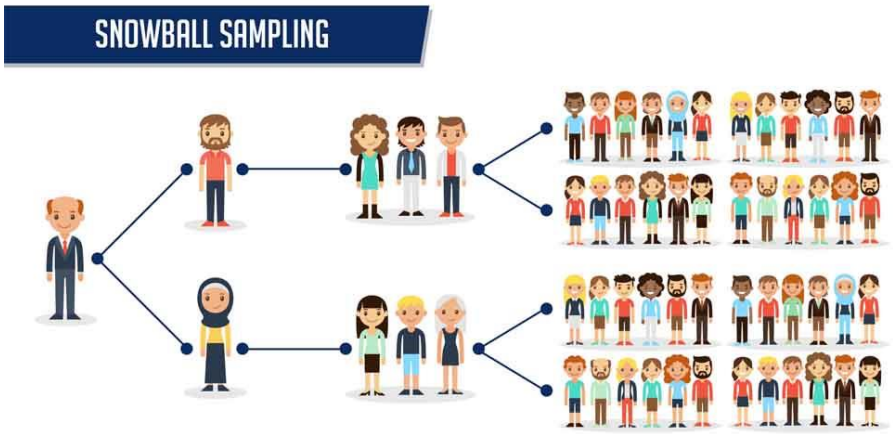


Figure 3-5. Snowball sampling

Judgement Sampling

Judgmental sampling, also known as purposive or authoritative sampling, is a non-probability sampling technique in which sample members are chosen solely based on the researcher's knowledge and judgment. Because the researcher's expertise is used to create a sample in this sampling technique, the results obtained are likely to be very accurate with a small margin of error.

The method of selecting a sample via judgemental sampling entails the researchers carefully selecting and selecting each individual to be included in the sample. Because the sample members are not chosen at random, the researcher's knowledge is essential in this sampling procedure.

Quota Sampling

Quota sampling begins by segmenting a population into mutually exclusive sub-groups. Then, based on a predetermined proportion, judgment is employed to select topics or units from each segment. For example, an interviewer may be instructed to

sample 200 females and 300 males aged 45 to 60. This means that individuals can specify who they want to sample (targeting).

The technique is non-probability sampling as a result of the second phase. Non-random sample selection occurs in quota sampling, which can be incorrect. To save time, interviewers may be tempted to interview persons on the street who appear to be the most helpful, or they may choose to employ unintentional sampling to ask those nearest to them. The issue is that these samples may be biased in ways that are difficult to quantify or compensate for. For example, if interviewers ask the first person they see, they may oversample tall respondents (who are more visible from a distance), resulting in an overestimation of average income. This non-random aspect raises questions about the true nature of the sample.

Figure 3-6. Quota Sampling

3.5.1.2. Probability Sampling

Random sampling can save time and resources such as money and personnel. However, for models that require high reliability, we need to use random sampling methods. In terms of theory and practice, the normal distribution is the most important distribution in statistics. Because there are many fluctuations in nature, industry, and society that follow (or almost follow) this normal distribution. Therefore, random sampling will help samples retain the original characteristics of real data. Here are some methods of random sampling.

Simple random sampling

Simple random sampling, where data has an equal probability of being selected for the sample. This is the simplest method in the random sampling group, and it is very easy to implement. However, if data has a low occurrence rate, for example, out of 10,000 data, there is one data that belongs to group Z, with an occurrence rate of 0.01%, and selecting 1% of the 10,000 data to create the sample, the probability of group Z appearing in the sample becomes lower.

Stratified sampling

This technique divides each individual in the population into smaller groups based on similarity, meaning individuals in the same group will be consistent with each other in

some aspect and will not be similar to other groups in that aspect. And we will randomly select individuals from each group. In this method, we need prior information about the population to create the subgroups

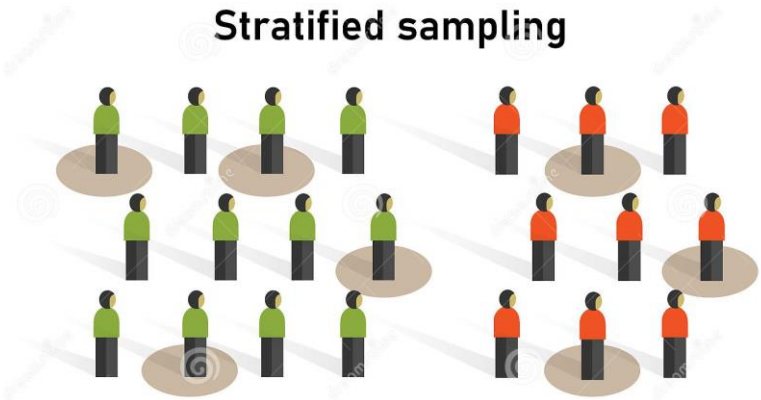


Figure 3-7. Stratied sampling

Weighted sampling

Weighted sampling is a method that assigns a weight to each group of similar data in the dataset, corresponding to the probability of that group being selected. This method requires knowledge of the domain. For example, if there are three groups A, B, and C, which make up 50%, 30%, and 20% of the actual data, respectively, with group C having the lowest proportion but we want it to appear more in the sample, we can choose weights as 0.3, 0.3, and 0.4.

Figure 3-8. Weighted sampling

Reservoir sampling

Reservoir sampling is a very good technique that is often used for data streams, which is then projected on the time axis, the data collected is called a set S with a very large number of n. Reservoir sampling will select k elements to create a sample.

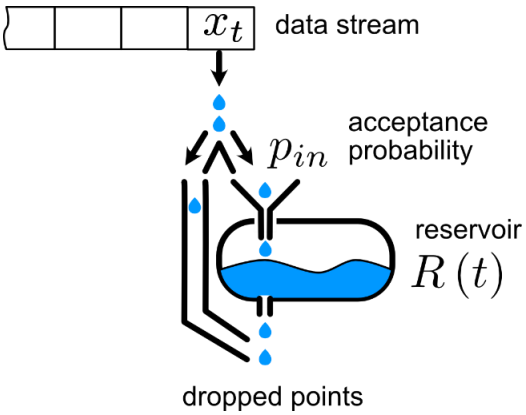


Figure 3-9. Revisor sampling

3.5.2. Feature Engineering

Feature extraction is the process of transforming the original data set into a set of attributes (features) that can better represent the original data, creating conditions to solve problems more easily, compatibility with specific prediction models, and also improving the accuracy of the current prediction model. According to Chip Huyen [8], the feature extraction stage of data is the most important stage, even top algorithms will give poor results if the appropriate features are not extracted. In the case of the current project, the hardware reading and data transmission has not seen cases such as missing data or non-compliant data formats, so in this topic, only scaling is presented and used.

3.5.2.1. Denoising

The data collection system using sensors, such as IoT systems, all have noise, noise from the hardware of the device or noise caused by the environment, which causes a deviation in the sensor value.

Because there are many types of noise like this, it requires specialized knowledge in the specific application to be able to filter the noise accurately. There are two forms of filtering noise, hardware filtering such as using a capacitor, ... or software filtering.

According to two articles [15] and [16], heart diseases are usually expressed in the frequency range from 0.5Hz to 80Hz, while some special diseases are in the frequency range to 100Hz or higher. Combining the two articles above, there is a need for a high-pass filter to reduce the signal levels with frequencies below 0.05Hz. A band-stop filter is used to remove the noise of the Vietnamese power grid, frequency 50Hz. For frequency ranges above 100Hz, specific cases need to be applied for appropriate filtering.

Finally, because human skin resistance changes depending on environmental conditions and the body's condition, ECG signals can vary as shown below. Band-pass and notch filters require specialized knowledge, so in this report, we only performed filtering to eliminate baseline wander noise. In addition, according to the article [18], ECG signals taken at a frequency of 250-500Hz are useful for analysis in both the time and frequency domains. However, ECG signals with a sampling frequency of 100Hz cannot be analyzed in the frequency domain. Signals from devices with a frequency of

128Hz cannot be applied with low-pass filters for 100Hz. Similarly, applying a 50Hz notch filter does not result in any change in the signal.

Figure 3-10. Baseline wander noise

3.5.2.2. Scaling

When trying to predict whether someone will purchase a house within the next year, it is important to take into account the data shown in **Table 3-2**. The ages of individuals in the data range from 27 to 40, while their annual incomes range from 50,000 to 150,000. However, when inputting these variables into a machine learning model, it may not understand the significance of the difference between 150,000 and 40. Instead, it may assign more importance to the larger number without considering which variable is more relevant to the prediction. To prevent this, it is crucial to scale the features to be within similar ranges before inputting them into the model.

Table 3-2. Buying house example

ID	Age	Gender	Annual income	Job	Buy
1	27	Female	50,000	Teacher	No
2	40	Male	120,000	Engineer	Yes
3	35	Male	150,000	Doctor	Yes
4	33	Male	100,000	Engineer	No

3.5.3. Labeling

Despite the promise of unsupervised ML, most ML models in production today are supervised, which means that they need labeled data to learn from. The performance of an ML model still depends heavily on the quality and quantity of the labeled data it's trained on. [11]

Hand labels

This is a phase that encounters a lot of difficulties and obstacles. Firstly, manual labeling is very time-consuming, try to imagine classifying and labeling simple objects like **Figure 3-11** for object recognition will be much easier than labeling different types

of bacteria in **Figure 3-12**. In some cases, the labeling phase requires the participation and consultation of experts, and these experts are usually very busy, asking for help is very difficult, and hiring them is very costly.

The second obstacle is when it comes to data that has very important information that we, managers, use data needs to be aware of is data security and privacy. Most real-world data has certain privacy requirements, the higher the privacy requirements, the higher the data security, to use that data source, we need to have a great responsibility. For example, patient medical records are a very high-privacy data source, it notes personal information of each person, we cannot give this data to a third party to participate in labeling or related activities that may reveal information.

The third thing is that manual labeling is very slow, that's the reason why an organization with a labeling data team takes a lot of personnel and time.

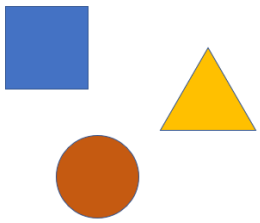


Figure 3-11. Simple objects classification

Figure 3-12. Bacteria classification

Natural labels

With the help of machines, we can reduce the workload of manual labeling for supervised models. Although this method requires a system that has already been operating and will output labeled data. Humans only need to edit those labels. For example, Google Maps will work by suggesting a route to the user, with the estimated time that Google Maps has calculated before. If the user takes that route, it will measure the time taken, after the trip is over, we have the estimated time and the actual time the user took.

3.5.4. Class imbalanced

Class imbalance is a common issue in classification tasks. The imbalance in the ratio of labels in the data set is often very large. Machine learning models, particularly deep learning, work well when the ratio of labels is balanced or not too different, but the performance will not be good when there is an imbalance between labels. **Figure 3-13** illustrates two cases: balanced data set and imbalanced data set.

Class imbalance can cause problems in the learning process for the reasons listed below.

The first reason is that labels with less data (the minority classes), if the model is only trained a few times on these labels, it will produce poor results or if the model has no chance to learn the minority classes, the model will inevitably make errors in the test set and in reality.

The next reason is that the imbalance in the data set causes the model to get stuck in suboptimal solutions, that is, the model will learn and exploit some simple features that are represented in the majority set, which will greatly affect the models that find the extreme values such as gradient descent.

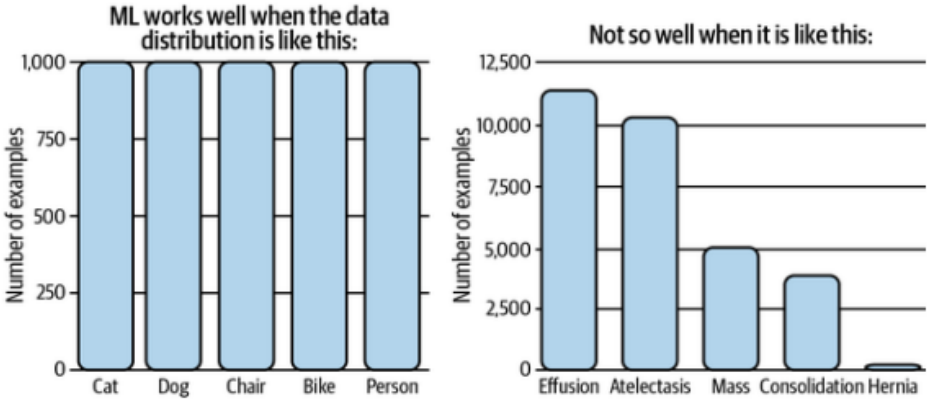


Figure 3-13. Class imbalance

The final reason is that class imbalance will make the cost of a prediction of the minority class higher. A simple example, suppose we have a dataset on classifying X-ray images of lung disease patients, we will get a smaller number of rare images than common diseases. Therefore, when the model makes a wrong classification, it will have a significant impact, even life-threatening.

There are many ways to limit the impact of class imbalance on the results of machine learning models, such as selecting appropriate metrics, or resampling. Some methods used in the project will be listed in the following sections.

3.5.4.1. Resampling

Resampling, also known as data re-sampling, is an important technique applied to address class imbalance in the data set. It brings high accuracy to the machine learning model. There are two types of resampling, namely bootstrap and cross-validation.

It is important to note that the data pre-processing step, in this case, feature extraction as presented in section 3.4.2, is applied to all three sets, train, evaluation and test set. However, when performing resampling, the evaluation set and test set should not be resampled. The aim of applying machine learning model is to handle real-world data. Therefore, the test set and evaluation set should remain unchanged after they were collected and pre-processed.

Bootstrap

F or the bootstrap technique, samples in the dataset are randomly selected, and then the number of those samples is replicated, which means that a data sample may appear multiple times. Alternatively, samples that are not selected may be discarded.

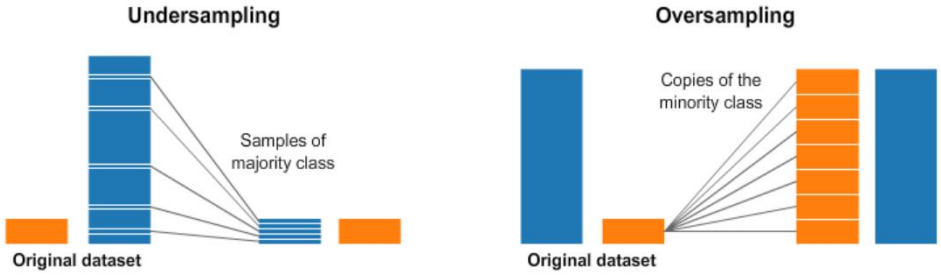


Figure 3-14. Bootstrap

Even though the data set is balanced, bootstrap also has its own disadvantages. Undersampling, removing samples that are not selected from labels that dominate large in the data set, this can risk making the data set lack important information causing underfitting.

Oversampling, duplicating labels with low weight, a data sample will appear multiple times which can cause overfitting. To counteract the above disadvantages, there have been some improvements such as applying clusters, ... and we will explore these improvements through visual examples below.

Use the method of the sciket-learn library to create a data set with class 0 having 90 samples, class 1 having 10 samples:

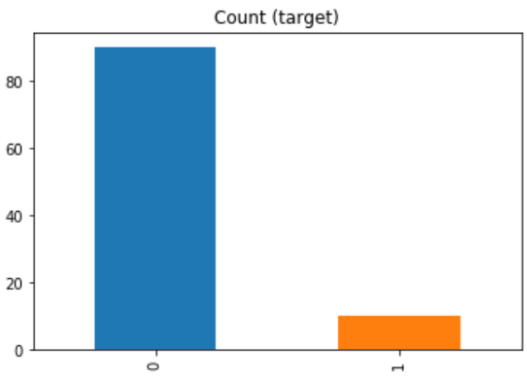


Figure 3-15. Bootstrap example data

As there are 20 features in X, we can't visualize a 20-dimensional space to represent X visually. Therefore, we use the Principal Component Analysis (PCA) method to reduce the dimension of X and draw the scatter plot:

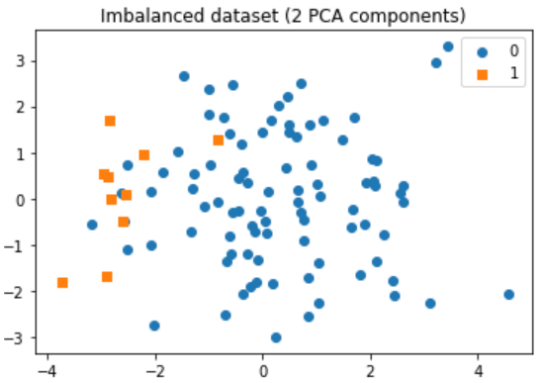


Figure 3-16. 2D scatter plot

Next, we will plot the data points after applying under-sampling to class 0.

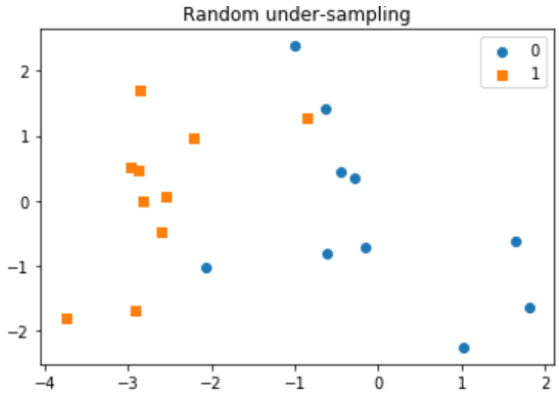


Figure 3-17. Example data after under-sampling

We can clearly see that the number of Xs has decreased significantly, as previously mentioned, this can cause important information loss in the features of the data, affecting the results of the model.

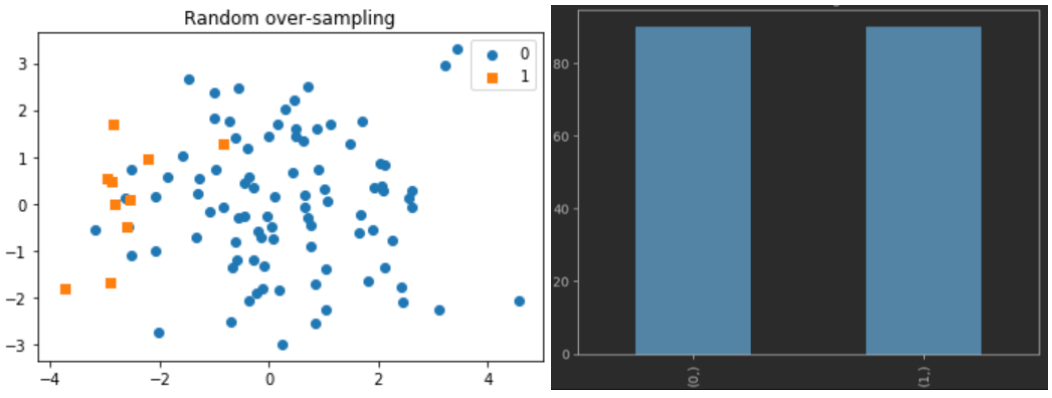


Figure 3-18. Random Over-sampling

As such, random over-sampling only increases the number of minority labels, without changing the characteristics of the new data.

Currently, there are many solutions commonly used with under-sampling and over-sampling, the content below lists the theory of the solutions used in this project.

- Under-sampling: Instance Hardness Threshold

Class overlap occurs when a region in the sample space contains samples from another class. Overlapping samples have a high degree of difficulty. When there is an imbalance and overlap in the data set, the classification problem becomes more difficult.

A library built based on the theory of Instance Hardness Threshold [18], mentioned above, exists. Samples classified with low probabilities will be removed from the data set. Then, the model will be built on the undersampled data.

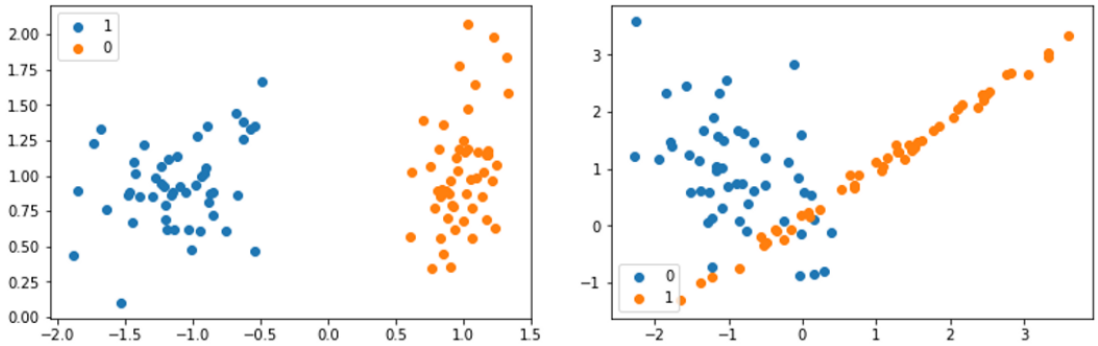


Figure 3-19. Class overlap between class 0 and 1

- Under-sampling: Edited Nearest Neighbours

The ENN algorithm, created by Wilson in 1972, operates by identifying the k closest points to each observed data point. It then examines whether the majority class among these k closest points is the same as the class of the observed point. If the majority class differs from the observed point's class, both the observed point and its k closest neighbors are removed from the dataset. By default, ENN uses k=3 nearest neighbors in its calculation.

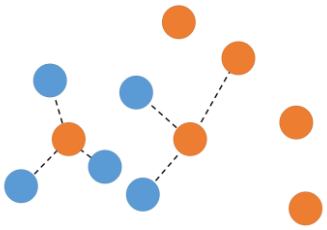


Figure 3-20. Undersampling Edited Nearest Neighbours

- Under-sampling: Repeated Edited Nearest Neighbours

This method repeats ENN algorithm many times as we want.

- Over-sampling: SMOTE

SMOTE stands for Synthetic Minority Oversampling TEchnique, or SMOTE for short. This technique was described by Nitesh Chawla, et al. in their 2002 paper named for the technique titled “SMOTE: Synthetic Minority Over-sampling Technique.”[19]

The SMOTE algorithm aims to balance class imbalance by generating synthetic samples. It does this by selecting a random minority class example and identifying k of

its nearest neighbors (usually k=5). A new sample is then created by choosing a random point along the line connecting the chosen example to one of its k-nearest neighbors in the feature space.

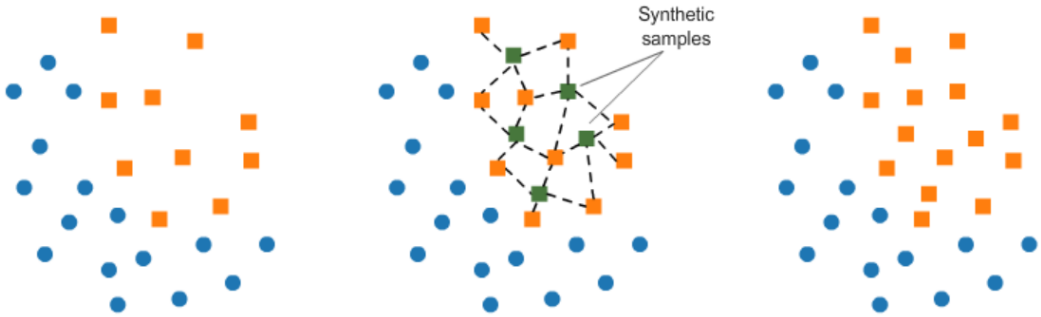


Figure 3-21. Over-sampling SMOTE

- Over-sampling: ADASYN

This technique is a refined version of SMOTE. It operates in a similar way to SMOTE, but with a slight improvement. After generating synthetic samples, it introduces small random variations to the points, making them more realistic. This means that the samples are not all perfectly correlated with the original data and have a bit more variation, or are slightly scattered.

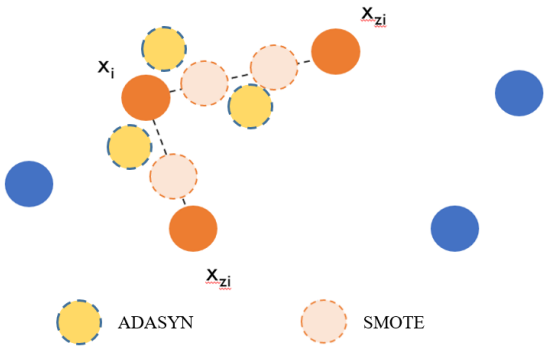


Figure 3-22. ADASYN vs SMOTE

- Combine over-sampling and under-sampling

As previously stated, oversampling techniques involve replicating or generating artificial samples within the minority class, while undersampling techniques involve removing or combining examples from the majority class. Either approach can be successful on its own, but a combination of both methods may result in even greater effectiveness.

Cross Validation

- K-fold Cross validation

Cross-validation is a technique used to evaluate the performance of machine learning models on a limited dataset by dividing it into smaller groups called "folds". The

number of folds is determined by the "k" parameter. This method is commonly referred to as "k-fold cross-validation".

For example, if k=10, it is known as "10-fold cross-validation." The goal of cross-validation is to estimate how the model will perform on new unseen data. It is a popular method because it is easy to understand and usually provides a more accurate estimate of the model's performance compared to other methods, such as train/test splits.

The general procedure is as follows:

1. Shuffle the dataset randomly.
2. Split the dataset into k groups
3. For each unique group:
    a. Take the group as a hold out or test data set
    b. Take the remaining groups as a training data set
    c. Fit a model on the training set and evaluate it on the test set
4. Retain the evaluation score and discard the model
5. Summarize the skill of the model using the sample of model evaluation scores

It is important to note that each data point is assigned to a specific group and remains in that group throughout the process. This means that each data point is used as a holdout set once and used to train the model k-1 times.

- Stratified k-fold Cross validation

Stratified k-fold cross validation is a variation of regular k-fold cross validation that ensures that the proportion of different classes in the target variable is maintained in each fold, as it is in the entire dataset.

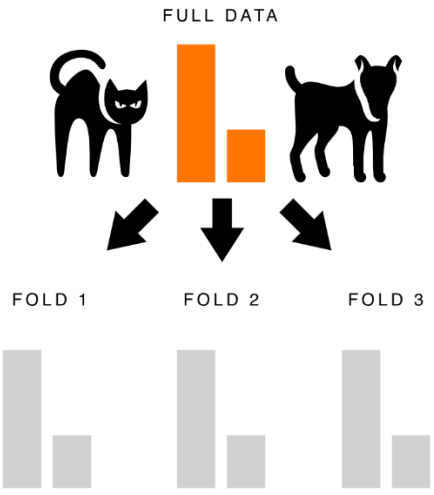


Figure 3-23. asd asd

For instance, if we are building a model to classify pictures of cats and dogs shown in **Figure 3-23**, and the dataset consists of 75% cat pictures and 25% dog pictures, using stratified k-fold cross validation ensures that each fold created has a similar 75/25 ratio.

3.5.5. Machine Learning model selection

We have learned about the types of machine learning algorithms through section 3.1. To be able to choose a machine learning model, we need to know the input data and the output results.

In the scope of this graduation project, the input data is a series of numbers representing an ECG, and the outcome we need to know is whether this ECG has hidden signs of illness or not. Therefore, the model we need is Classification model as mentioned in section 3.1. Science and technology development is very fast, every day there are many research results in general sciences and artificial intelligence in particular, using old algorithms will not bring desired results.

Once we know the type of model we need, the next step is to research the list of the best models (state of the art) currently available. There are many sources to refer to, such as reputable communities: Kaggle, stackoverflow, medium, paperswithcode, physionet, etc.

Based on the MIT-BIH Arrythmia data published on Physionet, we will find a Kaggle post. A dataset extracted from the MIT-BIH Arrythmia dataset using the method of [10] with over 150 notebooks, the machine learning models used are listed in the table below.

The table above lists some algorithms that have been used to classify ECG signals, along with the resources that the model needs to be able to calculate and produce results in the desired time (usually milliseconds) are CPU or GPU.

We know that basic classification models such as Random Forest or Support Vector Machine can be used to models that use neural networks. Because the goal of this project is to find a solution for ECG classification on IoT devices, IoT devices are small devices that come with requirements such as energy saving and low cost, so models that run well on the CPU are prioritized in the testing process.

The remaining algorithms are PrunedDTW, FastDTW, Random Forest, Support Vector Machine, TinyML, Catboost, and XGboost. Ignoring the most basic algorithm, Random Forest and SVM, as they will produce the worst results. The two most suitable models are Catboost and XGboost, which are built based on the boosting algorithm and are popular in real-world applications. In addition, boosting is a basic knowledge of Machine Learning that should be learned when still sitting in a university lecture. To develop complex things, we need to grasp basic knowledge, that's why in this graduation project will use one of the two algorithms Catboost or XGboost.

Table 3-3. Models

Model name	CPU	GPU
PrunedDTW	X	X
FastDTW	X	X
CNN LSTM Attention		X
Dense Neural Network		X
Multi-head Attention		X
Dilated CNN		X
Multi Layer Perceptron		X
Random Forest	X	X
Support Vector Machine	X	X
ANN		X
TinyML	X	X
Catboost	X	X
XGboost	X	X

To eliminate an algorithm, we need to consider other aspects, specifically the ability to build on multiple platforms, specifically the ability to install the environment on embedded computers. When building machine learning systems on electronic devices, we need hardware that can process and calculate machine learning models, embedded computers will assume this role.

There are two ways to install a Python environment on an embedded computer, the first is to install directly, the second is to package it with Docker. Because in the experimental phase, we do not know which method runs effectively and less resource-consuming, we will use models that can be installed in both ways. Catboost does not support direct installation of the environment on ARM chip-based machines, while XGboost can be easily installed on Windows, Linux, Mac OS, or embedded computers. Therefore, XGboost will be used in this project.

3.5.6. Validation

After selecting the machine learning model, the next important step is to select appropriate metrics to evaluate the effectiveness of the model to determine whether it meets our expectations or not.

3.5.6.1. Accuracy

The most simple and commonly used method is accuracy. This evaluation method simply calculates the ratio between the number of correctly predicted points and the total number of points in the test data set. For example, we can count that there are 6 data points that are correctly predicted out of a total of 10 points. Therefore, the accuracy of the model is 0.6 (or 60%).

3.5.6.2. Confusion matrix

However, using accuracy as the only method of evaluation only tells us the percentage of data that is classified correctly and does not indicate specifically how each

class is classified, which class is most accurately classified, and which class data is often misclassified into other classes. To evaluate these values, we use a matrix called a confusion matrix.

In essence, a confusion matrix shows the number of data points that actually belong to a class and are predicted to fall into that class.

A confusion matrix is a square matrix with the size of each dimension equal to the number of data classes. The value at row i, column j is the number of points that should actually belong to class i but are predicted to be in class j. Thus, by looking at row 1 (0), we can see that out of the four points that actually belong to class 0, only two points are classified correctly, while the other two points are misclassified into class 1 and class 2.

We can immediately conclude that the sum of all elements in this matrix is the number of points in the test set. The elements on the diagonal of the matrix are the number of correctly classified points of each data class.

Total: 10	Predicted as: 0	Predicted as: 1	Predicted as: 2	
True: 0	2	1	1	4
True: 1	1	2	0	3
True: 2	0	1	2	3

Figure 3-24. Confusion matrix

3.5.6.3. True/False Positive/Negative

This evaluation method is typically applied to binary classification problems. Specifically, in these two data classes, one class is more important than the other and needs to be predicted accurately. For example, in the problem of determining whether or not there is cancer, it is more important to not miss (miss) than to misdiagnose a negative as positive. In the problem of determining whether or not there is a mine under the ground, it is more important to miss than to alarm many false alarms. Or in the problem of filtering spam emails, it is important to mistakenly put important emails in the trash rather than identifying a spam email as a regular email.

In these problems, people often define the more important class that needs to be correctly identified as the Positive class (P-positive), and the other class is called the Negative class (N-negative). We define True Positive (TP), False Positive (FP), True Negative (TN), False Negative (FN) based on the unnormalized confusion matrix.

Researchers often focus on TPR, FNR, FPR, TNR (R - Rate) based on a normalized confusion matrix. The False Positive Rate is also known as the False Alarm Rate, and the False Negative Rate is also known as the Miss Detection Rate. In a mine detection problem, it is better to have a false alarm than to miss a mine, meaning that we can accept a high False Alarm Rate in order to achieve a low Miss Detection Rate.

	Predicted as Positive	Predicted as Negative
Actual: Positive	True Positive (TP)	False Negative (FN)
Actual: Negative	False Positive (FP)	True Negative (TN)

Figure 3-25. Unnormalized confusion matrix

	Predicted as Positive	Predicted as Negative
Actual: Positive	TPR = TP/(TP + FN)	FNR = FN/(TP + FN)
Actual: Negative	FPR = FP/(FP + TN)	TNR = TN/(FP + TN)

Figure 3-26. Normalized c

3.5.6.4. ROC – AUC

ROC

In some problems, increasing or decreasing FNR and FPR can be done by changing a threshold. For example, when using the Logistic Regression algorithm, the output of the model can be hard classes of 0 or 1, or it can also be values representing the probability that the input data belongs to class 1. When using the sklearn Logistic Regression library, we can get these probability values using the predict_proba() method. By default, the threshold used is 0.5, which means that a data point x will be predicted to fall into class 1 if the value of predict_proba(x) is greater than 0.5 and vice versa.

If now we consider class 1 as the Positive class and class 0 as the Negative class, the question is how to increase the false positive rate (FPR) in order to decrease the false negative rate (FNR)? Note that increasing FNR is equivalent to decreasing TPR because the sum of them is always 1.

A simple technique is to lower the threshold value from 0.5 to a smaller number. For example, if the threshold is chosen to be 0.3, then any points predicted with a probability greater than 0.3 will be predicted to belong to the Positive class. In other words, the proportion of points classified as Positive will increase, resulting in an increase in both False Positive Rate and True Positive Rate (column 1 in the matrix increases). As a result, both FNR and TNR decrease.

On the other hand, if we want to miss less and misclassify more, at some level, such as in the case of identifying spam emails, we need to increase the threshold to a number greater than 0.5. In this case, most of the data points will be predicted to belong to class 0, that is, Negative, and both TNF and FNR will increase, that is, TPR and FPR will decrease.

Thus, for each value of the threshold, we will get a pair of (FPR, TPR). Representing these points (FPR, TPR) on a graph when changing the threshold from 0 to

1 will give us a line called the Receiver Operating Characteristic curve or ROC curve. (Note that the range of threshold values is not necessarily from 0 to 1 in general problems. This range needs to be ensured to have the maximum or minimum TPR/FPR value that it can achieve).

AUC

Based on the ROC curve, we can indicate whether a model is effective or not. An effective model has a low FPR and a high TPR. Another metric used to evaluate the model, which I have used in this project, is called the Area Under the Curve or AUC. This is the area under the pink ROC curve. This value is a number between 0 and 1, with a larger value indicating a better model.

3.5.6.5. F1 Score

In a classification problem where the data of the classes are very different from each other, a metric is often used effectively is Precision-Recall.

First, consider a binary classification problem. We also consider one of the two classes as positive, and the other class is negative.

With a way to determine a class as positive, **Precision** is defined as the ratio of the number of true positive points among those points classified as positive (TP + FP).

Recall is defined as the ratio of the number of true positive points among those points that are actually positive (TP + FN).

A mathematical way, Precision and Recall are two fractions with the same numerator but different denominators:

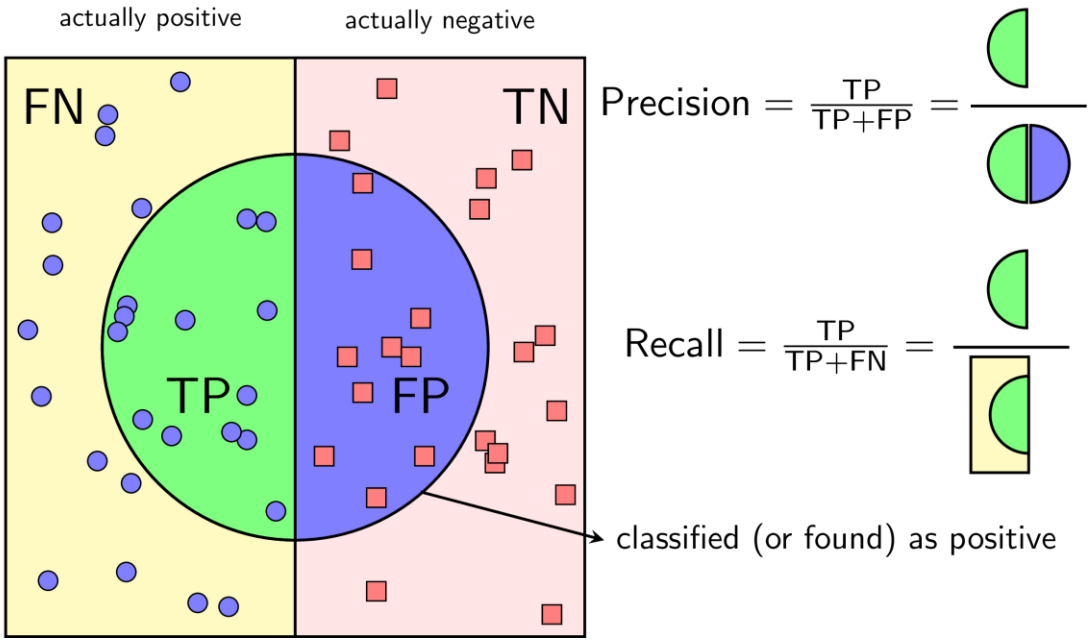


Figure 3-27. Precision – Recall

We can say that TPR and Recall are two equivalent measures. Additionally, both Precision and Recall are non-negative numbers less than or equal to one. A high

Precision means that the points found are accurate. A high Recall means that the True Positive Rate is high, meaning that the rate of missed points that are truly positive is low.

When Precision = 1, all the points found are truly positive, meaning there are no negative points mixed in the results. However, Precision = 1 does not ensure that the model is good, because the question is whether the model has found all the positive points or not. If a model only finds one point that is most certain, it cannot be called a good model.

When Recall = 1, all positive points are found. However, this measure does not measure how many negative points are mixed in. If the classification model classifies all points as positive, then the Recall is certainly 1, but it is easy to recognize that this is a very poor model.

A good classification model is one that has both high Precision and Recall, that is, the closer to one, the better. **F1-score**, is the harmonic mean of precision and recall (assuming that these two measures are not equal):

$$\frac{2}{F_1} = \frac{1}{\text{precision}} + \frac{1}{\text{recall}} \text{ hay } F_1 = 2\frac{1}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}} = 2\frac{\text{precion} \cdot recall}{\text{precision} + \text{recall}}$$

The F1-score has a value within (0,1]. The higher the F1, the better the classifier. When both recall and precision are equal to 1 (the best possible), F1 = 1.

3.6. Machine Learning model deployment

3.6.1. Feature engineering

This is the data pre-processing process, in which the electrocardiogram is recorded by the sensor and sent 16 times per second.

The output data from the device will go through all the steps in feature engineering as outlined in section 3.5.2. Because the requirement for the input data when training and the data to be put into the test must have the same format, including steps:

- Denoising
- Scaling

3.6.2. Database

Database is one of the basic knowledge when working with computer software. Due to the increasing amount of data generated, the need for managing that data arises. And databases have met and continue to meet the needs of people.

According to theory, a database is a set of organized data, commonly stored in computer systems. Simply put, it is information stored in a computer. There are two types of databases: relational and non-relational.

For a relational database, data is structured into tables, each table consisting of rows and columns, and each data added must ensure the structure specified by that table. Data is written using the Structured Query Language (SQL). Examples of relational databases include MS SQL Server, MySQL, Oracle, etc.

A NoSQL database (also known as a non-relational database) is a database that does not require a predefined structure, and each data can have different fields of information (the storage structure is similar to JSON). NoSQL databases do not use query languages. Some examples of NoSQL databases include MongoDB, CouchDB, etc.

To manage these databases, we use database management systems (DBMS). These systems provide us with the necessary tools to interact with the database, and most of the time, we work with the DBMS to manage the database.

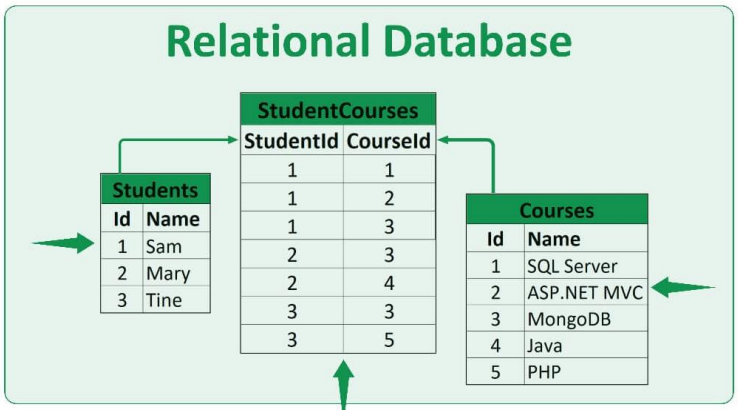


Figure 3-28. Relational Database

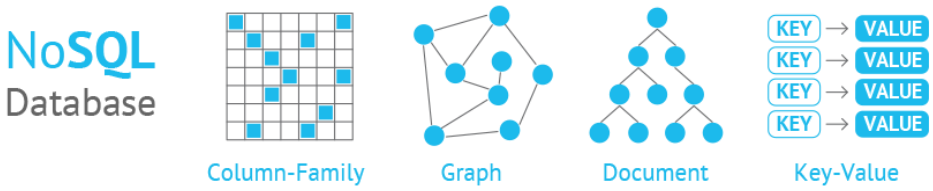


Figure 3-29. NoSQL Database

Besides, there are many other types of databases categorized into NoSQL such as time series database. Time series database is not a new technology, it has been widely applied in the industry. The strong development of hardware has led to the development of software, including database systems, so today many people have access to various types of databases.

Time series databases have some characteristics and architectures that make them different from other types of databases. For example, storing and compressing data according to timestamps, managing the lifecycle of data, summarizing data, the ability to process large amounts of data with many records in real time, and the ability to recognize timestamps when querying data.

3.6.2.1. InfluxDB

InfluxDB is a time series database, built on an open-source database platform developed by InfluxData. It is written in the Go programming language to store and retrieve time series data in fields such as monitoring activity, application data, Internet of Things sensor data, and real-time analysis.

InfluxDB is often chosen in cases where large amounts of data marked by time labels (DevOps monitoring, IoT sensor data, etc.) need to be stored and organized. The main features supported by InfluxDB include:

- Clear and high-performance read-write APIs.
- Plugins that support other data input protocols such as Graphite, etc.
- Queries that are similar to SQL, making it easy for those with a SQL background to use.
- Indexing by tags fields for fast queries.

Continuous queries that automatically calculate aggregated data to make frequent queries more efficient. Finally, InfluxDB has both open-source and enterprise versions.

3.6.2.2. SQLite

SQLite is a relational database management system (DBMS) similar to MySQL, ... The standout features of SQLite compared to other DBMSs are its compactness, lightweight, simplicity, and most importantly, the lack of a server-client model, which eliminates the need for installation, configuration, or startup, so there is no concept of user, password, or permissions in the SQLite Database. The data is also stored in a single file.

SQLite is not typically used with large systems, but for small and medium-sized systems, SQLite is no less functional or fast than other DBMSs. Because it does not require installation or configuration, SQLite is often used in development, testing, and so on, to avoid complications during installation.

The features of SQLite:

- Transactions in SQLite comply with the ACID principles even after system crashes and power outages.
- No configuration required: no setup or administration needed.
- SQLite supports full functionality with advanced capabilities such as single-part indexes, expression indexes, JSON, and common table expressions.
- A complete database is stored in a single, cross-platform file, suitable for use as an application file format.
- Support for terabyte-size databases and gigabyte-size strings.

- Simple, easy-to-use API.
- Fast: in some cases, SQLite is faster than direct I/O file systems.
- Written in ANSI-C.
- Bindings for many other languages available separately.
- Fully open-source and source-code available for 100% inspection.
- Cross-platform: SQLite is available on Android, Linux, Mac, Solaris, Windows, etc. and can be easily ported to other systems.

The main applications of SQLite:

- Database for Internet of Things.
- Application file format.
- Database for web.
- Stand-in for an enterprise RDBMS.

3.6.3. Create package

Why do we need to package? Because during the research process, the source code may not be written according to standards, may not have enough comments or the file structure/name may be confusing. Therefore, it is necessary to go through this step to standardize the source code, reorganize the project structure, etc. Finally, it can be packaged into an app using Docker or run directly on an already installed environment.

3.6.3.1. Configuration

This stage is creating a config file, the purpose is to store information in the form of key: value. The program will read and retrieve this information at some point during the running process. These information may change frequently, therefore for convenience in updating, we should separate it from the code in the program.

There are many formats including toml, yaml, json, ini or writing directly with python module. For example with python module, create a python file with configs as follows:

```
truck = dict(
    color = 'blue',
    brand = 'ford',
)
city = 'new york'
cabriolet = dict(
    color = 'black',
    engine = dict(
        cylinders = 8,
        placement = 'mid',
    ),
    doors = 2,
)
```

Then we can retrieve data:

```
import config
print(config.truck['color'])
```

This method uses the import module in python in combination with a dictionary to access values by key. It can be used in situations where users (those without programming expertise) do not need to edit the config file.

The structure of a config file in ini format is as follows:

```
[SectionOne]
Status: Single
FirstName: Derek
LastName: John
Value: Yes
Age: 30
Single: True

[SectionTwo]
FavoriteColor=Green

[SectionThree]
FamilyName: Johnson
```

A config file in yaml format has the following structure:

```
---
 doe: "a deer, a female deer"
 ray: "a drop of golden sun"
 pi: 3.14159
 xmas: true
 french-hens: 3
 calling-birds:
   - huey
   - dewey
   - louie
   - fred
 xmas-fifth-day:
   calling-birds: four
   french-hens: 3
   golden-rings: 5
   partridges:
     count: 1
     location: "a pear tree"
   turtle-doves: two
```

And finally, the structure of a json file:

```
{
  "squadName": "Super hero squad",
  "homeTown": "Metro City",
  "formed": 2016,
  "secretBase": "Super tower",
  "active": true,
  "members": [
    {
      "name": "Molecule Man",
      "age": 29,
      "secretIdentity": "Dan Jukes",
      "powers": ["Radiation resistance", "Turning tiny"]
    },
  ]
}
```

The two most commonly used formats are YAML and JSON because of their superior advantages. YAML is a superset of the JSON platform, however, there are some differences between them.

YAML has a readable structure, so users can comfortably edit it. But also because of this advantage, there is a disadvantage, for a computer program, the YAML structure is more complex, making the program run slower. And YAML has many other features.

The JSON format has a tight structure, in the form of a dictionary combined with a list including nested dictionaries and nested lists, that is key: value with value can be a different list or dictionary. Because of the advantages of running fast and users not having to pay attention to the config, in this project, I use the JSON format.

3.6.3.2. Docker

Docker is a platform for developers to develop, deploy, and run applications with containers. It allows creating isolated and separated environments to run and develop applications, which are called containers. When deploying to any server, you only need to run the Docker container and the application will run immediately.

It is not just for servers, nowadays deploying artificial intelligence applications on different platforms is relatively difficult, and Docker is the key for most of these cases, it can even run on mini computer like Raspberry 4.

- You can start a container on any system you want.
- Containers can be built and removed faster than virtual machines.
- It is easy to set up a working environment. Just set up the parameters once and never have to install dependencies again.
- If you change machines or have new people joining the project, you just need to give them the config.
- It keeps the workspace cleaner by removing the environment without affecting other parts.

To build a multi-platform application, we use the Docker Image feature. A Docker Image is an immutable file that contains the entire source code, libraries, dependencies, etc. of the application after it has been built. A Docker Image represents an application and a virtual environment to run that application at a specific point in time. Because the advantages of maintaining the state of this environment allows the software to run in a stable and consistent manner, programmers have taken advantage of Docker Images in testing processes.

Because of this nature, Docker Images are read-only and cannot run as an application, and we have a concept called a Container. Containers exist separately from Images, and when a Docker Container is launched, it creates a copy of the Image and adds a Container Layer that allows the modification of the entire copy of the Image.

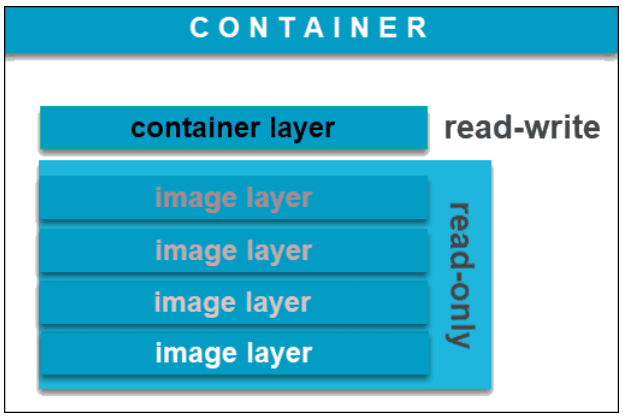


Figure 3-30. Docker contrainer

We can create unlimited Docker Images, every time we change the state of an Image, Docker will create a new Image by copying the old Image and adding the new states. Additionally, if we need to run multiple Docker Containers, we need a feature called Docker Compose.

3.6.4. Data Visualization

A dashboard is a type of graphical user interface; it can also be understood as a report on the process of data visualization by connecting to various sources of data. In business, dashboards not only provide in-depth data on business operations, but also provide an overview of the performance of each department, trends, activities, and key performance indicators (KPIs).

As data explosion has been mentioned, dashboards are increasingly used, not only for businesses or technical professionals and researchers, but also for users to understand how a product is operating.

Furthermore, dashboards can be divided into two types, a static dashboard that displays static data, and an interactive dashboard that is used to display continuous real-time data. When devices provide information, new data is added to the database and this process repeats continuously to create a data pipeline, the data in the data pipeline changes continuously.

3.6.4.1. Grafana

Grafana is an open-source platform that specializes in monitoring and evaluating collected data. As defined, we can see that the application of Grafana is very wide, not just in the IT sector. Any field that can collect data over time can be displayed optimally on Grafana. In addition to the ability to connect to a variety of data sources, the tool's interface is user-friendly. Easy to provide information and warnings.

Grafana can be used on personal computers because developers have developed applications that can run on multiple platforms other than mobile devices, it can be installed directly or through a Docker Image.

Grafana is very convenient because it can connect to many different data sources such as time series databases, logging and document databases, distributed tracing, etc.

And there are many different types of charts to use for different requirements such as time series, bar charts, stats, gauges, pie charts, etc. along with a user community spread all over the world that helps create more attractive charts.

Additionally, we can use Grafana Cloud to build Dashboards and share them online with many people at a reasonable price based on each need.

Figure 3-31. Grafana Desktop log in

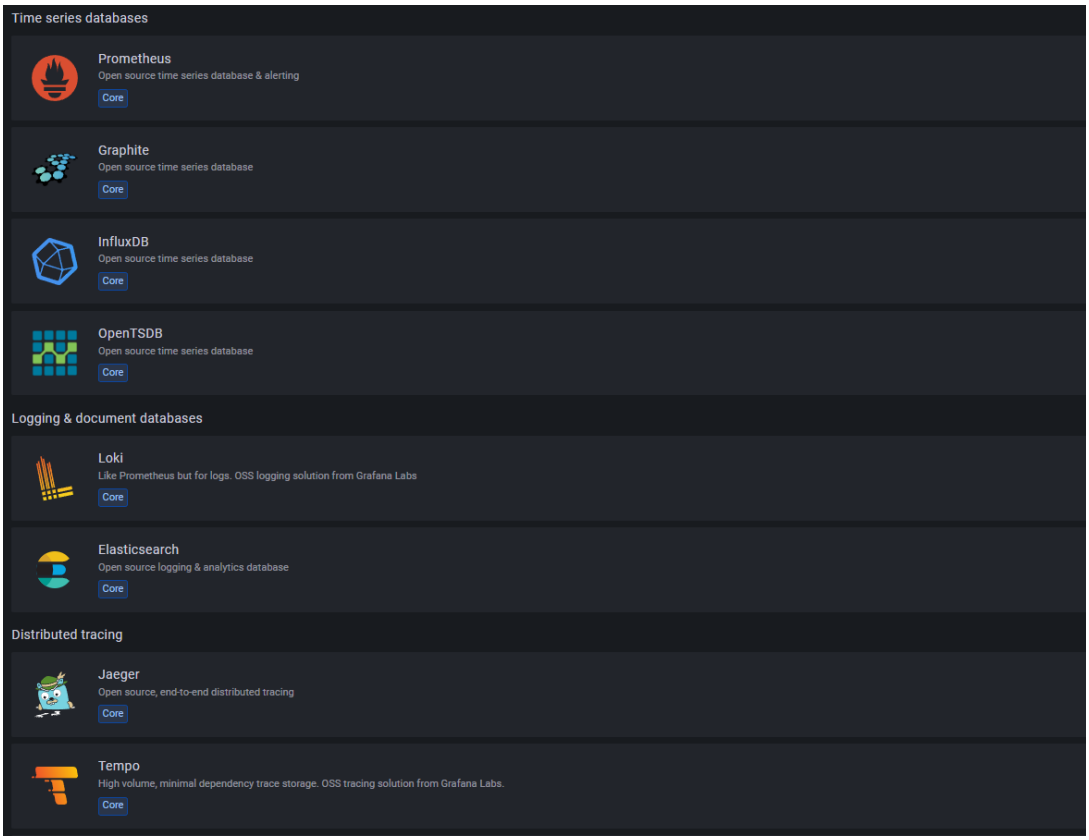


Figure 3-32. Data Sources in Grafana

3.6.4.2. Wandb

We can use the WanDB, Weights & Biases library as a useful tool to support various stages in the management of machine learning models (visualize, tracking, monitoring, model registry, etc.) in the experimentation phase.

This tool will track and visualize each part of the Machine Learninng pipeline, from data processing to model transformations into products.

Some applications of WanDB include:

- Displaying real-time results in the form of tables and charts that are managed by a simple and easy-to-read web interface.
- Helping to focus on building models, saving time tracking results with text or excel.
- Saving versions of data with W&B Artifacts when there are changes to the records, considering the level of impact on the model results.
- Backup models (code, hyperparameters, launch commands, input data, weight, metric, config, etc.).

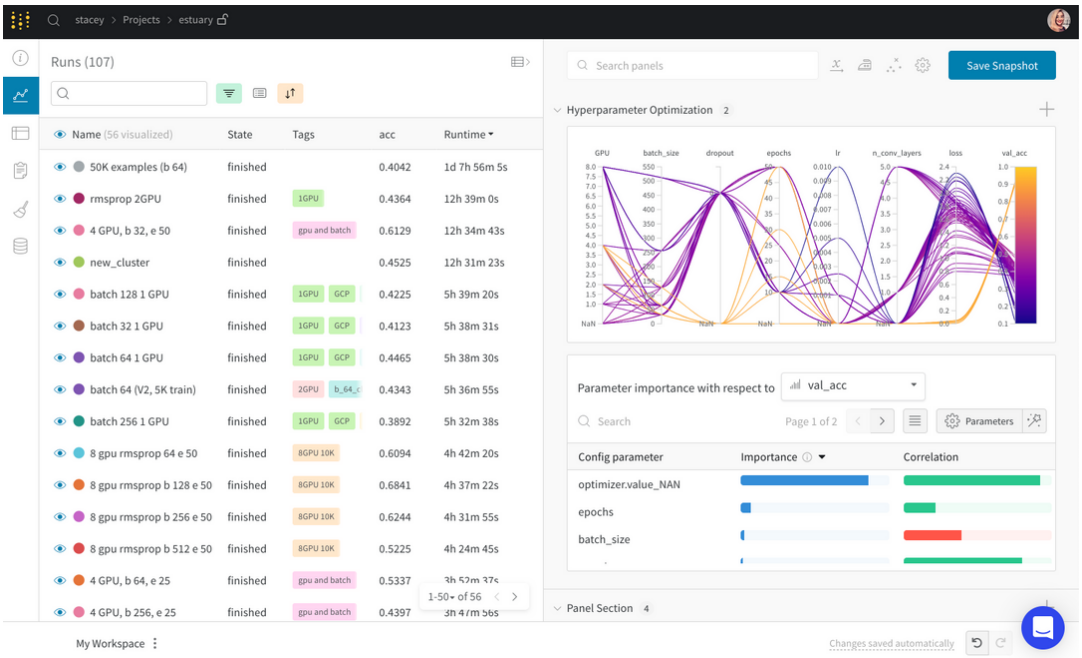


Figure 3-33. Example of WanDB

3.6.5. Continual Learning

One of the most common misconceptions data scientists make with machine learning is assuming their models will continue to perform properly after deployment. But what about the data, which is bound to change? A model in production that is left alone will not be able to respond to changes in data.

In reality, very few corporations really do it. Chip Huyen states in her book [11], there is two reasons for this situation. Initially, when using a neural network model, the process of learning with each new sample can lead to catastrophic forgetting. This refers to the tendency of the neural network to completely and suddenly lose previously acquired knowledge upon learning new information.

Another reason is that it can increase the cost of training - as many current hardware systems are optimized for processing multiple samples at once, processing only one sample at a time results in a significant loss of computing resources and the inability to use parallel processing techniques with data.

Companies that use continuous learning in manufacturing update their models in micro-batches. They may, for example, update the existing model every 512 or 1,024 examples—the appropriate number of examples in each micro-batch depends on the task.

The new model should not be deployed until it has been thoroughly tested. This means you shouldn't make direct changes to the existing model. Instead, you establish a replica of the old model and update it with new data, replacing the existing model only if the updated replica proves to be better.

CHAPTER 4: DATA ENGINEERING

4.1. MIT – BIH Arrhythmia dataset

The MIT-BIH Arrhythmia dataset is a compilation of ECG recordings that have been labeled with information about different types of heart rhythm disorders. The data was collected by MIT Laboratory for Computational Physiology and Beth Israel Deaconess Medical Center. The dataset has 48 30-minute ECG recordings from 47 patients and contains a total of 549,047 samples. The ECG signals were recorded from patients who were hospitalized for various heart conditions and annotated by cardiologists. This dataset is frequently used in research related to the identification and classification of arrhythmias and has been utilized in numerous studies and evaluations of detection algorithms for arrhythmias.

According to [10], the author extracted and classified the ECG signals of the MIT-BIH Arrythmia dataset into labels including: N, F, S, V, Q. Specifically, each label will include the following diseases:

- N: Normal, Left/Right bundle branch block, Atrial escape, Nodal escape.
- S: Atrial premature, Aberrant atrial premature, Nodal premature, Supra-ventricular premature.
- V: Premature ventricular contraction, Ventricular escape.
- F: Fusion of ventricular and normal
- Q: Paced, Fusion of paced and normal, Unclassifiable.

Figure 4-1. Example of MIT – BIH Arrythmia Dataset

4.2. Feature engineering – real-time data from device

In the data pre-processing section, the sequences of steps are followed as described in the paper [10], including:

1) Dividing the continuous ECG signal into 10-second segments, and selecting one segment from the ECG signal.

2) Adjusting the amplitude values to a range between 0 and 1.

3) Identifying all local maximums using zero-crossings of the first derivative.

4) Identifying R-peak candidates in the ECG signal by applying a threshold of 0.9 on the normalized local maximums.

5) Determining the median of R-R time intervals as the nominal heartbeat period for that segment (T).

6) For each identified R-peak, selecting a portion of the signal with a length of 1.2T.

7) Adding zeroes to each selected portion to reach a fixed, predefined length.

4.2.1. Preprocessing

4.2.1.1. Filtering

As mentioned in Chapter 3, it is not possible to apply a low-pass filter with a 100Hz cutoff frequency to data with a 128Hz sampling rate, and applying a 50Hz notch filter does not result in any changes in the signal. To determine the effectiveness of these filters, consultation with experts is necessary. Within the scope of this project, we only applied and presented results for the noise filter from motion artifact.

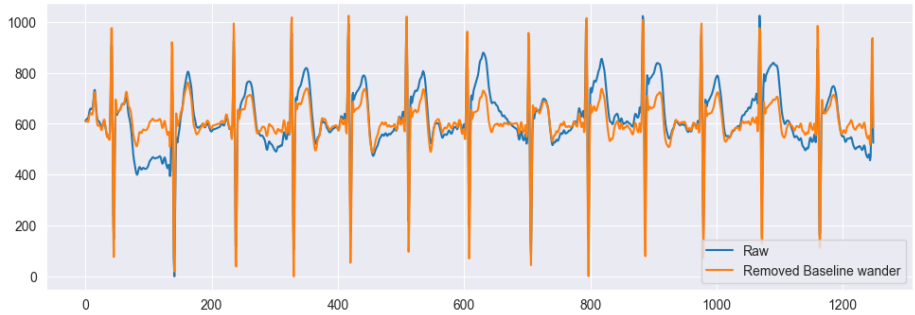


Figure 4-2. Results of motion artifact noise removal

By using the heartpy library in python, baseline wander is removed. The blue line is the original ECG signal from the device, the pink line is the signal after noise filtering.

4.2.1.2. Scaling

In **Figure 4-3**, the ECG signal has been filtered for noise, but it has not been normalized to values between 0 and 1. Using heartpy to normalize the signal results in the following:

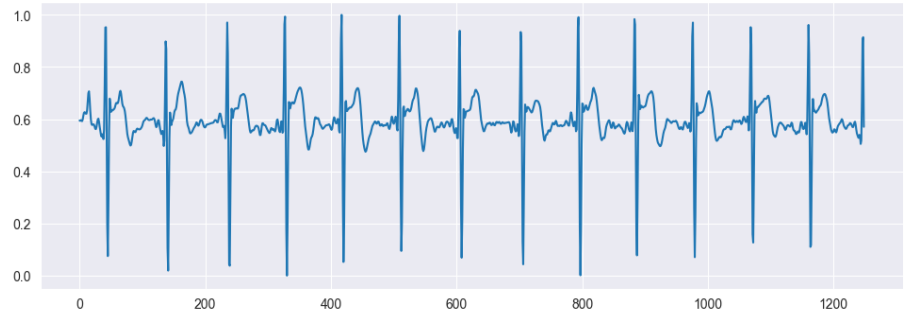


Figure 4-3. Normalized ECG signal

Thus, the highest R peak value is assigned as 1, the lowest S value is assigned as 0. The remaining values are normalized based on these highest R and lowest S values.

4.2.1.3. Features extraction

Features extraction methods vary based on the specific machine learning models and applications being used. In this specific case, for electrocardiogram classification, we followed steps 3 to 6 as described in the previous information.

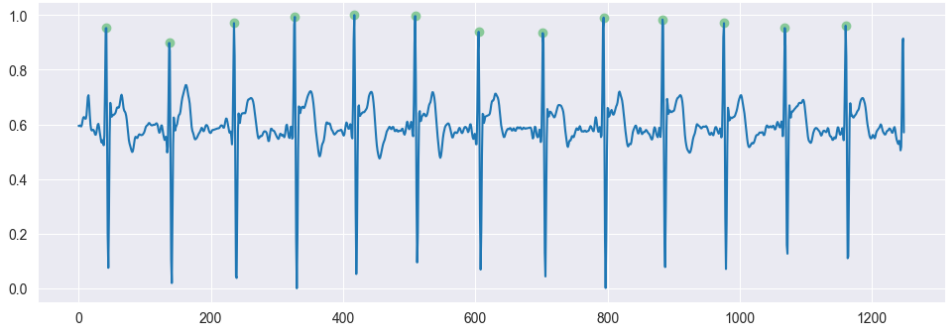


Figure 4-4. Finding R peaks

We found the R-peaks by counting the number of data points between them and then multiplied that by 7.8125ms to find the time interval in seconds. We then calculated all R-R pairs, took the total time and divided it by the number of pairs to get the final result as shown in **Figure 4-5**.

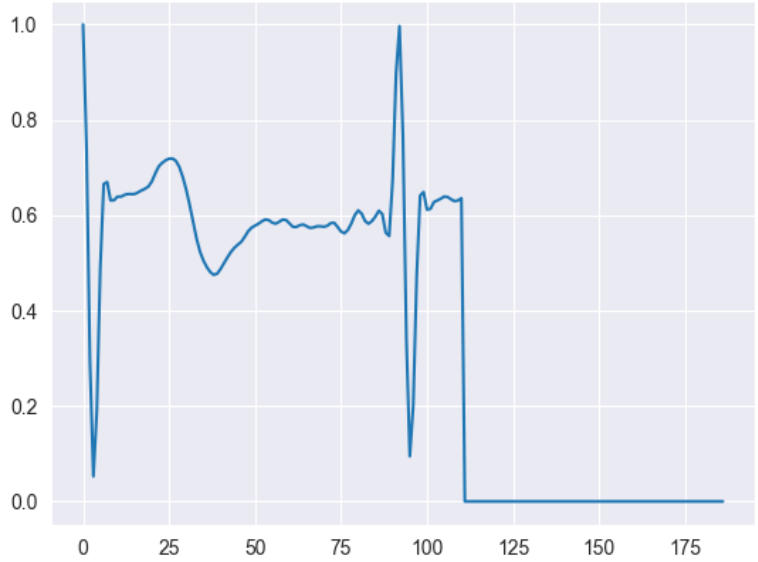


Figure 4-5. Extracted data

4.2.2. Storing data

4.2.2.1. Storing data into InfluxDB

Unlike structured databases, time series databases such as InfluxDB use concepts of measurement, tag, field, and time. Within an InfluxDB database, there can be multiple measurements, each containing one or more tags and fields. Each tag and field is stored as a column.

Tags help to extract and sort data, fields store values, in this case, sensor values. Time is a field for time, every data point written to InfluxDB must have a timestamp.

Currently, there are 3 measurements as follows:

- Heartrate, to record heart rate in the last 10 seconds.
- ECG, to record the ECG signal from the sensor.
- Log, to record the results of the AI model classification.

The tables and figures below will provide an overall view of the above 3 measurements.

Table 4-1. ECG signal measurement

Measurement: ECG signal		
Timestamp	Tag: id	Field: value

Where field: value will record the sensor value as a float data type, and tag: id will record the user's id as a string data type.

	time	id	value
1	2022-12-10 14:08:47.968	T1lsO	127.0
2	2022-12-10 14:10:55.807	T1lsO	127.0
3	2022-12-10 14:11:10.808	T1lsO	128.0
4	2022-12-10 14:13:16.347	T1lsO	128.0
5	2022-12-10 14:13:32.347	T1lsO	128.0
6	2022-12-10 13:58:57.105	T1lsO	128.0
7	2022-12-10 13:59:17.935	T1lsO	128.0
8	2022-12-10 14:01:16.612	T1lsO	128.0
9	2022-12-10 14:01:48.612	T1lsO	128.0
10	2022-12-10 14:03:37.257	T1lsO	128.0

Figure 4-6. Example of ECG signal measurement

Table 4-2. Heart rate measurement

Measurement: Heart rate		
Timestamp	Tag: id	Field: value

Where value is the heart rate, id is the user’s id.

	time	id	value
1	2022-12-10 14:08:47.968	T1lsO	127.0
2	2022-12-10 14:10:55.807	T1lsO	127.0
3	2022-12-10 14:11:10.808	T1lsO	128.0
4	2022-12-10 14:13:16.347	T1lsO	128.0
5	2022-12-10 14:13:32.347	T1lsO	128.0
6	2022-12-10 13:58:57.105	T1lsO	128.0
7	2022-12-10 13:59:17.935	T1lsO	128.0
8	2022-12-10 14:01:16.612	T1lsO	128.0
9	2022-12-10 14:01:48.612	T1lsO	128.0
10	2022-12-10 14:03:37.257	T1lsO	128.0

Figure 4-7. Example of Heart rate measurement

Table 4-3. Result measurement

Measurement: result		
Timestamp	Tag: id	Field: value

This measurement store output of xgboost model in field: value.

	time	id	result	value
1	2022-12-10 14:05:21.212	Ti1sO	[0.]	Normal
2	2022-12-10 14:28:03.321	Ti1sO	[0.]	Normal
3	2022-12-10 14:42:30.122	Ti1sO	[0.]	Normal
4	2022-12-10 14:56:26.578	Ti1sO	[0.]	Normal
5	2022-12-10 15:38:44.737	Ti1sO	[0.]	Normal
6	2022-12-10 15:51:41.787	Ti1sO	[0.]	Normal
7	2022-12-10 16:02:11.801	Ti1sO	[0.]	Normal
8	2022-12-10 16:15:14.827	Ti1sO	[0.]	Normal
9	2022-12-10 13:57:36.477	Ti1sO	[0.]	Normal
10	2022-12-10 14:15:36.075	Ti1sO	[0.]	Normal

Figure 4-8. Example of Result measurement

4.2.2.2. Storing data into SQLite

SQLite is used to record activities within the gateway, as it has many advantages. It is particularly used to store sensor data when there is no internet connection, as without a network connection, the gateway cannot send data to the cloud-based database (InfluxDB Cloud).

Currently, there are 2 databases in the SQLite database. One database is used to control the version of the machine learning model, and the other is used to store sensor signals.

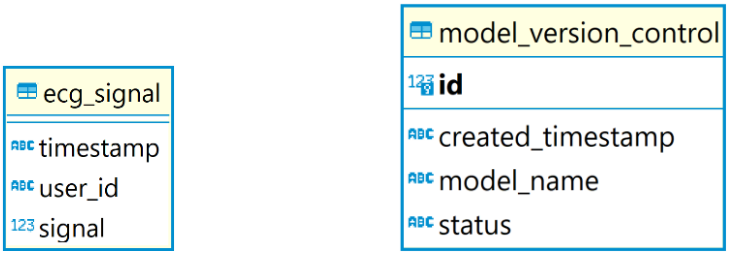


Figure 4-9. ER Diagram of databases in SQLite

	id	created_timestamp	model_name	status
1	1	2022-12-12T 11:20:44.131533Z	6JcP8S60QI.model	downloaded
2	2	2022-12-12T 11:20:58.560243Z	6JcP8S60QI.model	used
3	3	2022-12-12T 11:22:44.000202Z	xHnQTBxHmo.model	downloaded
4	4	2022-12-12T 11:22:47.620772Z	xHnQTBxHmo.model	used
5	5	2022-12-12T 11:58:14.599074Z	271cPVsejv.model	downloaded
6	6	2022-12-12T 11:58:22.230212Z	271cPVsejv.model	used
7	7	2022-12-12T 12:00:14.000364Z	ecqdsx9FvO.model	downloaded
8	8	2022-12-12T 12:00:17.995911Z	ecqdsx9FvO.model	used

Figure 4-10. Example of model_version_control

CHAPTER 5: MACHINE LEARNING IN RESEARCH

5.1. Introduction to XGBoost

5.1.1. Bias and Variance

Bias is defined as the difference between our model's average forecast and the correct value that we are attempting to predict. High bias models oversimplify the model and pay little attention to the training data It consistently leads to a significant number of mistakes in both training and test data.

The variability of model prediction for a specific data point or value tells us about the spread of our data. A high variance model focuses heavily on training data and does not generalize to data that it has not previously encountered. Because of this, such models have significant error rates when applied to test data yet perform quite well on training data.

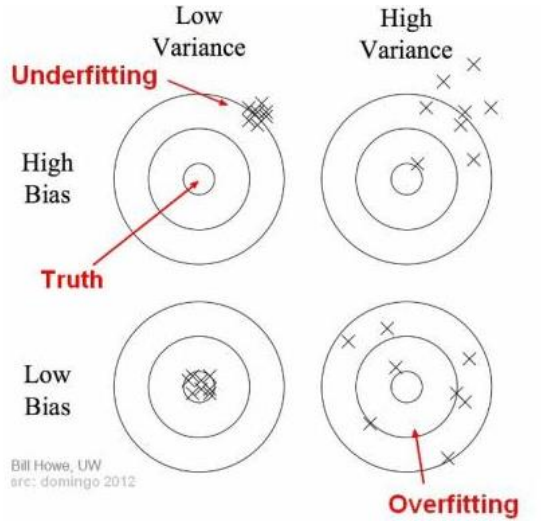


Figure 5-1. Bias – Variance

The bias and variance of a model may be significant if it is overly simplistic and contains few parameters. On the other hand, a complicated model will have a low bias and a high variance if it includes a lot of parameters. Therefore, we are trying to avoid both overfitting and underfitting the data.

5.1.2. Decision tree

Decision tree is an algorithm, a part of supervised learning technique, that can be used in many cases which can be categorized in two types: classification and regression. But in most cases, decision tree is used to solve classification problems.

It is a tree-structured classifier, where branches for the decision-making process, and each leaf node for the result.

The Decision Node and Leaf Node are the two different sorts of nodes that make up a decision tree. When making decisions, Decision nodes can be used, and they can have several branches, whereas Leaf nodes represent the results of those decisions and do not have any additional branches.

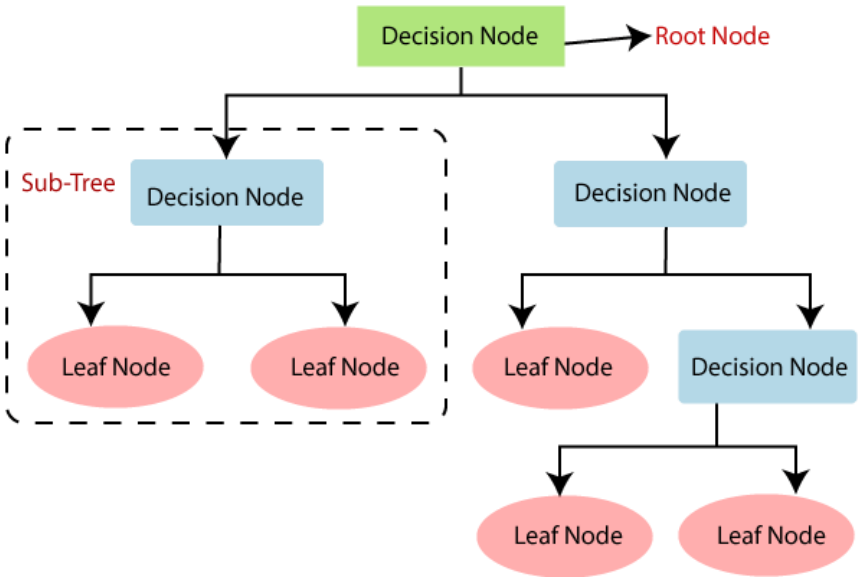


Figure 5-2. Decision tree structure

Root Node: The decision tree's root node is where the tree's decision-making process begins. It is a representation of the whole dataset, which is then split into two or more homogeneous sets.

Leaf Node: A leaf node marks the end of the output process; the tree cannot be further divided at this point.

Splitting: Splitting is the process of subdividing the root node/decision node into sub-nodes in accordance with the conditions specified.

Branch/subtree: A branch/subtree is a tree that is generated by splitting a larger tree.

Pruning: The removal of undesirable limbs from a tree is known as pruning.

Parent/Child node: The parent node in a tree is the root node, while the child nodes are the other nodes.

Let's think about the next illustration. Let's say an applicant has a job offer with a pay range of $50000 to $80000. He needs to make a decision regarding whether to accept or reject the offer. Decision trees are a useful tool to utilize in order to solve this issue.

The salary will be the root node. Based on the corresponding labels, the root node further divides into the next decision node (distance from the office) and one leaf node. The following decision node is further divided into a leaf node and a decision node (Cab facility). The decision node finally separates into two leaf nodes (Accepted offers and Declined offer).

As we can see, the decision tree can be ploted like **Figure 5-3**.

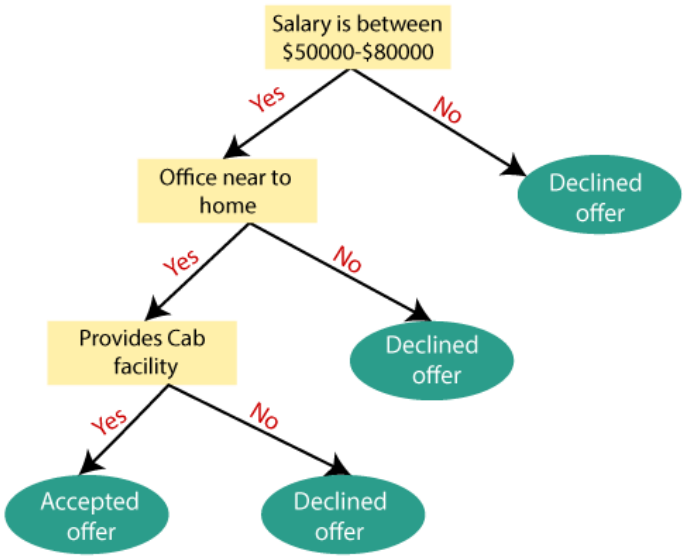


Figure 5-3. Example of decision tree

5.1.3. Bagging

Bagging, also known as Bootstrap aggregating, is an ensemble learning approach that contributes in improving the efficiency and precision of machine learning algorithms. It reduces a prediction model's variance and is used to solve bias-variance trade-offs. Bagging, especially decision tree approaches, is used to reduce overfitting of data in both regression and classification models.

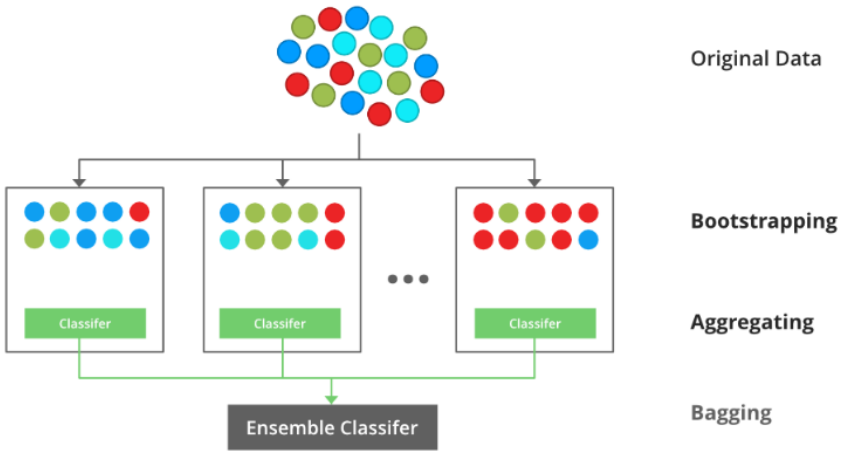


Figure 5-4. Bagging

5.1.4. Random Forest

Random forest is a supervised learning approach. It is a common machine learning algorithm that may be utilized for both classifier and regression issues.

Random Forest, as the name implies, is a classifier that uses a number of decision trees on different subsets of the provided dataset and averages them to increase the dataset's prediction accuracy. Instead of depending on a single decision tree, the random forest considers the forecast from each tree and guesses the result based on the predictions that have received the most votes.

The bigger the number of trees in the forest, the higher the accuracy and the lower the risk of overfitting.

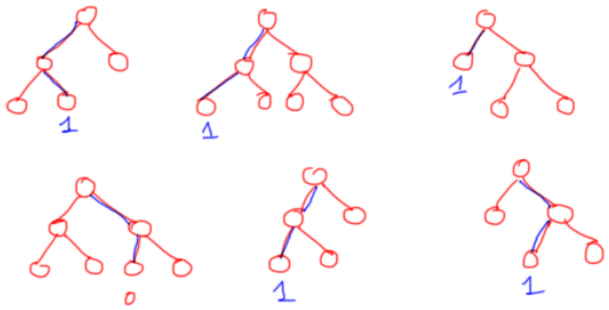


Figure 5-5. A random forest model

In **Figure 5-5**, the final result is the combination of all results above.

Assume there is a dataset with multiple fruit photos. The Random forest classifier is fed this dataset. Each decision tree receives a subset of the dataset. During the training phase, each decision tree forecasts a new data point, and the Random Forest classifier predicts the final option based on the majority of results. Consider the following image:

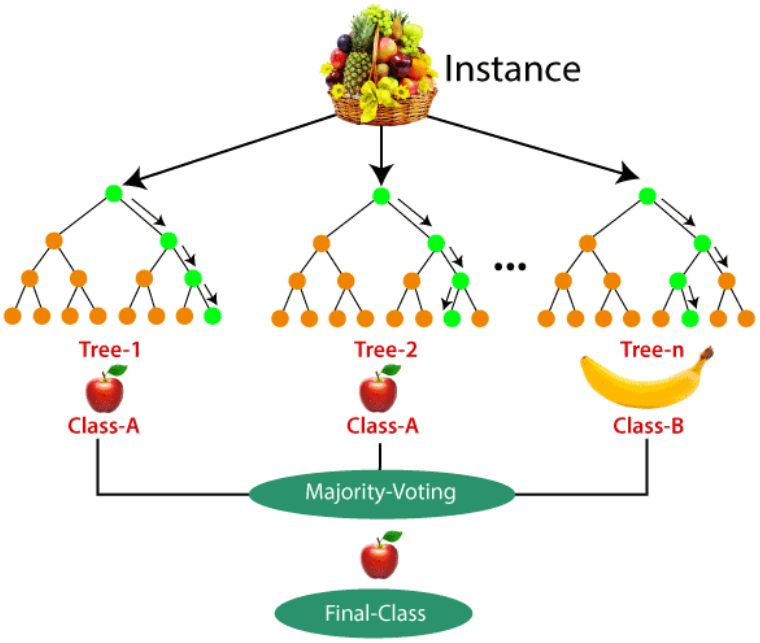


Figure 5-6. An example of Random Forest

5.1.5. Boosting

Boosting algorithms, like the theory discussed in earlier parts, strive to enhance prediction power by training a series of weak models, each correcting for the shortcomings of its predecessors.

To comprehend Boosting, it is critical to realize that it is a general algorithm rather than a specific model. Boosting requires you to specify a weak model, which is subsequently improved.

After that, it's time to look at several definitions of weakness and the algorithms that go with them. Adaptive Boosting (AdaBoost) and Gradient Boosting are the two main methods that will be discussed.

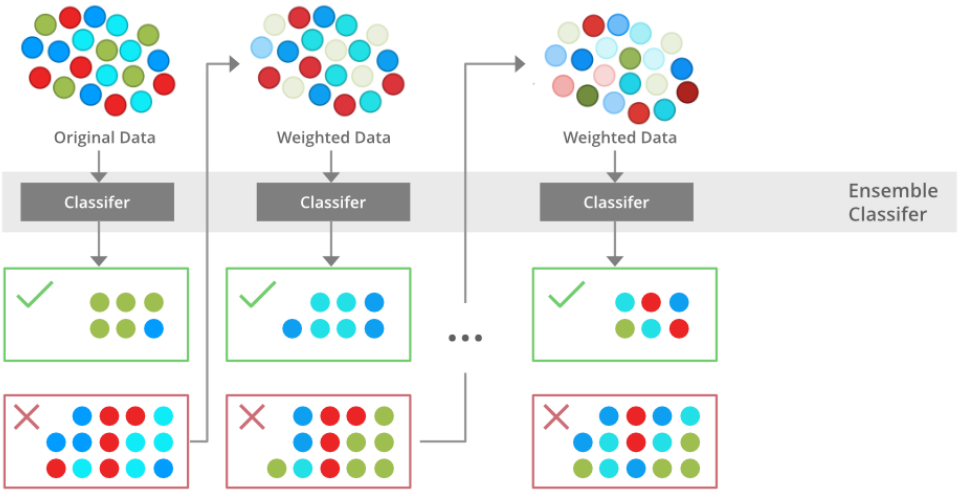


Figure 5-7. Boosting

5.1.5.1. Adaptive Boosting (AdaBoost)

AdaBoost was the first truly successful binary classification boosting technique. AdaBoost is short for Adaptive Boosting and is a very common boosting approach that combines numerous "weak classifiers" into a single "strong classifier". Yoav Freund and Robert Schapire created it.

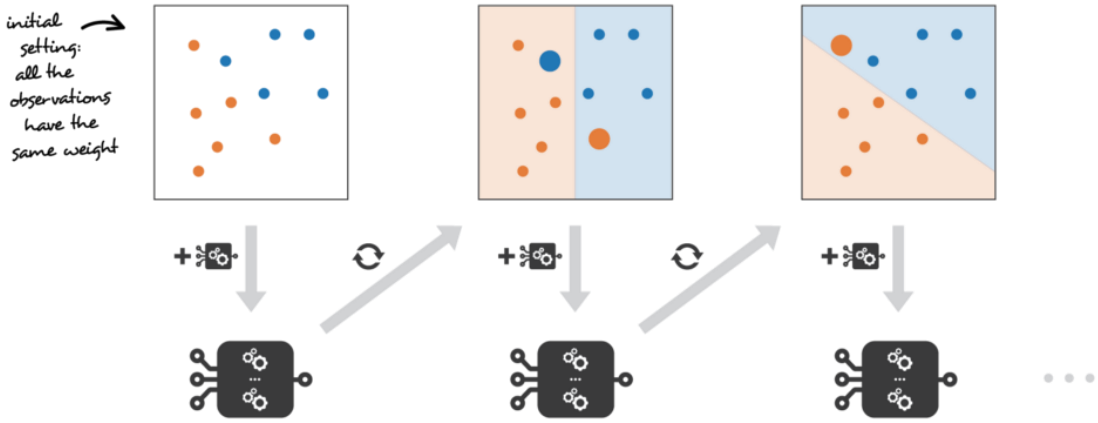


Figure 5-8. AdaBoost

One classifier might not be able to correctly forecast the class of an item, but by grouping several weak classifiers and having them gradually learn from each other's incorrectly classified objects, we can create one such strong model. The classifier indicated here might be any of your fundamental classifiers, ranging from Decision Trees (which are frequently the default) to Logistic Regression, and so on. A weak classifier outperforms random classifier but is nonetheless ineffective at assigning classes to objects.

A poor classifier, for example, may predict that everyone above the age of 40 cannot run a marathon, but those under that age can. You may achieve greater than 60% accuracy, but you will still misclassify a large number of data items!

AdaBoost may be implemented on top of any classifier to learn from its flaws and suggest a more accurate model rather than being a model in and of itself. It is commonly referred to as the "best out-of-the-box classifier" for this reason.

Let's look at how AdaBoost interacts with Decision Stumps. Decision Stumps are similar to trees in a Random Forest, except they are not "completely developed." They have two leaves and one node. Instead of trees, AdaBoost employs a forest of such stumps.

Stumps by themselves are not an useful approach to make decisions. A fully formed tree incorporates all variable decisions to anticipate the goal value. A stump, on the other hand, can only make a choice based on one variable. Let's attempt to figure out what's going on behind the scenes.

- **Step 1:** A weak classifier (e.g. a decision stump) is made on top of the training data based on the weighted samples. Here, the weights of each sample indicate how important it is to be correctly classified. Initially, for the first stump, we give all the samples equal weights.

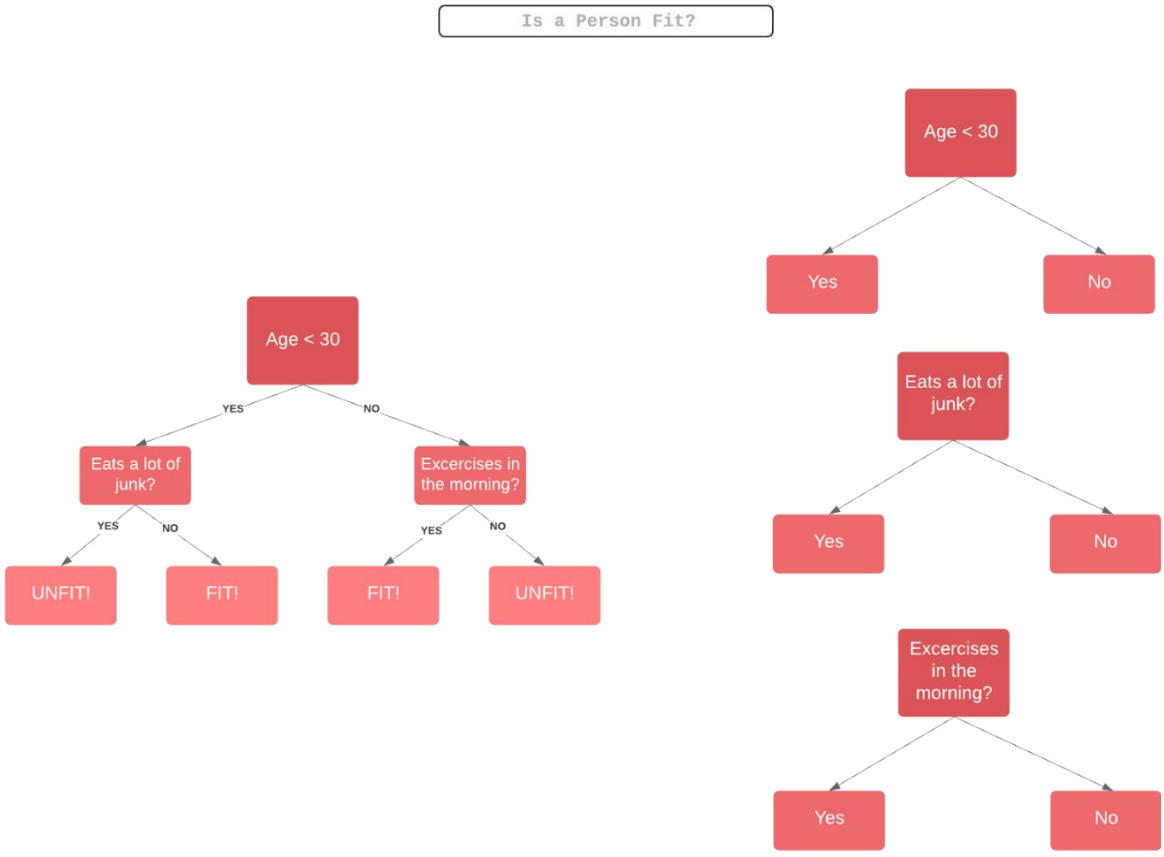


Figure 5-9. AdaBoost stumps

- **Step 2:** For each variable, we design a decision stump, and we test how well each stump assigns samples to the intended classes. For instance, we look at age, eating junk food, and exercise in the diagram below. For each individual stump, we would count the number of samples that were rightly or wrongly categorized as Fit or Unfit.

- **Step 3:** To ensure that the samples are accurately classified in the following decision step, more weight is given to the erroneously classified samples. Additionally, weight is assigned to each classifier depending on its accuracy, so high accuracy equals high weight.

- **Step 4:** Repeat Step 2 until either the maximum number of iterations has been achieved or all of the data points have been appropriately categorised.

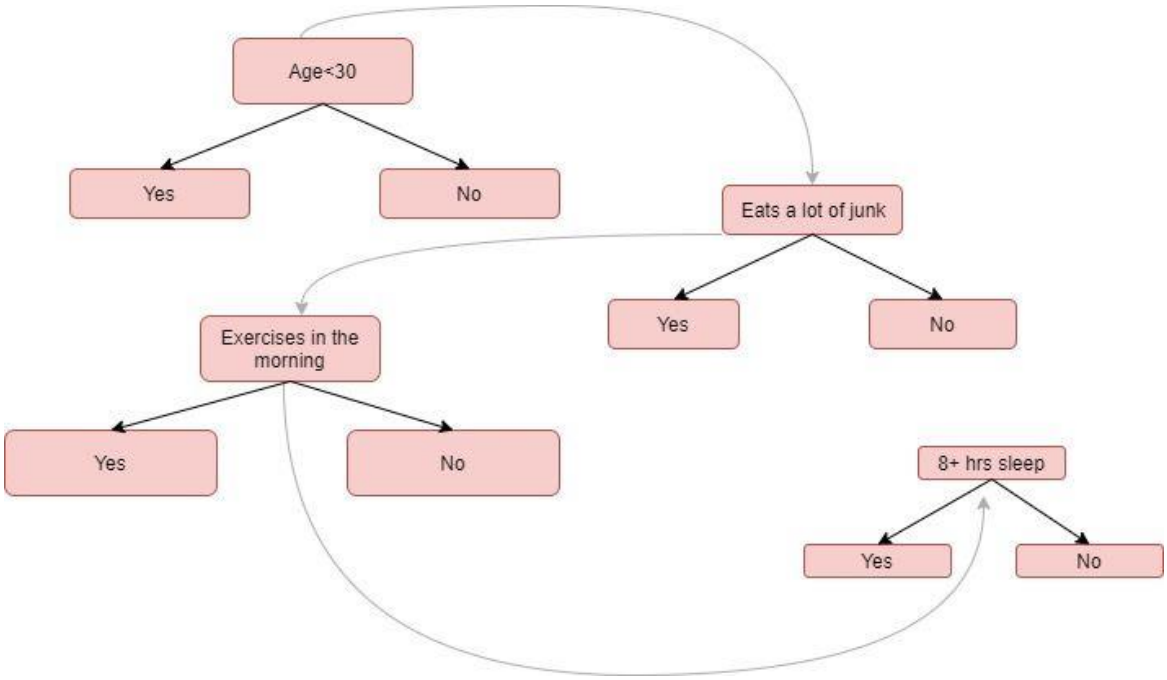


Figure 5-10. AdaBoost progress

5.1.5.2. Gradient Boosting

The first boosting algorithm with a specific loss function was called AdaBoost. Gradient Boosting, on the other hand, is a general method that aids in the search for approximate solutions to the additive modeling issue. Gradient Boosting is thus more adaptable than AdaBoost.

Gradient boosting generates prediction-based models by combining weak prediction models. Weak hypotheses are parameters that perform marginally better than random selections. When employed with appropriate cost functions, Leo Breiman, an American statistician, considered boosting to be an optimization technique. Cost function optimization is accomplished by iteratively selecting weak hypotheses or a function with a significantly negative gradient. Many improvements have been made to the gradient boosting approach in order to optimize cost functions.

The operation of gradient boosting focuses on three basic components. These are the following:

Loss function, Weak leaner, Addictive model.

Loss function

The primary goal here is to optimize the loss function. The loss function varies depending on the kind of problem. It is simple to define one's own standard loss function, but it must be differentiable.

As an example, regression may employ the squared error and classification can employ the algorithmic loss. One of the nicest things about gradient boosting is that each framework does not require a new boosting method for each loss function. As a result, a more general framework would suffice.

Weak leaner

Weak learners are there to help you make predictions. A decision tree is essentially a weak learner. For the real output values needed for splits, specific regression trees are applied. The reminders in the prediction models can be adjusted. Purity ratings like Gini identify the best split-points, which are then used to build the trees.

Addictive model

There are more trees added at once, but no changes are made to the model's already-existing trees.

A gradient descent approach reduces the losses when adding the trees. It reduces the number of parameters to a minimum. To reduce the error, the weights are only updated after the error has been calculated.

Parameters are replaced with weak learner sub-models. After computing the loss, we must add a tree to the model in order to decrease losses and perform the gradient descent technique. Finally, we may add the output to the tree sequence.

5.1.6. XGBoost

Extreme Gradient Boosting (XGBoost) is a scalable, distributed gradient-boosted decision tree (GBDT) machine learning framework. It is the top machine learning package for regression, classification, and ranking tasks, and it supports parallel tree boosting. It was proposed by University of Washington scholars.

This technique generates decision trees in a sequential fashion. Weights are very significant in XGBoost. All of the independent variables are given weights, which are subsequently put into the decision tree, which predicts results. The weight of factors that the tree predicted incorrectly is raised, and these variables are subsequently put into the second decision tree. These various classifiers/predictors are then combined to form a more powerful and precise model. It can do regression, classification, and ranking tasks.

5.2. Hyperparameters

This section's theory of xgboost's parameters is entirely based on the documentation available on the xgboost library webpage [1].

First, it's important to understand the distinction between parameters and hyperparameters. In a machine learning model, internal configuration variables and parameter values will change as the model learns more about the data.

Let's use the following instance:

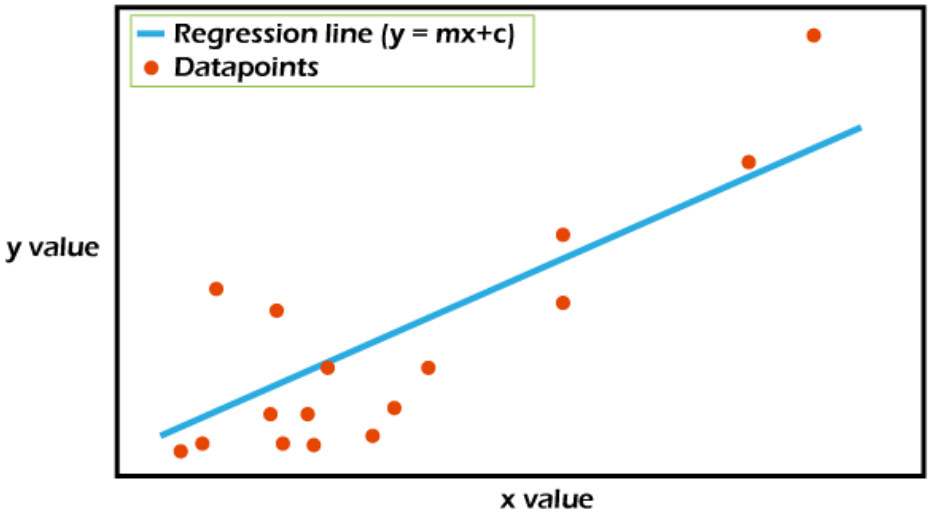


Figure 5-11. Simple Linear Regression

Simple linear regression is displayed in **Figure 5-1**. With an understanding of essential calculus, we can understand that the objective is to fit the best regression line to the available data in order to describe a relationship between the independent variables *x* and *y*, and that *y* is the dependent variable.

The relationship can be easily expressed using the equation $y = xm + c$. where *c* is the line's intercept and *m* is the line's slope. Model parameters are the two parameters determined by fitting the line while minimizing RMSE..

Meanwhile, hyperparameters are defined by the user to control the learning process.

- These are usually defined manually by the machine learning engineer.
- One cannot know the exact best value for hyperparameters for the given problem. The best value can be determined either by the rule of thumb or by trial and error (tuning).

Table 5-1. Comparison between parameters and hyperparameters

Parameters	Hyperparameters
Parameters are the configuration model, which are internal to the model.	Hyperparameters are the explicitly specified parameters that control the training process.
Parameters are essential for making predictions.	Hyperparameters are essential for optimizing the model.
These are specified or estimated while training the model.	These are set before the beginning of the training of the model.
It is internal to the model.	These are external to the model.
These are learned & set by the model by itself.	These are set manually by a machine learning engineer/practitioner.
These are dependent on the dataset, which is used for training.	These are independent of the dataset.
The values of parameters can be estimated by the optimization algorithms, such as Gradient Descent.	The values of hyperparameters can be estimated by hyperparameter tuning.
The final parameters estimated after training decide the model performance on unseen data.	The selected or fine-tuned hyper-parameters decide the quality of the model.

Let’s go into more detailed about XGBoost hyperparameters.

5.2.1. General parameters

These parameters guide the overall functioning of the XGBoost model.

booster [default= gbtree]: Which booster to use. Can be gbtree, gblinear or dart; gbtree and dart use tree based models while gblinear uses linear functions.

verbosity [default=1]: Verbosity of printing messages. Valid values are 0 (silent), 1 (warning), 2 (info), 3 (debug). Sometimes XGBoost tries to change configurations based on heuristics, which is displayed as warning message. If there’s unexpected behaviour, please try to increase value of verbosity.

validate_parameters [default to false, except for Python, R and CLI interface]: When set to True, XGBoost will perform validation of input parameters to check whether a parameter is used or not.

nthread [default to maximum number of threads available if not set]: Number of parallel threads used to run XGBoost. When choosing it, please keep thread contention and hyperthreading in mind.

disable_default_eval_metric [default= false]: Flag to disable default metric. Set to 1 or true to disable.

num_feature [set automatically by XGBoost, no need to be set by user]: Feature dimension used in boosting, set to maximum dimension of the feature.

5.2.2. Booster parameters

There are two types of booster parameters, tree booster and linear booster. Because the outperformace of the linear booster, we will discuss and use linear booster.

eta [default=0.3, alias: learning_rate]:

- Step size shrinkage used in update to prevents overfitting. After each boosting step, we can directly get the weights of new features, and eta shrinks the feature weights to make the boosting process more conservative.

- Range: [0, 1]

gamma [default=0, alias: min_split_loss]:

- Minimum loss reduction required to make a further partition on a leaf node of the tree. The larger gamma is, the more conservative the algorithm will be.

- Range: [0, ∞]

max_depth [default=6]

- Maximum depth of a tree. Increasing this value will make the model more complex and more likely to overfit. 0 indicates no limit on depth. Beware that XGBoost aggressively consumes memory when training a deep tree. exact tree method requires non-zero value.

- Range: [0, ∞]

min_child_weight [default=1]:

- It defines the minimum sum of weights of all observations required in a child.

- This is similar to min_child_leaf in GBM but not exactly. This refers to min "sum of weights" of observations while GBM has min "number of observations".

- It is used to control over-fitting.

- Higher values prevent a model from learning relations which might be highly specific to the particular sample selected for a tree.

- Too high values can lead to under-fitting.

- Hence, it should be tuned using CV.

- The larger min_child_weight is, the more conservative the algorithm will be.

- Range: [0, ∞]

max_delta_step [default=0]:

- In maximum delta step we allow each tree's weight estimation to be.
- If the value is set to 0, it means there is no constraint.
- If it is set to a positive value, it can help making the update step more conservative.
- Usually this parameter is not needed, but it might help in logistic regression when class is extremely imbalanced.
- Set it to value of 1-10 might help control the update.
- Range: [0, ∞]

subsample [default=1]:

- Subsample ratio of the training instances. Setting it to 0.5 means that XGBoost would randomly sample half of the training data prior to growing trees. and this will prevent overfitting. Subsampling will occur once in every boosting iteration.
- Range: (0, 1]

sampling_method [default= uniform]

- The method to use to sample the training instances.
- uniform: each training instance has an equal probability of being selected. Typically set subsample >= 0.5 for good results.
- gradient_based: the selection probability for each training instance is proportional to the regularized absolute value of gradients. subsample may be set to as low as 0.1 without loss of model accuracy. Note that this sampling method is only supported when tree_method is set to gpu_hist; other tree methods only support uniform sampling.

colsample_bytree, colsample_bylevel, colsample_bynode [default=1]:

- This is a family of parameters for subsampling of columns.
- All colsample_by* parameters have a range of (0, 1], the default value of 1, and specify the fraction of columns to be subsampled.
- colsample_bytree is the subsample ratio of columns when constructing each tree. Subsampling occurs once for every tree constructed.

- colsample_bylevel is the subsample ratio of columns for each level. Subsampling occurs once for every new depth level reached in a tree. Columns are subsampled from the set of columns chosen for the current tree.

- colsample_bynode is the subsample ratio of columns for each node (split). Subsampling occurs once every time a new split is evaluated. Columns are subsampled from the set of columns chosen for the current level.

lambda [default=1, alias: reg_lambda]:

- L2 regularization term on weights (analogous to Ridge regression).

- This is used to handle the regularization part of XGBoost.

- Increasing this value will make model more conservative.

alpha [default=0, alias: reg_alpha]:

- L1 regularization term on weights (analogous to Lasso regression).

- It can be used in case of very high dimensionality so that the algorithm runs faster when implemented.

- Increasing this value will make model more conservative.

tree_method string [default= auto]:

- The tree construction algorithm used in XGBoost. Choices*: auto, exact, approx, hist, gpu_hist.*

- XGBoost supports approx, hist and gpu_hist for distributed training. Experimental support for external memory is available for approx and gpu_hist.

- auto: Use heuristic to choose the fastest method. For small to medium dataset, exact greedy (exact) will be used. For very large dataset, approximate algorithm (approx) will be chosen. Because old behavior is always use exact greedy in single machine, user will get a message when approximate algorithm is chosen to notify this choice.

- exact: Exact greedy algorithm.

- approx: Approximate greedy algorithm using quantile sketch and gradient histogram.

- hist: Fast histogram optimized approximate greedy algorithm. It uses some performance improvements such as bins caching.

- gpu_hist: GPU implementation of hist algorithm.

scale_pos_weight [default=1]:

- It controls the balance of positive and negative weights,

- It is useful for imbalanced classes.
- A value greater than 0 should be used in case of high class imbalance as it helps in faster convergence.
- A typical value to consider: sum(negative instances) / sum(positive instances).

max_leaves [default=0]:

- Maximum number of nodes to be added.
- Only relevant when grow_policy=lossguide is set.
- There are other hyperparameters like sketch_eps, updater, refresh_leaf, process_type, grow_policy, max_bin, predictor and num_parallel_tree.

5.2.3. Learning parameters

These parameters are used to define the optimization objective the metric to be calculated at each step. They are used to specify the learning task and the corresponding learning objective.

objective [default=reg:squarederror] It defines the loss function to be minimized. Most commonly used values are given below:

- reg:squarederror : regression with squared loss.
- reg:squaredlogerror: regression with squared log loss. All input labels are required to be greater than -1.
- reg:logistic : logistic regression
- binary:logistic : logistic regression for binary classification, output probability
- binary:logitraw: logistic regression for binary classification, output score before logistic transformation
- binary:hinge : hinge loss for binary classification. This makes predictions of 0 or 1, rather than producing probabilities.
- multi:softmax : set XGBoost to do multiclass classification using the softmax objective, you also need to set num_class(number of classes)
- multi:softprob : same as softmax, but output a vector of ndata nclass, which can be further reshaped to ndata nclass matrix. The result contains predicted probability of each data point belonging to each class.

eval_metric [default according to objective]:

- The metric to be used for validation data.

- The default values are rmse for regression, error for classification and mean average precision for ranking.

seed [default=0]:

- The random number seed.
- This parameter is ignored in R package, use set.seed() instead.
- It can be used for generating reproducible results and also for parameter tuning.

5.3. Training XGBoost model

5.3.1. Resampling

We will combine oversampling with undersampling. As mentioned earlier, there are two types in the resampling process, bootstrap and cross validation. Both are good and help us solve the class imbalance problem while tuning hyperparameters.

We will discuss resampling in the following sections.

5.3.1.1. Bootstrap

We will combine oversampling with undersampling for a better result. The combinations in this project will be ADASYN and Instance Hardness Threshold, ADASYN and Repeated Edited Nearest Neighbors.

Input dataset will be represented below.

Table 5-2. Classes in orginal dataset

	Train	Validation	Test
0 (N)	48553	23918	18118
1 (F)	1516	707	556
2 (V)	3857	1931	1448
3 (Q)	421	220	162
4 (S)	4314	2117	1608

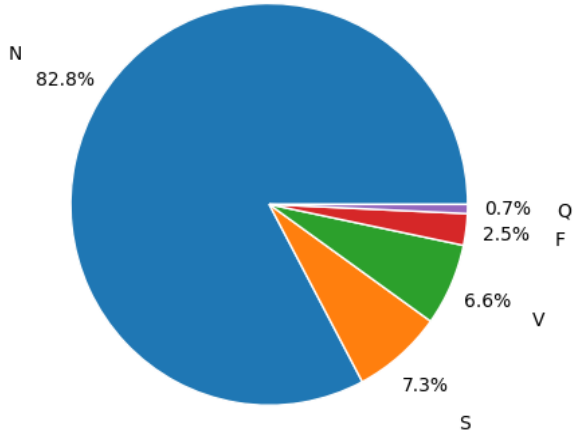


Figure 5-12. Class ratio before resampling

Use ADASYN to create more data points for class 1, 2, 3, 4. We will add up to 8000 for each class.

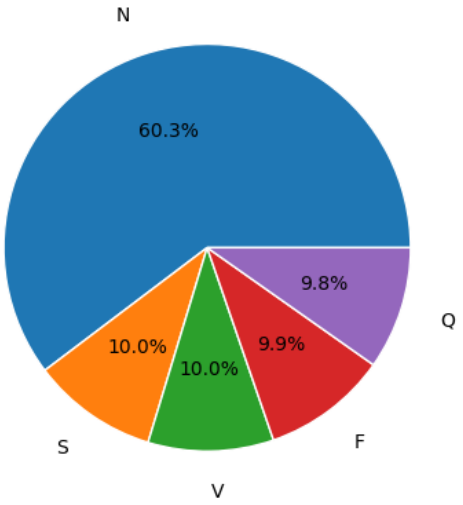


Figure 5-13. Class ratio after ADASYN sampling

After that, we will do Instance Hardness Threshold sampling process.

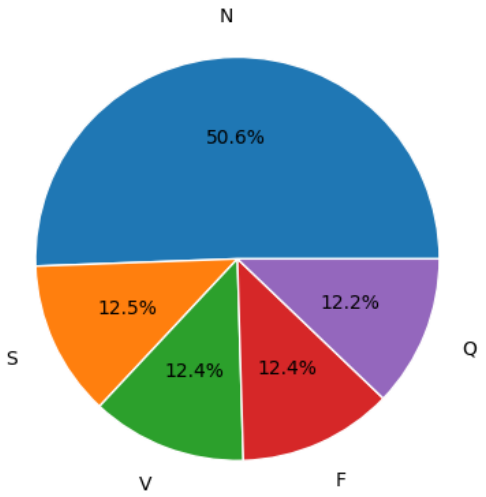


Figure 5-14. Class ratio after IHT

Then, we will sum all the samples in S, V, F, Q and name the class as abnormal in order to train binary classification.

Bootstrap has its weakness too, in some cases, the resampled dataset might give a bad result after training. So I will create another dataset with ADASYN and Repeated Nearest Neighbors and compare it with the previous dataset. Repeat the oversampling with ADASYN step.

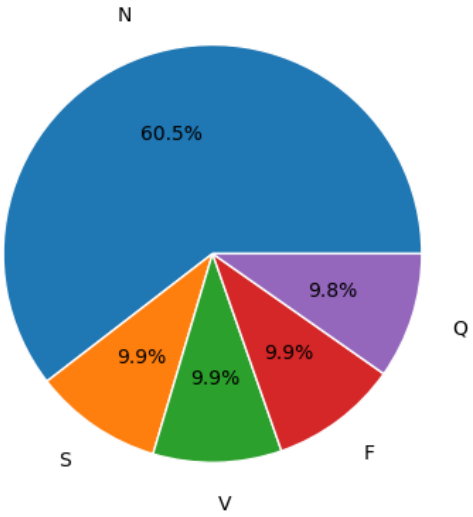


Figure 5-15. Oversampling again

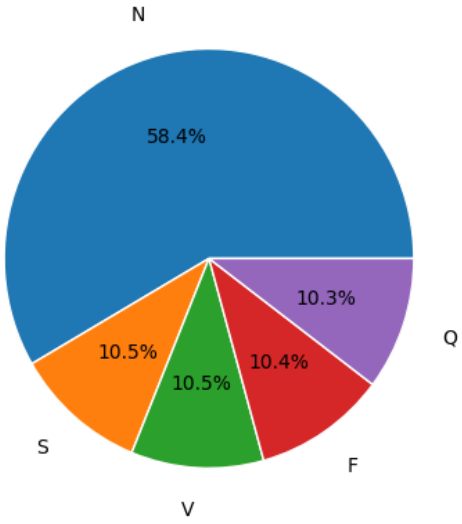


Figure 5-16. Dataset after undersampling with RENN

We will train XGBoost model with the resampled datasets, and group classes into 2 classes that are normal (N) case and abnormal (S, V, F, Q) case, use AUC metric to validate the training progress. Finally, we use F1-score on testset for camparison.

5.3.1.2. Cross Validation

In section 3.4.4.1, I briefly introduce cross validation method for solving class imbalanced problem. But it is just a simple method which will lead to overfitting, we should use nested cross-validation to prevent overfitting.

In nested cross validation, we have a double loop instead of one loop. An outer loop (that will serve for assessing the quality of the model), and an inner loop (that will serve for model/parameter selection). Those loops must be independent, so each step or layer of cross-validation does one and only one thing.

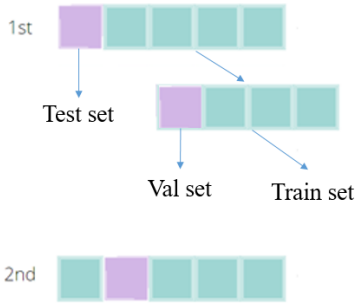


Figure 5-17. Nested cross-validation

In **Figure 5-7**, the outer loop is repeated 5 times, generating five different test sets. And for each iteration, the outer train set will be further split (in this case, into 2 folds). If there are 2 outer folds and 5 inner folds, as in the picture, the total number of trained models will be 10.

In this method, each model will learn various aspects of the dataset as the *k* fold increases. The outer layer will be used for estimating the quality of the models trained on the inner layer. In other word, the model returned from searching algorithm is then evaluated using the outer fold. This process is repeated *k* times (equal the number of folds in inner loop), and the final score is computed by taking the mean of all *k* scores (in this case *k* is 5).

In this project, outer loop will repeat 10 times and inner loop will also repeat 10 times according to Dr. Frank E. Harrell's comment in datamethods social community [..].

5.3.2. Tuning hyperparameters

During the training process, the model will not give the best result in most cases if we use default hyperparameters. Because a model can be used in many problems, such as the XGBoost algorithm can be used as a classifier, regression model or learning to rank model. We need to tune parameters for a specific usage scenario.

Firstly, we must know hyperparameters that can be tuned in xgboost library in python. Then we can use hand–tuning or experimentation, two ways to tune a model. Hand–tuning is a sequence that we modify hyperparameters manually and then write the record of each run. At experimentation, automated tuning refers to defining the optimal hyperparameters via a reproducible tuning approach or a loop of the tuning process. Many algorithms help us find the local or global minimum of the loss function.

We will detail the xgboost hyperparameters and wandb configuration used for tuning. XGBoost hyperparameters are used for tuning in this project:

- learning_rate
- n_estimators
- max_depth
- gamma

- min_child_weight
- colsample_bytree
- reg_alpha
- reg_lambda

Wandb sweep configuration example:

```
sweep_configuration = {
    'method': 'random',
    'name': 'sweep',
    'metric': {
        'goal': 'minimize',
        'name': 'validation_loss'
      },
    'parameters': {
        'batch_size': {'values': [16, 32, 64]},
        'epochs': {'values': [5, 10, 15]},
        'lr': {'max': 0.1, 'min': 0.0001}
     }
}
```

- method: Specify the search strategy.
- name: The name of the sweep, displayed in the W&B UI.
- metric: Specify the metric to optimize (only used by certain search strategies and stopping criteria).
- parameters: Specify parameters bounds to search.

Sweep	State	Created ↑	Creator	Run count	Est. Runs	Compute time
ADASYN-RENN sweep	Finished	11 hours ago	betelegeuse	100		2 hours
ADASYN-Instance Hardness Threshold sweep	Finished	13 hours ago	betelegeuse	100		1 hour
Nested Cross Validation sweep	Finished	16 hours ago	betelegeuse	100		10 hours

Figure 5-18. WanDB sweeps from 3 tuning process

Figure 5-1 shows three tuning processes with 300 run counts in total. The Nested Cross Validation sweep is the longest run because many models are needed to train. Because the size of the dataset resampled from the ADASYN – IHT method is smaller than the ADASYN – RENN method, the ADASYN – Instance Hardness Threshold sweep is faster than the ADASYN – RENN sweep, and it is the fastest sweep.

For monitoring, there are three plots in WanDB UI: scatter plot, Importance – Correlation, and Parallel Coordinates.

Scatter plots show F1-score and Validation AUC for each hyperparameter set.

Importance – Correlation, Correlation is the linear correlation between the chosen metric and hyperparameter. The high correlation score means that when the parameter has a higher value, the metric also has higher values and vice versa. Importance metric is calculated by a random forest model with inputs is the hyperparameters and the metric as the target output, then report the feature importance values for the random forest.

The final plot is Parallel Coordinates, and it summarizes the relationship between large numbers of hyperparameters and model metrics at a glance.

List of hyperparameters that score the best F1-score on the testset in each sweep:

Table 5-3. Hyperparameters of each sweep

	ADASYN – RENN	**ADASYN – IHT**	**Nested** CV
learning_rate	0.2394	0.2954	0.1429
n_estimators	337	970	655
max_depth	10	9	10
gamma	0.1379	0.2474	0.001828
min_child_weight	8.022	1.031	0.03738
colsample_bytree	0.7224	0.8874	0.8119
reg_alpha	0.0009261	0.0000655	0.0006148
reg_lambda	0.0006745	0.0004105	0.000223

As we can see from **Table 5-1**, hyperparameters are not the same in three sweeps, each of hyperparameter set gives a local minimum in the loss function.

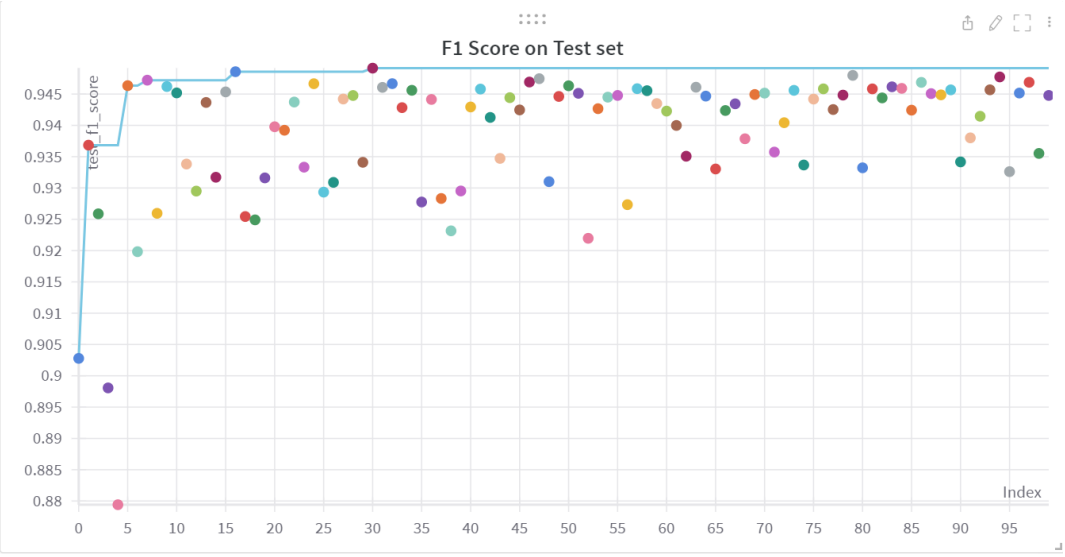


Figure 5-19. ADASYN – RENN F1-score

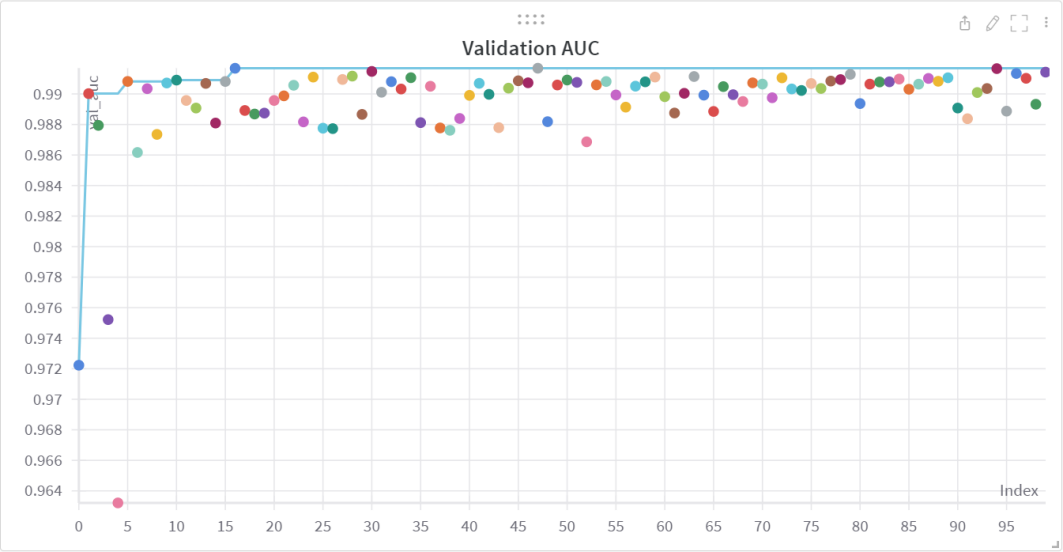


Figure 5-20. ADASYN – RENN Validation AUC

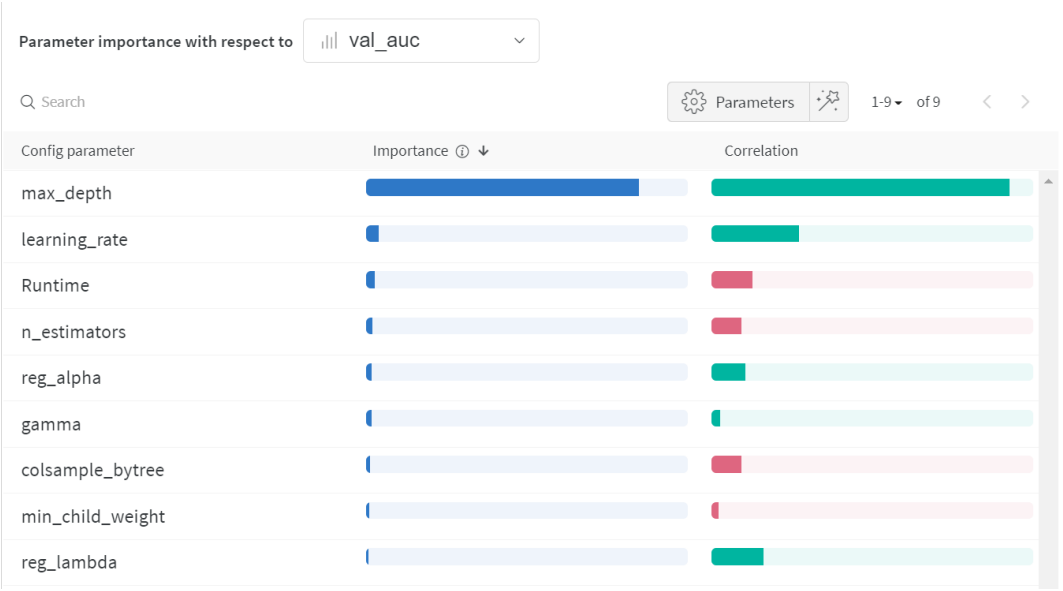


Figure 5-21. ADASYN – RENN Importance – Correlation

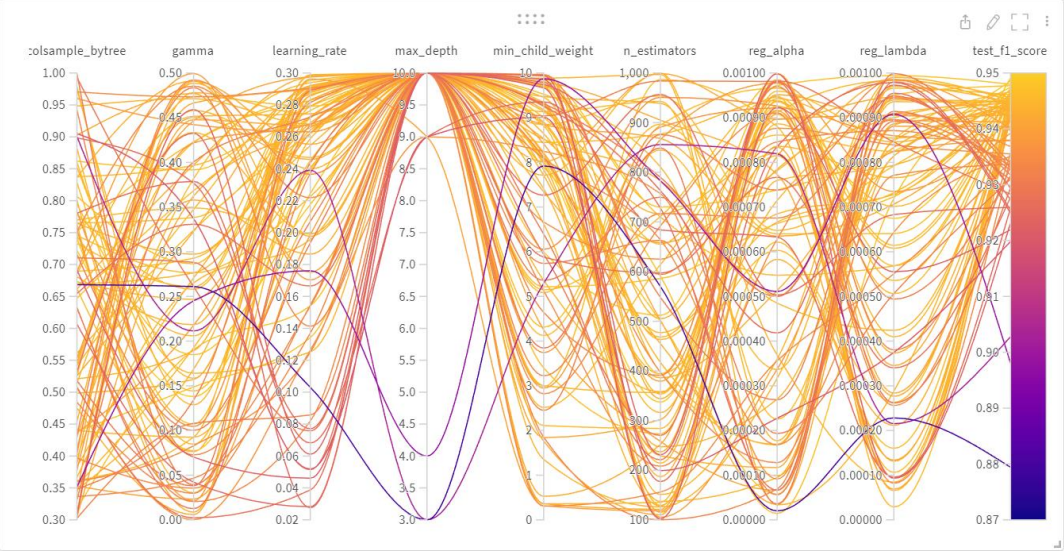


Figure 5-22. ADASYN – RENN F1-score Parallel Coordinates

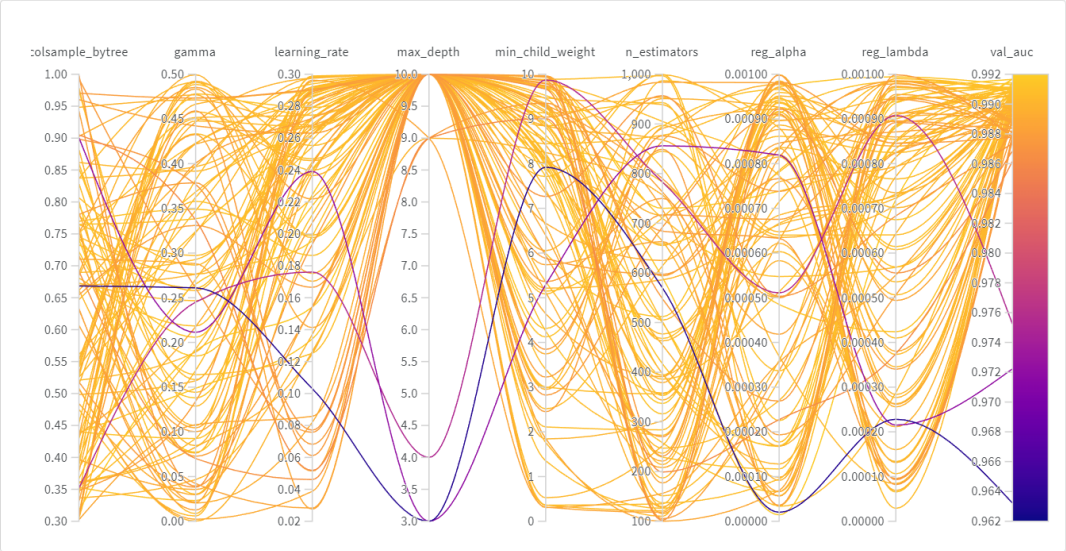


Figure 5-23. ADASYN – RENN Validation AUC Parallel Coordinates

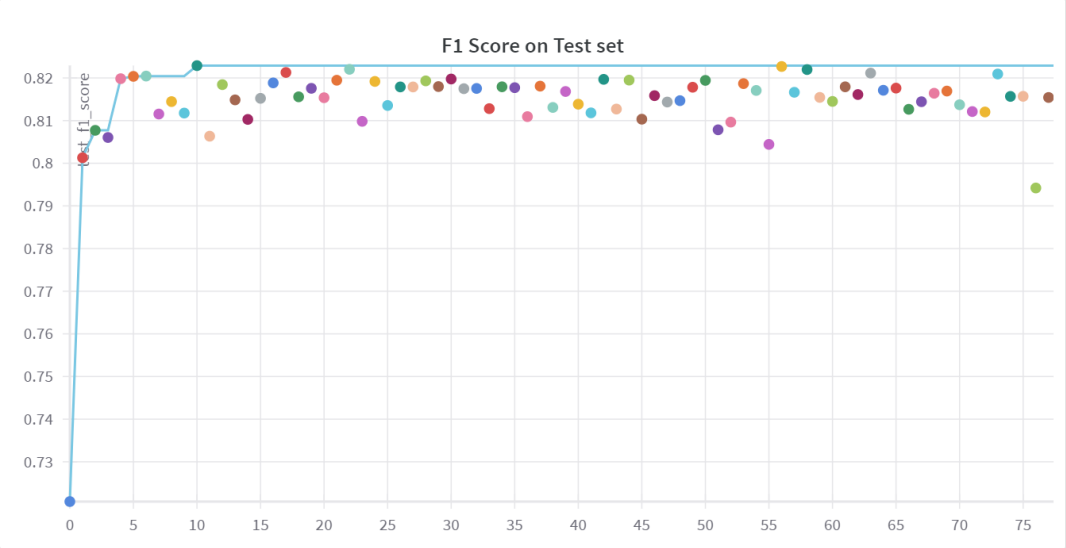


Figure 5-24. ADASYN – IHT F1-score

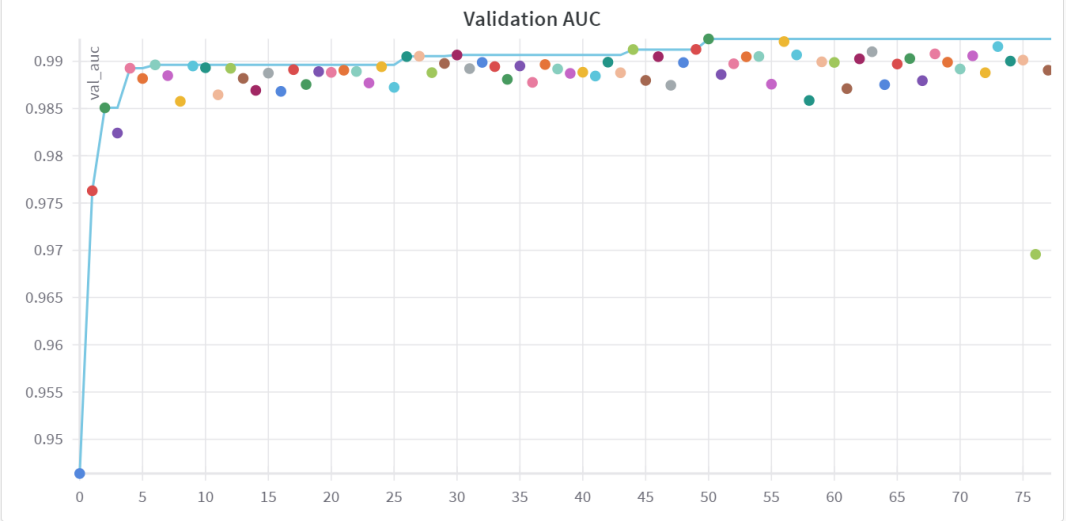


Figure 5-25. ADASYN – IHT Validation AUC

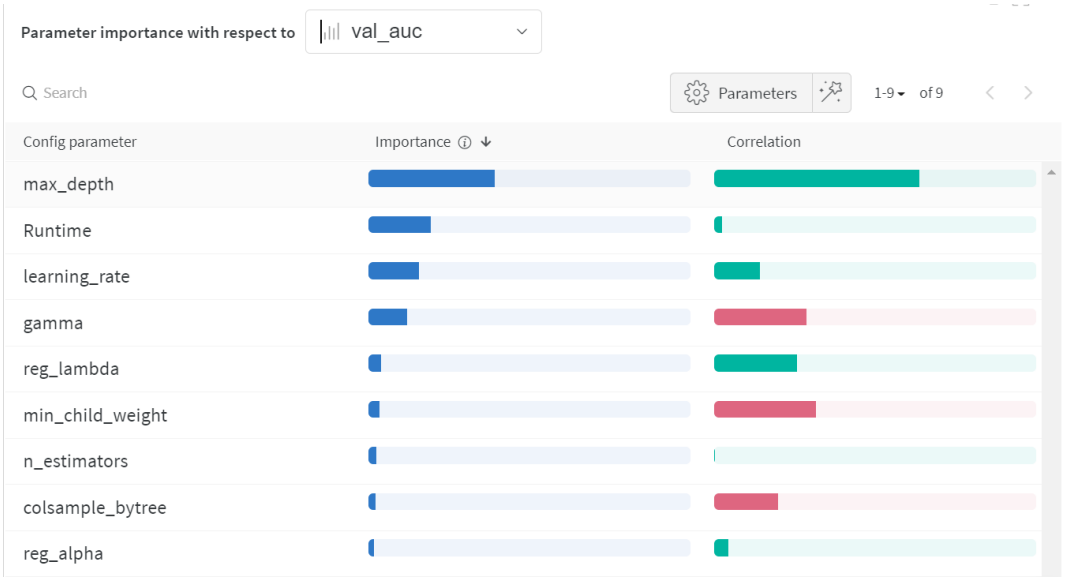


Figure 5-26. ADASYN – IHT Importance – Correlation

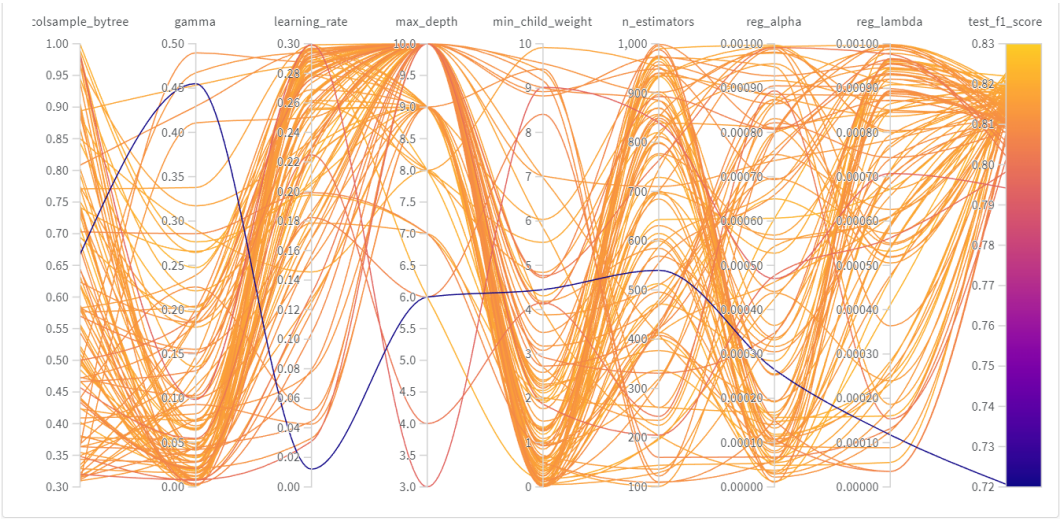


Figure 5-27. ADASYN – IHT F1-score Parallel Coordinates

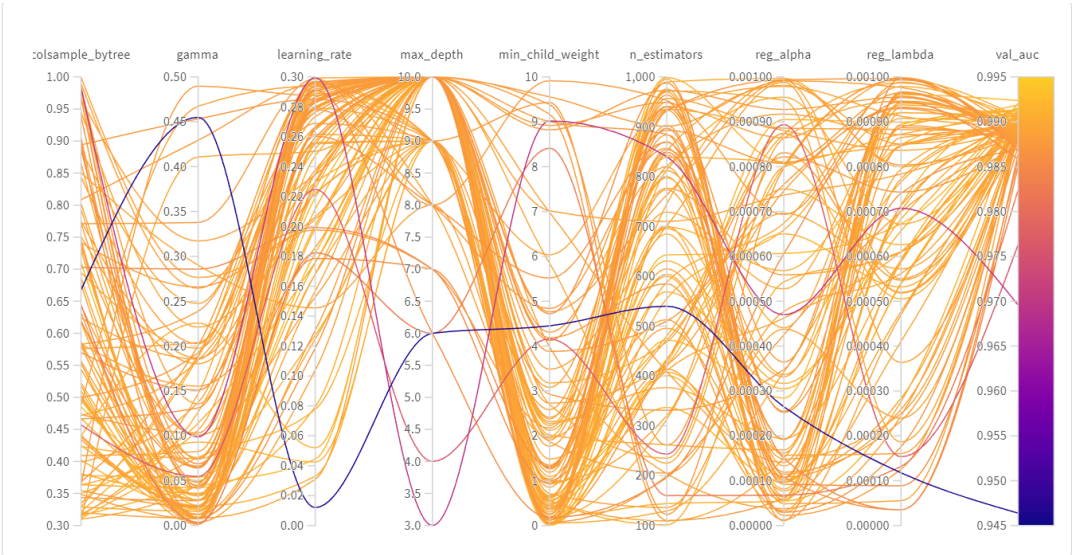


Figure 5-28. ADASYN – IHT Validation AUC Parallel Coordinates

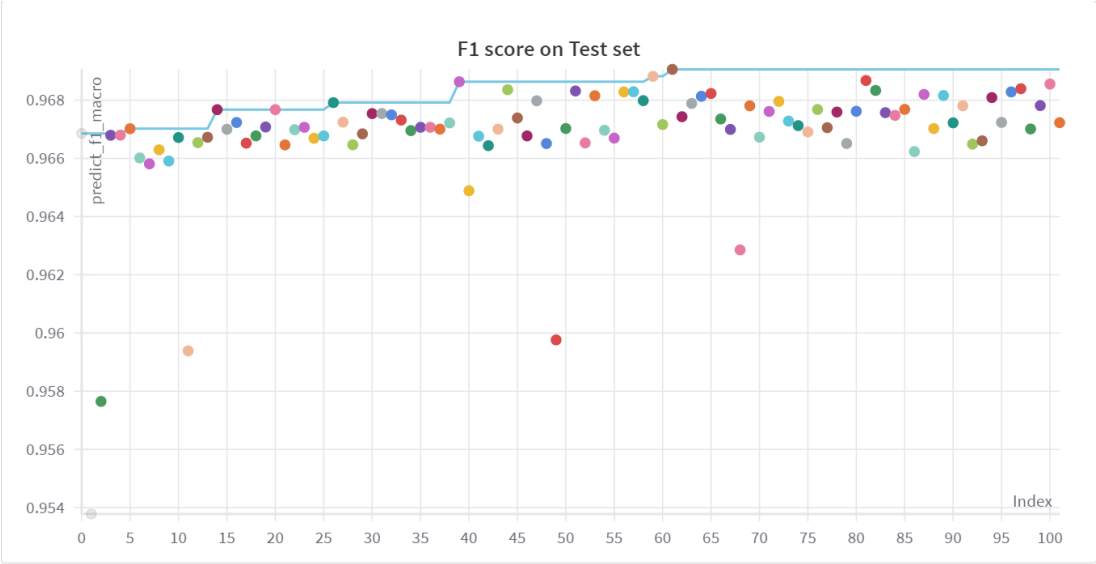


Figure 5-29. Nested CV F1-score

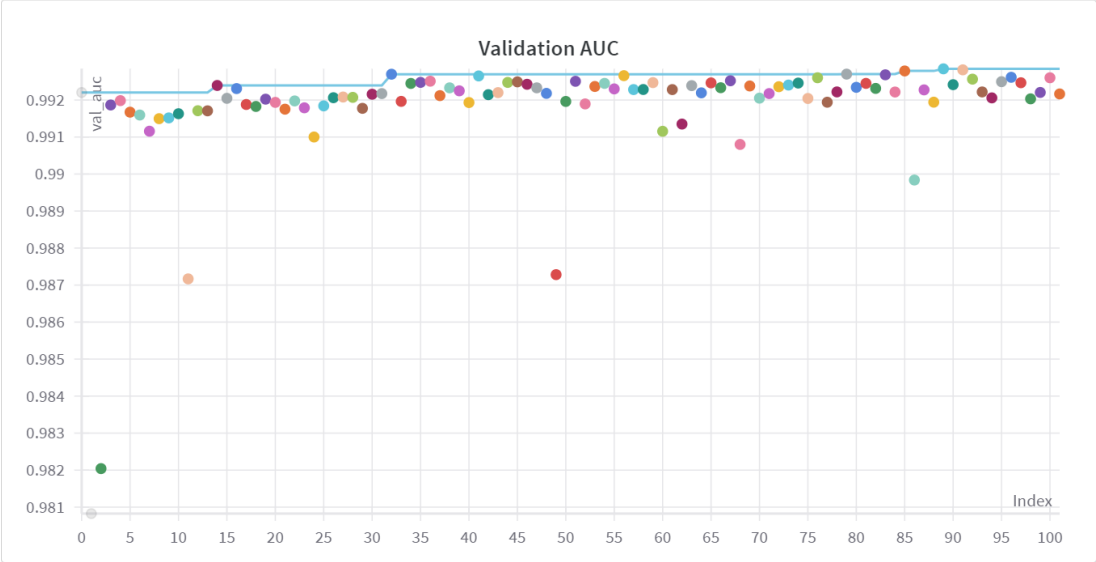


Figure 5-30. Nested CV Validation AUC

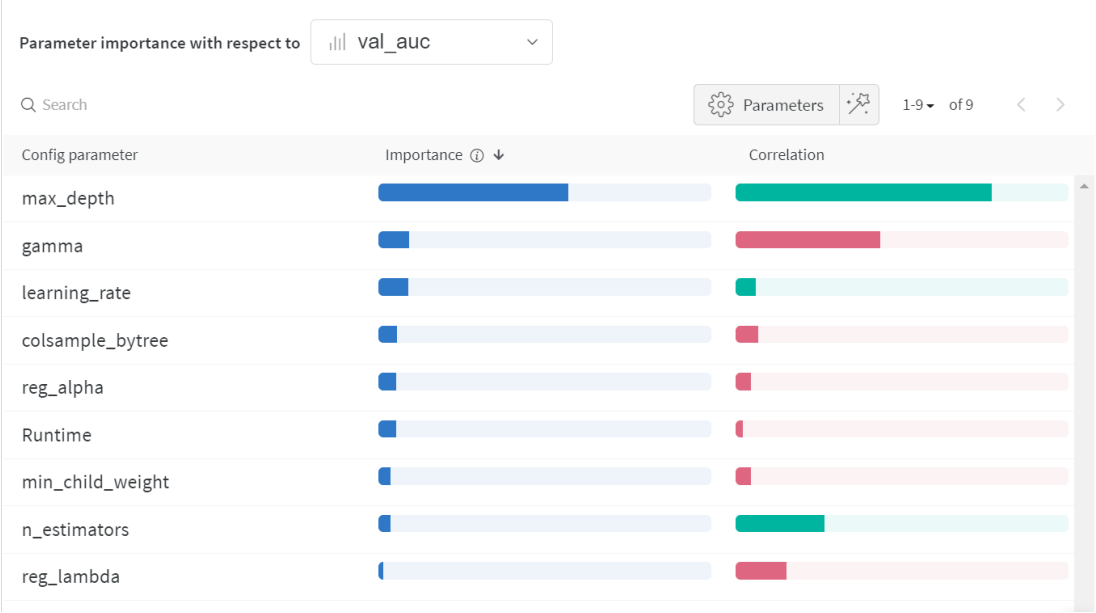


Figure 5-31. Nested CV Importance – Correlation

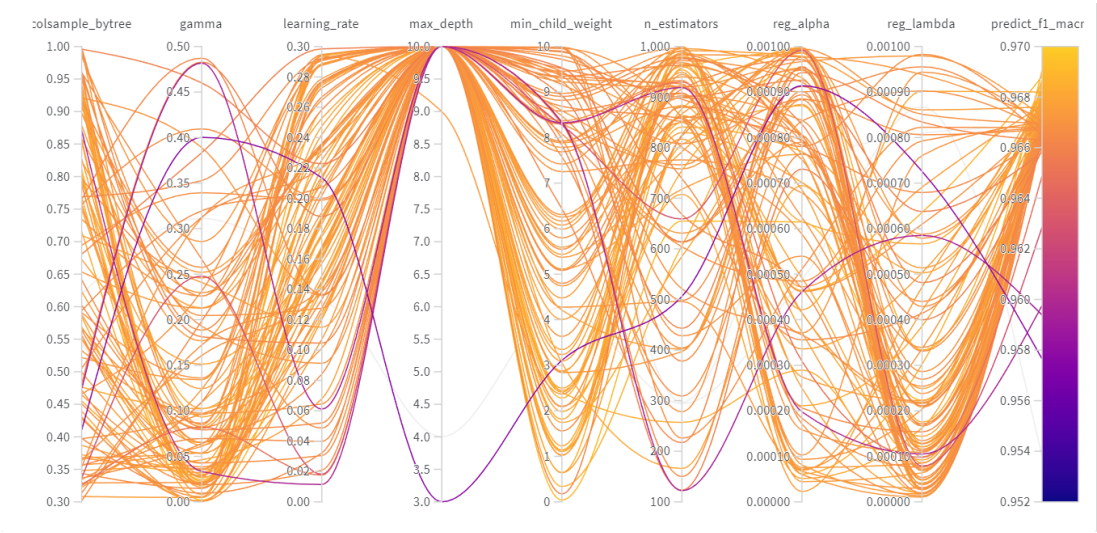


Figure 5-32. Nested CV F1-score Parallel Coordinates

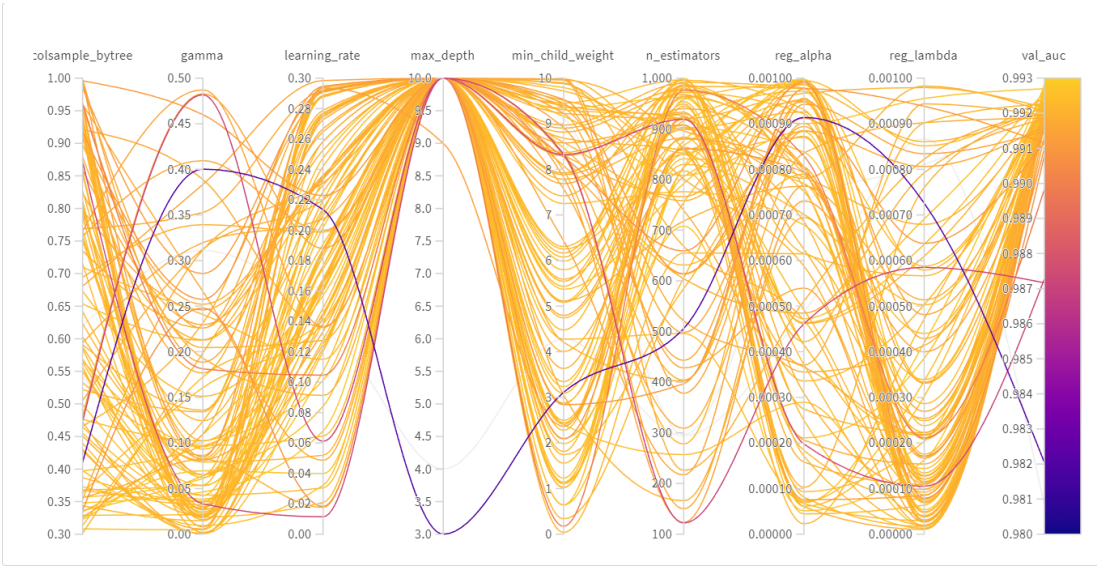


Figure 5-33. Nested CV Validation AUC Parallel Coordinates

CHAPTER 6: MODEL DEPLOYMENT

6.1. Configuration

There are two config files in this project, one is used for storing information for XGBoost model process, the last one is used for checking update process.

The following script is xgboost model process config file.

```
{
    "parameters": {
        "model": {
            "name": "ecqdsx9FvO.model",
            "path": "volume"
        },
        "influxdb": {
            "host": "192.168.50.225",
            "username": "",
            "password": "",
            "database": "iot_device"
        },
        "sqlite": {
            "name": "20221216.db",
            "path": "volume"
        }
    },
    "task": {
        "load_new_model": {
            "load_timestamp": "2022-12-13T 16:15:59.916993Z",
            "save_path": "volume"
        }
    }
}
```

The key “parameters” is storing InfluxDB and SQLite parameters. They are used to create a connect to 2 databases. While key “task” lists all main task of the program, in this case is updating new model task. And here is the update model config file:

```
{
  "parameters": {
    "google_api": {
      "client_secret_file": "client_secret.json",
      "api_name": "drive",
      "api_version": "v3",
      "scopes": ["https://www.googleapis.com/auth/drive"],
      "folder_id": {
        "model_version_launch": ""
      }
    },
    "sqlite": {
      "name": "20221216.db",
      "path": "volume"
    }
  },
  "task": {
    "update": {
      "update_timestamp": "2022-12-13T 16:15:59.916993Z"
    }
  }
}
```

Similar to xgboost model process config file, there are two parent keys (parameters, task). Key “google_api” stores values which needed to request to google api for drive. In “task”, key “update_timestamp” of key “update” indicates next timestamp that the program needs to check for updating.

6.2. Flowchart

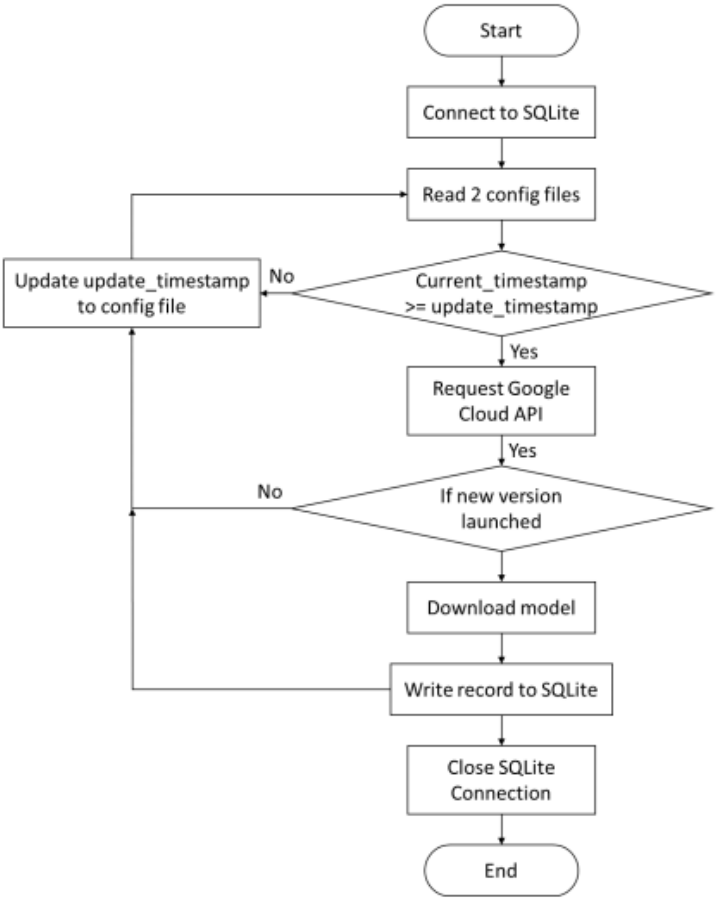


Figure 6-1. Update new model program

First, the program is going to create a connection to SQLite database. Then it reads 2 config files, which are used for storing values for update new model and classification program (update_config.json and model_config.json), to retrieve parameters.

If update_timestamp is less than current timestamp, the program will set update_timestamp to a future timestamp, then it will continue the loop. If update_timestamp is “overdue”, a request will be sent to google api, then google api will reponse some information that stored in google drive.

From the received information, the program will find out if new version is launched or not. If not, the program comeback to the infinite loop to check for new model. If new version is launched, the program will download that model and write record to SQLite.

Finally, update new model program will continue the loop or close the connection to SQLite and stop.

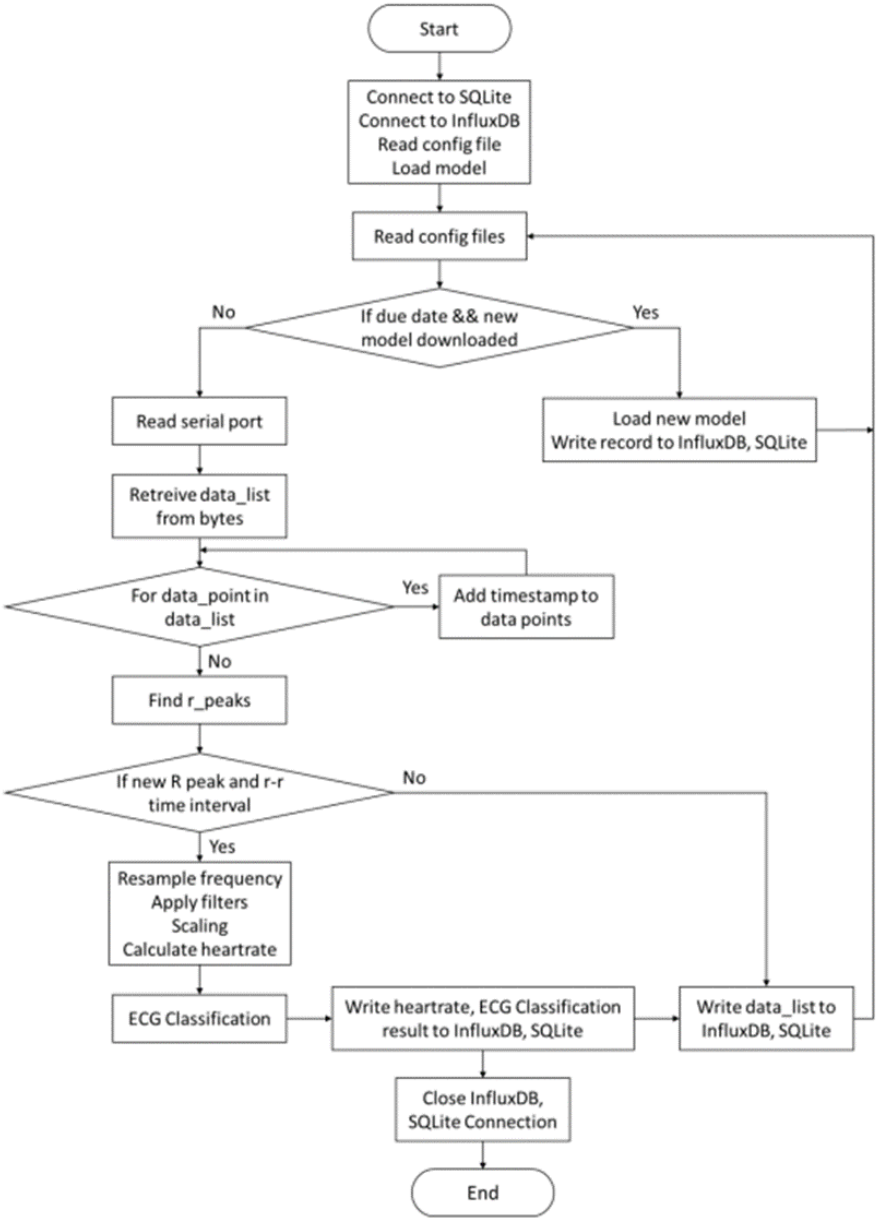


Figure 6-2. Classification program

Similar to update new model program, classification program initially reads 2 config files and create connections to databases, InfluxDB and SQLite. Then it need to reread those config files in the loop, because when the load new model process happens, config files will be updated.

After received information from many resources, this program is going to check if current timestamp is the deadline to load new model. If new model is downloaded, the program will load that model and write record to SQLite, InfluxDB and config files. If not, the program will reads serial port to get data in byte format. After successfully retreive data points from bytes, we need to add timestamp to each data point in order to write record to InfluxDB.

Then, classification program will seek for new R peak in ECG signal. If there is no new R peak, it will write ECG values into InfluxDB and SQLite. If new R peak appears, the program moves to data engineering step, it will resample the frequency,

apply filters, scale value and caculate heart rate. After that, xgboost model participates in the progress, this model will give the ECG classification result. Then the program writes records to InfluxDB and SQLite.

Finally, it will continue the infinite loop or close the previous connections and stop.

6.3. Setup

6.3.1. Hardware comparasion

This section is not mentioned in chapter 3 because it is a practical step. It is based on specific requirements for each project. Some projects will be launched in cloud service, so we do not have to set up the environment.

I am trying to deploy machine learning model in single-board computers. I have tried to deploy the model on single-board computers, as shown in **Figure 6-3**.

a. Orange Pi zero 2 **b. Raspberry Pi 3B** **c. Jetson Nano DevKit**

Figure 6-3. Single-board computers

There are specifications for each board, which will help us to know that board in detail. With that help, we can go for Jetson Nano DevKit, the most potential board on the list. However, why am I trying to deploy on many boards?

The main reason is optimization. In production, the hardware always has its limitation, so our goal is to deploy on available hardware or resources successfully. Firstly, I will deploy on mini pcs listed above, and then we will compare result in section 6.3.3. The list below will show some specifications that highly affect the AI model performance.

- Orange Pi zero 2:

AI Performance: 42 GFLOPs.

CPU: Allwinner H616 64-bit high-performance Quad-core Cortex-A53 processor.

GPU: Mali G31 MP2.

RAM: 1GB LPDDR3.

- Raspberry Pi 3 model B:

AI Performance: 48 GFLOPs.

CPU: Quad Core 1.2GHz Broadcom BCM2837 64bit.

GPU: 400 MHz VideoCore IV.

RAM: 1GB LPDDR2.

- Jetson Nano Developer Kit:

AI Performance: 472 GFLOPs.

CPU: Quad-core ARM A57 @1.43 GHz.

GPU: 128-core Maxwell™ GPU.

RAM: 4 GB LPDDR4.

A computer system with 1 gigaFLOPS (GFLOPS) can do one billion (109) floating-point operations per second. To equal the performance of a 1 GFLOPS computer system in one second, you'd have to complete one computation per second for 31.69 years. The bigger GFLOPS, the higher performance. It generally indicates that Jetson Nano DevKit is the best in this list, Raspberry Pi 3 model B is in the middle and Orange Pi zero 2 is the last one. But, we should know other specifications in detail.

Obviously, Ram is the most basis information that we can compare without in-depth knowledge. The numbers are listed previously is the size of the memory. In general, Jetson Nano DevKit has the most powerful memory, RPi3's RAM is the most ineffective one. However, there are some differences in their features, the table below lists these contrastive key features.

Table 6-1. LPDDR Key features comparison

Features	LPDDR4	LPDDR3	LPDDR2
Supply Voltage (V)	1.1	1.2	1.8
Speed (Mbps)	3200/4267	1600/2133	800/1066
Prefetch	16 bit	8 bit	4/2 bit
Burst Length	16 & 32	8 only	16, 8 & 4

DRAM memories are the "heart" of any computing device, such as smartphones, laptop computers, and servers. The LPDDR series was created primarily to improve memory performance and efficiency in small computer systems.

LPDDR4 memory architecture has been updated to obtain better bandwidth and reduced power consumption, which is a major demand of today's computer systems. By decreasing the supply voltage while expanding the bandwidth, it has lowered power consumption.

The purpose of prefetching is to make data available in the cache before the data consumer puts its request, disguising the latency of the slower data source below the cache. With 16-bit prefetch, Jetson Nano's I/O Frequency is greater compare to the others. The LPDDR2 has 4-bit or 2-bit prefetch depending on the version.

When a read/write command is given to the controller, the term "burst length" describes the volume of data that is read or written. High burst length value effectively reduces the latency for read/write operations..

The LPDDR3 is in the middle and the LPDDR2 is the least powerful architecture.

Then, let’s compare three cpus above [2] – [9].

Table 6-2. CPUs comparison

Specs	Broadcom BCM2837	Allwinner H616	Jetson Nano’s CPU
ISA	ARMv8-A (64-bit)	ARMv8-A (64-bit)	ARMv8-A (64-bit)
Microarchitecture	Cortex-A53	Cortex-A53	Cortex-A57
Lithography	28 nm	28 nm	-
Cores	4	4	4
Threads	4	4	4
Frequency	1.2 GHz	1.5 GHz	1.43 GHz
L1 cache (I-cache)	16 KB per core	32 KB per core	48 KB per core
L1 cache (D-cache)	16 KB per core	32 KB per core	32 KB per core
L2 cache (unified)	512 KB (shared)	512 (shared)	2 MB (shared)
L3 cache	None	None	None

As we can see, they are in the Cortex-A50 series. [4] shows that they are not literally different to each other. In some aspects, however, if we compare D-cache, Raspberry Pi 3B has the lowest size, when doing something related to manage data in cpu, we need to read/write or delete it properly in order to run out of cache. On other hand, Jetson Nano has largest cache size in both L1 cache and L2 cache. Or considering the cpu capacity (cpu frequency), which indicates how rapidly a CPU can process data, Allwinner H616 wins the race.

Lithography, often known as the manufacturing process, is used to measure the transistors that comprise a CPU. A CPU has billions of transistors that execute calculations by switching on and off electrical signals. In technology, a processor's nanometer is also known as a process node, or simply node. Lower nm is preferable for a machine. OPi zero 2 and RPi 3B have the same lithography, we can’t count Jetson Nano in because Nvidia does not provide information.

Overall, we can say Allwinner H616 and Jetson Nano’s CPU is better than Broadcom BCM2837. We cannot compare Allwinner H616 and Jetson Nano’s CPU due to the lack of information.

Finally, let’s dive into gpu specifications [2] – [9].

The GM20B in the Jetson Nano DevKit is believed to be superior based on performance metrics. The number of execution units and shading units utilized for

rendering in the Mali G21 MP2 is half that of the VideoCore IV's. However, the difference in performance metrics between the Mali G31 MP2 and VideoCore IV is only 6 GFLOPS.

Table 6-3. GPUs comparison

Specs	Mali-G31 MP2	VideoCore IV	Jetson Nano's GPU
Architecture	Bifrost	-	Maxwell 2.0
Release date	Q1 2018	Q1 2010	Q3 2016
Base clock	650 MHz	250 MHz	640 MHz
Boost clock	-	400 MHz	921 MHz
Execution units	2	4	-
Shading units	32	64	128
Performance	42 GFLOPS	48 GFLOPS	472 GFLOPS

This is due to the fact that the base clock of the OPi zero 2 is significantly better than the boost clock of the RPi 3B, indicating the speed at which a GPU can complete a task.

Additionally, the VideoCore IV was released in Q1 2010, while the Mali G31 MP2 was launched in Q1 2018. This means that the Bifrost architecture is a more advanced technology compared to the unnamed architecture of the VideoCore IV.

In other words, the superior performance is a result of quality over quantity. It is important to note that this section cannot rank these computers, but in a later section, tasks will be performed on each single board, and they will be ranked based on their output and performance.

6.3.2. Install OS and python libraries

There are several operating systems available for single-board computers, some of which are supported or provided by the organizations that manufacture the boards. For example, the Raspberry Pi 3 model B is equipped with a 64-bit CPU. While there are numerous 64-bit Linux distributions (such as SUSE, Arch, and Fedora) that have images available for the RPi3, many issues may still remain unresolved. Therefore, one may need to try and test different solutions to find the best one.

I have attempted to use various operating systems such as Raspian, Orange Pi OS, Jetson SDK, and Armbian. While some attempts were unsuccessful, I was ultimately able to deploy my model on a single-board computer.

To install an operating system on an SD card, I use balenaEtcher, an open-source and cross-platform tool that enables the flashing of OS images. This tool can assist in the installation of a variety of operating systems such as macOS, Windows, Armbian, Raspbian, and Orange Pi OS.

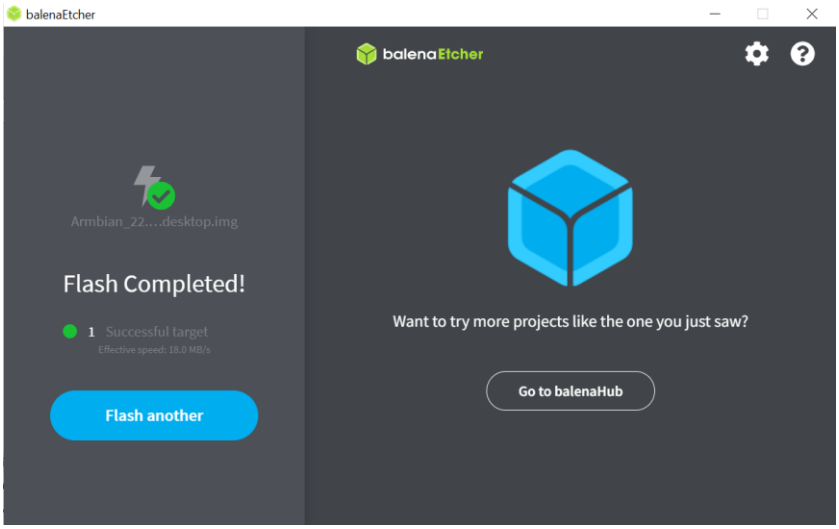


Figure 6-4. balenaEtcher GUI

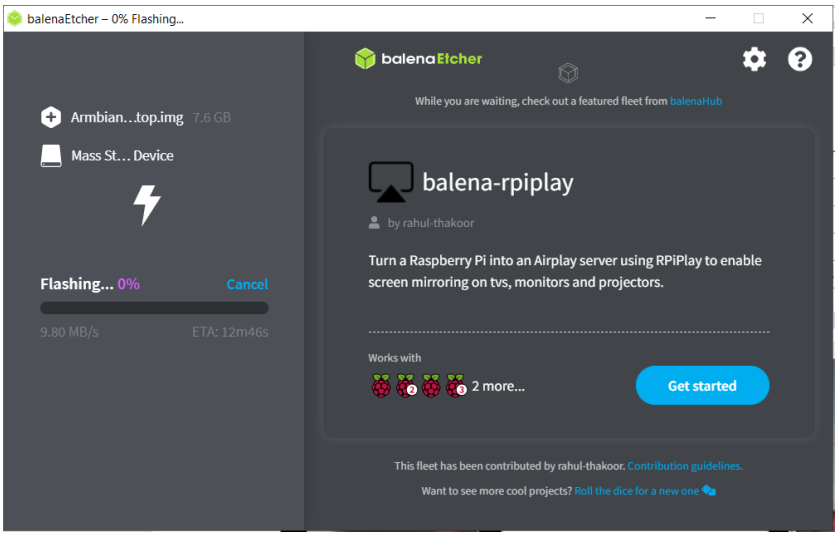


Figure 6-5. balenaEtcher Flashing progress

Then we can transfer files to single-board computers via VNC.

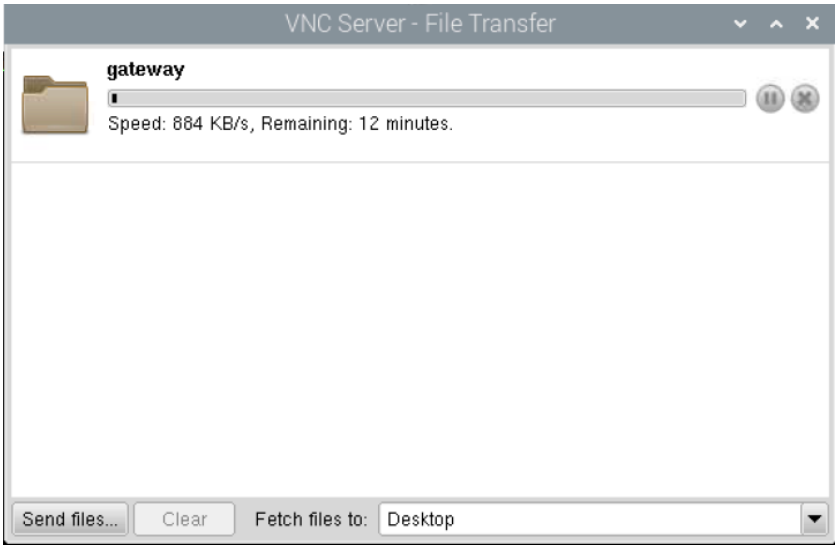


Figure 6-6. Transfer file via VNC Viewer

After that, we need to install python libraries. Here is some examples:

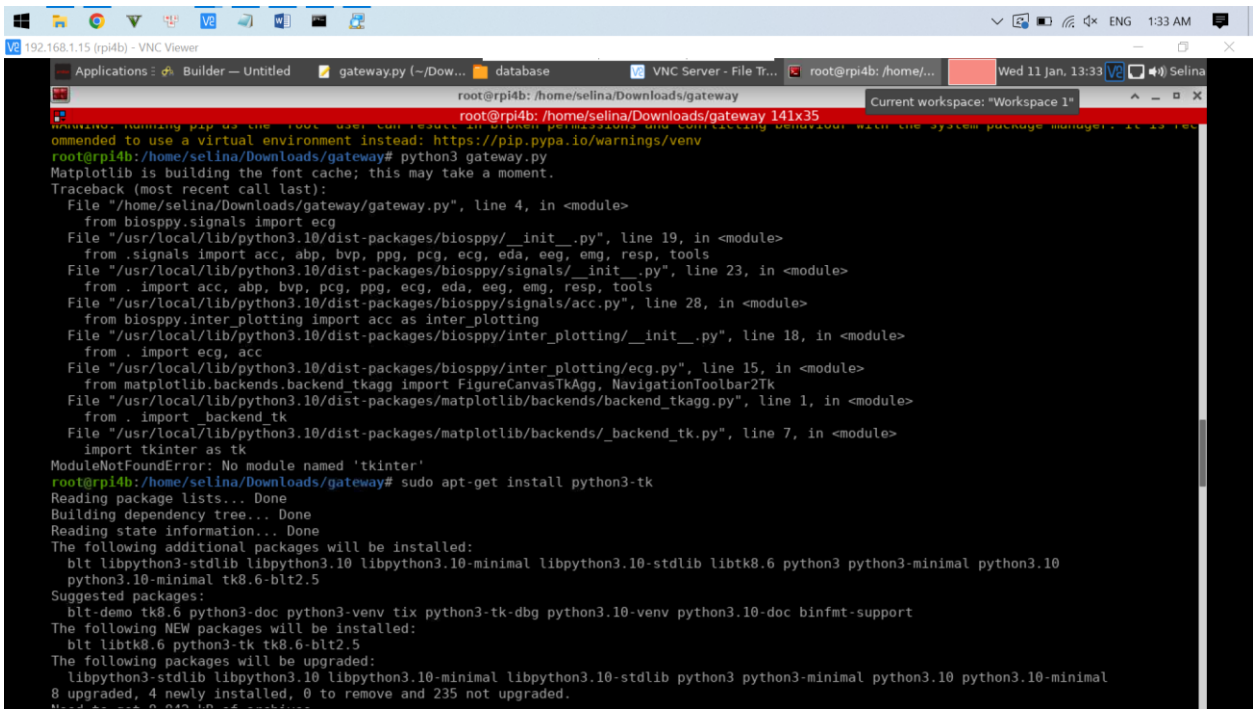


Figure 6-7. Install python3-tk package on RPi 3B

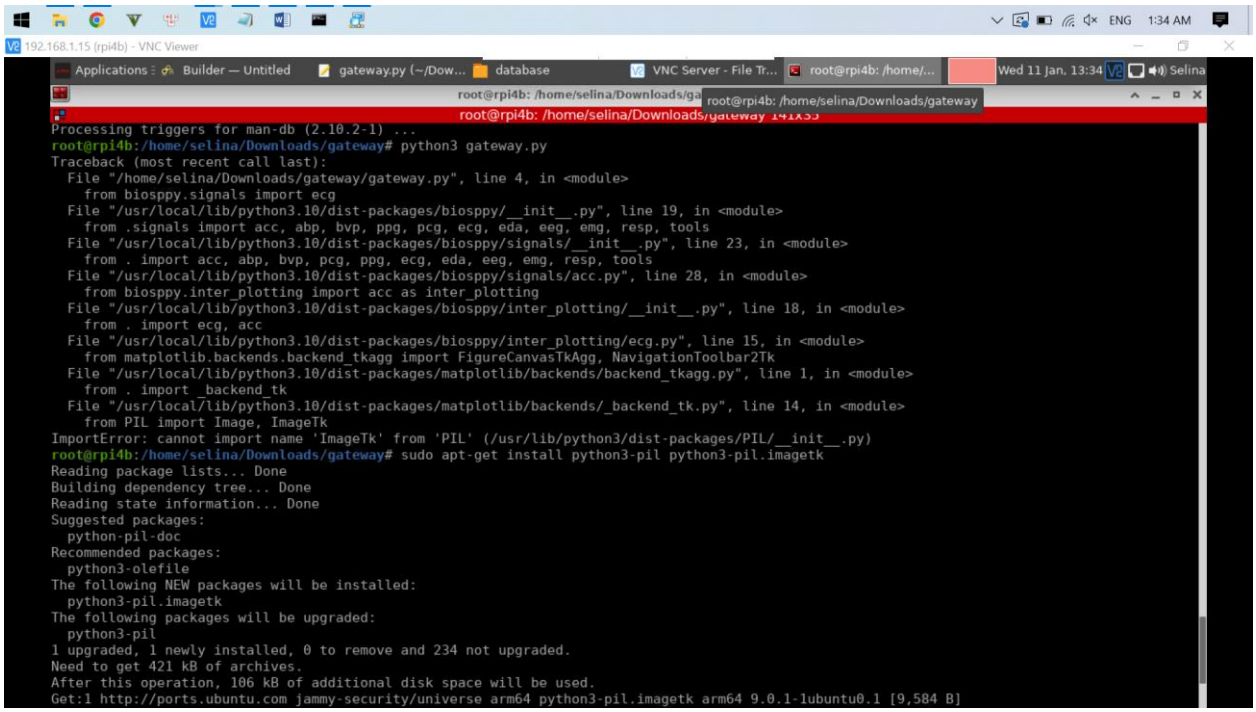


Figure 6-8. Install PIL library on RPi 3B

6.3.3. Optimization

After deploying machine learning model into single-board computers, let's compare their performance.

In the research stage, I use a computer with the following specs. I use this specs to see how a single-board computer runs AI model in realtime compare to a powerful computer.

Table 6-4. My PC specifications

CPU	Intel(R) Core(TM) i7-4770 CPU @ 3.40GHz
GPU	Nvidia Quadro K2200 4GB
RAM	16GB LPDDR3

How much time does xgboost model need to classify 21892 samples in test set? This metric shows the average time interval needed between each sample. The values below is recorded during 100 runs. The unit of time in **Table 6-5** is second.

Table 6-5. Compare time progressing

	Load model time				Inference time			
	PC	**OPi z2**	**RPi 3B**	**Jetson**	**PC**	**OPi z2**	**RPi 3B**	**Jetson**
Min	0.0429	0.0580	0.0736	0.0513	0.5027	2.3581	2.9535	1.7611
Max	0.0519	0.0656	0.0867	0.0722	0.6207	3.0433	3.4555	1,9957
Avg	0.0483	0.0591	0.0744	0.0655	0.5315	2.4855	3.1486	1.8465

The following table show the time needed to classify a sample.

Table 6-6. Milisecond per sample of single-board computers

	OPi z2	**RPi 3B**	**Jetson**
Min	0.1077	0.1349	0.0804
Max	0.1390	0.1578	0.0911
Avg	0.1135	0.1438	0.0843

We have **Table 7-6** by dividing inference time by 21892 samples.

From **Table 7-6**, Jetson Nano is the best and following by Orange Pi zero 2, than the worst is Raspberry Pi 3 model B. It is true compare to AI performance specification values shown in section 6.3.1, we can generally choose a single-board computer based on this metric. But we need to consider more aspects. When comparing load model time, Jetson Nano is slower than Orange Pi zero 2, that is strange. Why is it? Let's see some figures:

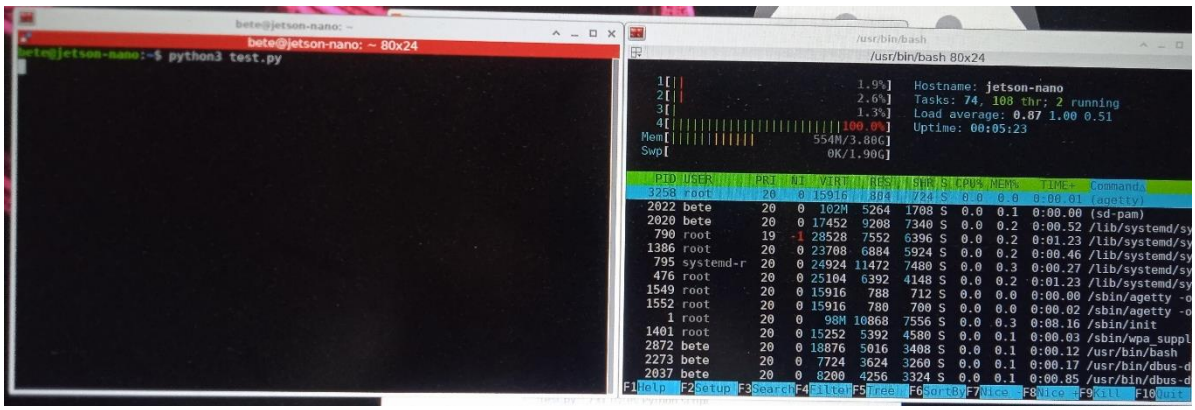


Figure 6-9. Jetson Nano loads model

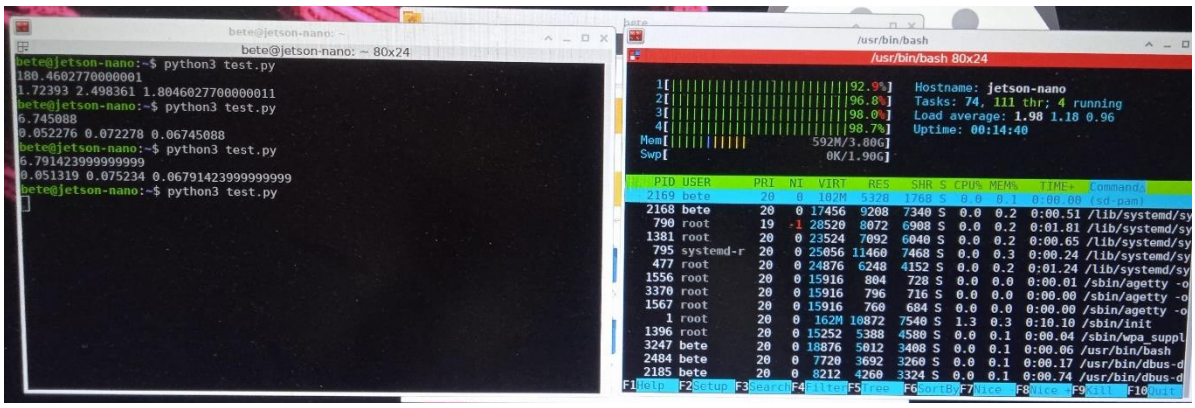


Figure 6-10. Jetson Nano classifies samples

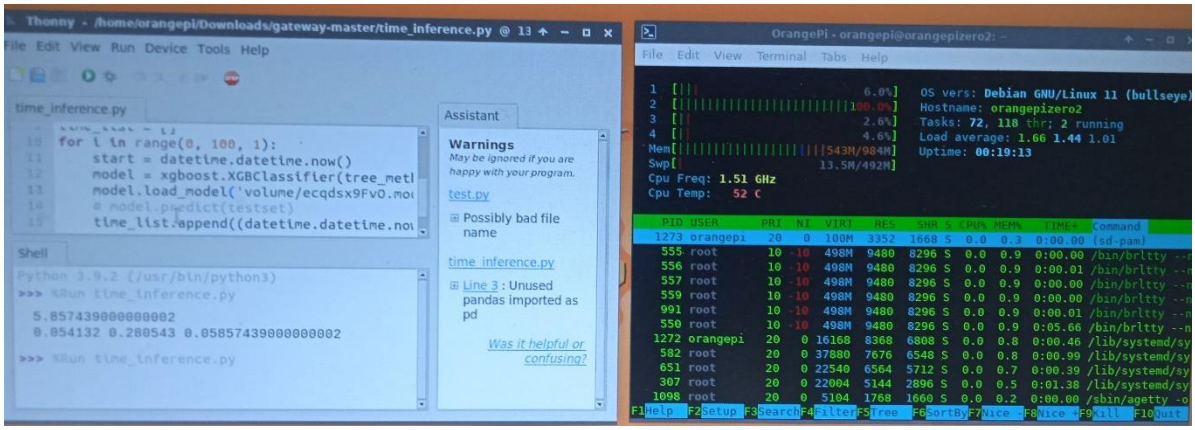


Figure 6-11. Orange Pi zero 2 loads model

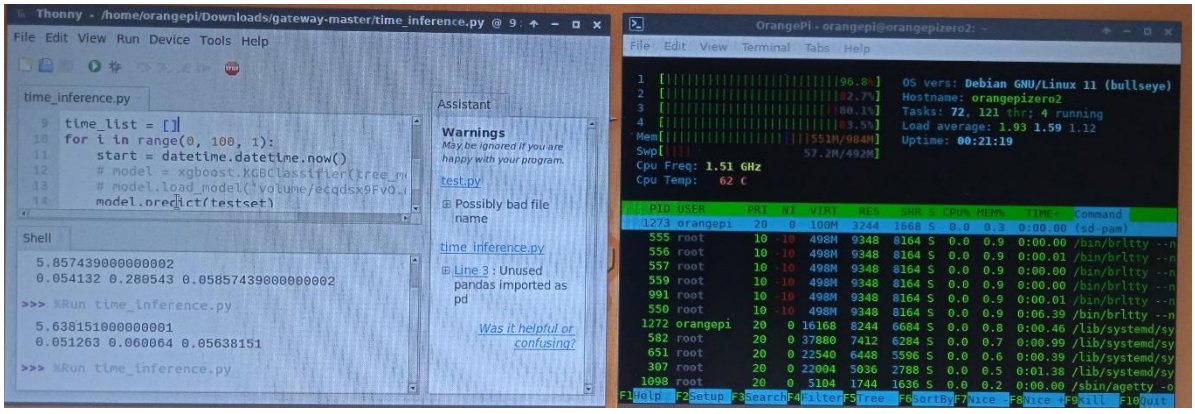


Figure 6-12. Orange Pi zero 2 classifies samples

When classifying samples, XGBoost likely keep the CPU cores fully occupied then the work is fed into GPU for acceleration]. Because Jetson Nano's GPU is the most powerful, Jetson Nano performs very quickly on classification. But, when we try to load a model, only CPU and RAM work. In both cases, two RAMs do not help much, as we can see each single-board computer uses approximately 0.5GB RAM, only one CPU core is fully occupied. We can conclude that Orange Pi zero 2's CPU is more capable of performance than Jetson Nano's. This is the reason why OPi zero 2 loads model more quickly.

Finally, we can rank these single-board computers for optimization:

Table 6-7. Ranking single-board computers

Name	Rank
Jetson Nano DevKit	1
Orange Pi zero 2	2
Raspberry Pi 3 model B	3

6.4. Deploy Machine learning model

Here is some images show the progress that I create docker image.

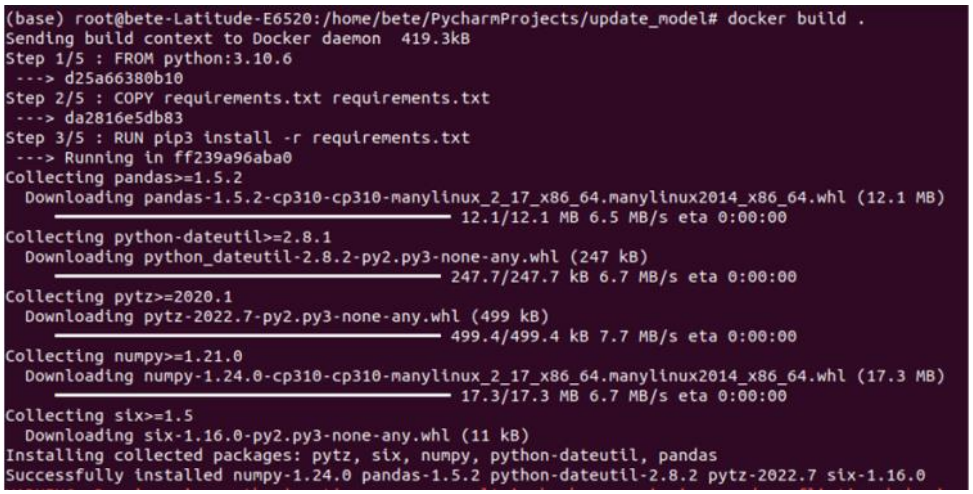


Figure 6-13. Create update model program image

When it done, there is a image created recently (18 seconds ago). It is our image for update model program.

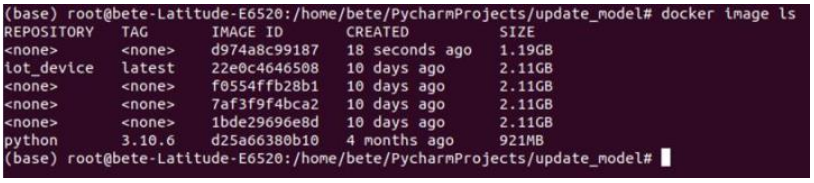


Figure 6-14. Result of create docker image in PC

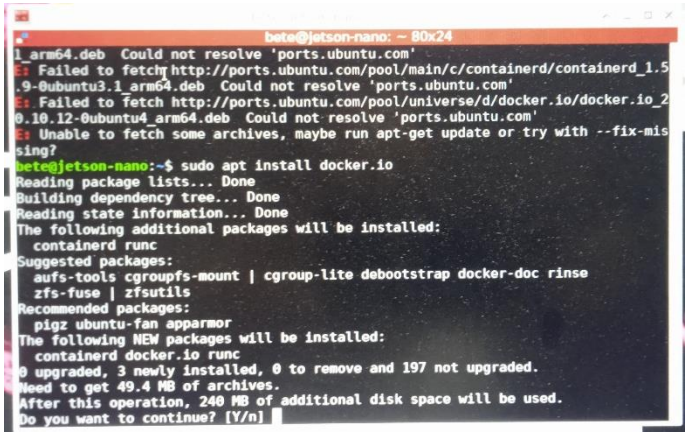


Figure 6-15. Install docker on Jetson Nano DevKit

Finally, we need to install docker in single-board computer and run docker image.

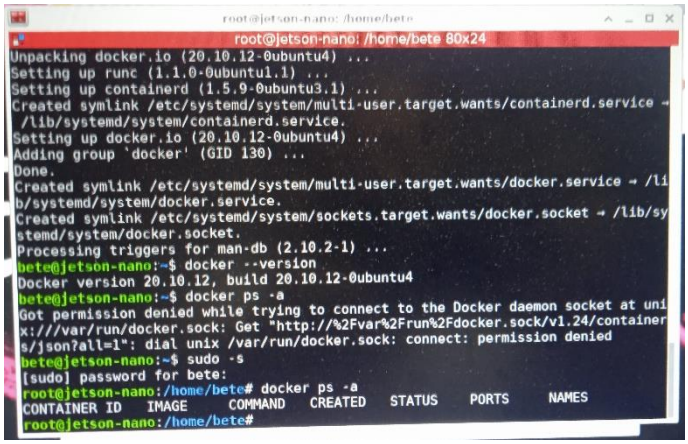


Figure 6-16. Install docker successfully

6.5. Retraining during deployment

6.5.1. Google drive api

Users can use the Google Drive API to develop apps that use Google Drive cloud storage. Using the Drive API, we can construct comprehensive functionality in our application and design apps that interface with Drive.

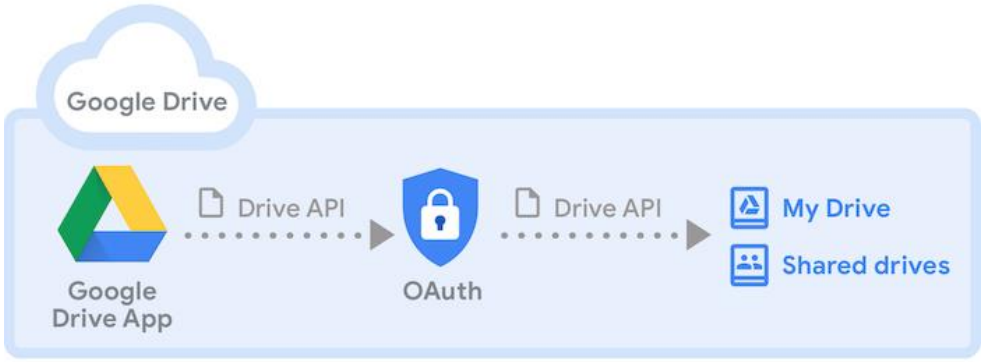


Figure 6-17. Google Drive API relationship diagram

The following terms define the key elements depicted in **Figure 6-17**:

- Google Drive: Google's cloud file storage service offers customers a personal storage space called My Drive as well as the ability to access collaborative shared folders known as shared drives.

- Google Drive API: This is a REST API for accessing Drive storage from within your program.

- Google Drive app: An program that uses Google Drive for storage.

- Google Drive UI: Google Drive's user interface for managing files. If your app is an editor, such as a spreadsheet or word processor, you may use the Drive UI to generate and open files from within your app.

- My Drive: A Drive storage location owned by a certain user. Files in My Drive can be shared with other users, but ownership of the material remains with the user.

- OAuth 2.0: The authentication protocol required by Google Drive API to authenticate your app users. Sign In With Google manages the OAuth 2.0 flow and application access tokens for your application.

- Shared drive: A Drive storage location that owns files that multiple users collaborate on. Any user with access to a shared drive has access to all files it contains. Users can also be granted access to individual files inside the shared drive.

We can do numerous things such as create/move/copy/delete files and folders, share files, etc. In short, my program need authentication files to request google drive api, then receive response from google drive api. My program will read files that stored inside drive and download model if needed.

6.5.2. Launch new version and download model

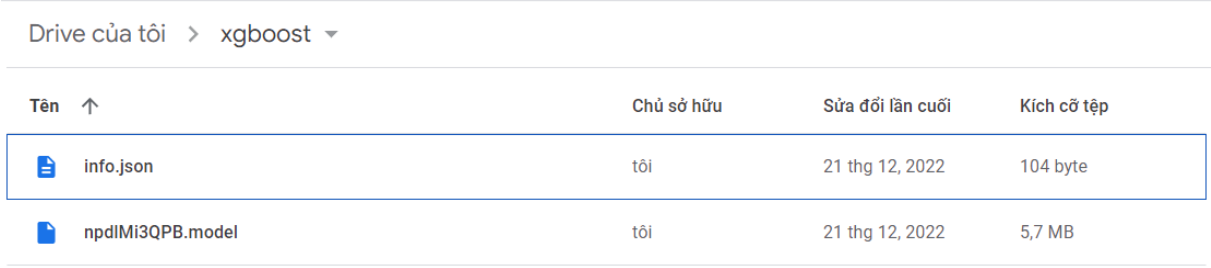


Figure 6-18. Model location

Figure 6-18 shows where I will launch new version. In order to request to API, we need authentication files, I will show it right below:

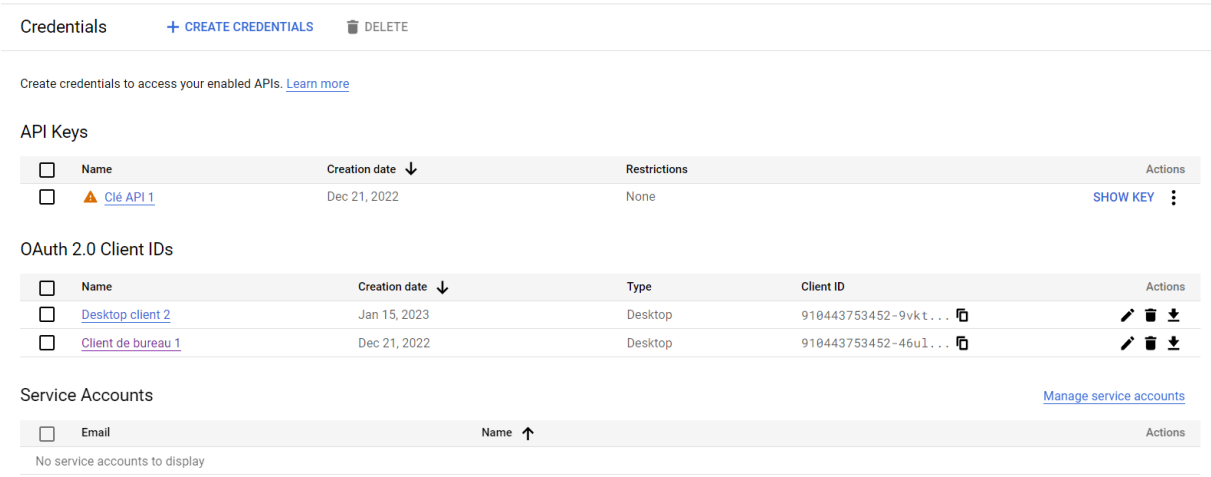


Figure 6-19. Credentials

We need to download Oauth 2.0 Client IDs as shown above. There are 2 files named “Desktop client 2” and “Client de bureau 1”, we can download one of them or create new one.

Figure 6-20 shows update model program progress, it can download model through internet.

```
MSG: Check for update.
client_secret.json-drive-v3-(['https://www.googleapis.com/auth/drive'],)
['https://www.googleapis.com/auth/drive']
drive service created successfully
MSG: Downloading...
{'kind': 'drive#file', 'mimeType': 'application/octet-stream', 'id': '19SesyNWdaXF_0bSN_Ei9l3OSxKQeiWuB', 'name': 'npdlMi3QPB.model'}
```

Figure 6-20. Update model program

6.6. Create Dashboard

6.6.1. ECG Signal – Line chart

The input data given is the heart rate measured by the device, which has not been converted to millivolts yet. The above graph is a time series, a type of line chart, with the only difference being that the x-axis of the chart shows time instead of integers like a traditional line chart. Time series displays the value of a data point at the time it was recorded, making it suitable for displaying heart rate in real-time. This is because we need to understand the value and shape of the ECG over time, rather than the quantity of data received as in Arduino IDE's Serial Plotter.

Figure 6-21. ECG signal chart

The chart in **Figure 6-21** displays the heart rate signal over the last 15 seconds from the current time. This will help the user understand their heart rate and heartbeats over recent time, which is particularly useful when they experience any abnormal symptoms in their heart such as a rapid heartbeat or when the device is unable to measure ECG.

6.6.2. Heart rate – Gauge chart and Stat chart

a **b** **c**

Figure 6-22. Heart rate chart

The previous chart displays the average heart rate, which serves to reinforce the chart above. When observing the ECG chart, it may be difficult to determine the average heart rate. In this gauge chart, Grafana has the feature of setting thresholds.

Typically, the heart rate of an adult falls within the range of 60 to 120 beats per minute. The chart will appear in green as shown in **Figure 6-22a**.

If the heart rate exceeds 120 beats per minute, the background color of the chart will turn red, as shown in **Figure 6-22b**. This indicates a heart rate above the normal range. Similarly, if the heart rate of the wearer drops below 60 beats per minute, the chart will appear in purple as shown in **Figure 6-22c**. The Min and Max stats charts display the lowest and highest heart rate recorded in the last 10 seconds respectively.

6.6.3. Classification history – Log

Finally, the log feature will display the results of the Machine Learning model in the form of a historical record, including timestamps. The reason for including a log is that users often do not understand the ECG signals displayed on the Signal graph, so we will add a log to provide more easily understandable results of "normal" and "abnormal" heartbeats.

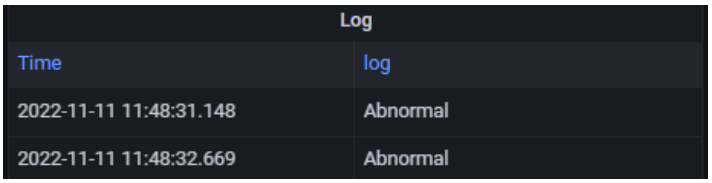


Figure 6-23. Log

Depending on the situation, the log may show all instances of normal and abnormal results as in **Figure 6-23** or only instances of abnormal results. The log will automatically jump to the most recent timestamp and allow us to see the results in real-time as the Machine Learning model outputs them.

CHAPTER 7: RESULTS

7.1. Training results

In chapter 5, I list hyperparameters of XGBoost model and how to tuning a model with wandb. However, testset which is used in nested cross validation and bootstrap different from each other, so we cannot compare the training results.

In this chapter, I will train each parameter set that scores highest F1-score during tuning process with same dataset.

Table 7-1. Original dataset

	Train	Validation	Test
0 (N)	48553	23918	18118
1 (F)	1516	707	556
2 (V)	3857	1931	1448
3 (Q)	421	220	162
4 (S)	4314	2117	1608

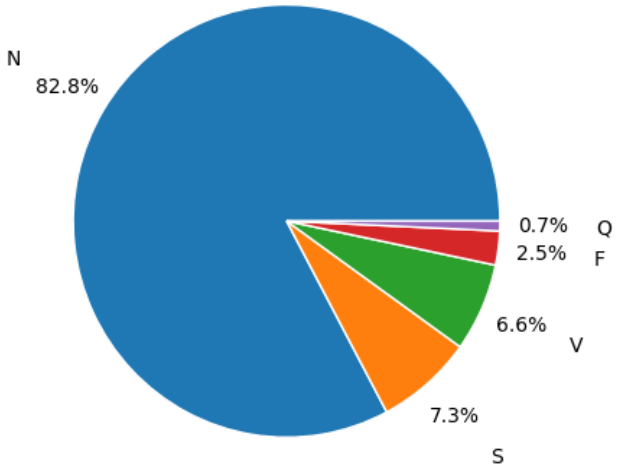


Figure 7-1. Classes ratio

Table 7-2. Number of samples after grouping classes

	Train	Validation	Test
0 (A)	10108	4975	3774
1 (N)	48553	23918	18118

Table 7-3. TPR, FPR, FNR, TNR

	ADASYN – RENN	ADASYN – IHT	Nested Cross validation
TPR	0.911	0.905	0.910
FPR	0.003	0.003	0.002
TNR	0.997	0.997	0.998
FNR	0.089	0.095	0.090
F1-score	0.946	0.943	0.948

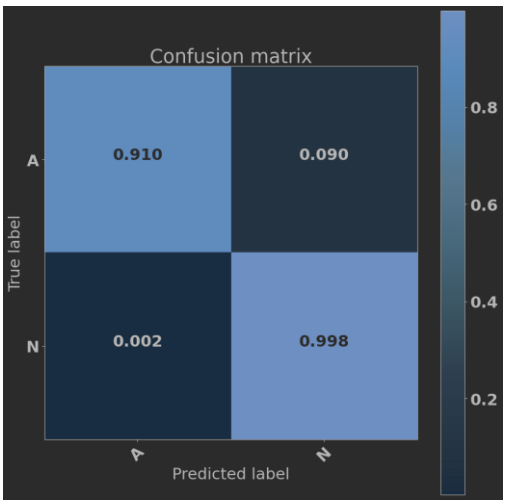


Figure 7-2. Nested Cross Validation Confusion matrix

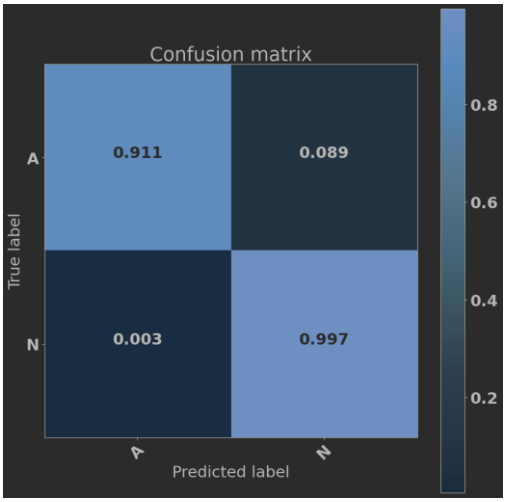


Figure 7-3. Confusion matrix của ADASYN – RENN Bootstrap

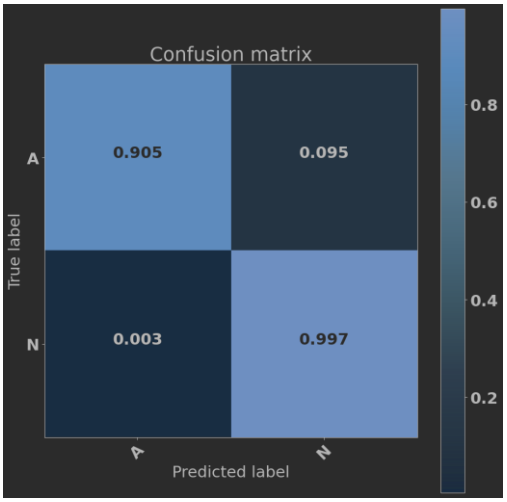


Figure 7-4. Confusion matrix của ADASYN – IHT

As the result table above, ADASYN – IHT method generates dataset that scores the worst score in confusion matrix. This combination might be bad in this case. On other hand, False Positive Rate and True Positive Rate in ADASYN – IHT are higher.

This scenario is actually introduced in chapter 3 when choosing validation metrics. In order to increase the accuracy on abnormal cases, the accuracy of normal cases will decrease.

In general, Nested Cross validation gives the best result.

7.2. Machine learning model in IoT system

Jetson Nano is running and classifying ECG signal.

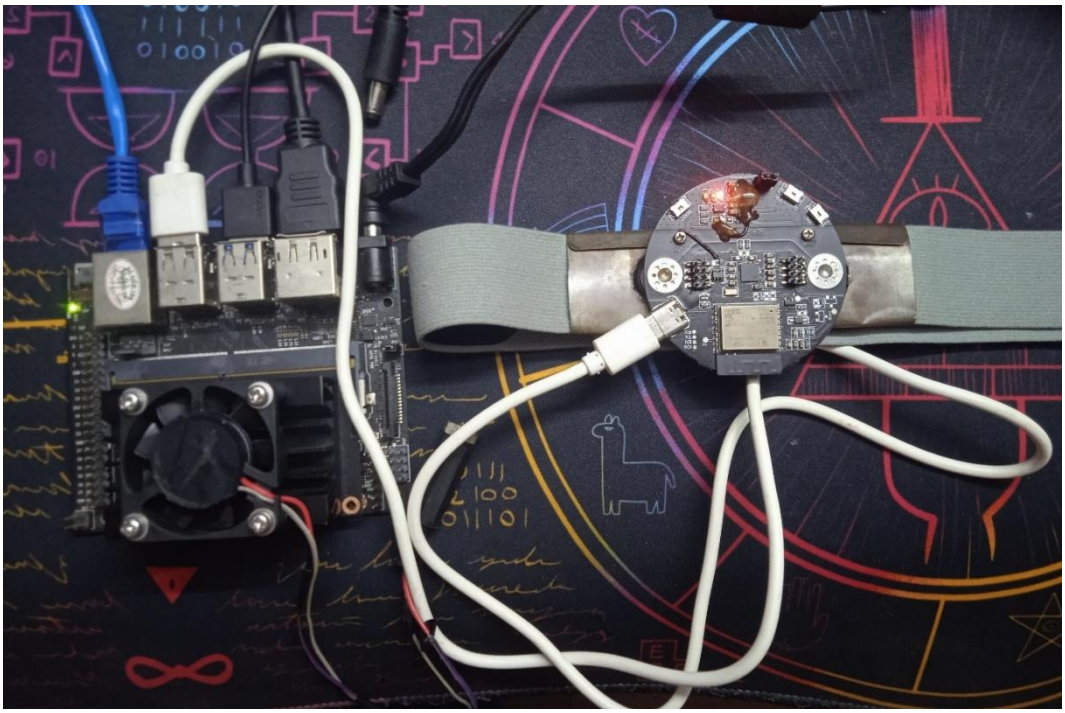


Figure 7-5. Jetson Nano reads data via USB port

Figure 7-6. Jetson Nano with Armbian OS

7.3. Dashboard for users

Section 6.6 only explains the meaning of each chart. After create each single chart, I rearrange it in dashboard and finally get the following result.

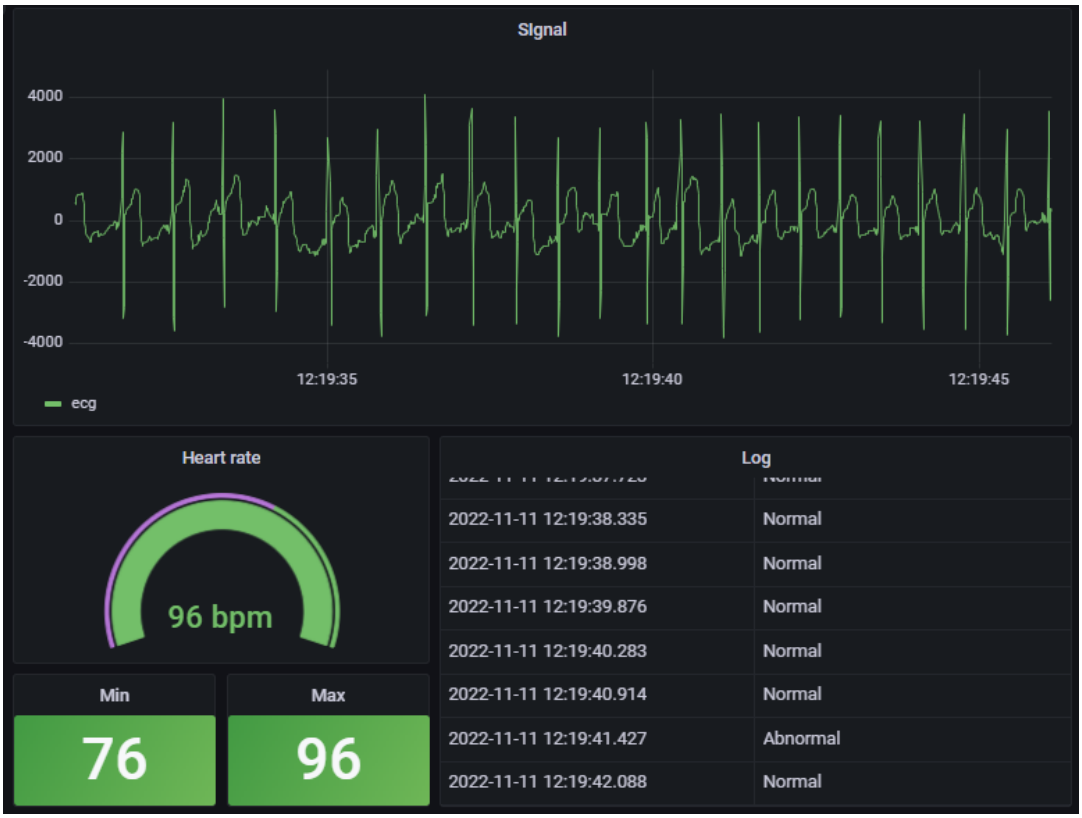


Figure 7-7. Final Dashboard

CHAPTER 8: CONCLUSION AND RECOMMENDATIONS

8.1. Concluding remarks

8.1.1. Result Summary

- Know how to analyze the problem and understand the final target to choose good models.
- Can implement nested cross validation and bootstrap from research papers and books, then using it in tuning hyperparameters with wandb.
- Can create Docker image and run Docker container on multiple platforms such as Windows, Linux, Armbian, and MacOS.
- There is one solution for storing real-time data with a time series database (InfluxDB) and storing data when offline with SQLite.
- One solution for retraining during deployment.
- There is an interactive and attractive dashboard for end users.
- Can deploy the model in Jetson Nano DevKit, Raspberry Pi 3 model B and Orange Pi zero 2.

8.1.2. Limitations

- Data used in the training model process is not the output data from the IoT device, so the result does not actually indicate reliability.
- Local InfluxDB is used instead of InfluxDB Cloud due to lack of funding.
- Local dashboard is used (Grafana Desktop) due to lack of funding.
- There are not enough filters for denoise processing.
- Just read and classify one device at the same time.

8.2. Suggestion for further studies

- Add more filters such as bandpass filter for frequency between 0.05Hz and 49Hz or 51Hz and higher frequency.
- Running data stream on Cloud.
- Create a system monitoring dashboard.
- With the help of experts in the department of cardiology or someone in related fields, we can minimize the gap between the research stage and the industry stage.

REFERENCES

[1]. dmlc XGBoost, *XGBoost parameters*:

https://xgboost.readthedocs.io/en/latest/parameter.html, accessed Oct 11th 2022.

[2]. Raspberry Pi Ltd, *Raspberry Pi 3 datasheet*:

https://bit.ly/finalthesis-ref2, accessed Jan 01st 2023.

[3]. Nvidia, DATA SHEET NVIDIA Jetson Nano System-on-Module:

https://bit.ly/finalthesis-ref3, accessed Jan 01st 2023.

[4]. ARM Developer, Arm Cortex-A Processor Comparison Table:

https://developer.arm.com/documentation/102826/latest/, accessed Jan 01st 2023.

[5]. ARM Developer, Arm Cortex-A53 MPCore Processor Technical Reference Manual r0p4:

https://bit.ly/finalthesis-ref5, accessed Jan 02nd 2023.

[6]. Allwinner, Allwinner H616 datasheet:

https://bit.ly/finalthesis-ref6, accsessed Jan 01st 2023.

[7]. Allwinner, Allwinner H6 V200 datasheet:

https://bit.ly/finalthesis-ref7, accessed Jan 01st 2023.

[8]. ARM Developer, Arm Cortex-A57 MPCore Processor Technical Reference Manual r1p0:

https://bit.ly/finalthesis-ref8, accessed Jan 01st 2023.

[9]. The MagPi, Raspberry Pi 3 Specs, benchmarks & testing:

https://bit.ly/finalthesis-ref9, accessed Jan 03nd 2023.

[10]. Mohammad Kachuee, Shayan Fazeli, Majid Sarrafzadeh (2018), *ECG Heartbeat Classification: A Deep Transferable Representation*.

[11]. Chip Huyen (2022), *Designing Machine Learning Systems an Iterative Process for Production-Ready Applications*, O'reilly.

[12]. Tweet by Geoffrey Hinton (@geoffreyhinton), February 20, 2020:

https://oreil.ly/KdfD8, accessed Nov 11th 2022.

[13]. Stanford HAI, *The 2019 AI Index Report.*

[14]. Saunders, M. Lewis, P. Thornhill (2012). Research Methods for Business Students (6th ed.).

[15]. Bardia Baraeinejad, Masood Fallah Shayan et al. (2022), *Design and Implementation of an Ultra-low-Power ECG Patch and Smart Cloud-Based Platform*.

[16]. Ha Luu, NguyenThangm, et al. (2008), *Loc nhieu tin hieu ECG thoi gian thuc tren dsPIC*.

[17]. Ohhwan Kwon, Jinwoo Jeong, et al. (2018), *Electrocardiogram Sampling Frequency Range Acceptable for Heart Rate Variability Analysis*.

[18]. Michael R. Smith, Tony Martinez, Christophe Giaud-Carrier (2015), *An instance level analysis of data complexity*.

[19]. Nitesh Chawla, et al. (2002), "SMOTE: Synthetic Minority Over-sampling Technique."

[20]. Khoa M. Truong (2022), *Thiet ke thiet bi do nhip tim, dien tim deo quanh nguc theo doi suc khoe qua dien thoai thong minh cho nguoi mac benh tim mach*, Final year project, Faculty of Electrical and Electronics Engineering, HCM University of Technology and Education.

[21]. Hung M. Pham, *Lam sang tim mach hoc*, Ministry of Health, Institut du Coeur.