ESPORT VALORANT: OPTIMIZING HYPERPARAMETERS OF WIN RATE PREDICTION MODEL USING DERIVATIVE FREE METHOD

BY

CHANG YANG

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONOURS)

Faculty of Information and Communication Technology

(Kampar Campus)

JUNE 2024

UNIVERSITI TUNKU ABDUL RAHMAN

REPORT STATUS DECLARATION FORM

Title: ESPORT VALORANT: OPTIMIZING HYPERPARAMETERS OF WIN RATE PREDICTION MODEL USING DERIVATIVE FREE METHOD

Academic Session: JUNE 2024

I CHANG YANG

(CAPITAL LETTER)

declare that I allow this Final Year Project Report to be kept in
Universiti Tunku Abdul Rahman Library subject to the regulations as follows:

1. The dissertation is a property of the Library.
2. The Library is allowed to make copies of this dissertation for academic purposes.

Verified by,

[illegible] [illegible]

(Author's signature) (Supervisor's signature)

Address:
29, LORONG 2/SS5,
BANDAR TASEK MUTIARA
14120, SIMPANG AMPAT

Tseu Kwan Lee
Supervisor's name

Date: 18/07/2024

Date: 9/9/2024

Universiti Tunku Abdul Rahman			
Form Title : **Sample of Submission Sheet for FYP/Dissertation/Thesis**			
Form Number: **FM-IAD-004**	Rev No.: **0**	Effective Date: **21 JUNE 2011**	Page No.: **1 of 1**

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

UNIVERSITI TUNKU ABDUL RAHMAN

Date: 18/7/2024

SUBMISSION OF FINAL YEAR PROJECT /DISSERTATION/THESIS

It is hereby certified that ______***CHANG YANG***_____ (ID No: __***19ACB05073***__) has completed this final year project entitled "____ESPORT VALORANT: OPTIMIZING HYPERPARAMETERS OF WIN RATE PREDICTION MODEL USING DERIVATIVE FREE METHOD_" under the supervision of __Ms. Tseu Kwan Lee_ (Supervisor) from the Department of __Computer Science__, Faculty of __Information and Communication Technology .

I understand that University will upload softcopy of my final year project / dissertation/ thesis* in pdf format into UTAR Institutional Repository, which may be made accessible to UTAR community and public.

Yours truly,

[illegible]

CHANG YANG

*Delete whichever not applicable

DECLARATION OF ORIGINALITY

I declare that this report entitled "**ESPORT VALORANT: OPTIMIZING HYPERPARAMETERS OF WIN RATE PREDICTION MODEL USING DERIVATIVE FREE METHOD**" is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Signature	:	[illegible]
Name	:	CHANG YANG
Date	:	18/07/2024

ACKNOWLEDGEMENTS

I would like to express sincere gratitude to Ms. Tseu Kwan Lee, my supervisor, for her guidance and support throughout the project. Her expertise and feedback were very helpful in developing the project successfully. I appreciate her always pushing me to improve my work.

I also want to thank my family and friends for their encouragement during this project. Their support helped me stay focused and motivated.

ABSTRACT

This research investigates the application of machine learning to predict outcomes in Valorant, a rapidly-growing first-person shooter game within the esports field. This project developed two predictive models, neural network and XGBoost, and optimized using Bayesian optimization and random search to optimize their hyperparameters. The findings shows that the efficiency and capability of Bayesian optimization over random search in terms of both model performance and computational efficiency. To enhance public accessibility and usability, this project has created web APIs and a user-friendly graphical interface. This research contributes significantly to the field of esports analytics and provides practical tools for predicting Valorant gameplay outcomes. By offering a robust and efficient predictive system, this research aims to support informed decision-making, enhance the overall experience for pro-players and enthusiasts of this popular game, and potentially suggest strategic development within the Valorant community. Furthermore, the findings of this project may serve as a valuable reference for future research exploring the application of machine learning to other esports games. Specifically, this research provides insights into the effectiveness of Bayesian optimization in optimizing machine learning models for complex tasks. By comparing Bayesian optimization to random search, it highlights the benefits of Bayesian optimization's ability to intelligently balance between explore and exploit the hyperparameter search space, leading to more efficient and effective model optimization. Additionally, our study noted that the importance of developing user-friendly interfaces and APIs to make win rate prediction tools accessible to a wider audience, fostering collaboration and innovation within the esports community.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

α	Learning rate
β	Momentum

LIST OF ABBREVIATIONS

FPS	First Person Shooting
CS2	Counter-Strike 2
CS:GO	Counter-Strike: Global Offensive
GUI	Graphical User Interface
API	Application Programming Interface
BO-GP	Bayesian Optimization with Gaussian Process
ROC	Receiver-Operating Characteristic curve
AUC	Area Under the Curve
JSON	JavaScript Object Notation
HTML	Hypertext Transfer Markup Language
SGD	Stochastic Gradient Descent
BGD	Batch Gradient Descent
MGD	Mini-batch Gradient Descent
HTTP	HyperText Transfer Protocol
PIL	Python Image Library
PRC	Precision-Recall Curve

Chapter 1

Introduction

Valorant is a 5 versus 5 First Person Shooting (FPS) video game, where 10 players are split into 2 teams, each players select different agent with different ability to cast in game. A team will play to defend site while another will attack site to plant the spike. After 12 rounds, the attacker and defender will swap. The game ends when one of the team reach 13 round wins with the condition that the number of round wins by the team is 2 rounds leading to their opponent. Valorant is released in mid-2020 by Riot Games and it has won the 2022 "Best E-sport Game" award at the The Game Award (TGA). It shows that the game has recognized by the FPS community, and it starts gaining attentions by millions of players worldwide.

On the other hand, another FPS game that have the same popularity among the community is the Counter Strike 2 (CS2 formerly known as CS:GO) launched in 2012 by Valve, which is 8 years earlier than Valorant. There are a lot of studies are done within these 8 years by the researchers especially from the Computer Science and Machine Learning field. For example, there are research are made based on CS2 where the researchers developed a machine learning model to predict optimal buying strategy to have higher win rate [1]. The model able to suggest the players to buy wisely instead of always fully spend the money on equipment or guns. In contrast, currently there are not many studies and research available based on Valorant.

The gameplay style of CS2 is similar to Valorant, However, unlike Valorant, all the players in CS2 does not have different roles and abilities, all players can buy the same equipment such as smoke, flashbang, grenade, etc. While in Valorant, there are 23 different agents with unique ability set currently in the agent pool, each agent can be categorized into a role such as duelist, initiator, controller, or sentinel. Each role have their own responsibilities in game, for example, duelist responsibilities is to initiate a 1 versus 1 fight with the enemies, using their abilities to create advantages and eventually take down the opponents, creating the number advantages, while initiator are responsible for helping teams to initiate the fight with enemies by using a flashbang or location-revealing abilities, causing enemies to lose sight for a few second and reveal the location of the enemies, create time advantages for the teams to win the fight. Therefore, the composition of the agent in teams and the unique abilities created many unpredictable yet complex elements in Valorant compared to CS2.

This project aimed to build a machine learning model that able to take players performance data in past games in Valorant and predict the win rate of the team to win the game with the agent and the map selected by the players, using Neural Network model and XGBoost Tree model. In order to let the models to achieve it optimal performance, the model needs to undergo hyperparameter optimization. However, due to the complexity of the Neural Network model and XGBoost Tree model, commonly used method such as Grid Search and Random Search might not be the best fit solution. Therefore, this project also will implement different optimization techniques to investigate how does different optimization techniques affect the optimization process.

1.1 Problem Statement and Motivation

There are a few issues that causing the model unable to perform well. Based on the previous research paper, the author mentioned that the models are not tuned to its optimum form since they are only searched within a small search space. Therefore, the authors suggested to explore more hyperparameters and also try wider search space to further improve the models [2].

In the previous research paper, the authors use common search method such as Grid Search or Random Search to perform hyperparameter optimization, however these common search methods are inefficient especially when the dimension and the search space of hyperparameter is complex [3]. These search methods emphasize only on exploration on the search space but does not exploit specific area where possibly consist of the global minimum point of the entire search space [4]. Therefore, there are higher chance to miss the optimum point for the model to perform at its best. Some papers suggested an effective optimization method to optimize models with hyperparameters that are high dimension and wide search space, meanwhile it balanced both exploration and exploitation to the search space, which is Bayesian Optimization using Gaussian Process.

1.2 Objectives

The objective of the project is to investigate the impact to the model's performance when applying different optimization methods on complex models such as Neural Network and XGBoost Tree. The optimization focuses on enhancing the validation accuracy of the models, in order to improve the ability of the model to predict the result of the game based on unforeseen data.

This project will also compare different optimization methods in terms of time effectiveness and performance improvement when applied on Neural Network and XGBoost. Both of the models are having sensitive and complex hyperparameters that required to be tuned carefully. Hyperparameter optimization usually require high computing power and time consuming, therefore this project aim to find out which optimization method are more effective to optimize complex models. This can be evaluated based on some model evaluation metrics such as accuracy, precision, recall, etc., as well as the time taken for the optimization to complete.

1.3 Project Scope and Direction

The project will deliver optimized machine learning models that takes Valorant gameplay data as input and predict the win rate of both teams. The models will be deployed as web services, user or developer can access the services through API when required, the API will be free access and available to everyone.

The project will also develop a Graphical User Interface (GUI) that allow user to select preferred model, input prediction data and visualize the prediction results. The GUI act as a basic interface that using the API to access the web service and let the user to access to the model easily.

1.4 Contributions

This project able to contribute to multiple fields in the E-sports as well as the research area. First of all, the model can serve as a reference for game developers to balance the game environment. Valorant is a complex game and usually it is hard to balance in terms of the strength of agent’s abilities, map preferences, weapon strength, economy growth, player distribution, etc. The models from this project can be a useful tool for game developers to adjust the game to become more balance, so that all the players can have a better gaming experience.

Meanwhile for professional players and teams that plays in international event such as Valorant Champion Tour (VCT), which is one of the biggest tournaments hosted by Riot Games, the models and the GUI from this project are powerful tools for them. The teams can utilize the GUI to predict the probability to win the game, and further optimize the win rate of the team by changing the team compositions and map selection.

This project also able to contribute to machine learning and hyperparameter optimization field. This project experimented and compared different optimization methods on 2 different models, Neural Network and XGBoost Tree model when performing prediction using high dimensional E-sports data. Hence, the findings of this project can act as a reference for future research especially in E-sport machine learning and hyperparameter optimization field.

1.5 Report Organization

This report is organized into 7 chapters: Chapter 1 Introduction, Chapter 2 Literature Review, Chapter 3 System Methodology, Chapter 4 System Design, Chapter 5 System Implementation, Chapter 6 System Evaluation and Discussion, and Chapter 7 Conclusion and Recommendation.

Chapter 1 introduced the problem statement, objectives, scope of the project. Chapter 2 reviewed some previously done research and discussed how different optimization methods work in details, then further compare each methods algorithm and implementation complexity. Chapter 3 gives an overview of how the project is going to be developed, including model training and API/GUI development, it also briefly discussed how each machine learning model works and introduced the hyperparameters chosen to be tuned in this project. Chapter 4 discussed in detail about how each part of the project is developed, including flowchart of each script used in the project. Chapter 5 is about how to implement the parts mentioned in chapter 4, including the hardware/software setup, and challenges faced during implementation phase. Chapter 6 reviewed the result based on the performance metrics, and also discussed the project challenges. Chapter 7 concluded the project and recommended future work that can further improve this project.

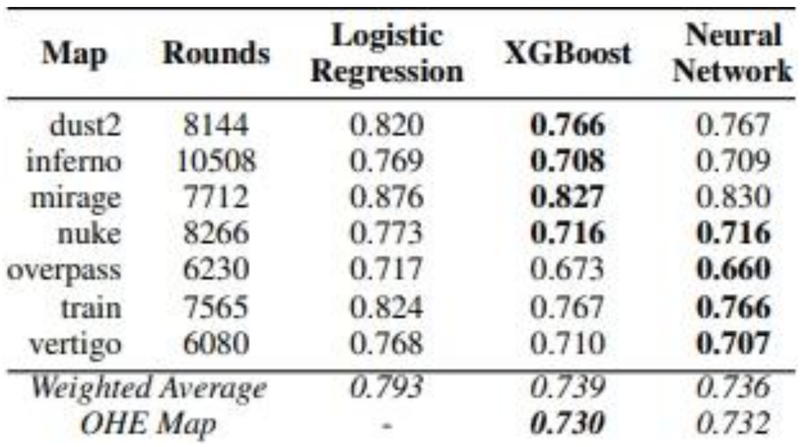
Chapter 2

Literature Review

2.1 Previous Works Review

2.1.1 Optimal Spending Decision based on win probability

Several win rate prediction projects have been conducted prior to this study. For instance, research in [1], the authors developed multiple prediction models for CS:GO, using machine learning models such as logistic regression, XGBoost, and neural networks. The primary goal of this project was to establish guidelines for players to make optimal weapon purchase decisions based on predicted win probability. The authors initially predicted game win rates using various classification algorithms and subsequently utilize these probabilities to assess optimal spending strategies. A metric termed Optimal Spending Error was introduced by the authors to evaluate the discrepancy between player decisions and optimal choices. Results indicated that XGBoost and neural networks outperformed logistic regression in terms of log-loss.



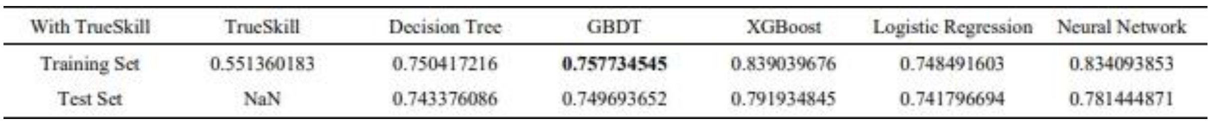
Map	Rounds	Logistic Regression	XGBoost	Neural Network
dust2	8144	0.820	**0.766**	0.767
inferno	10508	0.769	**0.708**	0.709
mirage	7712	0.876	**0.827**	0.830
nuke	8266	0.773	**0.716**	**0.716**
overpass	6230	0.717	0.673	**0.660**
train	7565	0.824	0.767	**0.766**
vertigo	6080	0.768	0.710	**0.707**
Weighted Average		*0.793*	*0.739*	*0.736*
OHE Map		-	***0.730***	*0.732*

Figure 2.1.1: Log-loss result by map for candidate models [1]

Based on Figure 2.1.1. the log-loss results of both XGBoost and Neural Network for each map are lower than logistic regression, meaning that both of these models are performing better than logistic regression. The performance between XGBoost and Neural Network can be said as similar. However, the result shows that Neural Network are slightly better than XGBoost.

2.1.2 Predicting round result in CS:GO using Machine Learning

Another research [2] has been made based on the research [1] to further improve the win rate prediction models, the authors modified the training dataset by implementing TrueSkill value to improve dataset quality. TrueSkill is a ranking system developed by Microsoft based on ELO rating system to estimate the quality of the game. Most of the games utilized this algorithm to match up players with similar skills [2]. The aim of this project is to investigate which models are able to have better accuracy when performing win rate predictions.



With TrueSkill	TrueSkill	Decision Tree	GBDT	XGBoost	Logistic Regression	Neural Network
Training Set	0.551360183	0.750417216	**0.757734545**	0.839039676	0.748491603	0.834093853
Test Set	NaN	0.743376086	0.749693652	0.791934845	0.741796694	0.781444871

Figure 2.1.2: Accuracy of each classification algorithms with TrueSkill values [2]

In the research, the authors used 5 common models including Decision Tree, GBDT, XGBoost, Logistic Regression, and Neural Network. Figure 2.1.2 shows the results of the experiment. Based on the authors of [2], generally the model will have improvement of 0.2% on train accuracy while 1% on test accuracy, while for neural network, the train and test accuracy both increased by 1%. Hence the implementation of TrueSkill value able to increase the dimension of the data to provide more information, therefore improve the model performance.

2.1.3 Previous works issues and proposed solutions

Several limitations are pointed out within the models presented in the research papers. Firstly, the models have not yet reached their optimal performance, hence the models require further hyperparameter optimization [1]. Additionally, the datasets employed could be enhanced by including prediction results from other machine learning models into the datasets [2].

Furthermore, optimizing hyperparameters for complex models such as XGBoost and Neural Networks using traditional methods like grid search and random search is computationally expensive and often yields suboptimal hyperparameters [3]. These methods exhaustively explore the hyperparameter space, leading to inefficiency of optimization. Hence, using grid search or random search are not the best solution for complex model optimization [5]. Alternative optimization techniques, such as Bayesian optimization, have shown the potential to achieve high-performing hyperparameters more efficiently. Unlike grid search and random search methods that only focusing on exploration on the wide search space, Bayesian Optimization balanced both exploration and exploitation of the search space [4], therefore it able to find a comparable point where the hyperparameters can provide decent performance, meanwhile consume only a short period of time. It also has higher chance to find out the global optimum point due to its exploitation nature where it will search within niche spaces that possibly contain optimum point.

2.2 Hyperparameter Optimization Techniques

2.2.1 Exhaustive Search (Grid Search / Random Search)

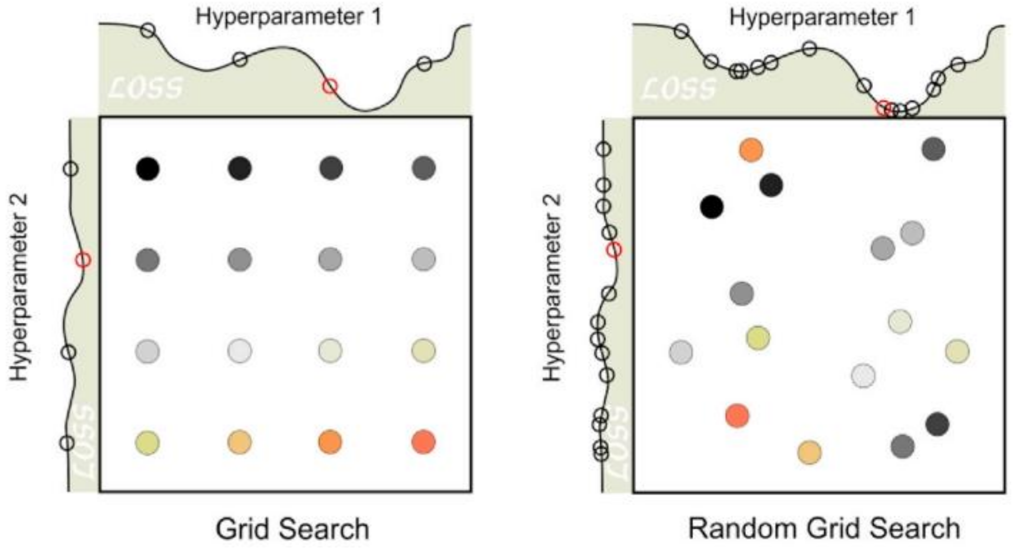


Figure 2.2.1(a): Illustration of Grid Search and Random Search [6]

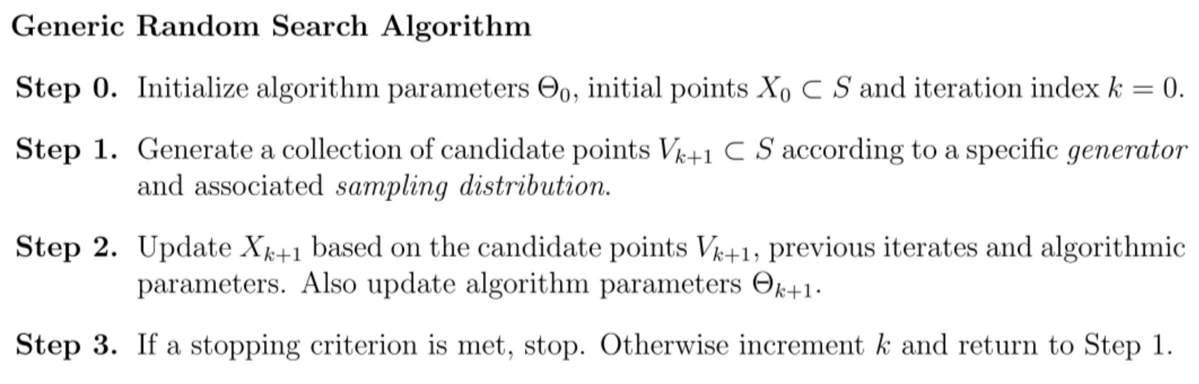
Both grid search and random search are commonly used by researchers to optimize models because they are easier to implement in code and have a great ability to explore wide search space. However, there are some weaknesses on these methods when optimizing complex models.

Grid search is an optimization method that optimize model by trying every possible combination of hyperparameters from a defined search space. Figure 2.2.1 shows the hyperparameters are chosen in a uniform distribution and the objective is to obtain hyperparameters that have lowest loss. It is able to observe that grid search is computationally expensive because it has to try all combination of hyperparameters. Many values of hyperparameters might have to be probed before finding a value that leads to improved model performance [6]. Moreover, there is no guarantee if the optimal hyperparameters was included in the defined search space [6]. As the search space dimension grow, the required exploration of search space grows exponentially as well. This is inefficient especially to optimize complex models that have many hyperparameters to optimize.

Random search is similar to grid search. However, random search fixed the issues of high number of combinations in grid search by randomly select hyperparameters from search space. In random search, user can define number of trials for the search algorithm to sample hyperparameters from the search space by either uniform or normal distribution, this allows user to control the time consumed by the algorithm to perform optimization. Figure 2.2.1(b) shows the generic random search algorithm. According to the authors of [7], the random search algorithm consists of two basic procedures, which is generator that generate candidates points and update the generator for next candidate generation. Random search usually uses single-point generator to generate candidate point based on the combination of current point and previous points. The generator algorithm can be express as the equation:

$$V_{k+1} = X_k + S_k D_k \ [7]$$

Where candidate point is generated by taking step S_k from current point X_k in direction D_k on iteration k. The direction D_k can be determined by gradient information or can be generated according to a uniform distribution on a hypersphere, while step S_k might be the result of a line search. [7].



Generic Random Search Algorithm

Step 0. Initialize algorithm parameters Θ_0, initial points $X_0 \subset S$ and iteration index $k = 0$.

Step 1. Generate a collection of candidate points $V_{k+1} \subset S$ according to a specific *generator* and associated *sampling distribution*.

Step 2. Update X_{k+1} based on the candidate points V_{k+1}, previous iterates and algorithmic parameters. Also update algorithm parameters Θ_{k+1}.

Step 3. If a stopping criterion is met, stop. Otherwise increment k and return to Step 1.

Figure 2.2.1(b): Generic Random Search Algorithm [7]

However, despite random search is better than grid search given that user able to define the number of trials, it is still not an effective solution to optimize models that have wide search space. Since the random search point is sampled randomly using uniform distribution, the algorithm does not promise the next sampled point has better performance. Random search does not take previous generated points into account to consider which point to choose in the next iteration. Compared to sophisticated methods such as Bayesian optimization, random search does not have exploit the knowledge of well-performing search space [4].

2.2.2 Bayesian Optimization

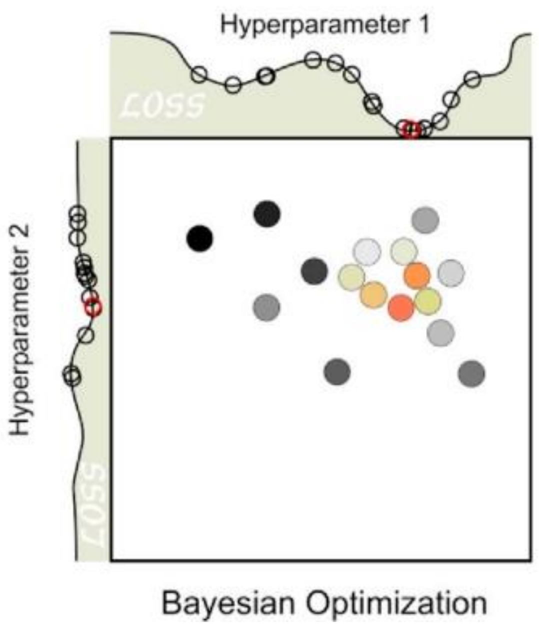
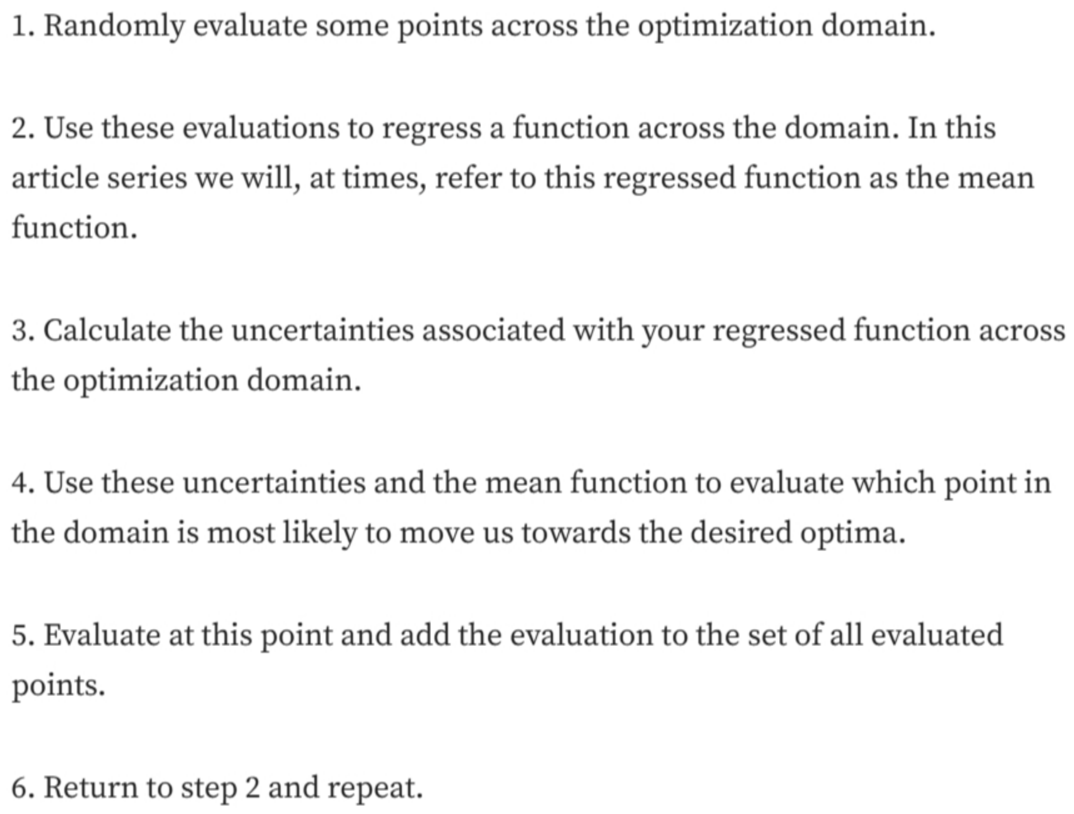


Figure 2.2.2(a): Illustration of Bayesian Optimization [6]

Bayesian optimization is widely used to optimize complex models that have high dimensional hyperparameters such as neural networks and XGBoost. Bayesian Optimization learn from previous points and based on the knowledge to sample next point that most probably will provide a promising result. From Figure 2.2.2(a), it is observed that the sampled point clustered at the minimum loss point of both hyperparameter. Compared to random search, the sampled point is scattered around the entire search space, which means that Bayesian Optimization has higher chance to search for the most optimum point. This is due to the nature of balance between exploration and exploitation of search space in Bayesian Optimization. In Bayesian Optimization, the model not only will explore the search space, if certain area of search space is getting promising optimization results, it will start exploiting that area to find out the best point within that area.



1. Randomly evaluate some points across the optimization domain.

2. Use these evaluations to regress a function across the domain. In this article series we will, at times, refer to this regressed function as the mean function.

3. Calculate the uncertainties associated with your regressed function across the optimization domain.

4. Use these uncertainties and the mean function to evaluate which point in the domain is most likely to move us towards the desired optima.

5. Evaluate at this point and add the evaluation to the set of all evaluated points.

6. Return to step 2 and repeat.

Figure 2.2.2(b): Bayesian Optimization algorithm [8]

From Figure 2.2.2(b), notice that the Bayesian optimization algorithm is unlike grid search and random search that search the entire search space in a brute force way or randomly sample points and hope it able to find the optimal point. Bayesian optimization relies on a regressor function, sometimes referred as mean function [8], or surrogate function. This project will focus only on using Gaussian Process Regressor as the surrogate function, hence Bayesian optimization with Gaussian Process usually will be written as BO-GP. In the first few trial, BO-GP will randomly select a few points from the search space and evaluate the performance of the hyperparameter selected. Then, Gaussian Process Regressor will simulate and predict the overall pattern of the optimization objective based on the search space. This method able to reduce the computational cost to model the validation accuracy of the entire search space. The Gaussian Process Regressor will train multiple models to fit the point selected in the first few trials.

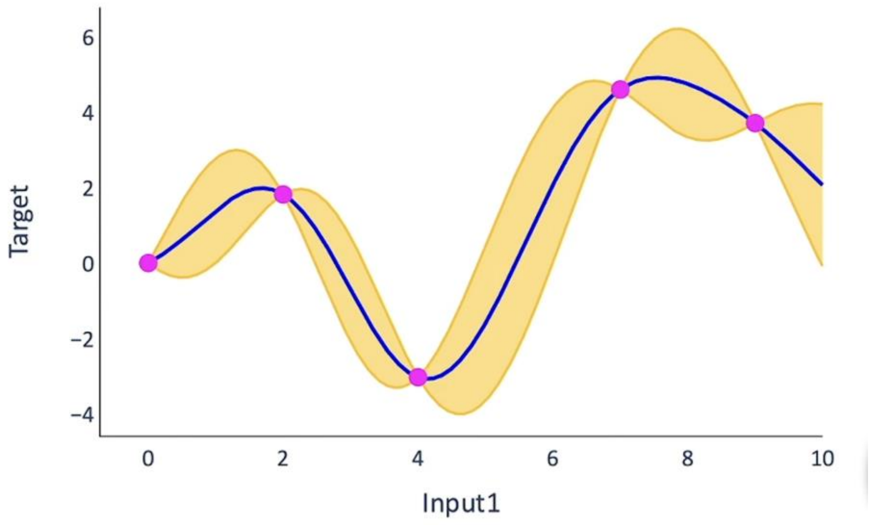


Figure 2.2.2(c): Regressed function (also known as Surrogated Function) [9]

Figure 2.2.2(c) shows the functions generated by the regressor, the pink points are the sampled points in the first few trials, the blue line indicates the mean of all the models train from the regressor and the yellow region indicate the uncertainty or standard deviation of the functions. The uncertainty region became smaller when nearer to the sampled point as it become more predictable.

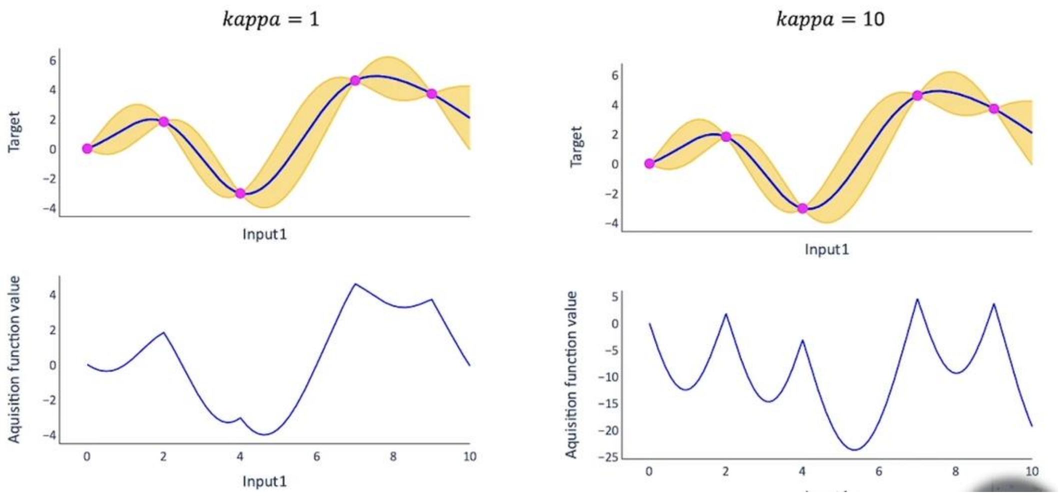


Figure 2.2.2(d): Acquisition function with different kappa value [9]

Acquisition function is a function modeled from the surrogate model where it provides information for the algorithm to sample the possible optimal point. In Figure 2.2.2(c), the authors wanted to minimize the target, therefore the lower confidence bound of the surrogate model are chosen as acquisition function. The acquisition function is expressed as the equation below:

$$A(x) = f(x) - kappa \times std\big(f(x)\big)[9]$$

Where *kappa* is a hyperparameter of the acquisition function that will determine the optimization process tends to be locally or globally [9]. The hyperparameter *kappa* act as a magnifier of the uncertainties. When the uncertainties are magnified for 10 times (*kappa = 10*), the uncertainties in global scope will arise as well. BO-GP algorithms strategy is to search for region that have lowest acquisition function value, where the region is most probably have the optimal point. From Figure 2.2.2(d), in *kappa = 10*, the region with very low acquisition function value is spread over the entire search space, compared to *kappa = 1* which the acquisition function is smoother and the region with low acquisition function value is between 4 and 6 only. In other words, when *kappa = 1*, the algorithm tends to exploit the region to find the local optima point, while when *kappa = 10*, the algorithm tends to explore than exploit the search space and eventually able to find the global optima point.

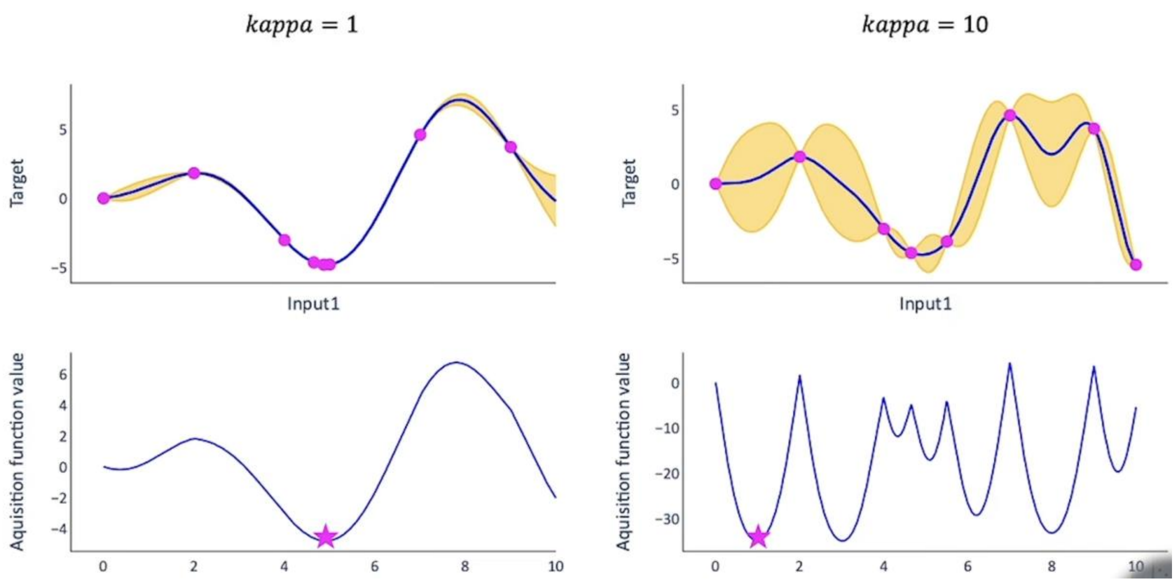


Figure 2.2.2(e): Sampling process of BO-GP with different kappa value [9]

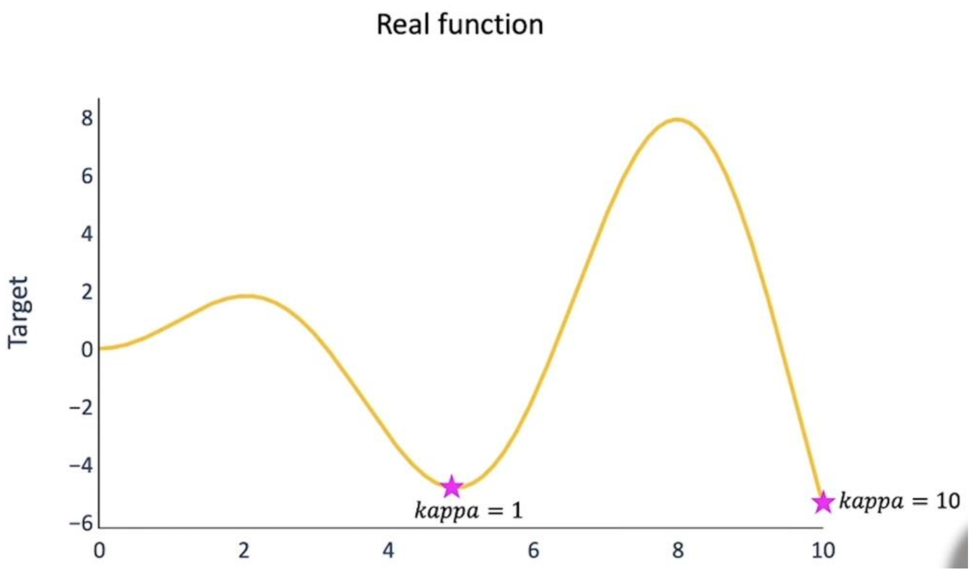


Figure 2.2.2(f): Final optimization result for different kappa value [9]

Figure 2.2.2(f) shows the final optimized results for different *kappa* value. The *kappa = 1* able to obtain the local optima where the value sits between 4 and 6, while for *kappa = 10*, it able to obtain the global optima where the target value is slightly lower than the first result. Therefore, *kappa* value is important to determine the quality and the time required for the optimization. If the *kappa* is too low, it tends to be stuck at local optima and it is hard to escape, while if *kappa* is too high, the algorithm tends to explore instead exploiting the search space, and it will take longer time or more trials to obtain the global optima. It is important to balance between quality of the results and the efficiency of the optimization.

2.3 Comparison of Hyperparameter Optimization techniques

Methods	**Grid Search**	**Random Search**	**Bayesian Optimization**
Implementation	Easy	Easy	Complex
Optimization Strategy	Brute force by trying each combination of hyperparameter	Random sample hyperparameters from search space	Based on surrogated model and acquisition function to make smart guess of next point
Computational Power required	Expensive for high dimensional hyperparameter	Depends on number of trials	Balanced between consumption and result quality

Algorithm	1. start optimization from the first point in the defined search space 2. obtain the result 3. obtain next point in the search space 4. repeat from step 2 until all combination is tried.	1. start optimization by randomly sample a point from defined search space 2. obtain the result 3. randomly sample next point from the search space 4. repeat step 2 until defined number of trials.	1. start optimization by randomly sample few points from search space 2. obtain the result 3. regress a surrogated model 4. obtain acquisition function 5. sample the point with optimum acquisition function value 6. repeat step 2 until defined number of trials.

Table 2.2.3: Comparison between different optimization methods

Table 2.2.3 shows the comparison between the optimization methods mentioned above, the table compared multiple aspect of the optimization algorithms. To determine whether the optimization method is suitable for optimizing complex models like neural network or XGBoost, the algorithm efficiency and the computational power required need to be focused. Under the circumstances of high dimensional hyperparameters, Grid search has the simplest algorithm, but it requires more computational power in order to perform model training for each combination. While for Bayesian Optimization, it also consumes computational power due to its complex algorithm needs to regress a new surrogate model and calculate latest acquisition function based on the previously evaluated points, but the complex algorithm helped to reduce the number of iterations needed to obtain quality hyperparameters. While for Random Search, although the algorithm is not as complex as Bayesian optimization, and the power consumption can be controlled by user by defining number of iterations, however it does not always produce promising results due to its random nature. Therefore, random search and Bayesian Optimization are better ways to optimize hyperparameter of complex models efficiently. This leads to the project problem statement, which methods works better in this project? The answer will be revealed in the following chapters.

Chapter 3

System Methodology/Approach

3.1 Project Overview

This project contains 2 major parts, which is model optimization and API/GUI development. The main focus will be on optimizing models and compare the performance of each model. Meanwhile this project also will develop an Application Programming Interface (API) that allow developers to access and use the optimized model, as well as a Graphical User Interface (GUI) to ease normal user to interact with the optimized model and perform game predictions.

3.1.1 Model Optimization

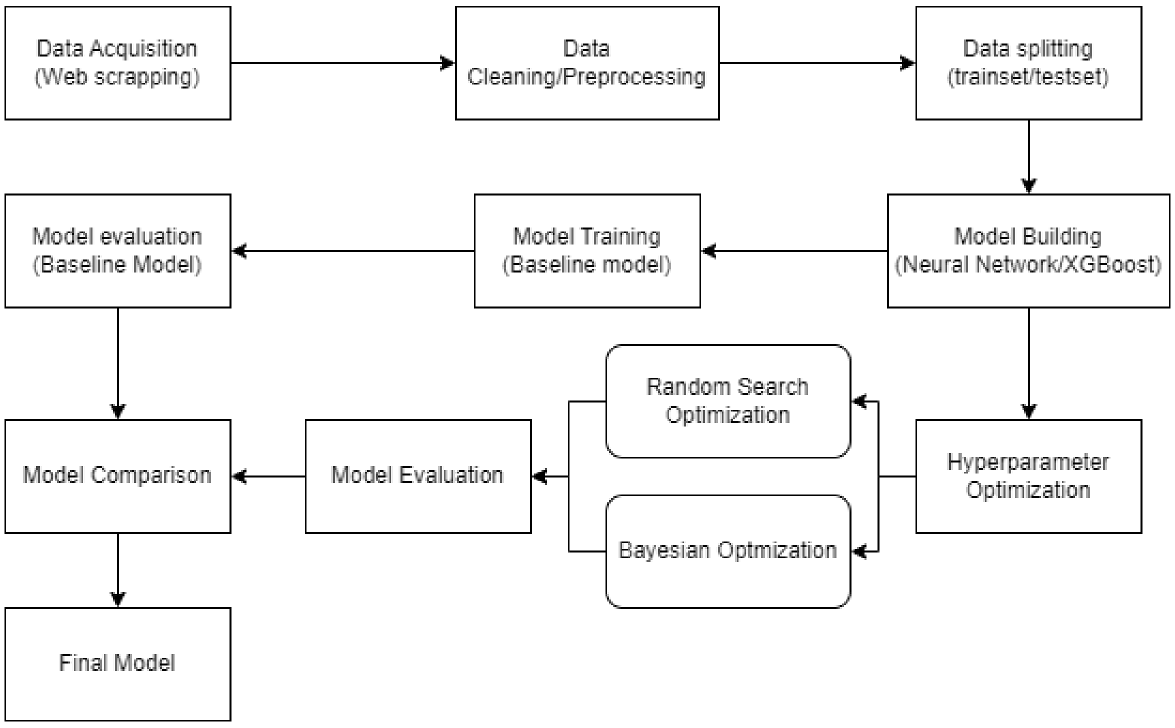

Data Acquisition (Web scrapping)
Data Cleaning/Preprocessing
Data splitting (trainset/testset)
Model evaluation (Baseline Model)
Model Training (Baseline model)
Model Building (Neural Network/XGBoost)
Random Search Optimization
Model Comparison
Model Evaluation
Hyperparameter Optimization
Bayesian Optmization
Final Model

Figure 3.1.1: Flowchart of the model training process

The first part of the project will be focused on creating the models. Before the models can be trained, the data needs to be well prepared. The preparation steps include data acquisition, data cleaning and train/test set splitting. Once these steps are completed, the model is ready to be trained. First, models that is highly customizable such as neural network, need to define its structure such as number of layers, number of neurons, learning rate, momentum, etc. After a default model is decided, the model go through normal training process without any optimization involved will act as baseline model. In the meantime, another exact same model is created to go through hyperparameter optimization using Random Search and Bayesian Optimization. The optimized models and baseline models will go through comparison and finally, 2 best model will be selected as the final model of the project.

3.1.2 API and GUI development

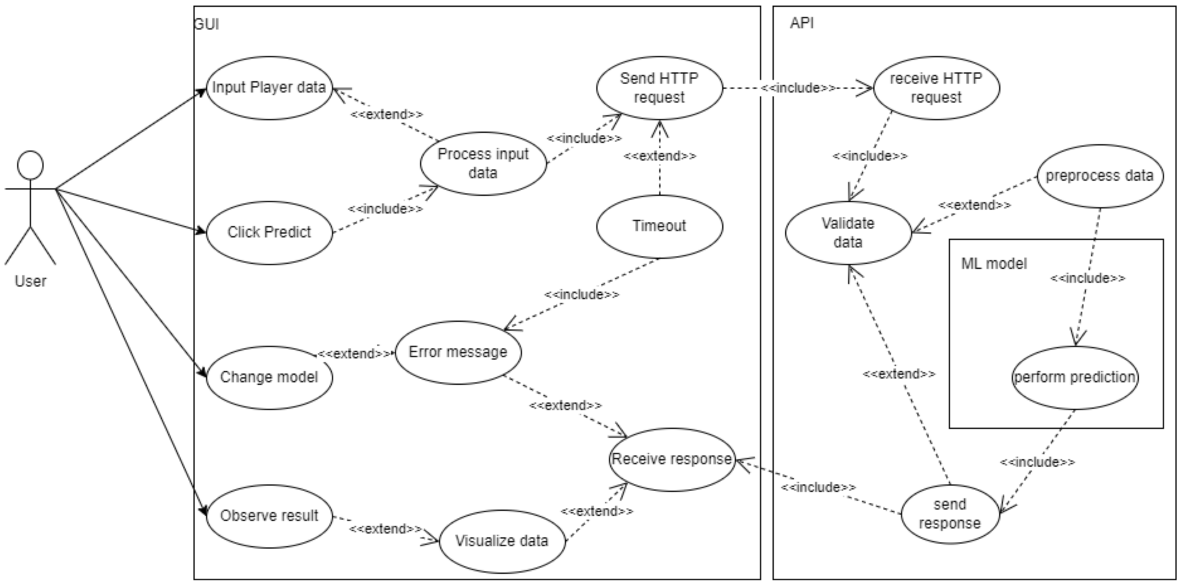


Figure 3.1.2: Use case diagram of the system

Use case name:	Win rate prediction with GUI
Actor:	User
Summary Description:	This use case describes the process of predicting player data using a machine learning model. The user interacts with a graphical user interface (GUI) to input player data and initiate the prediction process. The system then sends an HTTP request to an API, which validates and processes the data before invoking ML model for prediction. The predicted results are returned to the user for visualization.
Priority:	Optional
Status:	Medium level of details
Pre-condition:	The API and ML model is online and ready to serve

Post-condition:	The predicted player data is displayed to the user
Normal flow:	1. User Input: The user enters player data into the GUI. 2. Initiate Prediction: The user clicks the "Predict" button. 3. Send HTTP Request: The system sends an HTTP request to the API with the player data. 4. Receive HTTP Request: The API receives the HTTP request. 5. Preprocess Data: The API preprocesses the received player data. 6. Validate Data: The API validates the preprocessed data for correctness and completeness. 7. Invoke ML Model: The API invokes the ML model to perform the prediction. 8. Perform Prediction: The ML model processes the validated data and generates a prediction. 9. Send Response: The API sends a response to the system containing the predicted results. 10. Receive Response: The system receives the response from the API. 11. Display Results: The system displays the predicted win rate to the user.
Alternative flow:	3a: Timeout: If the API does not receive a response from the ML model within a specified timeout, it sends a timeout message to the system, which is then displayed to the user. 10a: Error Message: If the API encounters an error during data validation or model invocation, it sends an error message to the system, which is then displayed to the user. 12: Change Model: The user may choose to change the ML model being used for prediction. This would involve selecting a different model from the GUI and repeating the prediction process.

Table 3.1.2: Use case description

After the final model is done, the model is saved as a pretrain model file. To create a web service that allows user to access the model's prediction function, the model needs to be loaded into the web services program. In the program, an API is created that accept user input in JSON format, perform preprocessing, prediction on the input, and finally return the prediction result of the model in JSON format to user. The web service program is then uploaded to web service hosting server and the service is ready to be accessed publicly. Besides, a GUI is developed with basic function included such as allow user input, interact with the web services, and display the result returned by the server and visualize the result. This purpose of this GUI is to allow people with or without IT knowledge to access to the model easily.

Figure 3.1.2 is the use case diagram of the system. The figure shows the processes involved in each part of the systems including GUI and API. User input player data into GUI. GUI will process the input data to suit the data format used by HTTP. Then the data is sent to web service API using POST method. When the data is received by the web service hosted, the processed player data is extracted and preprocessed to fit in the model and perform prediction. The result is returned to GUI. GUI received the result and will visualize the result to user.

3.2 Machine Learning Models and hyperparameters

From the previous research [1], [2], both authors trained multiple kinds of machine learning model to predict the winning teams of the game. These models included decision tree, logistic regression, Gradient Boosted Decision Tree (GBDT), XGBoost and Neural Network. The result from both research shows that neural network and XGBoost classifier are having the top performance among the models. This subchapter will discuss about the models' structures and their hyperparameters.

3.2.1 Neural Network

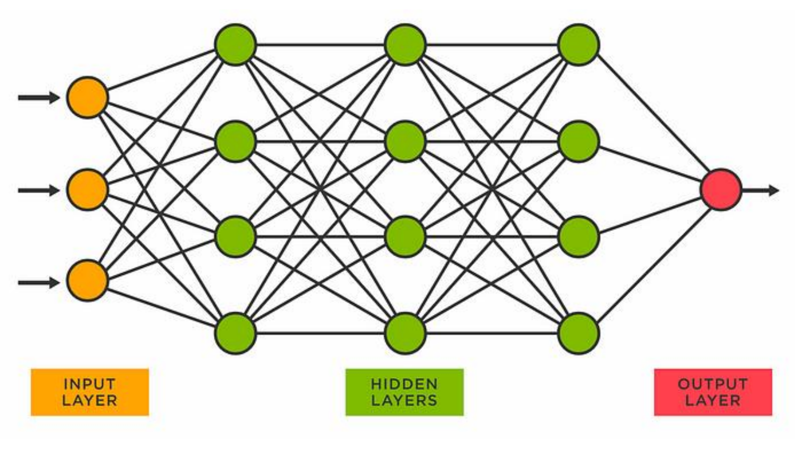
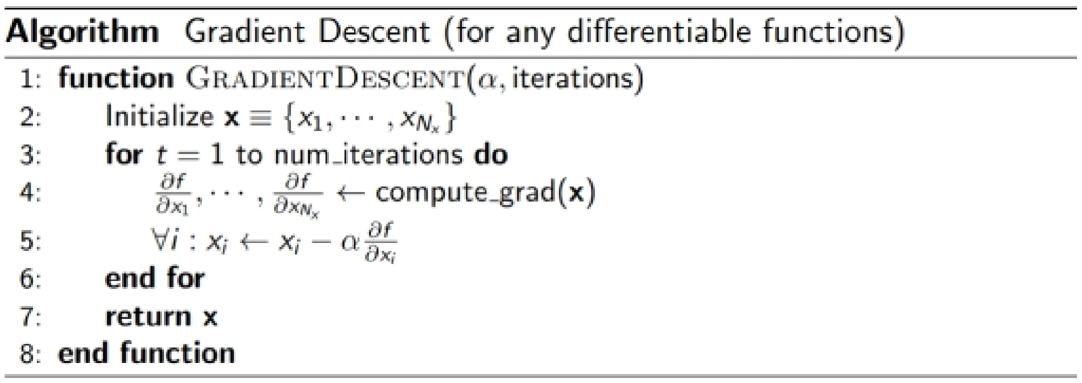


Figure 3.2.1(a): General Structure of a neural network [10]

The figure above shows the general structure of a neural network. It consists of 3 major parts, which are input layer, hidden layers, and output layer. Input layer is a single layer, and it is the first layer of every neural network, it is the input from user. The shape of input layer is depending on how many features in a data contains. The last layer of the neural network is the output layer, this is the layer that will output the decision, or the prediction. The shape of the output layer is depending on the task, or the objective of the neural network. For example, if the neural network is used to perform single class binary classification, then the output layer should be a single neuron. However, if the neural network is used to perform multi-class classification, then the output should be multiple neurons.

The layers between the input and the output layer are called hidden layers. Hidden layers are where the neural network extract and learn high level features from the input layer. It is the hyperparameters that determine the structure and the modeling ability of the network. Therefore, the wider and deeper the hidden layer, the model most likely able to learn higher level features and improve the ability of the model to perform prediction. However, high ability to learn complex features also means that it also possibly to have overfitting issues, where the model learnt low level pattern of the datasets, causing the model unable to perform well on unforeseen dataset.

Beside number of hidden layers and number of neurons for each hidden layer, the learning rate, batch size, number of epochs, and momentum are important hyperparameters as they determined the training efficiency of the model. In the neural network training process, gradient descent is used to optimize the parameters in the neural network by using the gradient information, it is an optimization algorithm to find a local minimum of a differentiable function [11].



```
Algorithm Gradient Descent (for any differentiable functions)
1: function GradientDescent($\alpha$, iterations)
2:     Initialize $\mathbf{x} \equiv \{x_1, \cdots, x_{N_x}\}$
3:     for $t = 1$ to num_iterations do
4:         $\frac{\partial f}{\partial x_1}, \cdots, \frac{\partial f}{\partial x_{N_x}} \leftarrow$ compute_grad($\mathbf{x}$)
5:         $\forall i : x_i \leftarrow x_i - \alpha \frac{\partial f}{\partial x_i}$
6:     end for
7:     return $\mathbf{x}$
8: end function
```

Figure 3.2.1(b): Algorithm of Gradient Descent [11]

Figure 3.2.1(b) shows the algorithm of gradient descent. In the algorithm, the learning rate α is multiplied with the gradient and the product is subtracted with the current x_i to get the x_i for next iteration. Therefore, the learning rate plays an important role in order to control the efficiency of the training process. Figure 3.2.1(c) shows the impact of different learning rate. An optimum learning rate allows the model to converge fast in a short time, increasing the training efficiency.

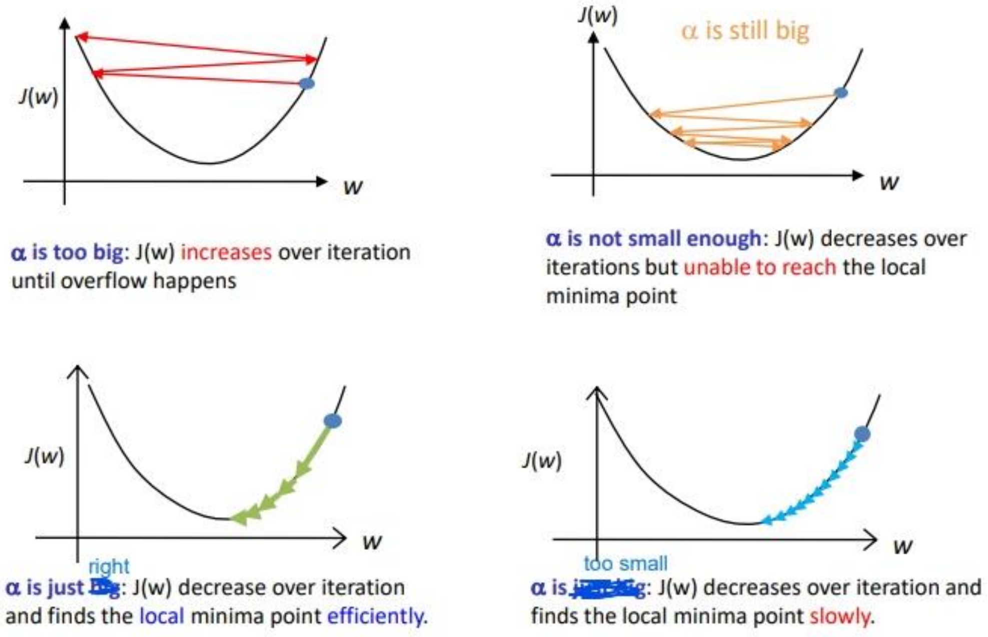


Figure 3.2.1(c): Impact of different learning rate α [11]

The hyperparameter batch size determine the size of data per batch. A basic batch gradient descent (BGD) updates the parameters by computes the activation and gradient of all sample. This method is slow especially when the number of samples is huge [12]. Hence, mini-batch gradient descent (MGD) is more commonly used method, where the whole dataset is split into mini-batches, the model parameter will update once a mini-batch of data is trained. An optimum batch size can have regularization effect, making the model less overfitting to the dataset [12]. Another hyperparameter, number of epochs, is the number of iterations that the neural network needs to go through the entire dataset. A neural network that trained with multiple epochs able to reinforce the knowledge learnt by the model. However, too many epochs will lead to overfitting issues as well.

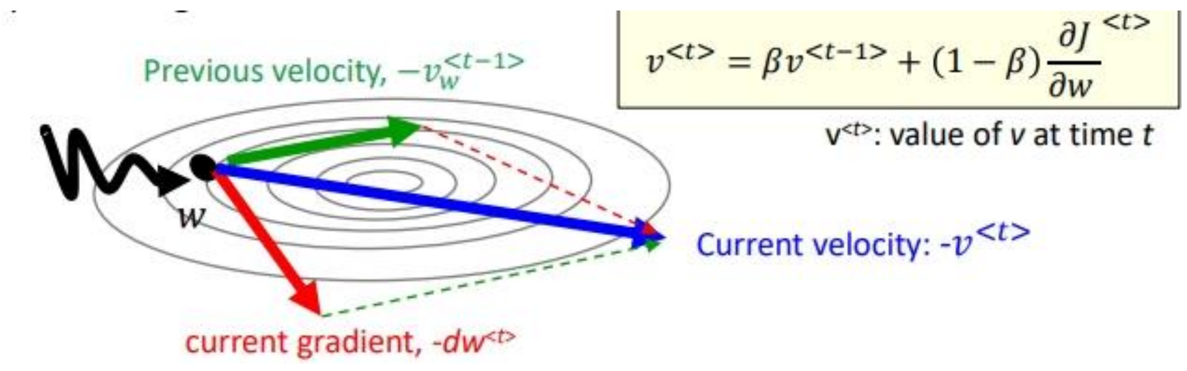

Previous velocity, $-v_w^{<t-1>}$
$v^{<t>} = \beta v^{<t-1>} + (1-\beta)\frac{\partial J}{\partial w}^{<t>}$
$v^{<t>}$: value of v at time t
w
Current velocity: $-v^{<t>}$
current gradient, $-dw^{<t>}$

Figure 3.2.1(d): Illustration of Momentum and momentum equation [12]

Meanwhile, momentum β is an optional tuning hyperparameters in neural networks. The existence of momentum is to tackle the issue of gradient oscillation in orthogonal direction especially in plateau area of the space [12]. By introducing momentum in the training process, it able to use exponentially weighted average of previous gradient to make the gradient update smoother by reducing the oscillation effect of normal update [12].

There are 6 hyperparameters that affect the performance of neural network the most, which are the number of hidden layers, number of neurons for each layer, learning rate (α), momentum (β), batch size, and number of epochs. In order to focus on comparing optimization algorithms, the project decided to fix the number of hidden layers at 5 layers to reduce the optimization time, in the meantime it ensures the ability of the model to learn high-level features through these layers. The rest of the hyperparameters will be tuned by the algorithms.

3.2.2 XGBoost Classifier

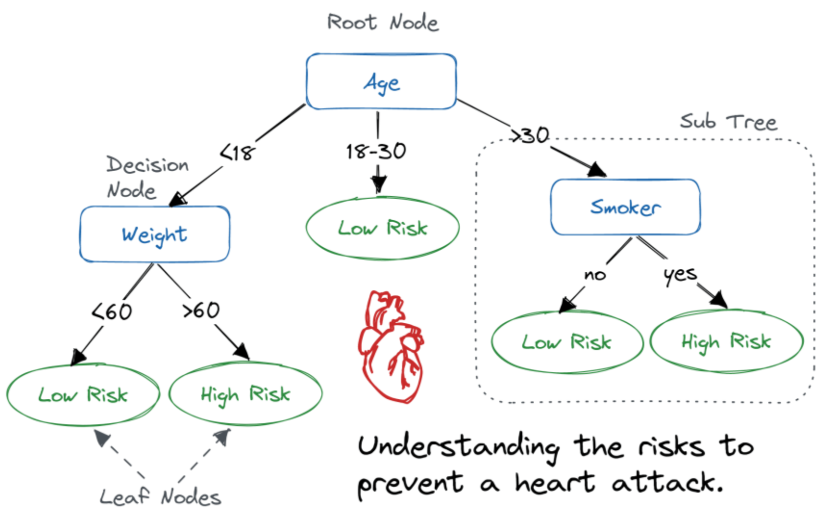


Figure 3.2.2(a): Illustration of Decision Tree [13]

XGBoost (acronym of eXtreme Gradient Boosting) is an enhanced machine learning model based on decision tree model. From Figure 3.2.2(a), a decision tree made its decision based on a series of conditions. In a tree model, it consists of a root node that act as the initial condition for classifying instance into category, then the classified instance will be further fed into decision node that consists of another rules [13]. The steps iteratively go through the decision tree until it reach the leave nodes that provide the final classification of the instance [13]. In particular, XGBoost is a decision tree that using Gradient Boosting algorithm to train the model, usually it is simplified as GBDT.

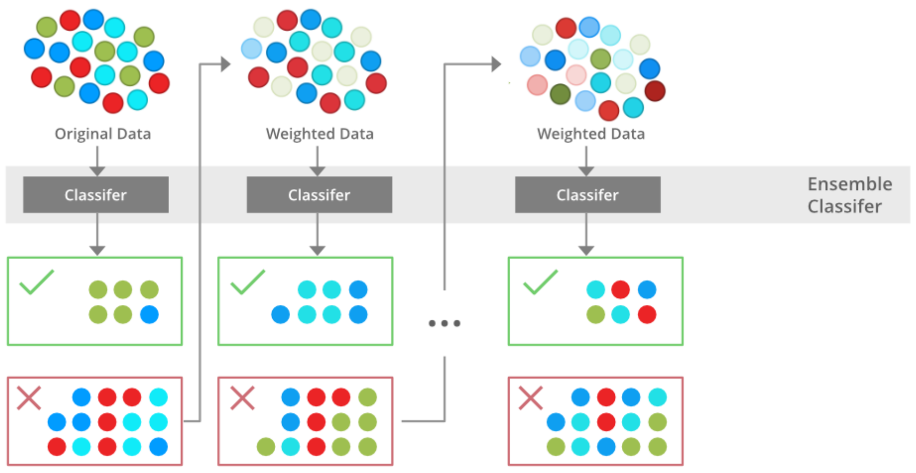
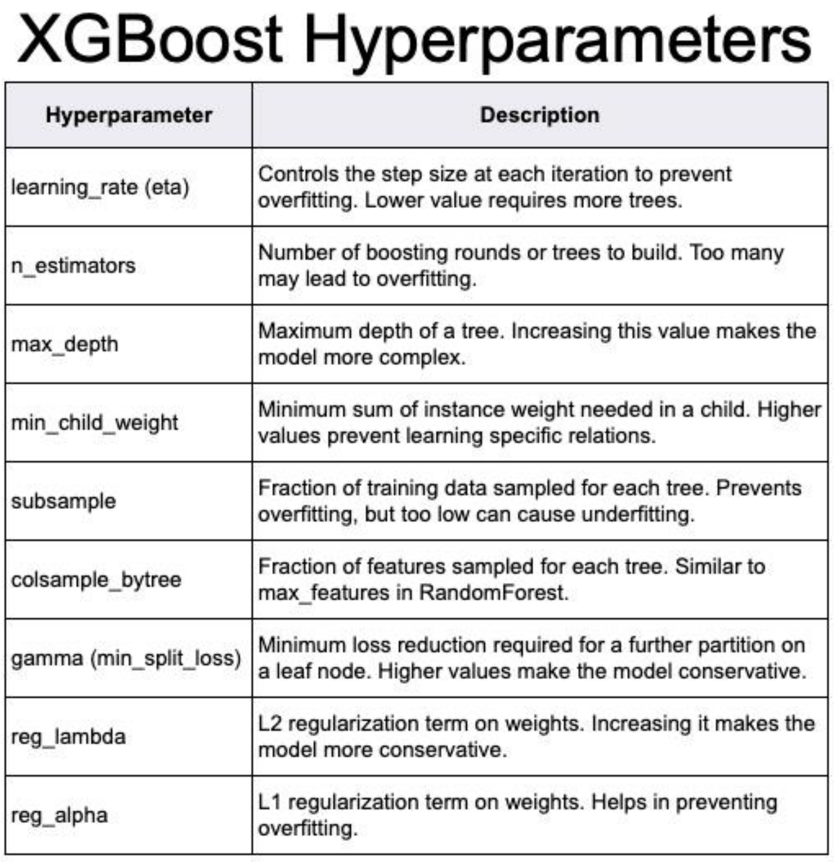


Figure 3.2.2(b): Illustration of Boosting [14]

Boosting is an ensemble modelling technique that build a classifier from weak classifiers [14]. An initial model is built from training data, then based on the errors in the first model, the second model is built, tries to correct the errors in the first model, and the steps are repeated until the maximum number of models built [14]. According to the author of [15], gradient boosting is like playing golf, every trial is to correct the direction and of the ball moving to the desired goal. In XGBoost, weights are assigned to all independent variables and fed into the decision tree to predict the result, the weight of the variables that are wrongly predicted is increased and the wrongly predicted variable is fed into the second tree to improve the model performance [14].


XGBoost Hyperparameters

Hyperparameter	Description
learning_rate (eta)	Controls the step size at each iteration to prevent overfitting. Lower value requires more trees.
n_estimators	Number of boosting rounds or trees to build. Too many may lead to overfitting.
max_depth	Maximum depth of a tree. Increasing this value makes the model more complex.
min_child_weight	Minimum sum of instance weight needed in a child. Higher values prevent learning specific relations.
subsample	Fraction of training data sampled for each tree. Prevents overfitting, but too low can cause underfitting.
colsample_bytree	Fraction of features sampled for each tree. Similar to max_features in RandomForest.
gamma (min_split_loss)	Minimum loss reduction required for a further partition on a leaf node. Higher values make the model conservative.
reg_lambda	L2 regularization term on weights. Increasing it makes the model more conservative.
reg_alpha	L1 regularization term on weights. Helps in preventing overfitting.

Figure 3.2.2(c): XGBoost hyperparameters according to importance and description [16]

Figure 3.2.2(c) shows the list of XGBoost hyperparameters by importance based on the author of [16]. According to the statement of the author [16], learning rate affect the stability of the model the most. In this project, 5 of the hyperparameters will be tuned as well, the top 3 hyperparameters was selected to tuned because they determined the structure of the model. However, XGBoost prone to overfitting especially when trained on small dataset or too many trees used in the model. Therefore, instead of selecting the top 5 most important hyperparameters, this project chooses to tune L1 and L2 regularization term, which is reg_lambda and reg_alpha to reduce the overfitting issues. In short, the target hyperparameters to tuned in this project include learning rate, n_estimators, max_depth, reg_lambda, and reg_alpha.

3.3 Gantt Chart

Phase	FYP1														FYP2													
Task / Week	1	2	3	4	5	6	7	8	9	10	11	12	13	14	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Data Acquisition	🟨	🟨	🟨	🟨																								
Data Preprocessing					🟨	🟨																						
Model Building and Training (NN)							🟨	🟨																				
Hyperparameter Optimization (NN)									🟨	🟨																		
Model Evaluation (NN)											🟨																	
Report Writing												🟨																
Presentation													🟨															
Model building and Training (XGBoost)															🟨	🟨												
Hyperparameter Optimization																	🟨											
Model Evaluation																	🟨											
API development																		🟨	🟨									
GUI development																			🟨	🟨								
System testing																				🟨	🟨	🟨						
Report Writing																						🟨	🟨	🟨	🟨	🟨		
Presentation																											🟨	

Figure 3.3: Gantt Chart of the project

Chapter 4

System Design

4.1 Data Acquisition

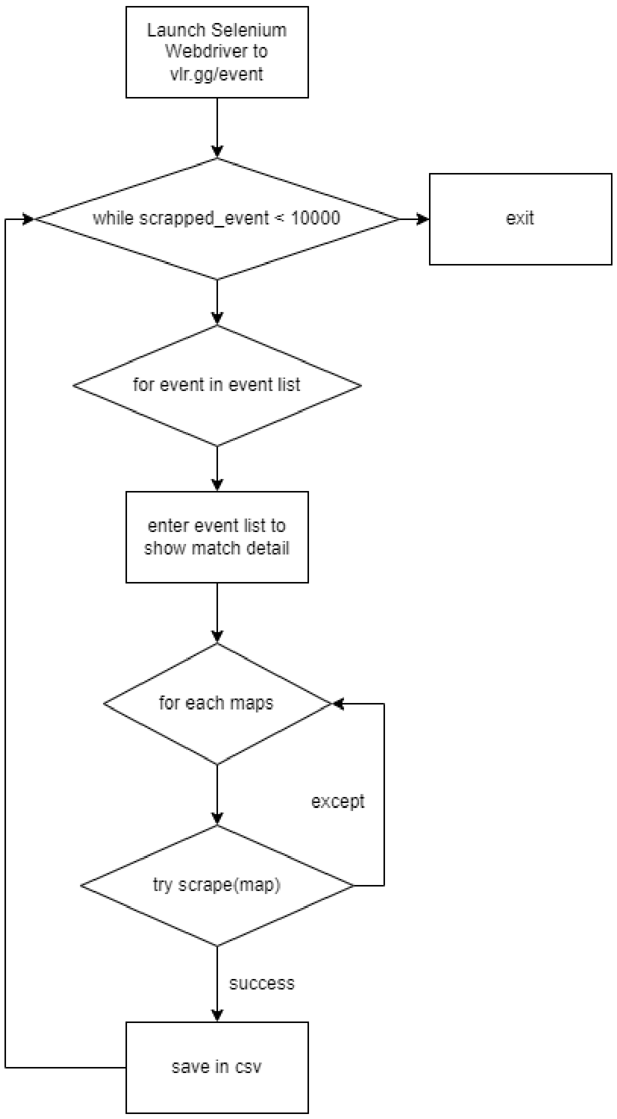
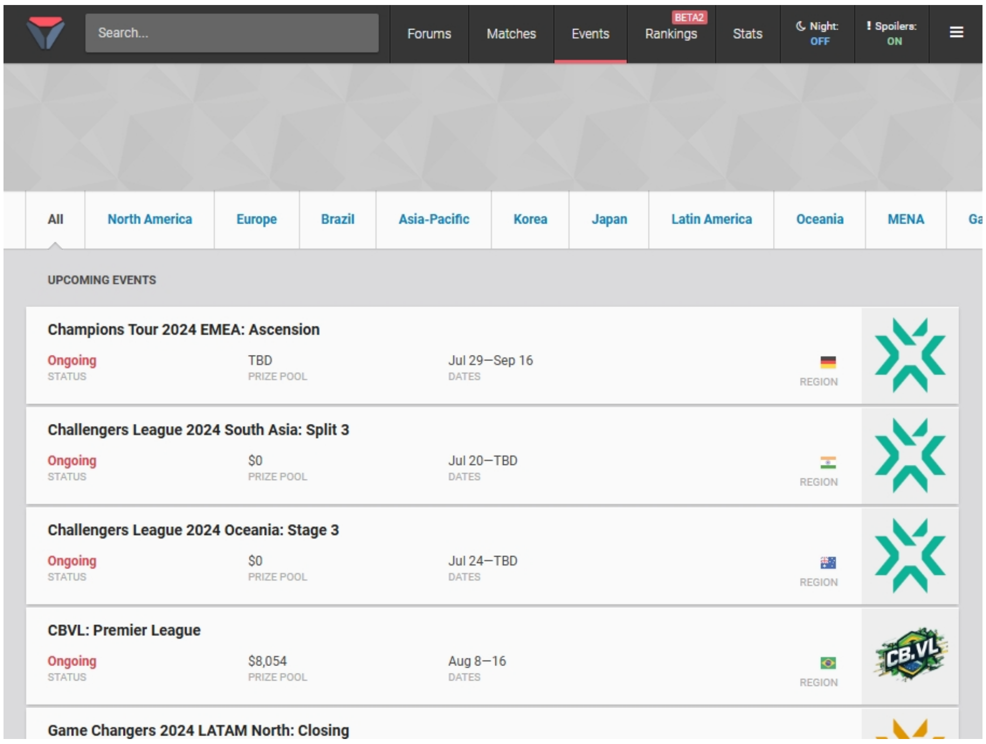

Launch Selenium Webdriver to vlr.gg/event
while scrapped_event < 10000
exit
for event in event list
enter event list to show match detail
for each maps
except
try scrape(map)
success
save in csv

Figure 4.1(a): Flowchart of the Python Scrapping script

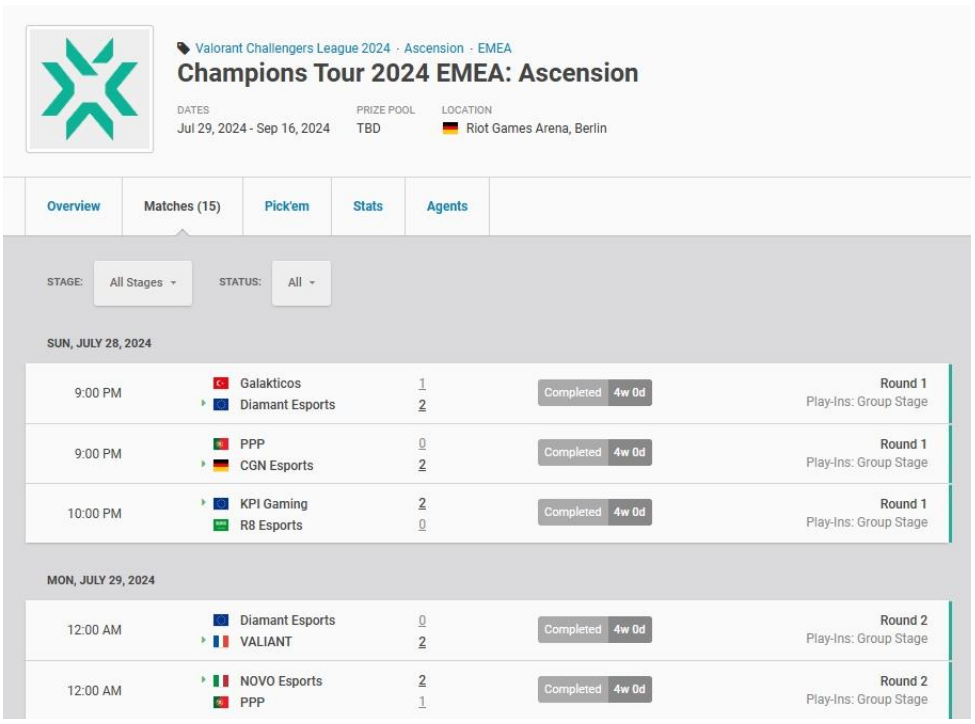
Data acquisition has been done during FYP 1, the scrapping process is done by using a Python script with Selenium and BeautifulSoup (BS4) to automate the scrapping process. The target event data is 10000 matches, and the script will stop scrape for new match.



UPCOMING EVENTS

Event	Status	Prize Pool	Dates	Region
Champions Tour 2024 EMEA: Ascension	Ongoing	TBD	Jul 29—Sep 16	
Challengers League 2024 South Asia: Split 3	Ongoing	$0	Jul 20—TBD	
Challengers League 2024 Oceania: Stage 3	Ongoing	$0	Jul 24—TBD	
CBVL: Premier League	Ongoing	$8,054	Aug 8—16	
Game Changers 2024 LATAM North: Closing				

Figure 4.1(b): Event tab shows all ongoing/completed international events [17]



Champions Tour 2024 EMEA: Ascension

DATES Jul 29, 2024 - Sep 16, 2024
PRIZE POOL TBD
LOCATION Riot Games Arena, Berlin

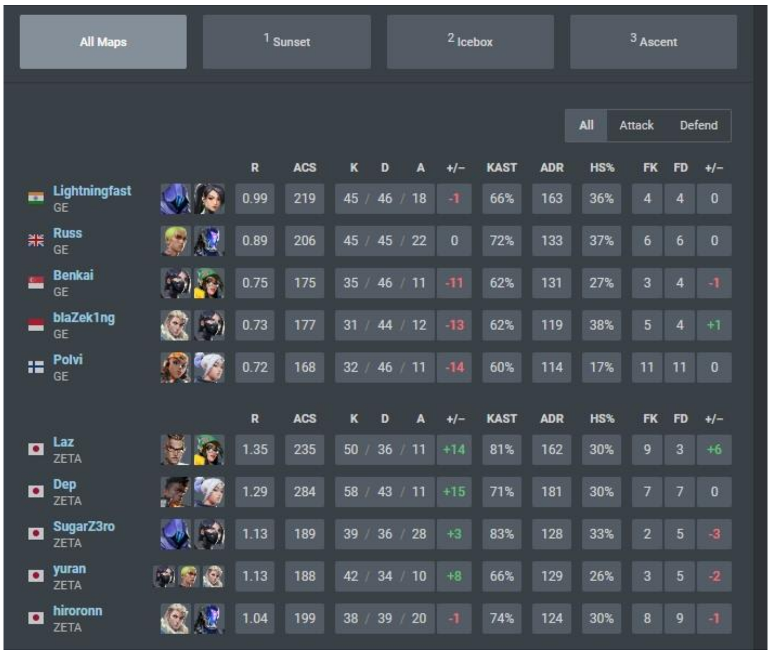
SUN, JULY 28, 2024

Time	Teams	Score	Status	Stage
9:00 PM	Galakticos / Diamant Esports	1 / 2	Completed 4w 0d	Round 1, Play-Ins: Group Stage
9:00 PM	PPP / CGN Esports	0 / 2	Completed 4w 0d	Round 1, Play-Ins: Group Stage
10:00 PM	KPI Gaming / R8 Esports	2 / 0	Completed 4w 0d	Round 1, Play-Ins: Group Stage

MON, JULY 29, 2024

Time	Teams	Score	Status	Stage
12:00 AM	Diamant Esports / VALIANT	0 / 2	Completed 4w 0d	Round 2, Play-Ins: Group Stage
12:00 AM	NOVO Esports / PPP	2 / 1	Completed 4w 0d	Round 2, Play-Ins: Group Stage

Figure 4.1(c): Matches for all stages in each event [17]

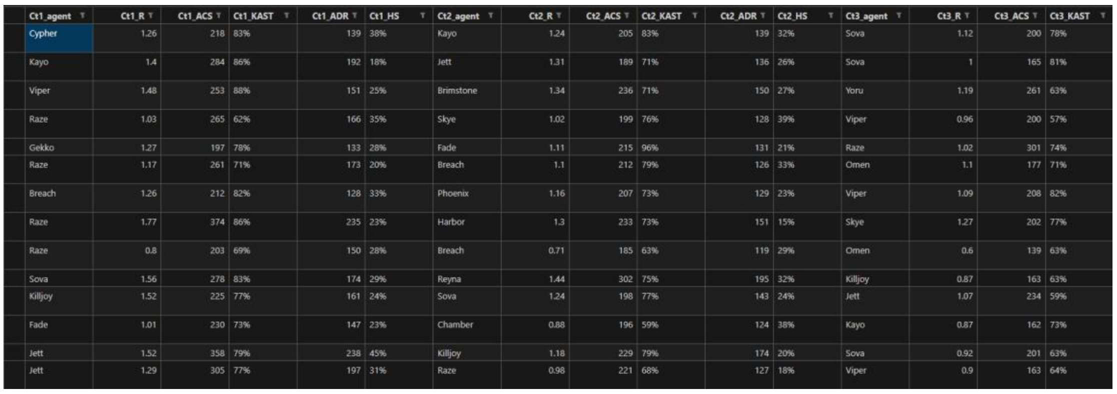


	R	ACS	K	D	A	+/–	KAST	ADR	HS%	FK	FD	+/–
Lightningfast GE	0.99	219	45	46	18	-1	66%	163	36%	4	4	0
Russ GE	0.89	206	45	45	22	0	72%	133	37%	6	6	0
Benkai GE	0.75	175	35	46	11	-11	62%	131	27%	3	4	-1
blaZek1ng GE	0.73	177	31	44	12	-13	62%	119	38%	5	4	+1
Polvi GE	0.72	168	32	46	11	-14	60%	114	17%	11	11	0

	R	ACS	K	D	A	+/–	KAST	ADR	HS%	FK	FD	+/–
Laz ZETA	1.35	235	50	36	11	+14	81%	162	30%	9	3	+6
Dep ZETA	1.29	284	58	43	11	+15	71%	181	30%	7	7	0
SugarZ3ro ZETA	1.13	189	39	36	28	+3	83%	128	33%	2	5	-3
yuran ZETA	1.13	188	42	34	10	+8	66%	129	26%	3	5	-2
hiroronn ZETA	1.04	199	38	39	20	-1	74%	124	30%	8	9	-1

Figure 4.1(d): Sample table of gameplay data and maps from vlr.gg [17]

Once the Selenium WebDriver launched, the page in Figure 4.1(b) will be shown. Using Selenium to locate the event list, the script will go through all the event to scrape the matches happened in the event. Figure 4.1(c) shows the page after an event is clicked. In an event, there usually have a group stage to select seeded teams, and a playoff stage, the official matches that decide the winner of the tournament. The script target to scrape all matches in group stage and playoff stage, therefore, the stage filter needs to show all stages. Then, the script will further locate the match for everyday as shown in Figure 4.1(c) and click the match. Figure 4.1(d) shows the page after the match is clicked. The target match data is in each map. Hence, the script will go through the map element above, and scrape the data below as shown in Figure 4.1(d). All the scrapped data will be stored into a .csv file as shown in Figure 4.1(e).



Ct1_agent	Ct1_R	Ct1_ACS	Ct1_KAST	Ct1_ADR	Ct1_HS	Ct2_agent	Ct2_R	Ct2_ACS	Ct2_KAST	Ct2_ADR	Ct2_HS	Ct3_agent	Ct3_R	Ct3_ACS	Ct3_KAST
Cypher	1.26	218	83%	139	38%	Kayo	1.24	205	83%	139	32%	Sova	1.12	200	78%
Kayo	1.4	284	86%	192	18%	Jett	1.31	189	71%	136	26%	Sova	1	165	81%
Viper	1.48	253	88%	151	25%	Brimstone	1.34	236	71%	150	27%	Yoru	1.19	261	63%
Raze	1.03	265	62%	166	35%	Skye	1.02	199	76%	128	39%	Viper	0.96	200	57%
Gekko	1.27	197	78%	133	28%	Fade	1.11	215	96%	131	21%	Raze	1.02	301	74%
Raze	1.17	261	71%	173	20%	Breach	1.1	212	79%	126	33%	Omen	1.1	177	71%
Breach	1.26	212	82%	128	33%	Phoenix	1.16	207	73%	129	23%	Viper	1.09	208	82%
Raze	1.77	374	86%	235	23%	Harbor	1.3	233	73%	151	15%	Skye	1.27	202	77%
Raze	0.8	203	69%	150	28%	Breach	0.71	185	63%	119	29%	Omen	0.6	139	63%
Sova	1.56	278	83%	174	29%	Reyna	1.44	302	75%	195	32%	Killjoy	0.87	163	63%
Killjoy	1.52	225	77%	161	24%	Sova	1.24	198	77%	143	24%	Jett	1.07	234	59%
Fade	1.01	230	73%	147	23%	Chamber	0.88	196	59%	124	38%	Kayo	0.87	162	73%
Jett	1.52	358	79%	238	45%	Killjoy	1.18	229	79%	174	20%	Sova	0.92	201	63%
Jett	1.29	305	77%	197	31%	Raze	0.98	221	68%	127	18%	Viper	0.9	163	64%

Figure 4.1(e): View of the scrapped dataset

4.2 Data Cleaning and Preprocessing

Once the .csv file contains 10000 data, the next steps are to perform data cleaning and preprocessing using a Python script with Pandas data frame. In data cleaning process, the main steps are to remove and replace unknown values. For example, an empty cell in the website table is represented as a null symbol, and it appears in the .csv file as "\xa0" string. The string is removed and leaved empty for further process. Hence, the dataset now consists of data that have null value. The next step is to remove the entire row that contain null value instead of filling in with mean or median value, which is commonly used in data cleaning steps. Then, the numeric data that are scrapped as string, are converted to float type data.

After data cleaning, the data will undergo preprocessing. Before performing any preprocessing, categorical data needed to be encoded using label encoder. Then, the correlation and distribution between all the numerical data is calculated. Figure 4.2(a) shows the correlation heatmap between all numerical data, visualized using matplotlib.

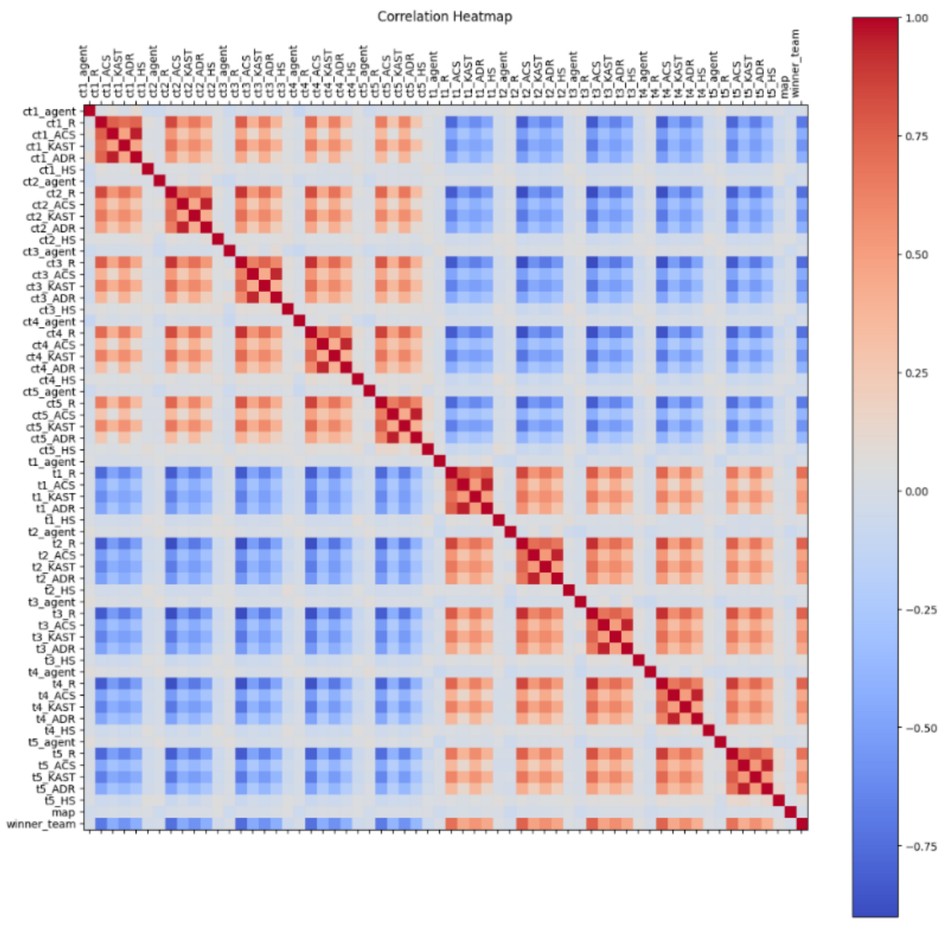


Figure 4.2(a): Correlation Heatmap for numerical data

The correlation heatmap contains all numerical data and some categorical data that are encoded as numerical label. The numerical labels are ignored. Noticed that in numerical data, only the Headshot percentage (HS%), is not correlated to any data in the entire dataset. Therefore, HS% is dropped to reduce the dimension of dataset. By observing the visualized chart of numerical data distribution, data such as Rating (R), Average Combat Score (ACS), Average Damage per Round (ADR) is normalized, while Kill, Assist, Survived, Traded percentage (KAST) is not normalized based on the skewed pattern in all KAST graph. According to [18], normalized data able to improve model performance by speeding up the convergence and also speed up gradient descent. Normalization process is done to KAST using MinMaxScaler from Scikit-learn. Then, all the numerical data that are not represented in percentage undergo standardization process using StandardScaler from Scikit-learn. This is because standardized data able to improve the convergence speed by ensuring all features have the same scale [18]. Standardizing the data also can facilitate Gradient Descent by preventing the gradient descent optimization skewed due to different feature scale [18]. These data include R, ACS, ADR. Then, all the Label Encoder, Standard Scaler, and MinMax Scaler object is stored as a .pkl (pickle) file for future use.

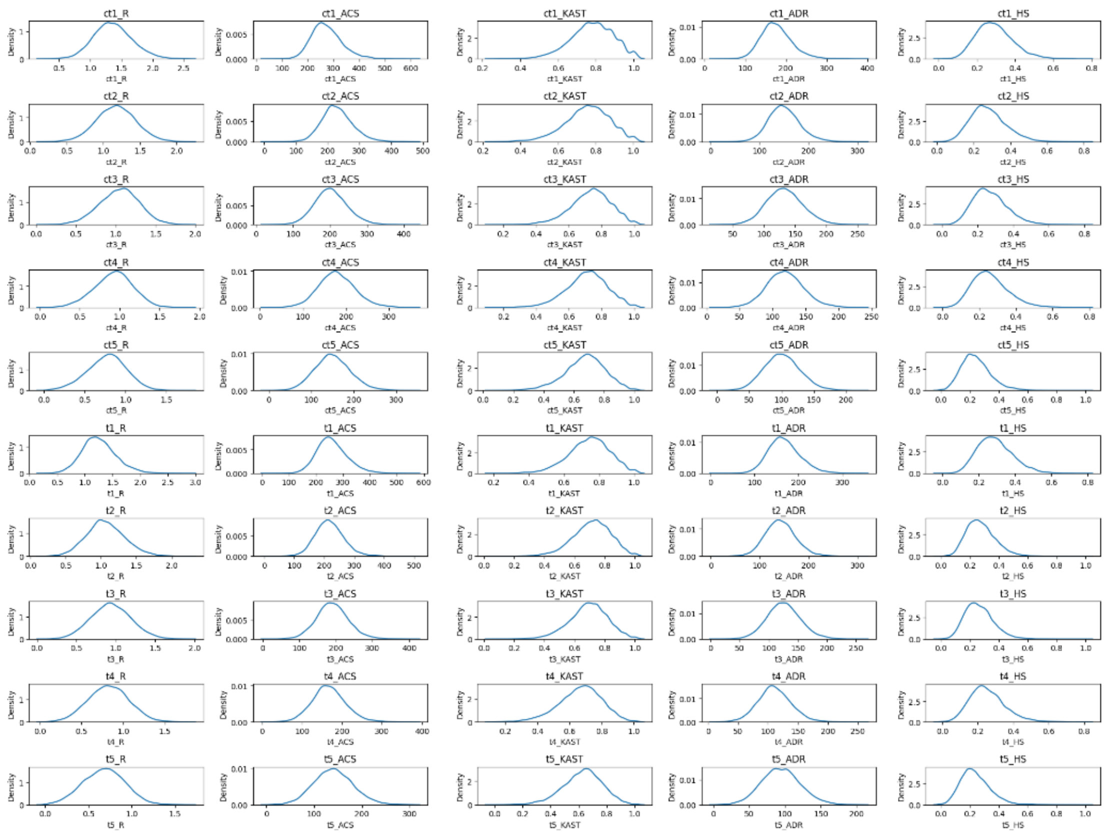


Figure 4.2(b): Data distribution of numerical data

4.3 Model Building, Training, and Hyperparameter Optimization

4.3.1 General Procedures

This subchapter introduces the general steps that have to be done regardless of the model type. Before training any model, the data has to be prepared properly. First of all, the preprocessed data needs to be loaded as a Pandas data frame. Then, the features and the result of the data is splitted as X (features) and y (ground truth). The X and y are then further split into train set and test set, where the ratio of train set versus test set is 80:20 with stratify to ensure the distribution of the data in both datasets are same. The test set will not involve in any training process. The purpose of test set is to evaluate the model under unforeseen data. The training and validation process will be performed only using the train set. In cross validation, the train set will be further split into ratio of 80:20 for 5-fold as shown in Figure 4.3.1. To facilitate the development process, some helper functions are created to reduce the redundancy of codes. The helper functions included perform prediction function, evaluate performance, plot curves, train functions, cross validation, etc.

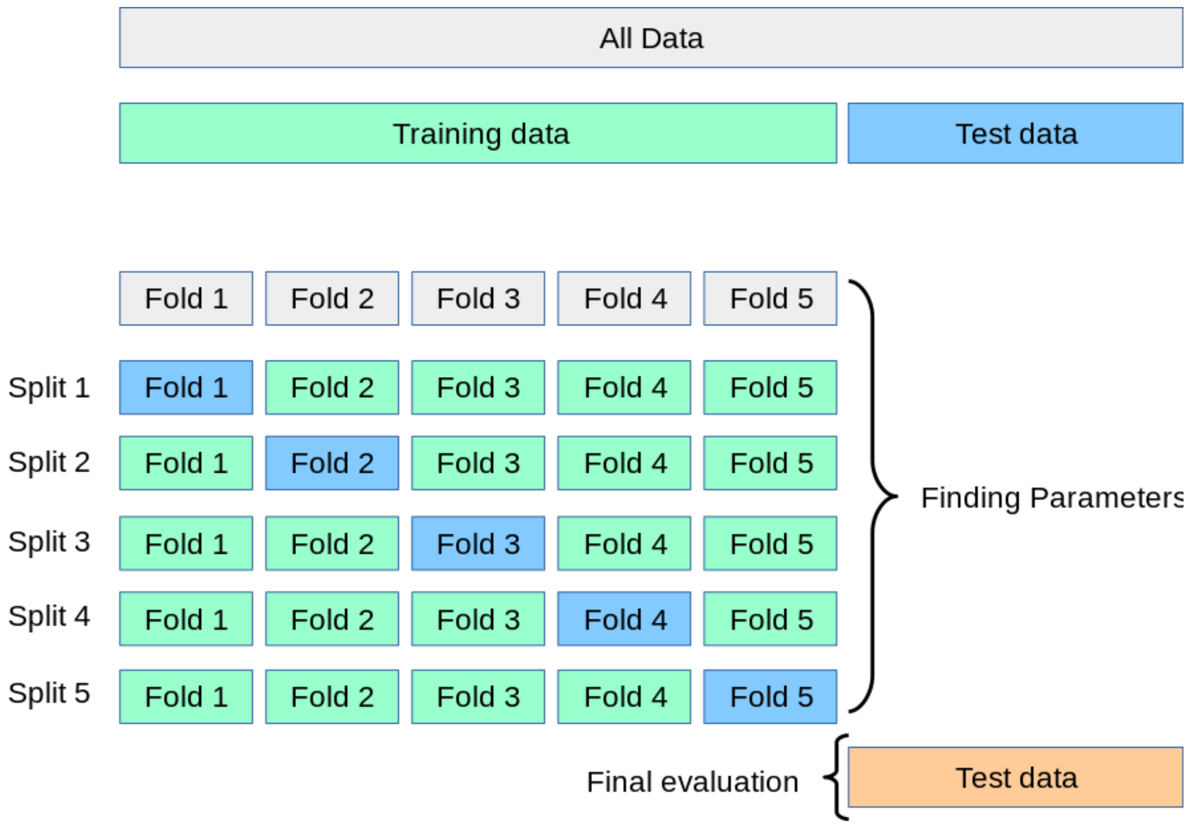


Figure 4.3.1: Illustration of 5-fold cross validation [19]

4.3.2 Neural Network

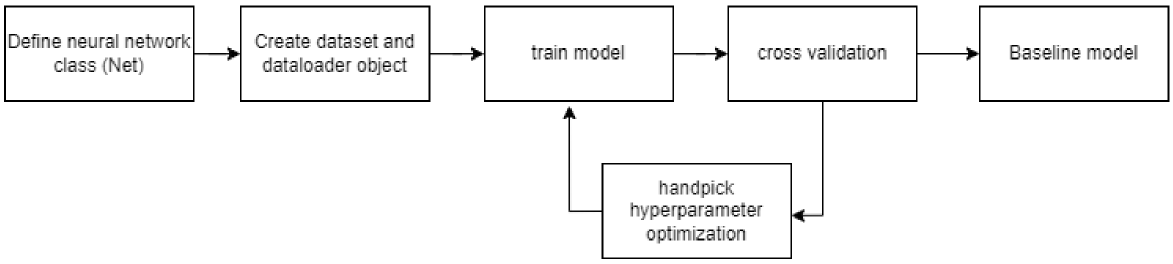

Define neural network class (Net)
Create dataset and dataloader object
train model
cross validation
Baseline model
handpick hyperparameter optimization

Figure 4.3.2: Flowchart of neural network development

The main library used in this project for neural network is PyTorch. To start building and training the base model of neural network, the dataset needs to be loaded into a data loader as the project implement mini-batch training for the neural network. To achieve this, the X_train (features in training set) and the y_train (ground truths of training set) is packed as a Dataset object. Then, the Dataset object is loaded into the DataLoader object with the specific batch size.

Next, a customized class, Net, that allows layer size customization is created to make the hyperparameter optimization easier. The class take an integer tuple as parameter and append the neuron layers with specified number of neurons into the neural network. Then, the first neural network can be trained, with the handpicked hyperparameters (number of epochs, learning rate, batch size, and number of neurons for each layer. Momentum default = 0.9). Using the train function in the helper functions to train the model. Then, using the evaluate function to evaluate the model. The result obtained will be the baseline result for the project, and the optimized model should have a better result than the baseline model.

4.3.3 XGBoost Classifier

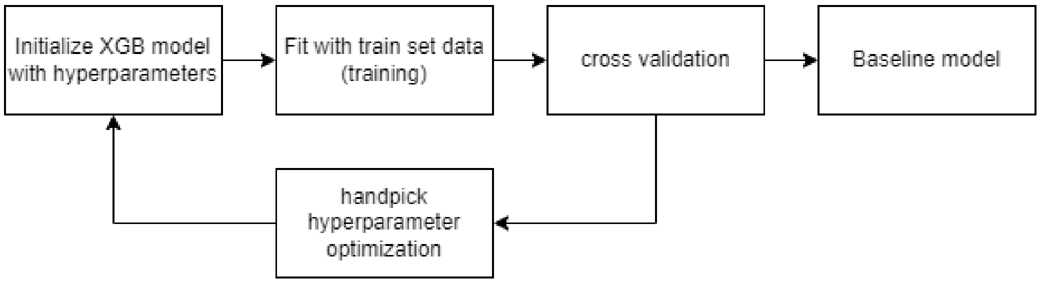

Initialize XGB model with hyperparameters
Fit with train set data (training)
cross validation
Baseline model
handpick hyperparameter optimization

Figure 4.3.3: Flowchart of XGBoost model development

XGboost library is mainly used in this part of the project. Unlike neural network, training XGBoost is simpler than training a neural network. The XGBoost library provides useful API to ease the training process. XGBoost also does not implement mini-batch training method, therefore the training of XGBoost and its development is relatively easy. In the beginning, a baseline model is initialized with handpicked hyperparameters (n_estimators and max_depth). The cross-validation result of baseline model is recorded for future reference.

4.3.4 Hyperparameter Optimization

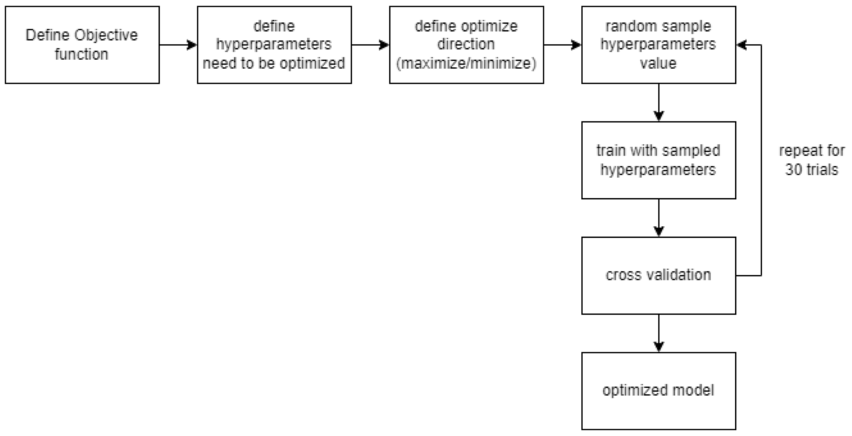

Define Objective function
define hyperparameters need to be optimized
define optimize direction (maximize/minimize)
random sample hyperparameters value
train with sampled hyperparameters
repeat for 30 trials
cross validation
optimized model

Figure 4.3.4(a): Flowchart of Random Serach hyperparameter optimization using Optuna

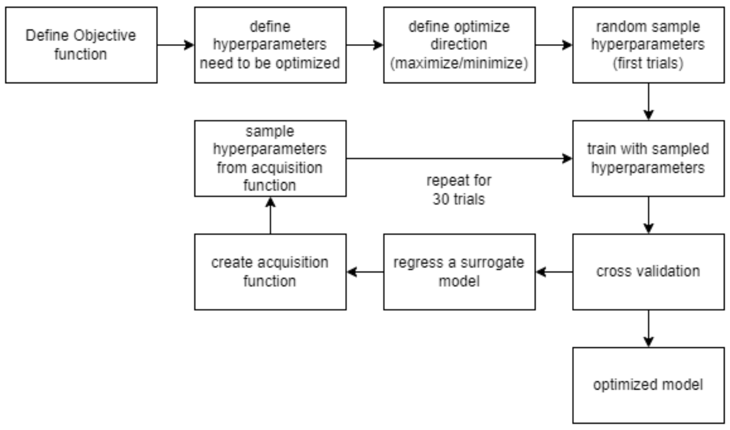

Define Objective function
define hyperparameters need to be optimized
define optimize direction (maximize/minimize)
random sample hyperparameters (first trials)
sample hyperparameters from acquisition function
train with sampled hyperparameters
repeat for 30 trials
create acquisition function
regress a surrogate model
cross validation
optimized model

Figure 4.3.4(b): Flowchart of Bayesian hyperparameter optimization using Optuna

The project uses Optuna hyperparameter optimization framework to perform optimization. The library provides both random search optimization and Bayesian optimization, as well as optimization process visualization to allow user to have a better view and able to compare the differences between both methods.

First, an objective function is required to decide the hyperparameters need to be tuned and the objective value that determine the result. Optuna only support single objective hyperparameters optimization for Bayesian optimization, hence the project chooses to maximize the model's average validation accuracy in 5-fold as the ultimate objective of the optimization. The number of trials is set to 30 trials per optimization.

Before the optimization starts, an Optuna storage object is created and passed into the optimization function. The purpose of the storage is to record and store the results of each trial, the storage is also used to visualize the optimization results for analysis purpose. The time of start and end of the optimization is recorded and compared between different models and optimization methods.

After 30 trials of optimization, the best trial hyperparameters is accessed and the models are trained again with the best hyperparameters as the optimized model. Then, using the evaluation helper function, evaluate the optimized model with entire train set and test set data to check whether the models suffer from overfitting issues. Other than validation accuracy, the other metrics such as ROC-AUC, precision-recall curve, confusion matrix, etc., are evaluated and plotted using matplotlib for better visualization of the model's capabilities. The optimized model is saved using PyTorch **save()** method for neural network and XGB model object **save_model()** method for XGBoost for API implementation in the next step.

4.4 API Development

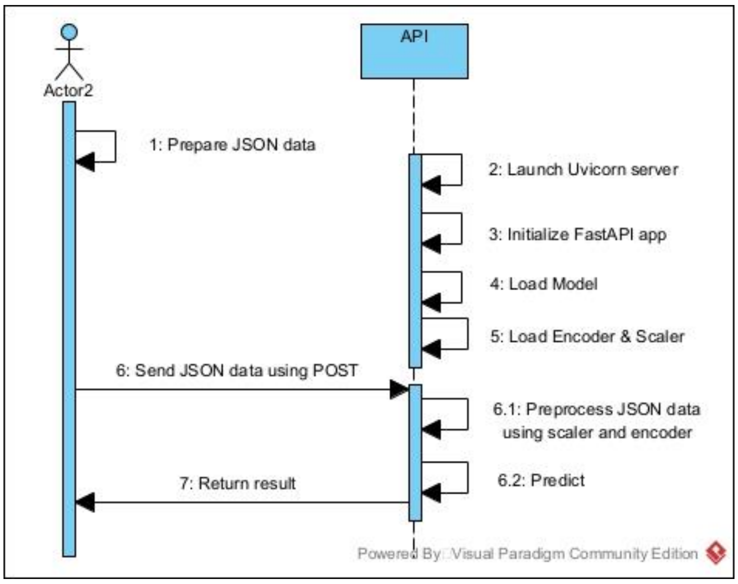

Actor2
API
1: Prepare JSON data
2: Launch Uvicorn server
3: Initialize FastAPI app
4: Load Model
5: Load Encoder & Scaler
6: Send JSON data using POST
6.1: Preprocess JSON data using scaler and encoder
6.2: Predict
7: Return result
Powered By Visual Paradigm Community Edition

Figure 4.4: Sequence diagram of API development

The API development process uses multiple libraries such as FastAPI as API development framework, Pydantic for data input verification, Uvicorn as web server, and so on. Before the API can launch, the saved models, encoders and scalers in the previous subchapter need to be loaded. The encoders and scalers are to ensure the new input data is preprocessed identically as in the training dataset. When the server receive data from user, the API first preprocess the data using the encoders and scalers. Then the model able to take the preprocessed data to perform prediction. The prediction is then return to user in JSON format. Once the API method is defined, the Uvicorn server is launched and hosted on IP address of 0.0.0.0 and on port 8000.

To make the API available on the Internet instead of only hosting it on a machine locally, the API code is uploaded to GitHub and deployed on a web hosting service called Render [20]. Render is a company that provides web hosting services, and it has a free option to host any web services with basic performance, which is suitable for this project. After ensuring the API can run properly on local machine, a requirements.txt, which describe the current Python environment, is generated and uploaded to Render to ensure that the environment on Render is same as local machine so it can run as expected. Once the API is deployed successfully on Render, the API is tested with dummy data using Postman to ensure the API is working and it can return the result to the user.

4.5 GUI Development

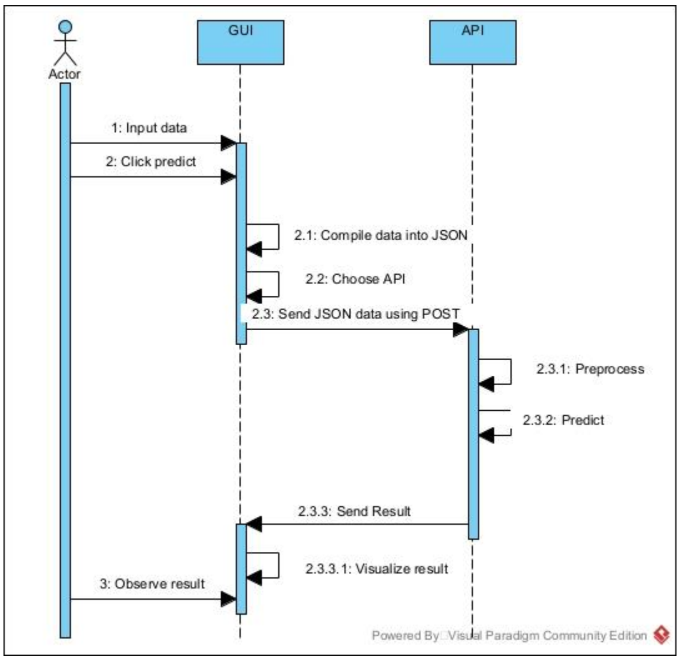

Actor
GUI
API
1: Input data
2: Click predict
2.1: Compile data into JSON
2.2: Choose API
2.3: Send JSON data using POST
2.3.1: Preprocess
2.3.2: Predict
2.3.3: Send Result
2.3.3.1: Visualize result
3: Observe result
Powered By Visual Paradigm Community Edition

Figure 4.5(a): Sequence diagram of GUI operation

Figure 4.5(b) shows the wireframe of the GUI design. The interface has a map input field, a model selection input field, and defender and attacker field. Inside defender and attacker consist of 5 players, each players have 5 input field including Agent, Rating, ACS, KAST, and ADR. Below the fields are a predict button that will compile the input data and convert it from plain text to JavaScript Object Notation (JSON) format. Then, the GUI will send the compiled JSON data using POST method to the respective API of the selected model. After receiving the response from the API, the GUI will utilize the response data and visualize the win rate into a horizontal bar chart using matplotlib.

Note that the main purpose of developing this GUI is to ease the normal user to access the model using a simple graphical interface, it is not the main focus of this project. Instead, the API is the main deliverable of the project. Therefore, the GUI only provide simple function like communicate with selected API, display prediction result and visualize the result in bar chart.

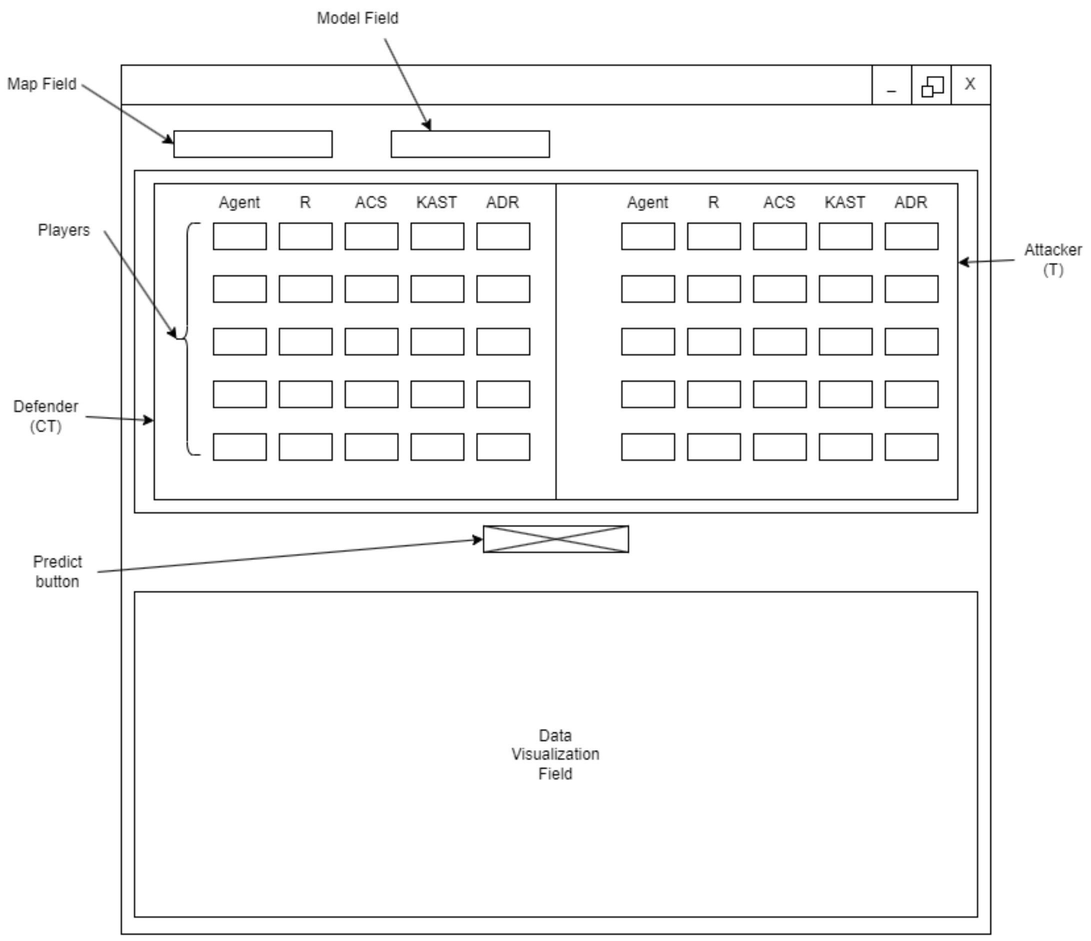


Figure 4.5(b): GUI wireframe design

Chapter 5

System Implementation

5.1 Hardware Setup

Table 5.1(a) shows the specification of the laptop used in this project. Besides using this laptop, this project also utilized some cloud computing instance such as in Google Colab and Render. The specification of both instance is listed in Table 5.1(b) and (c).

Description	Specifications
Model	Asus VivoBook A510U
Processor	Intel i5-8250U
Operating System	Windows 11 Home Edition
Graphic	Intel UHD Graphics 620 + NVIDIA GeForce MX150
Memory	16GB DDR4 2400MHz
Storage	512 GB SATA SSD + 1TB HDD

Table 5.1(a): Specifications of laptop

Description	Specifications
Hardware accelerator	CPU
Runtime type	Python 3
System RAM	12.7GB
Disk	107.7GB

Table 5.1(b): Google Colab instance specifications

Description	Specificaitons
Runtime (Language)	Python 3
Region	Singapore (SEA)
RAM	512MB
CPU	0.1 compute unit

Table 5.1(c): Render.com web service instance specifications

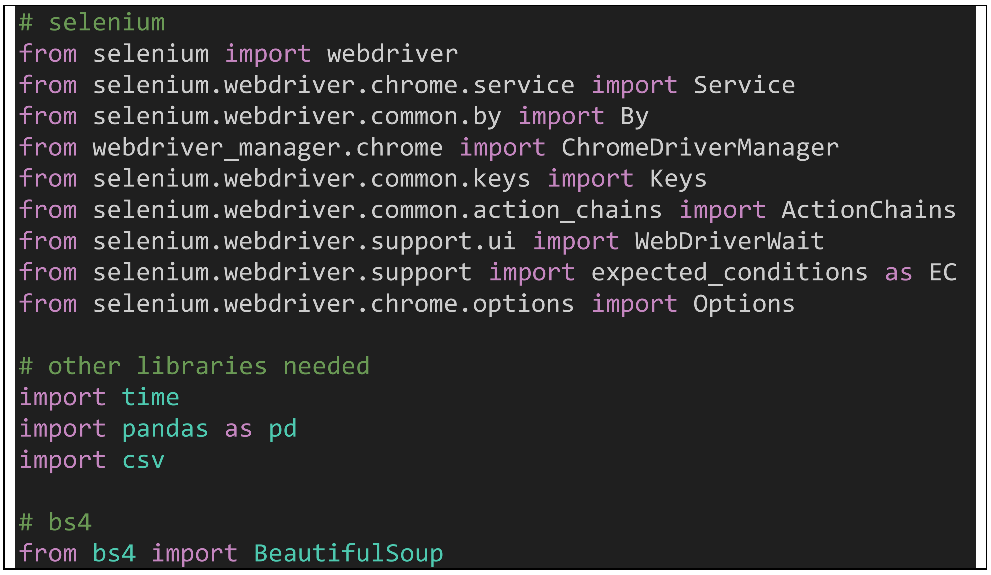
Note that the Google Colab and Render instance used in this project is free version. A free version of instance usually only provides basic functionality and sufficient system resources for a simple application. For Google Colab, the free CPU instance is used to train the models to ensure the computational power is consistent along the project, so the comparison of the optimization methods and models training time in this project is more referrable.

While for Render hosting, this project is using a free version as well. The only downside for the free version of Render is the instance will be suspended after some time of idle, and restart the instance will take up some time, it will make the GUI to be not responding for the first few minutes, after the instances is restarted, everything will work as normal and responsive (respond time around 200ms).

5.2 Software Setup and Configuration

In this project, the main programming language is Python 3, to be specific, the version of Python used is 3.11.9 from Anaconda for the entire project to ensure that the compatibility of the code. The editor used are Microsoft Visual Studio Code (VSCode) with the Python and Jupyter Notebook extensions installed for better development experience. The source codes are uploaded to GitHub public repository and the link to the repository is attached in the appendix of this report.

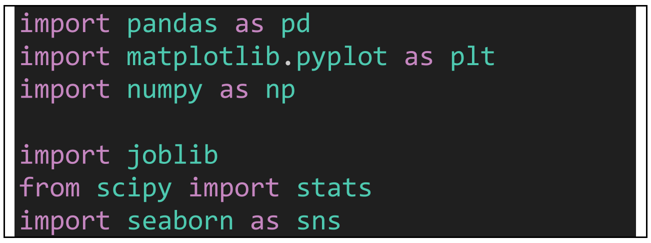
5.2.1 Data Acquisition and Preprocessing



```
# selenium
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.common.by import By
from webdriver_manager.chrome import ChromeDriverManager
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.action_chains import ActionChains
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.chrome.options import Options

# other libraries needed
import time
import pandas as pd
import csv

# bs4
from bs4 import BeautifulSoup
```

Figure 5.2.1(a): Library required to scrape data from vlr.gg

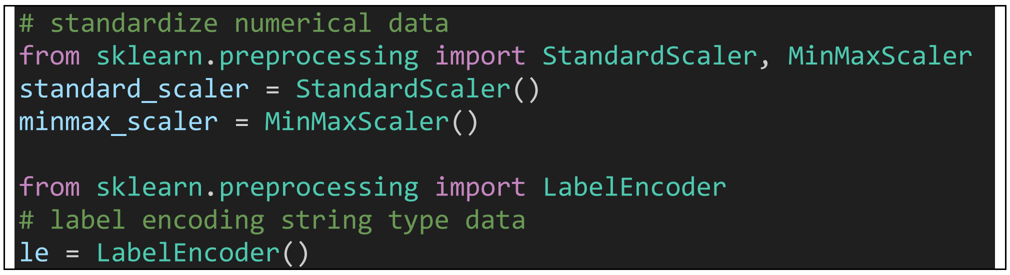
Figure 5.2.1(a) shows the required library for scrapping the data from vlr.gg. Selenium is the library that automate the browser action using the web driver. The time library is to create some short pause within the script when the page is redirected to ensure that the script can run smoothly. While pandas and csv library are to arrange the data in desired format and save into .csv file. BeautifulSoup library allows the script to interact with HTML elements on the website and extract the desired data from the HTML.



```python
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

import joblib
from scipy import stats
import seaborn as sns
```

Figure 5.2.1(b): Library required for data cleaning and data distribution visualization

After the data acquisition is completed, the data needs to undergo cleaning and preprocessing. In data cleaning, the required library is shown in Figure 5.2.1(b), the pandas library here is to load the raw .csv file into a data frame so that the following process is easier to be performed. Meanwhile, matplotlib and seaborn is to visualize the data distribution and correlation heatmap, so that the irrelevant data can be removed, and determine how the preprocessing steps is going to be implemented.



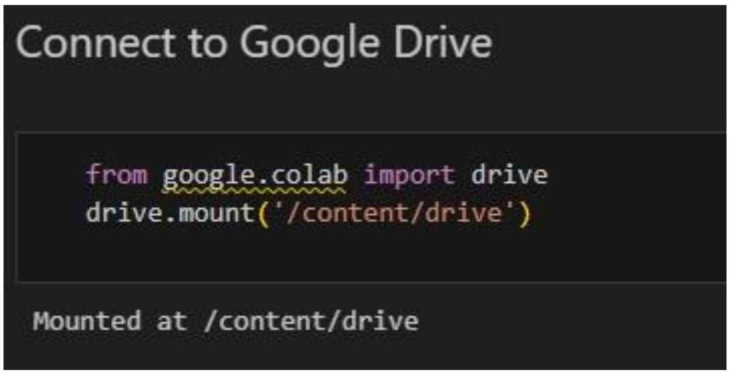
```python
# standardize numerical data
from sklearn.preprocessing import StandardScaler, MinMaxScaler
standard_scaler = StandardScaler()
minmax_scaler = MinMaxScaler()

from sklearn.preprocessing import LabelEncoder
# label encoding string type data
le = LabelEncoder()
```

Figure 5.2.1(c): Library required for data preprocessing

In data preprocessing, all the categorical data is converted to integer label using LabelEncoder from Scikit-learn. Using the matplotlib and seaborn generated correlation heatmap and data distribution chart, certain features will undergo normalization and standardization using MinMaxScaler and StandardScaler. The preprocessed data is then exported as another .csv file to avoid confusion with raw data file. After preprocessing is done, the Scalers and Encoder objects are saved as .pkl file using joblib library. The objects are needed in API development.

5.2.2 Model Training and Hyperparameter Optimization

The model training and hyperparameter optimization steps are done on Google Colab, therefore the Jupyter notebook has to be connected to Google Drive first with personal account signed in as shown in Figure 5.2.2(a), in order to utilize the CPU power from Google Colab instance.

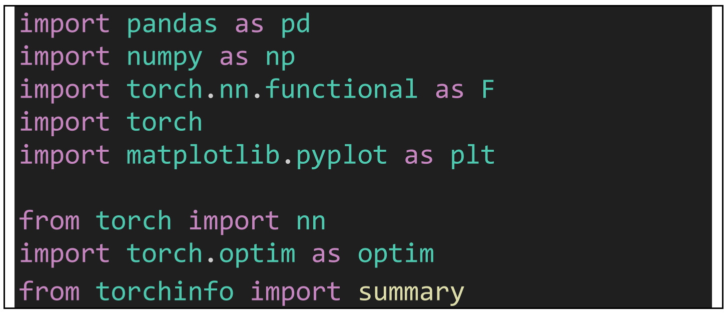

Connect to Google Drive

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

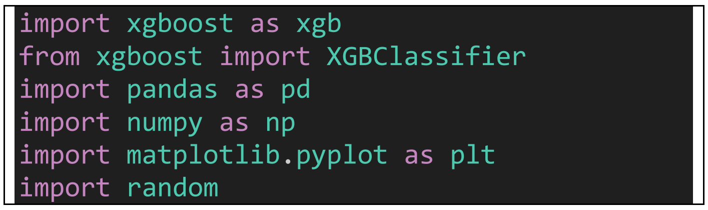
Figure 5.2.2(a): Connect to Google Drive in Google Colab

Since the Google Colab does not necessarily have the library required by our project, some of the libraries such as scikit-plot, torchinfo, optuna, etc., are installed using “! pip” command directly in Google Colab. Before start, ensure all the required files such as the preprocessed data .csv file are in the Google Drive, and directory is changed as the .csv file using “cd” command.



```
import pandas as pd
import numpy as np
import torch.nn.functional as F
import torch
import matplotlib.pyplot as plt

from torch import nn
import torch.optim as optim
from torchinfo import summary
```

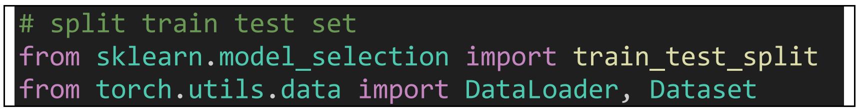
Figure 5.2.2(b): Libraries required for training neural network



```python
import xgboost as xgb
from xgboost import XGBClassifier
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import random
```

Figure 5.2.2(c): Libraries required for training XGBoost Classifier

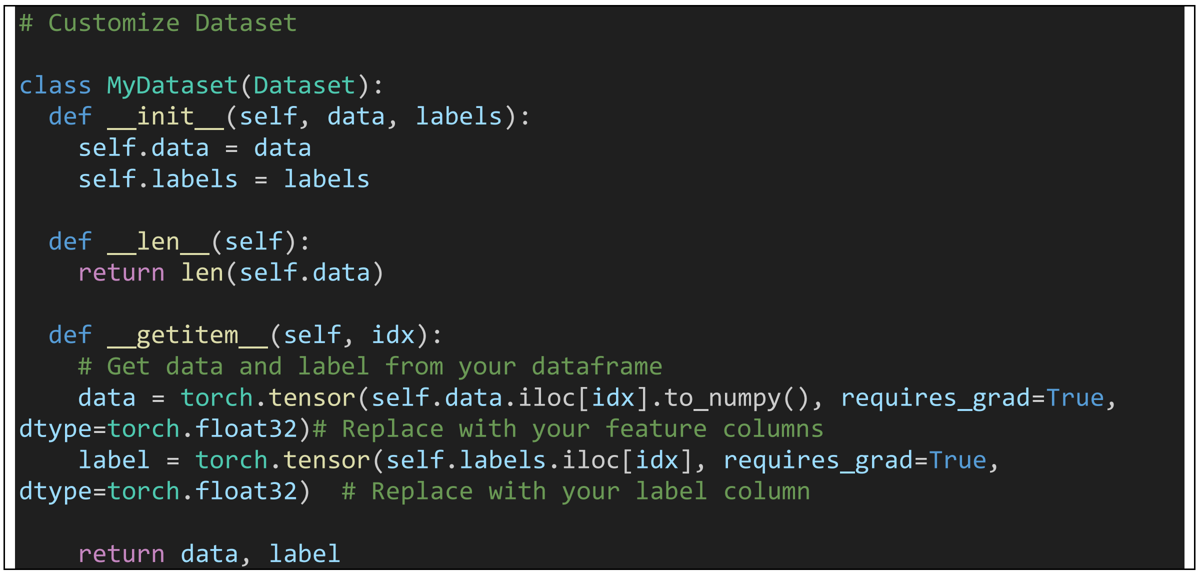
Figure 5.2.2(b) shows the main library used in the neural network training process. Besides pandas and numpy for reading the .csv file, the PyTorch (torch) library is the main framework to build and train a neural network, including “nn.functional” for cross entropy functionality, “torch.optim” for the SGD optimizer, and the “torchinfo” for a brief view of the neural network created. While for XGBoost, the required libraries are relatively lesser than neural network. As shown in Figure 5.2.2(c), the only difference is the XGBoost library is used instead of multiple torch libraries.



```python
# split train test set
from sklearn.model_selection import train_test_split
from torch.utils.data import DataLoader, Dataset
```

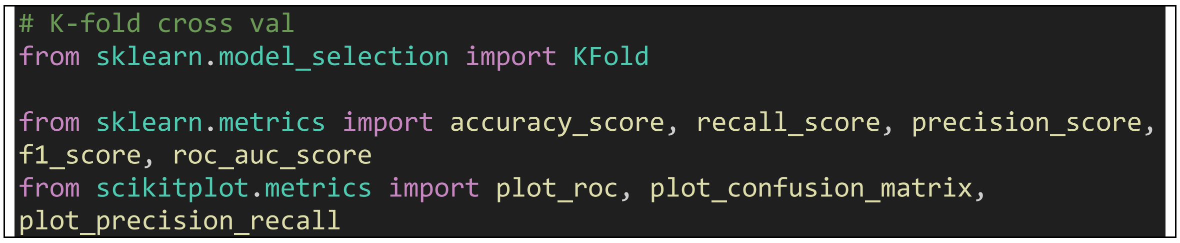
Figure 5.2.2(d): Libraries required for data splitting and data batching

Figure 5.2.2(d) shows the libraries and classes required to import to the notebook to perform data splitting. The main function required is the train_test_split from sklearn. The purpose of the function is to split the dataset into a train set and a test set by a given ratio and the distribution of the data in both set is equal. For neural network model, it requires further action to convert the train set into batches using DataLoader and Dataset classes. The batch class is shown in Figure 5.2.2(e).



```
# Customize Dataset

class MyDataset(Dataset):
  def __init__(self, data, labels):
    self.data = data
    self.labels = labels

  def __len__(self):
    return len(self.data)

  def __getitem__(self, idx):
    # Get data and label from your dataframe
    data = torch.tensor(self.data.iloc[idx].to_numpy(), requires_grad=True,
dtype=torch.float32)# Replace with your feature columns
    label = torch.tensor(self.labels.iloc[idx], requires_grad=True,
dtype=torch.float32)  # Replace with your label column

    return data, label
```

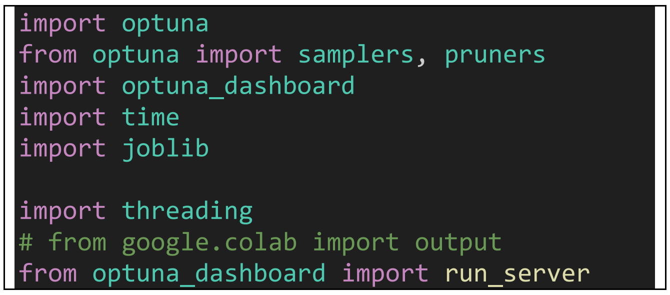
Figure 5.2.2(e): Customized batch dataset class for neural network



```
# K-fold cross val
from sklearn.model_selection import KFold

from sklearn.metrics import accuracy_score, recall_score, precision_score,
f1_score, roc_auc_score
from scikitplot.metrics import plot_roc, plot_confusion_matrix,
plot_precision_recall
```

Figure 5.2.2(f): Evaluation functions

Figure 5.2.2(f) shows libraries and functions required to perform model evaluation. The model evaluation and training process is written as a helper functions, so that the function can be reused in future easily. The KFold class is used to perform cross validation on the train set. Once the cross validation is completed, the performance of each validation is recorded, and average of the performance is calculated. Besides, some plots such as ROC, confusion matrix, precision-recall curve, is plotted to have a better view of how well the model performs.



```
import optuna
from optuna import samplers, pruners
import optuna_dashboard
import time
import joblib

import threading
# from google.colab import output
from optuna_dashboard import run_server
```

Figure 5.2.2(g): Libraries required for hyperparameter optimization

The library used in hyperparameters optimization is Optuna. The samplers class is to decide which sampler to be used in the optimization process, in this case is sampler.RandomSampler() for Random Search and sampler.GPsampler() for Bayesian Optimization (Gaussian Process). The threading and optuna dashboard are used to visualize the optimization process. A dashboard will be created and stored in an optuna.storage object. The object can be saved using joblib. The threading library is to create a thread to host the dashboard website, so that the optimization results can be shown as a website as shown in Figure 5.2.2(h).

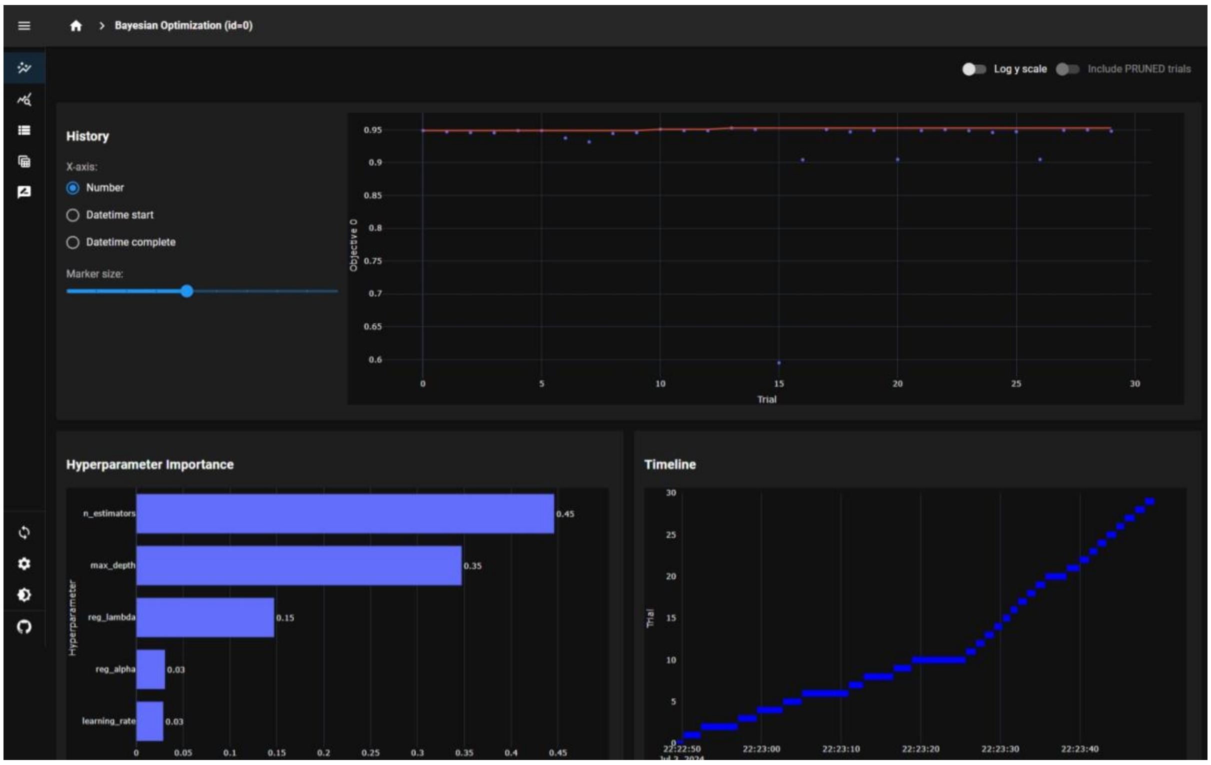


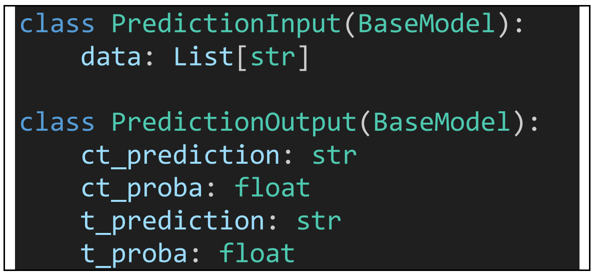
Figure 5.2.2(h): Optuna Dashboard for optimization process visualization

5.2.3 Web Service API

```
from fastapi import FastAPI, HTTPException
from pydantic import BaseModel
import torch
from torch import nn
import numpy as np
from typing import List
from sklearn.preprocessing import MinMaxScaler, StandardScaler, LabelEncoder
import joblib
import pandas as pd
```

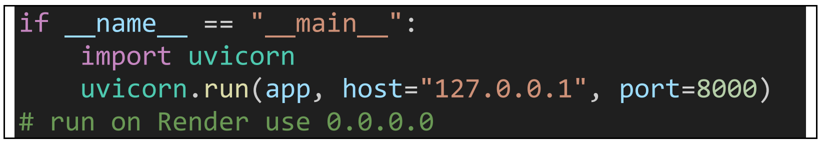
Figure 5.2.3(a): Libraries required to develop model API

In web service API development, instead of using Jupyter Notebook, a standard Python script (.py) is used. FastAPI is the main framework used to initialize the script as an API application. The BaseModel from Pydantic and the List is used to validate the user input and standardize the output of the API returned to the user. The class is shown in Figure 5.2.3(b). The torch library is used to load the neural network model in the API. If the API intend to serve XGBoost model, then the library here should be XGBoost library. The joblib and the preprocessing library is used here as well in order to load the previously saved Scaler and Encoder object and perform same preprocessing to the new input data from user.



```python
class PredictionInput(BaseModel):
    data: List[str]

class PredictionOutput(BaseModel):
    ct_prediction: str
    ct_proba: float
    t_prediction: str
    t_proba: float
```

Figure 5.2.3(b): Input and output format validation

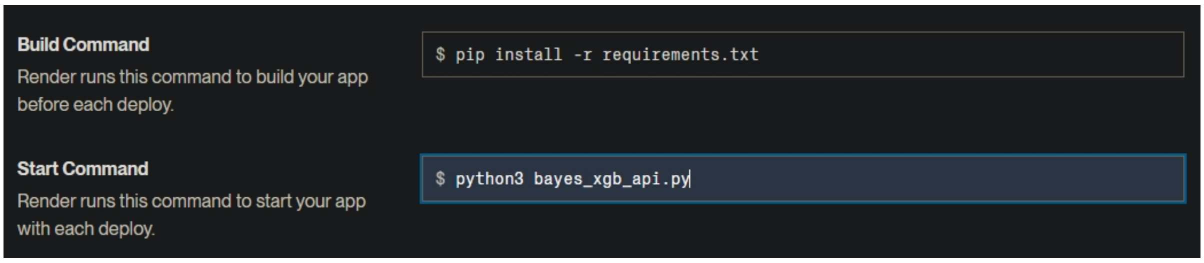


```python
if __name__ == "__main__":
    import uvicorn
    uvicorn.run(app, host="127.0.0.1", port=8000)
# run on Render use 0.0.0.0
```

Figure 5.2.3(c): Start server on specified IP address and port using Uvicorn

In Figure 5.2.3(c) shows the library used to run the API on a Uvicorn server. By default, the application should be hosted at address 127.0.0.1 on port 8000 if the hosting is on local machine. The port number can be different if multiple API applications required to run on the same time on local machine. However, if the application is ready to deploy on Render, the address should be changed to 0.0.0.0 to ensure success deployment.

Next, to deploy the API to Render, first login to Render using the GitHub account where the API application code is pushed to the repository. Then, the website will prompt to connect GitHub repository from the existing in the GitHub account. Select the one where the API application code is. After that, back to the Python script and open a command prompt to export a requirements.txt that includes the required environment and libraries. The requirements.txt is then uploaded to the same repository as the application code. Then, in the Build Command section, type in the command as shown in Figure 5.2.3(d) to let Render's Python runtime install the required libraries. In the start command section, type in command as shown in the figure below. Finally, select "Deploy Web Service" button, the service will deploy automatically and startup right after deployment is successful. The link to access the API will be shown in the web service dashboard.


Build Command
Render runs this command to build your app before each deploy.

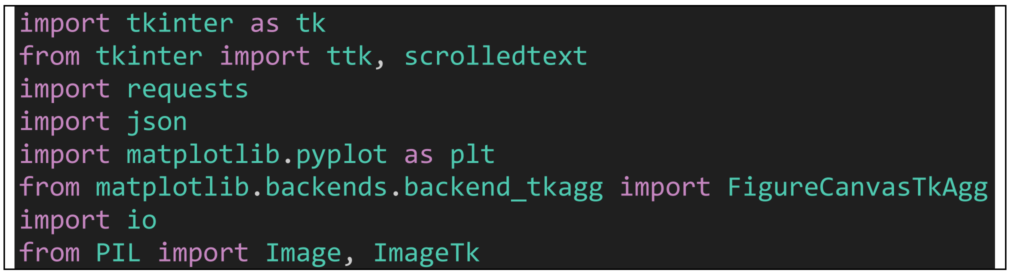
```
$ pip install -r requirements.txt
```

Start Command
Render runs this command to start your app with each deploy.

```
$ python3 bayes_xgb_api.py
```

Figure 5.2.3(d): Command to start web services on Render

5.2.4 GUI



```
import tkinter as tk
from tkinter import ttk, scrolledtext
import requests
import json
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import io
from PIL import Image, ImageTk
```

Figure 5.2.4(a): Libraries required for GUI development

Figure 5.2.4(a) shows the libraries used to develop a simple GUI. The main library used is TKinter to create the GUI as shown in Figure 4.5(b). The request library is used to send HTTP POST request together with user input data in JSON format to the API created in previous subchapter. Once the response from API is received, using matplotlib library to plot a bar chart and embed the bar chart to the GUI using FigureCanvasTkAgg. The Python Image Library (PIL) is to load image as the GUI application logo. The logo will show as the application logo on the top left corner of the window as well as on the taskbar.

5.3 Implementation Issues and Challenges

The issues encountered during the project is Python Environment management. This happened Between FYP1 and FYP2 where Numpy released new version (Numpy 2.0) and it does not compatible with the previous work done in FYP1. Therefore, the Numpy version needs to be manually downgraded back to Numpy 1.24.3 instead of using the new Numpy 2.0 version.

Besides Numpy, Python also has been updated to version 3.12, and some libraries such as scikit-plot is not compatible with the new versions, therefore it also required manual downgrade to Python 3.11.9 to ensure the compatibility of the entire project.

Another issue is regarding to the API deployment. During deployment testing phase, the API is tested on local machine which everything works fine. However, when deploy on Render, some of the code is not functioning as it was on local machine. Specifically, the issue is the datatype used in neural network needs to be specified as float datatype instead of leaving it undefined, which is usually done when coding in Python. It is believed that the instance is running on different operating systems such as Linux, causing the datatype needs to be specified.

Chapter 6

System Evaluation and Discussion

6.1 System Evaluation and Performance Metrics

The objective of model testing is to observe the improvement from baseline model to optimized model, as well as between models optimized using different optimization method to compare the efficiency. The metrics include accuracy, recall, precision, F1-score, and ROC-AUC score.

Accuracy indicates the number of correct predictions over the total number of predictions made. Recall is the ratio of true positive samples over all the actual positive samples. While precision is the ratio of true positive samples over the samples that are predicted as positive. To balance out both precision and recall, the ultimate metric to evaluate model is the F1-score, it takes both precision and recall into account [21]. Receiver-Operating Characteristic curve (ROC) is a curve that represent the model performance across all thresholds using True Positive Rate versus False Positive Rate [22]. The curve is drawn by calculating the True Positive Rate and False Positive Rate at every threshold possible [22]. From ROC, another important metric can be read which is the Area Under the Curve (AUC). AUC represent the probability of a model to differentiate positive and negative samples [22].

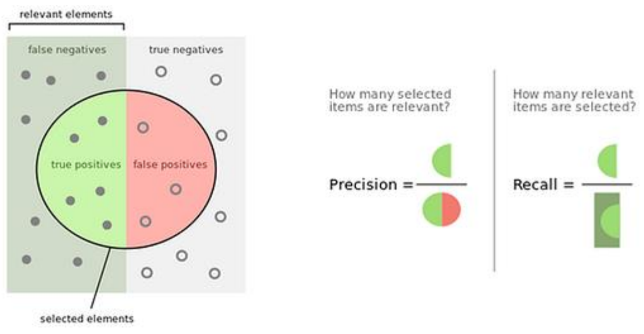


Figure 6.1: Illustration of Precision and Recall [21]

While for the API and GUI testing, the objective is relatively simple, the API must be able to receive HTTP POST response, validate the user input, perform prediction, and return the prediction result as JSON. While for GUI, it must allow user to input data, send data to the correct web services based on the model selected, receive response from API, visualize and display the result correctly as a bar chart.

6.2 Testing Result

To test the system, the testing is separated into 2 parts, the model testing and the API/GUI testing.

6.2.1 Model Evaluation

1) Neural Network

The baseline model, random optimized model, and Bayesian optimized model result of neural networks are shown in Table 6.2.1(a) and Table 6.2.1(b) below.

Model	**Baseline model (Handpick hyperparameters)**	**Random optimized model**	**Bayesian optimized model**

Hyper params.	`lr: 0.005,`
	`num_neuron: (200, 300, 100, 70, 50),`
	`momentum: 0.9,`
	`batch_size: 128,`
	`num_of_epoch: 24`
		`lr: 0.0657571637680928,`
		`num_layer_1: 99,`
		`num_layer_2: 192,`
		`num_layer_3: 149,`
		`num_layer_4: 75,`
		`num_layer_5: 52,`
		`momentum: 0.7279355045846245,`
		`num_of_epoch: 54,`
		`batch_size: 44`
			`lr: 0.062392167602705856,`
			`num_layer_1: 89,`
			`num_layer_2: 117,`
			`num_layer_3: 78,`
			`num_layer_4: 143,`
			`num_layer_5: 200,`
			`momentum: 0.553127569798693,`
			`num_of_epoch: 42,`
			`batch_size: 75`

Avg. Training metrics	`Avg training accuracy: 0.8876`
	`Avg training recall: 0.8092`
	`Avg training precision:0.9141`
	`Avg training f1: 0.8497`
	`Avg training ROC score:0.9730`
		`Avg training accuracy: 0.9084`
		`Avg training recall: 0.8854`
		`Avg training precision:0.8979`
		`Avg training f1: 0.8879`
		`Avg training ROC score:0.9792`
			`Avg training accuracy: 0.9126`
			`Avg training recall: 0.9007`
			`Avg training precision:0.8873`
			`Avg training f1: 0.8928`
			`Avg training ROC score:0.9774`

Cross Validation (5-fold)	`Avg validation accuracy: 0.8827`
	`Avg validation recall: 0.7992`
	`Avg validation precision:0.9056`
	`Avg validation f1: 0.8399`
	`Avg validation ROC score:0.9695`
		`Avg validation accuracy: 0.9003`
		`Avg validation recall: 0.8689`
		`Avg validation precision:0.8871`
		`Avg validation f1: 0.8742`
		`Avg validation ROC score: 0.9741`
			`Avg validation accuracy: 0.9041`
			`Avg validation recall: 0.8878`
			`Avg validation precision:0.8758`
			`Avg validation f1: 0.8800`
			`Avg validation ROC score: 0.9728`

Test Set inference	N/A	`recall score: 0.9666`
		`Precision score: 0.7526`
		`F1 score: 0.8463`
		`ROC_AUC score: 0.9712`
		`Accuracy Score: 0.8577`
			`recall score: 0.9399`
			`Precision score: 0.8421`
			`F1 score: 0.8883`
			`ROC_AUC score: 0.9719`
			`Accuracy Score: 0.9042`

Opt. time	N/A	`2516.2435 seconds`	`1829.28 seconds`

Table 6.2.1(a): Model Evaluation for neural networks

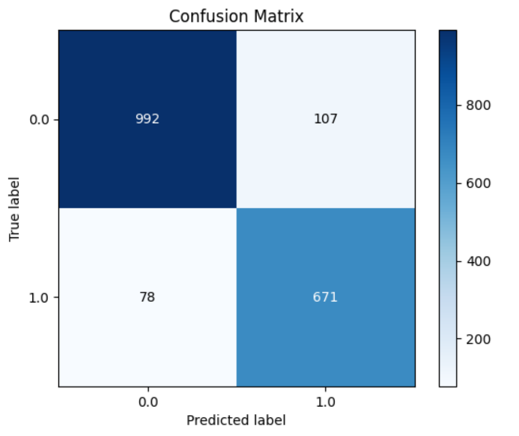
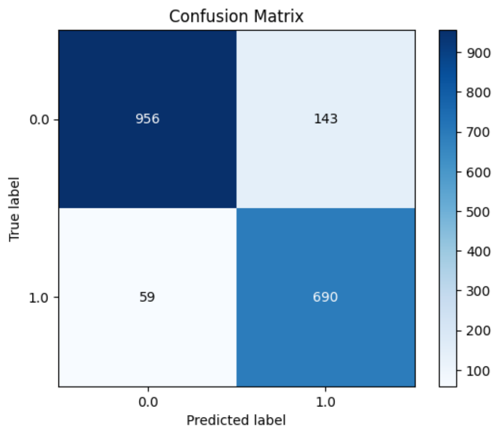
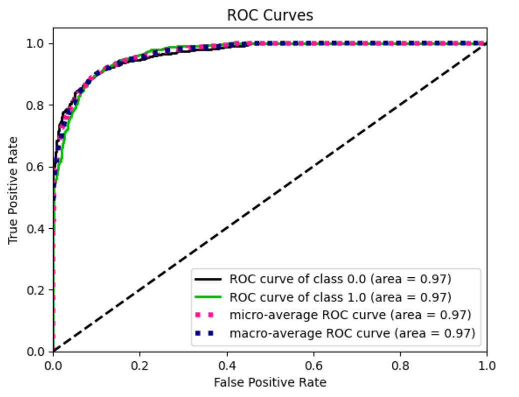
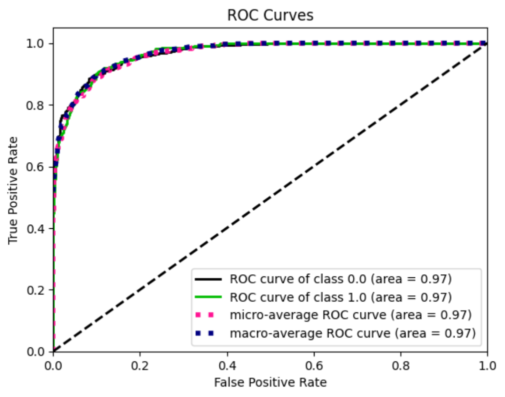
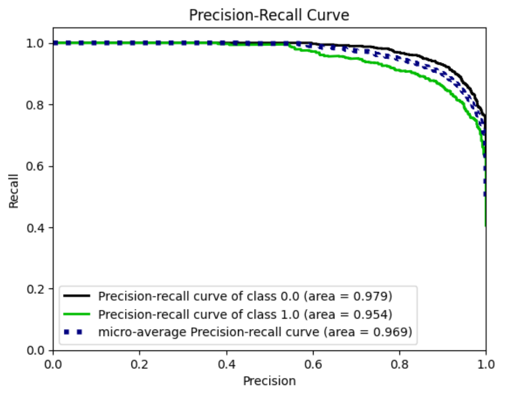
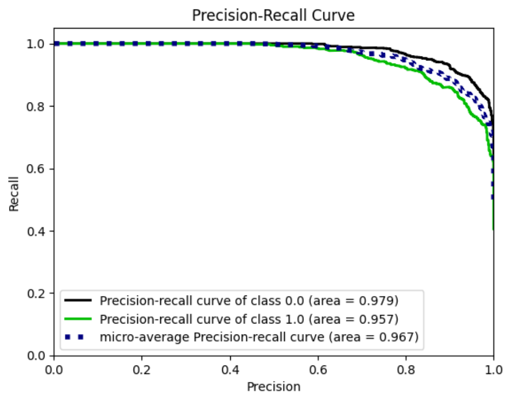
Model	**Random Optimized model**	**Bayesian Optimized model**
Conf. Matrix		
ROC		
PRC		

Table 6.2.1(b): Table of metrics plot using test set for neural networks

By comparing the results of baseline model and both optimized models, the optimized models have improvement around 2% on training and validation accuracy. There are also improvements on other metrics such as recall, F1-score, and ROC score.

Comparing both optimized models, the Bayesian optimized model has better performance in terms of accuracy. Other metrics such as recall, precision, F1-score and ROC-score are almost similar. When comparing the time taken to optimize the model, random optimization takes 2516 seconds to complete, which is taking lot more time compared to Bayesian optimization (1829 seconds). Based on the training and validation accuracy of both models, both models are suffering from a very minor overfitting issues that can be ignored.

In baseline model, although it already has acceptable performance based on its accuracy (0.88) and ROC score (0.97). However, it has lowest recall (0.7992) among all the models. On the other hand, random optimized model has significant improvement in accuracy, recall, precision and ROC score (highest among the models). Basically, it outperforms baseline model in multiple metrics. Bayesian optimized model have a more significant improvement compared to random optimized model and baseline model. It has better accuracy, recall, and F1-score compared to random optimized model, and better ROC score than baseline model.

Based on the metrics plot from Table 6.2.1(b), the confusion matrix and ROC of both neural network models are having almost similar result. From the confusion matrix, it can be observed that Random search optimized model is slightly more capable to classify class “0” samples, while Bayesian optimized model are more balanced in classifying both classes. From the Precision-Recall Curve (PRC), the curve of class “1” is not as smooth as it does in class “0” for random search optimized model. While Bayesian optimized model are having quite balance performance for both class “0” and class “1” based on the curve.

Both optimized models are performing well and able to output promising predictions on unforeseen data (test set), but Bayesian optimized model are performing better based on the test set accuracy. In terms of efficiency, Bayesian optimization perform better since it uses lesser optimization time to obtain a comparable results as random search. In short, Bayesian optimization is the preferred method to optimize neural network based on the comparison above.

2) XGBoost Classifier

The baseline model, random optimized model, and Bayesian optimized model result of XGBoost classifier are shown in Table 6.2.1(c) and Table 6.2.1(d) below.

Model	**Baseline model (default hyperparameters)**	**Random optimized model**	**Bayesian optimized model**

Hyper params.	`lr: 0.3(default)`
	`n_estimator: 0`
	`max_depth: 6`
	`reg_alpha: 0`
		`lr: 0.26111698321090004,`
		`n_estimators: 16,`
		`max_depth: 3,`
			`lr: 0.31502659178445475,`
			`n_estimators: 20,`
			`max_depth: 2,`
			`reg_lambda: 0.001,`

	reg_lambda: 0	reg_lambda: 0.7605182325138011, reg_alpha:0.1138252888021954	reg_alpha: 0.10630287998071727

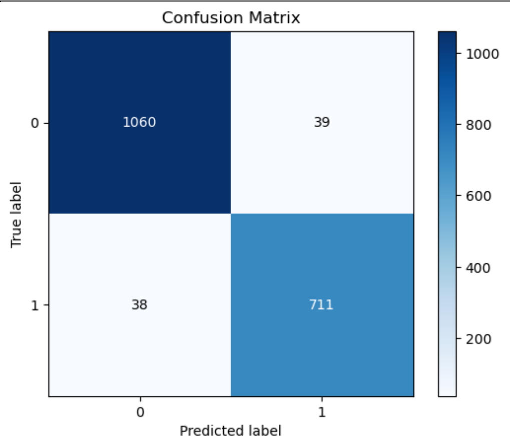
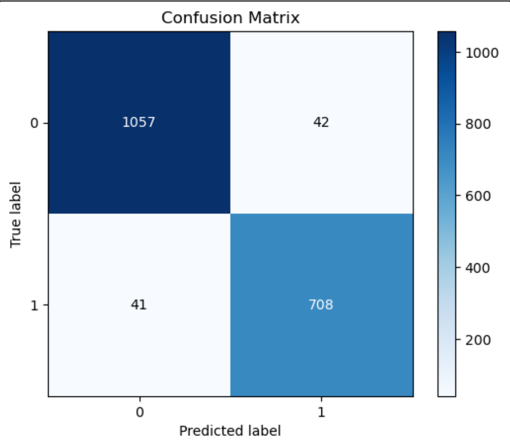
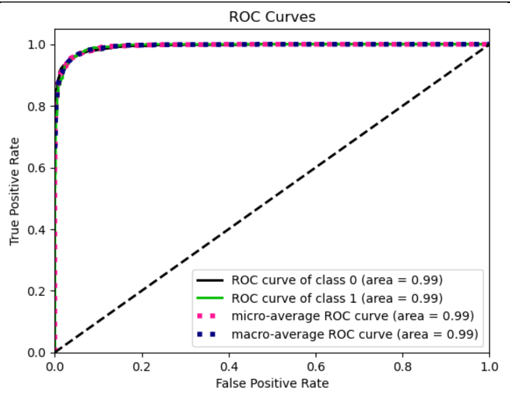
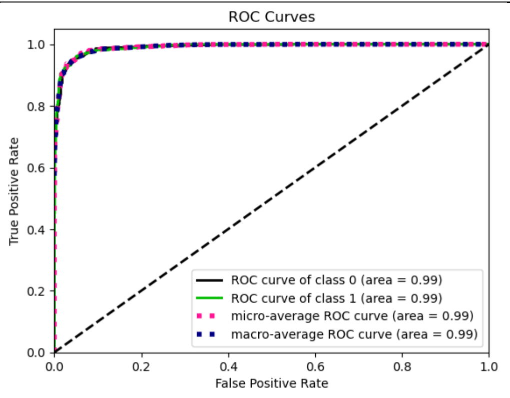
Avg. Training metrics	`Avg training accuracy: 0.9939`
	`Avg training recall:    0.9929`
	`Avg training precision: 0.9922`
	`Avg training f1:        0.9925`
	`Avg training ROC score: 0.9998`
		`Avg training accuracy: 0.9614`
		`Avg training recall:    0.9516`
		`Avg training precision: 0.9531`
		`Avg training f1:        0.9523`
		`Avg training ROC score: 0.9943`
			`Avg training accuracy: 0.9585`
			`Avg training recall:    0.9471`
			`Avg training precision:0.9502`
			`Avg training f1:        0.9486`
			`Avg training ROC score:0.9933`

Cross validation (5-fold)	`Avg validation accuracy: 0.9490`
	`Avg validation recall: 0.9378`
	`Avg validation precision: 0.9365`
	`Avg validation f1:      0.9371`
	`Avg validation ROC score: 0.9908`
		`Avg validation accuracy: 0.9506`
		`Avg validation recall: 0.9379`
		`Avg validation precision: 0.9400`
		`Avg validation f1:      0.9390`
		`Avg validation ROC score: 0.9909`
			`Avg validation accuracy: 0.9528`
			`Avg validation recall: 0.9409`
			`Avg validation precision: 0.9424`
			`Avg validation f1:      0.9416`
			`Avg validation ROC score: 0.9910`

Test Set inference	N/A	`recall score:    0.9493`
		`Precision score:0.9480`
		`F1 score:        0.9486`
		`ROC_AUC score:  0.9934`
		`Accuracy Score: 0.9583`
			`recall score:    0.9453`
			`Precision score:0.9440`
			`F1 score:        0.9446`
			`ROC_AUC score:  0.9921`
			`Accuracy Score: 0.9551`

Opt. time	N/A	`55.0504 seconds`	`59.9904 seconds`

Table 6.2.1(c): Model Evaluation for XGBoost Classifier

Model	**Random Optimized model**	**Bayesian Optimized model**
Conf. Matrix		
ROC		

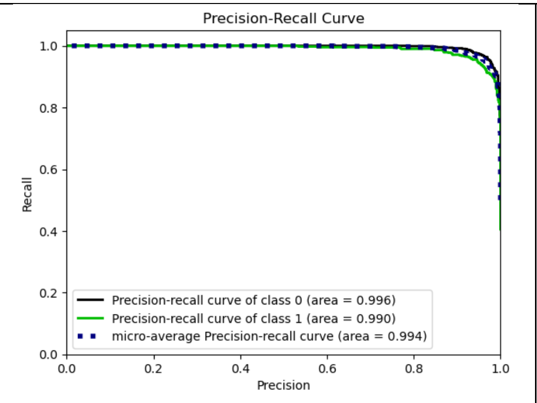
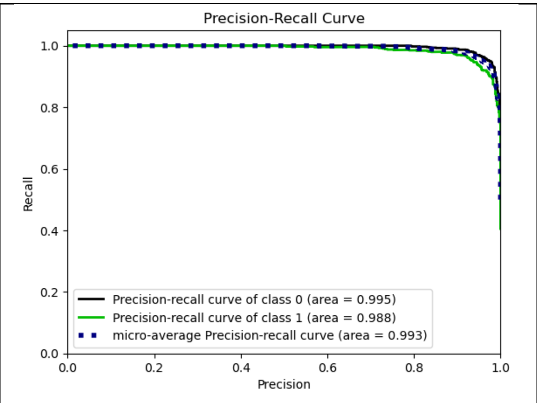
PRC		

Table 6.2.1(d): Table of metrics plot using test set for XGBoost

From the comparison table above, the baseline model of XGBoost Classifier is suffering severe overfitting issues, most of the metrics in baseline model hits 0.99 but all of the metrics drop significantly during validation.

After optimized, both optimized models having relatively better validation performance compared to baseline model. Also, the overfitting effect are reduced in both optimized models, especially for Bayesian optimized model, the gap between training and validation metrics is the smallest. Random search optimized model overfitting issues are slightly worse than Bayesian optimized model, but it is still way better than the baseline model. Bayesian optimized model outperformed random search model in all of the validation metrics. This indicates that Bayesian optimized model are better in every aspect when encountering unforeseen data.

From Table 6.2.1(d), based on the confusion matrix of both models, the capability to classify both class “0” and class “1” is similar. Even the pattern of the ROC and PRC is having almost similar pattern. The area under PRC for Bayesian optimized model is having a micro-disadvantage compared to the random optimized model, however the difference is too small, and the performance based on PRC is indistinguishable. Both models are having decent performance.

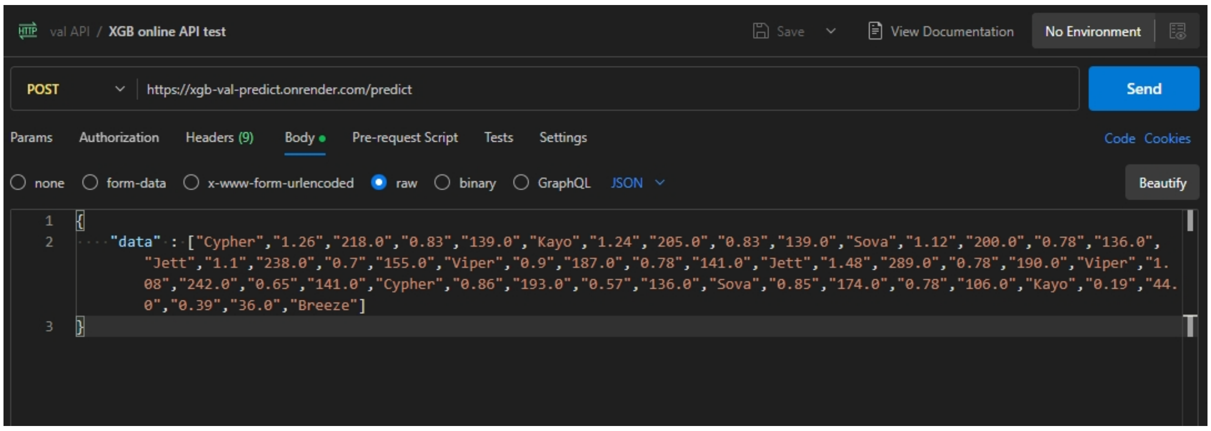
Comparing both optimized model, Bayesian optimized model are slightly better than random optimized model in every metrics. In test set inference, both models are having similar performance in terms of test accuracy. In terms of optimization time, random optimization uses 5 seconds lesser than Bayesian optimization, which is only a small difference.

When optimizing XGBoost, although both random optimization and Bayesian optimization are having good overall performance. However, in average validation, Bayesian optimized model's accuracy is better and it only cost slightly more optimization time. In other words, Bayesian optimized model has averagely better performance when dealing with unforeseen data. Hence, Bayesian optimization is a more preferred method when tuning XGBoost model.

6.2.2 API and GUI Testing

1) API testing

To test the API deployed on Render, a HTTP post request needs to be sent to the API. The testing tool for testing an API is using Postman, an API development platform. Using Postman to customize the request as shown in Figure 6.2.2(a). Select "POST" method and enter the API link. In the request body, select "raw" and insert dummy JSON data that follows the format allowed by the API.

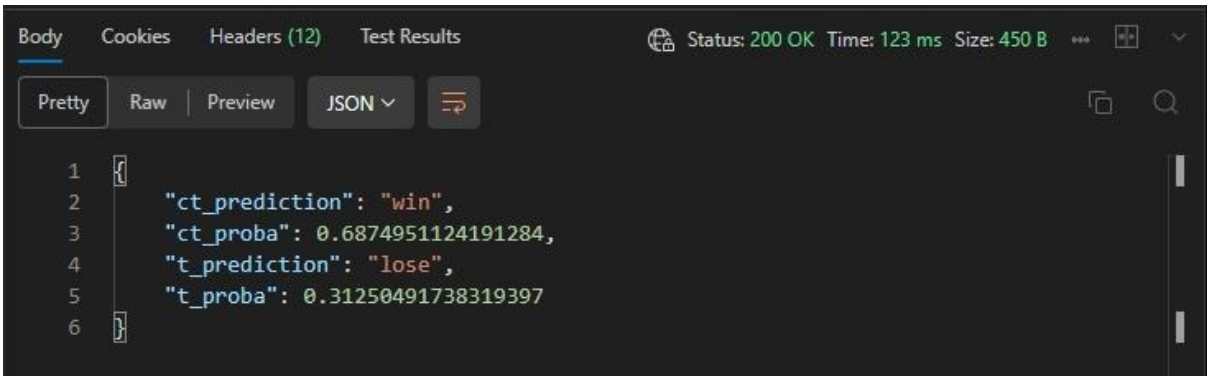


POST https://xgb-val-predict.onrender.com/predict

raw JSON

```
{
    "data" : ["Cypher","1.26","218.0","0.83","139.0","Kayo","1.24","205.0","0.83","139.0","Sova","1.12","200.0","0.78","136.0","Jett","1.1","238.0","0.7","155.0","Viper","0.9","187.0","0.78","141.0","Jett","1.48","289.0","0.78","190.0","Viper","1.08","242.0","0.65","141.0","Cypher","0.86","193.0","0.57","136.0","Sova","0.85","174.0","0.78","106.0","Kayo","0.19","44.0","0.39","36.0","Breeze"]
}
```

Figure 6.2.2(a): API testing demonstration



Status: 200 OK Time: 123 ms Size: 450 B

```
{
    "ct_prediction": "win",
    "ct_proba": 0.6874951124191284,
    "t_prediction": "lose",
    "t_proba": 0.31250491738319397
}
```

Figure 6.2.2(b): API response with win rate

If the API is online, it will respond the win rate for both teams as shown in Figure 6.2.2(b). However, as mentioned in Chapter 5.1, Render does not ensure the web service instance always online and ready to serve. Therefore, in the testing phase of API, it will wait for the response from API for a very long time (around 1 – 5 minutes). Once the service is online, the response time will be as low as 120ms shown in Figure 6.2.2(b).

2) GUI testing

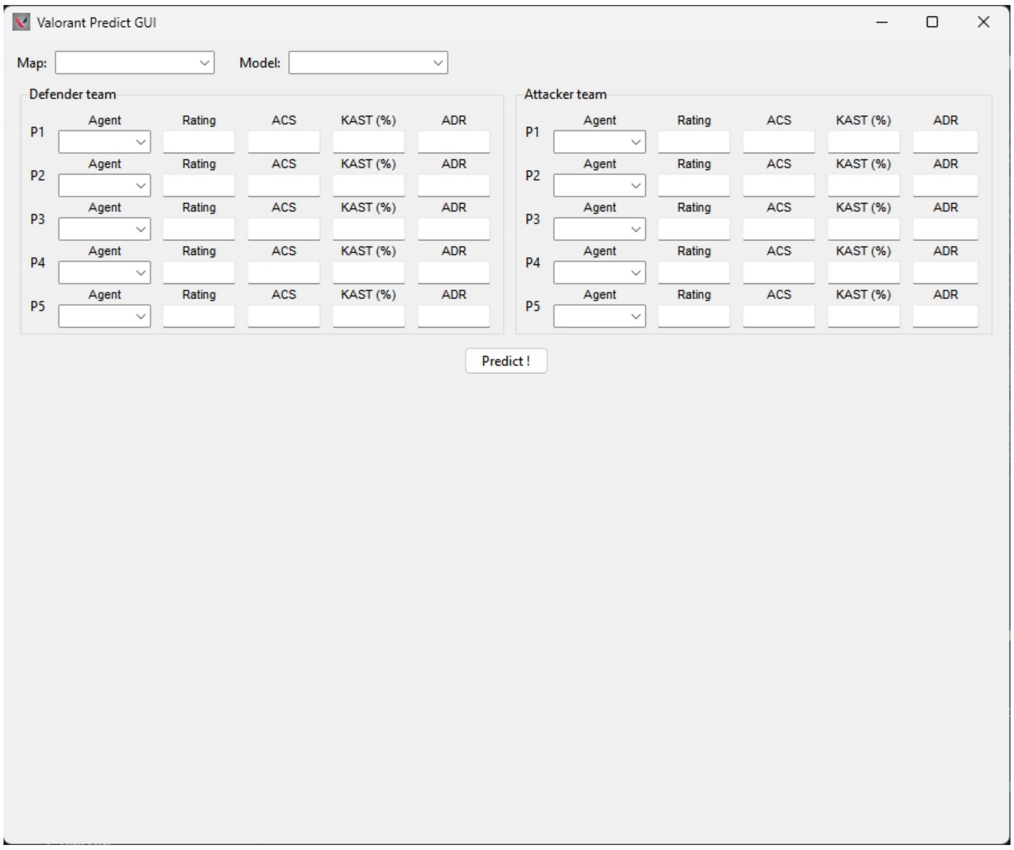


Figure 6.2.2(c): GUI of the project

Figure 6.2.2(c) shows the GUI developed based on the wireframe showed in Figure 4.5(b). Each input fields are labeled for better understanding and enhanced user experience. The fields with drop down menu are categorical data which only allows specific inputs. Note that the GUI allows to connect to API hosted on local machine as shown in Figure 6.2.2(d) to avoid API downtime issue caused by Render instance. Besides, to avoid the API to stuck on waiting for response from offline Render instance, the timeout is set to 10 seconds as shown in Figure 6.2.2(e) to allow users to change to local hosted API.

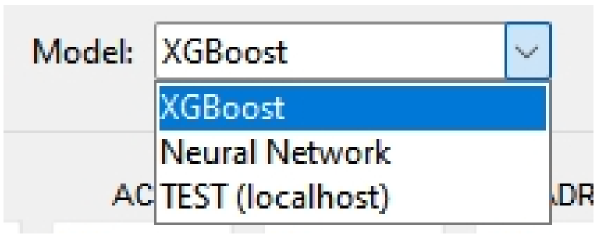
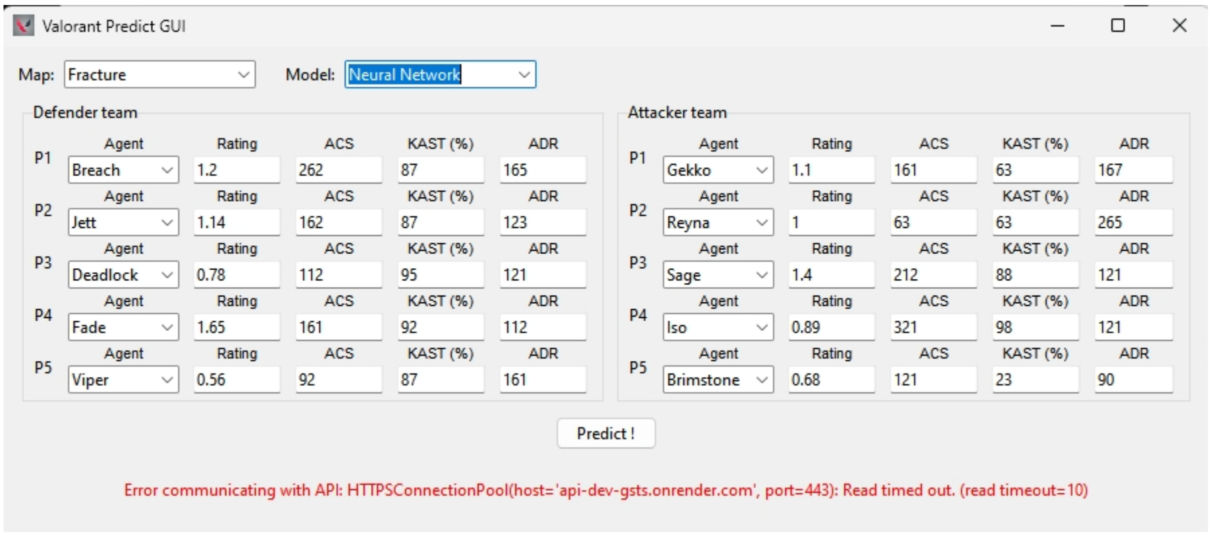


Figure 6.2.2(d): Model selection including localhost API


Valorant Predict GUI

Map: Fracture Model: Neural Network

Defender team

	Agent	Rating	ACS	KAST (%)	ADR
P1	Breach	1.2	262	87	165
P2	Jett	1.14	162	87	123
P3	Deadlock	0.78	112	95	121
P4	Fade	1.65	161	92	112
P5	Viper	0.56	92	87	161

Attacker team

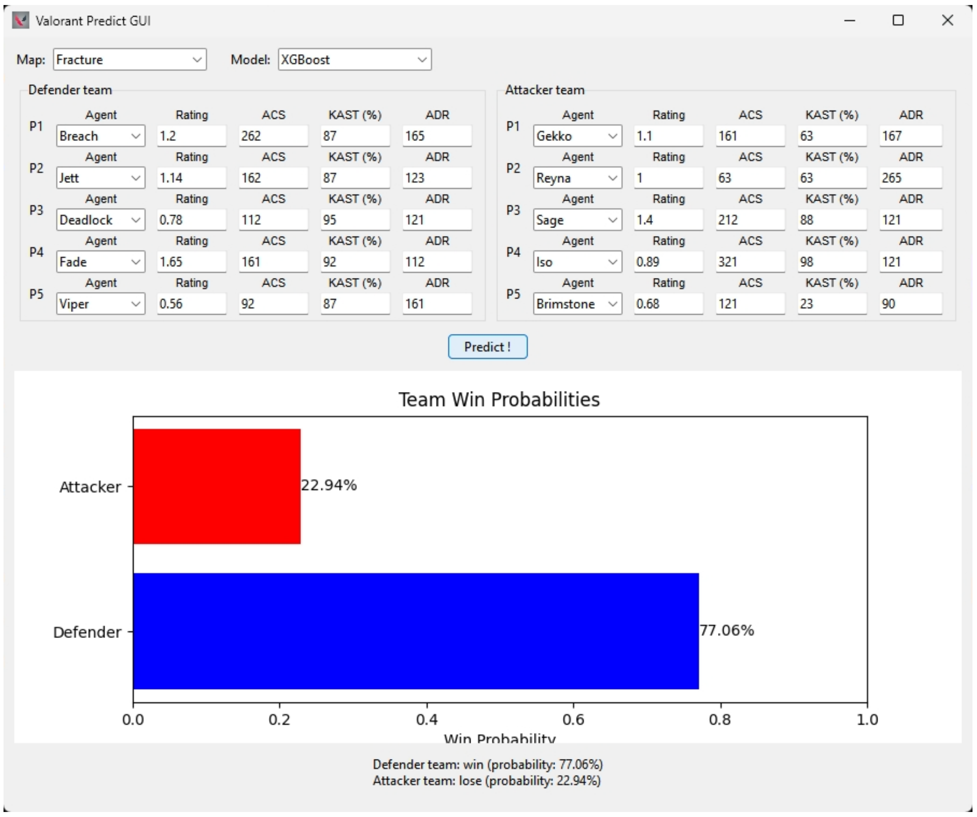
	Agent	Rating	ACS	KAST (%)	ADR
P1	Gekko	1.1	161	63	167
P2	Reyna	1	63	63	265
P3	Sage	1.4	212	88	121
P4	Iso	0.89	321	98	121
P5	Brimstone	0.68	121	23	90

Predict !

Error communicating with API: HTTPSConnectionPool(host='api-dev-gsts.onrender.com', port=443): Read timed out. (read timeout=10)

Figure 6.2.2(e): Timeout if no respond from API

Figure 6.2.2(f) shows the result display in bar chart, the bar chart shows the win rate of each team. Blue color indicates the winning team. The simplified result in text format also shown below the bar chart.



Map: Fracture Model: XGBoost

Defender team

	Agent	Rating	ACS	KAST (%)	ADR
P1	Breach	1.2	262	87	165
P2	Jett	1.14	162	87	123
P3	Deadlock	0.78	112	95	121
P4	Fade	1.65	161	92	112
P5	Viper	0.56	92	87	161

Attacker team

	Agent	Rating	ACS	KAST (%)	ADR
P1	Gekko	1.1	161	63	167
P2	Reyna	1	63	63	265
P3	Sage	1.4	212	88	121
P4	Iso	0.89	321	98	121
P5	Brimstone	0.68	121	23	90

Defender team: win (probability: 77.06%)
Attacker team: lose (probability: 22.94%)

Figure 6.2.2(f): Prediction result visualization and display on GUI

6.3 Project Challenge

The first challenge faced in this project is the system design, specifically when designing the optimization process. Due to lack of understanding of the properties of both optimization method, the number of trials for each optimization method was set to 150 trials in the beginning, without considering the time efficiency of optimization. These issues lead to the optimization process is too lengthy and show not much difference between both random search and Bayesian optimization. This is because in 150 trials, both random search and Bayesian optimization are able to explore the search space equally. This will reduce the advantages of Bayesian optimization, where it balances between exploitation and exploration. Also due to high number of trials, the time taken by Bayesian optimization is a lot longer than random search due to its complexity of algorithms is higher than random search. After some research and studies, the number of trials is reduced to 30 trial per optimization. This is to observe which optimization method can perform better under a stricter constraint of 30 trials. Result prove that Bayesian optimization is still a better option for optimizing complex model due to its efficiency and hyperparameters quality.

The second challenge is budget limitation. The entire project is developed without spending any money, including the model training process and web service hosting, which usually require service subscription. Also, the project wishes to allow public to use the API and GUI for free, therefore the project have to sacrifice user experience by choosing free services such as Render web hosting, that does not guarantee 100% availability time.

6.4 Objective Evaluation

As a recap, the objective of this project is to:

1) Investigate the impact of optimization method on model performance (Baseline model vs. optimized model).

2) Compare different optimization methods in terms of time efficiency and performance improvement (Random search vs. Bayesian Optimization).

For first objective, the impact of both optimization methods toward baseline model is observable from its performance metrics before and after optimized. The model does not only improve in accuracy, meanwhile it also reduced the overfitting issues occurred on the model. By reducing the overfitting issues in the model, the model able to produce a more promising result. Besides, based on the improvement of metrics such as F1-score and ROC-score, it also shows that the overall ability of the model when dealing with unforeseen data is getting significant improvements from optimizing their hyperparameters.

For the second objective, the efficiency of the optimization methods is depending on the complexity of the model hyperparameters. Although neural network seems to have same number of hyperparameters as XGBoost, however the number of neurons for each layer in neuron network is treated as multiple hyperparameters as there are multiple layers, and each layer is considered as a hyperparameter. Therefore, in terms of dimension of hyperparameters in this project, neural network is definitely having higher dimension of hyperparameters compared to XGBoost. As the result, when optimizing neural network, Bayesian Optimization is taking lesser time to obtain better result than Random search optimized model. In contrast, when optimizing XGBoost, the Random search takes less time to achieve similar result as Bayesian optimized model. However, no matter which model it is, Bayesian optimized model are getting better result in average. Hence, Bayesian optimization is still the preferred option when performing optimization.

Chapter 7

Conclusion and Recommendation

7.1 Conclusion

In conclusion, this project briefly introduced FPS game and the potential growth of FPS game in e-sports. The FPS game that was investigated in this project is Valorant which is a 5 versus 5, tactical gunfight game which does not have much research done yet based on this game. This project also reviewed previously done research, and further discussed about different machine learning model used by other researchers on E-sport prediction field, especially in FPS game, and the well-performed model include neural network and XGBoost classifier. In previous research on other FPS game, most of the author mentioned that the final model is not well-optimized, and they recommended to improve the model performance by optimizing the hyperparameters.

There are multiple ways to optimize models and the most common methods are not efficient when dealing with complex models. Some of the research suggested to use Bayesian optimization method instead of common methods such as grid search and random search due to its nature of exploration and exploitation. The objective of this project is to develop 2 prediction model using neural network and XGBoost. Then, the model will be further optimized using 2 different methods, which is Random Search and Bayesian optimization. The performance of each model is compared. At the end of the project also concluded which method is more suitable for optimizing complex models.

The project starts with obtaining the data from internet, cleaning data, and preprocess the data. After that the models are trained with the data using handpicked / default hyperparameters as baseline model. Then, the models undergo hyperparameters optimization using Bayesian optimization and random search. For each model, the baseline model, Bayesian optimized model and random search optimized model is evaluated and compared. The comparison shows that the Bayesian optimized model perform better in terms of performance and time efficiency. Besides the model, this project also developed 2 web API that allows developers to access the best optimized models from this project. The project even developed a GUI to allow normal users to access the model easily.

7.2 Recommendation

Due to time constraint, the project has to sacrifice some features and ideas that possibly able to improve the project quality. Therefore, this subchapter will discuss the possible future improvement to this project.

1) Deeper data analysis

The models in this project only able to provide a simple info, which is the winning team win rate. In future, the model can be improved by further analyze the data, or increase the feature of the data and further combines the features to provide useful suggestions such as team compositions, map and agent ban/pick (future update), strategy suggestion, etc. These features can be useful to professionals' team and players to make wise decisions based on their opponent they are playing. It will be a game changer if these features can be introduced to the e-sports gaming scene.

3) Improve GUI

With the new information provided in point (1) above, the GUI also have to be optimized to display enriched information. The GUI also can be improved by adding new features such as dark mode, developer options to modify some settings in the GUI and so on.

2) Introduce new optimization method

As discussed in previous chapters, different optimization methods have their own properties. For example, random search only focuses on search space exploration, while Bayesian optimization balance between exploration and exploitation. In future, maybe these methods can be combined and used together to optimize a model. With the wide exploration property from random search, combines with high efficiency and explore-exploit balanced advantages from Bayesian optimization, it might be the next state-of-the-art optimization methods used by everyone in machine learning field.

REFERENCES

[1] P. Xenopoulos, B. Coelho, and C. Silva, “Optimal Team Economic Decisions in Counter-Strike,” Sep. 2021, [Online]. Available: http://arxiv.org/abs/2109.12990

[2] W. Xu Huang, J. Wang, and Y. Xu, “Predicting Round Result in Counter-Strike: Global Offensive Using Machine Learning,” in *2022 7th International Conference on Intelligent Computing and Signal Processing, ICSP 2022*, Institute of Electrical and Electronics Engineers Inc., 2022, pp. 1685–1691. doi: 10.1109/ICSP54964.2022.9778597.

[3] RITHP, “Optimizing XGBoost: A Guide to Hyperparameter Tuning,” medium.com. Accessed: Sep. 01, 2023. [Online]. Available: https://medium.com/@rithpansanga/optimizing-xgboost-a-guide-to-hyperparameter-tuning-77b6e48e289d#:~:text=Bayesian%20optimization%20is,a%20more%20sophisticated

[4] S. Shekhar, A. Bansode, and A. Salim, “A Comparative study of Hyper-Parameter Optimization Tools,” Jan. 2022, [Online]. Available: http://arxiv.org/abs/2201.06433

[5] hiepnguyen034, “Improving neural network’s performance with Bayesian Optimization,” Medium.com. Accessed: Sep. 11, 2023. [Online]. Available: https://medium.com/@hiepnguyen034/improving-neural-networks-performance-with-bayesian-optimization-efbaa801ad26

[6] D. Passos and P. Mishra, “A tutorial on automatic hyperparameter tuning of deep spectral modelling for regression and classification tasks,” Apr. 15, 2022, *Elsevier B.V.* doi: 10.1016/j.chemolab.2022.104520.

[7] Z. B. Zabinsky, “Random Search Algorithms,” 2009.

[8] Wim de Villiers, “Bayesian Optimization with Gaussian Processes Part 1,” Medium.com. Accessed: Apr. 19, 2024. [Online]. Available: https://medium.com/@de.villiers.wim/bayesian-optimization-with-gaussian-processes-part-1-83bcc291802b

[9] paretos, “Bayesian Optimization (Bayes Opt): Easy explanation of popular hyperparameter tuning method,” 2021. Accessed: Aug. 13, 2024. [Online]. Available: https://youtu.be/M-NTkxfd7-8?si=3AOy0gbLEPPR7bvL

[10] Amzhao, “Quantum Neural Networks,” MIT 6.s089 - Intro to Quantum Computing. Accessed: Aug. 22, 2024. [Online]. Available: https://medium.com/mit-6-s089-intro-to-quantum-computing/quantum-neural-networks-7b5bc469d984

[11] Tan Hung Khoon, “Lecture 1: Introduction to Neural Network with Logistic Regression,” 2024.

[12] Tan Hung Khoon, “L09: Training the Neural Network,” 2024.

[13] Shruti Misra, “Interpretable AI: Decision Trees,” Medium.com. Accessed: Aug. 24, 2024. [Online]. Available: https://medium.com/@shrutimisra/interpretable-ai-decision-trees-f9698e94ef9b

[14] pawangfg, “XGBoost,” GeeksforGeeks. Accessed: Aug. 24, 2024. [Online]. Available: https://www.geeksforgeeks.org/xgboost/

[15] Super Data Science: ML & AI Podcast with Jon Krohn, “What is XGBoost,” 2023. Accessed: Aug. 24, 2024. [Online]. Available: https://www.youtube.com/watch?v=BC99nCeZ7t8

[16] Matt Dancho, “XGBoost: Tuning the Hyperparameters (My Secret 2 Step Process in R),” Business Science. Accessed: Aug. 24, 2024. [Online]. Available: https://www.business-science.io/code-tools/2024/01/12/xgboost-hyperparameter-tuning.html

[17] “Vlr.gg.” Accessed: Aug. 26, 2024. [Online]. Available: https://www.vlr.gg/

[18] Vikram Singh, “Normalization vs Standardization,” Shiksha Online. Accessed: Aug. 28, 2024. [Online]. Available: https://www.shiksha.com/online-courses/articles/normalization-and-standardization/#:~:text=Improves%20Convergence%20Speed%3A%20Standardization%20can,features%20have%20the%20same%20scale.

[19] scikit-learn developers, “3.1. Cross-validation: evaluating estimator performance,” Scikit-Learn. Accessed: Aug. 28, 2024. [Online]. Available: https://scikit-learn.org/stable/modules/cross_validation.html

[20] “Render,” Render. Accessed: Sep. 04, 2024. [Online]. Available: https://render.com/

[21] Christopher Riggio, “What’s the deal with Accuracy, Precision, Recall and F1?,” Medium.com. Accessed: Sep. 08, 2024. [Online]. Available: https://towardsdatascience.com/whats-the-deal-with-accuracy-precision-recall-and-f1-f5d8b4db1021

[22] “Machine Learning: ML Concepts,” Google for Developers. Accessed: Sep. 08, 2024. [Online]. Available: https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc

APPENDIX

GitHub link of the project:

1) Data Acquisition, preprocessing scripts and datasets:

https://github.com/changyang666/Vlrgg-scrapper-project-file.git

2) Neural Network training / optimizing script, pretrain models:

https://github.com/changyang666/Neural-network.git

3) XGBoost training / optimizing script, pretrain models:

https://github.com/changyang666/XGBoost.git

4) API development scripts:

https://github.com/changyang666/API-dev.git

5) GUI development scripts and .exe file for instant access to GUI:

https://github.com/changyang666/GUI.git

Render web services API access link:

1) Neural network (Bayesian optimized):

https://api-dev-gsts.onrender.com/predict (POST)

2) XGBoost Classifier (Bayesian optimized):

https://xgb-val-predict.onrender.com/predict (POST)

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Y4S1	**Study week no.: 1&2**
Student Name & ID: Chang Yang 1905073	
Supervisor: Ms. Tseu Kwan Lee	
Project Title: ESPORT VALORANT: OPTIMIZING HYPERPARAMETERS OF WIN RATE PREDICTION MODEL USING DERIVATIVE FREE METHOD	

1. WORK DONE - study and build XGBoost model
2. WORK TO BE DONE - run optimization on XGBoost model
3. PROBLEMS ENCOUNTERED - unable to determine which hyperparameters to choose
4. SELF EVALUATION OF THE PROGRESS - problem resolved, done research on optimizing XGBoost and concluded top 5 hyperparameters to optimize

[illegible]
Supervisor's signature

[illegible]
Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Y4S1	**Study week no.: 3&4**
Student Name & ID: Chang Yang 1905073	
Supervisor: Ms. Tseu Kwan Lee	
Project Title: ESPORT VALORANT: OPTIMIZING HYPERPARAMETERS OF WIN RATE PREDICTION MODEL USING DERIVATIVE FREE METHOD	

1. WORK DONE - optimized XGBoost model with 2 different methods - evaluated models
2. WORK TO BE DONE - API development
3. PROBLEMS ENCOUNTERED - XGBoost easily overfitted and need to be tuned carefully - optimization result not tally with research done
4. SELF EVALUATION OF THE PROGRESS - problem resolved. Done some research on exploration and exploitation, understand the concept and decided to reduce number of trials. Manage to achieve result as in research

[illegible]
Supervisor's signature

[illegible]
Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Y4S1	**Study week no.: 5&6**
Student Name & ID: Chang Yang 1905073	
Supervisor: Ms. Tseu Kwan Lee	
Project Title: ESPORT VALORANT: OPTIMIZING HYPERPARAMETERS OF WIN RATE PREDICTION MODEL USING DERIVATIVE FREE METHOD	

1. WORK DONE - API developed and tested
2. WORK TO BE DONE - GUI development and testing
3. PROBLEMS ENCOUNTERED - most of the hosting service require payment subscription
4. SELF EVALUATION OF THE PROGRESS - Problem solved, manage to find free hosting services and successfully deployed API.

[illegible]

Supervisor's signature

[illegible]

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Y4S1	**Study week no.: 7&8**
Student Name & ID: Chang Yang 1905073	
Supervisor: Ms. Tseu Kwan Lee	
Project Title: ESPORT VALORANT: OPTIMIZING HYPERPARAMETERS OF WIN RATE PREDICTION MODEL USING DERIVATIVE FREE METHOD	

1. WORK DONE - GUI development and testing
2. WORK TO BE DONE - Project refinement
3. PROBLEMS ENCOUNTERED - Hard to decide which framework to choose when developing GUI
4. SELF EVALUATION OF THE PROGRESS - Problem solved, choosed tkinter as GUI development framework.

[illegible]
Supervisor's signature

[illegible]
Student's signature

FINAL YEAR PROJECT WEEKLY REPORT
(Project II)

Trimester, Year: Y4S1	**Study week no.: 9&10**
Student Name & ID: Chang Yang 1905073	
Supervisor: Ms. Tseu Kwan Lee	
Project Title: ESPORT VALORANT: OPTIMIZING HYPERPARAMETERS OF WIN RATE PREDICTION MODEL USING DERIVATIVE FREE METHOD	

1. WORK DONE - added features to GUI such as change model, allow local hosting API for offline purpose
2. WORK TO BE DONE **-** Report writing
3. PROBLEMS ENCOUNTERED **-** Render free hosting suspend instance after period of inactivity, causing GUI no responding on startup.
4. SELF EVALUATION OF THE PROGRESS **-** add in timeout to GUI, allow user to switch to local hosted API

[illegible]
Supervisor's signature

[illegible]
Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

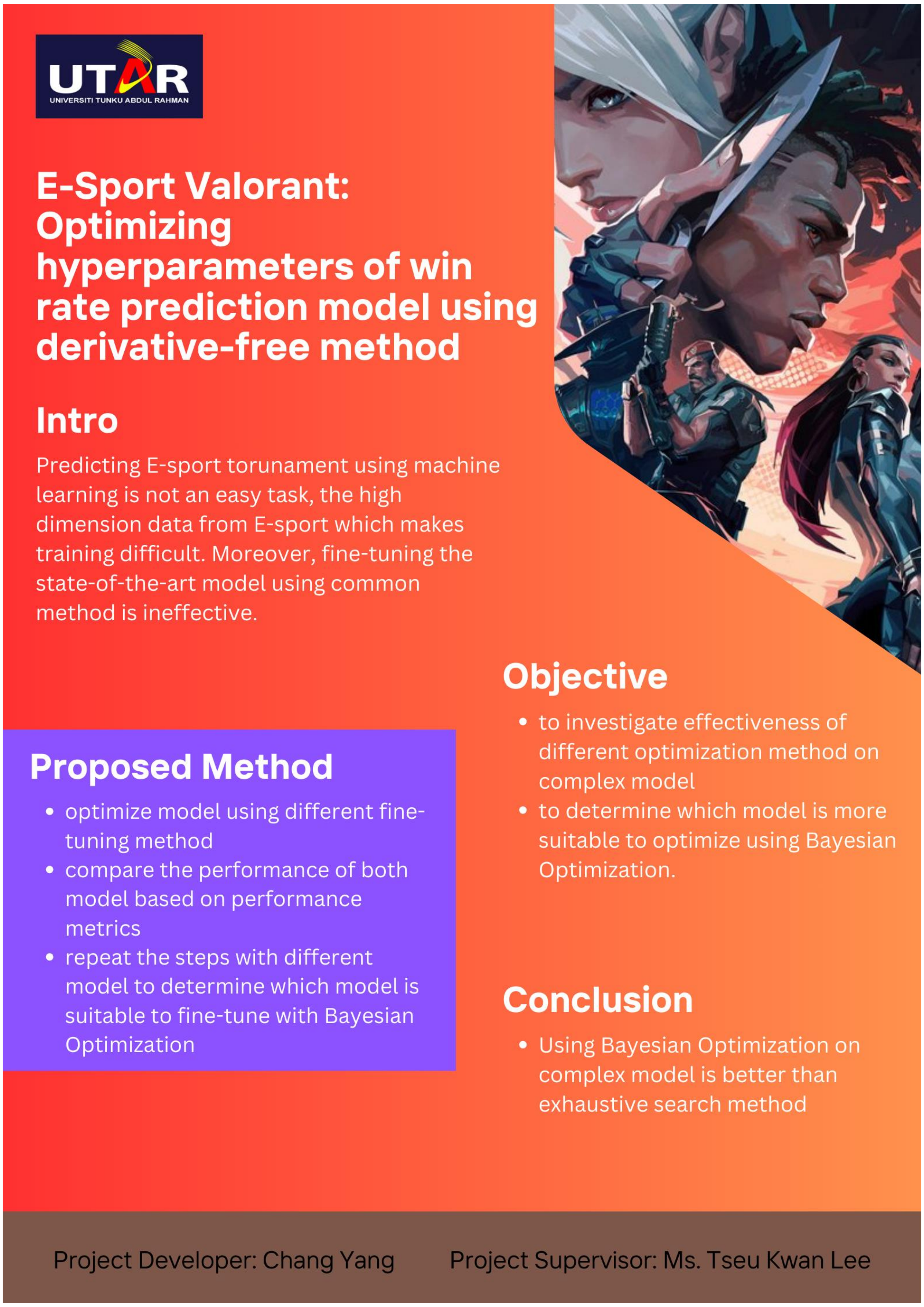
Trimester, Year: Y4S1	**Study week no.: 11&12**
Student Name & ID: Chang Yang 1905073	
Supervisor: Ms. Tseu Kwan Lee	
Project Title: ESPORT VALORANT: OPTIMIZING HYPERPARAMETERS OF WIN RATE PREDICTION MODEL USING DERIVATIVE FREE METHOD	

1. WORK DONE **-** report writing - presentation slides preparation
2. WORK TO BE DONE **-**
3. PROBLEMS ENCOUNTERED **-**
4. SELF EVALUATION OF THE PROGRESS **-** progressed as planned

[illegible]
Supervisor's signature

[illegible]
Student's signature

POSTER



UTAR UNIVERSITI TUNKU ABDUL RAHMAN

E-Sport Valorant: Optimizing hyperparameters of win rate prediction model using derivative-free method

Intro

Predicting E-sport torunament using machine learning is not an easy task, the high dimension data from E-sport which makes training difficult. Moreover, fine-tuning the state-of-the-art model using common method is ineffective.

Objective

- to investigate effectiveness of different optimization method on complex model
- to determine which model is more suitable to optimize using Bayesian Optimization.

Proposed Method

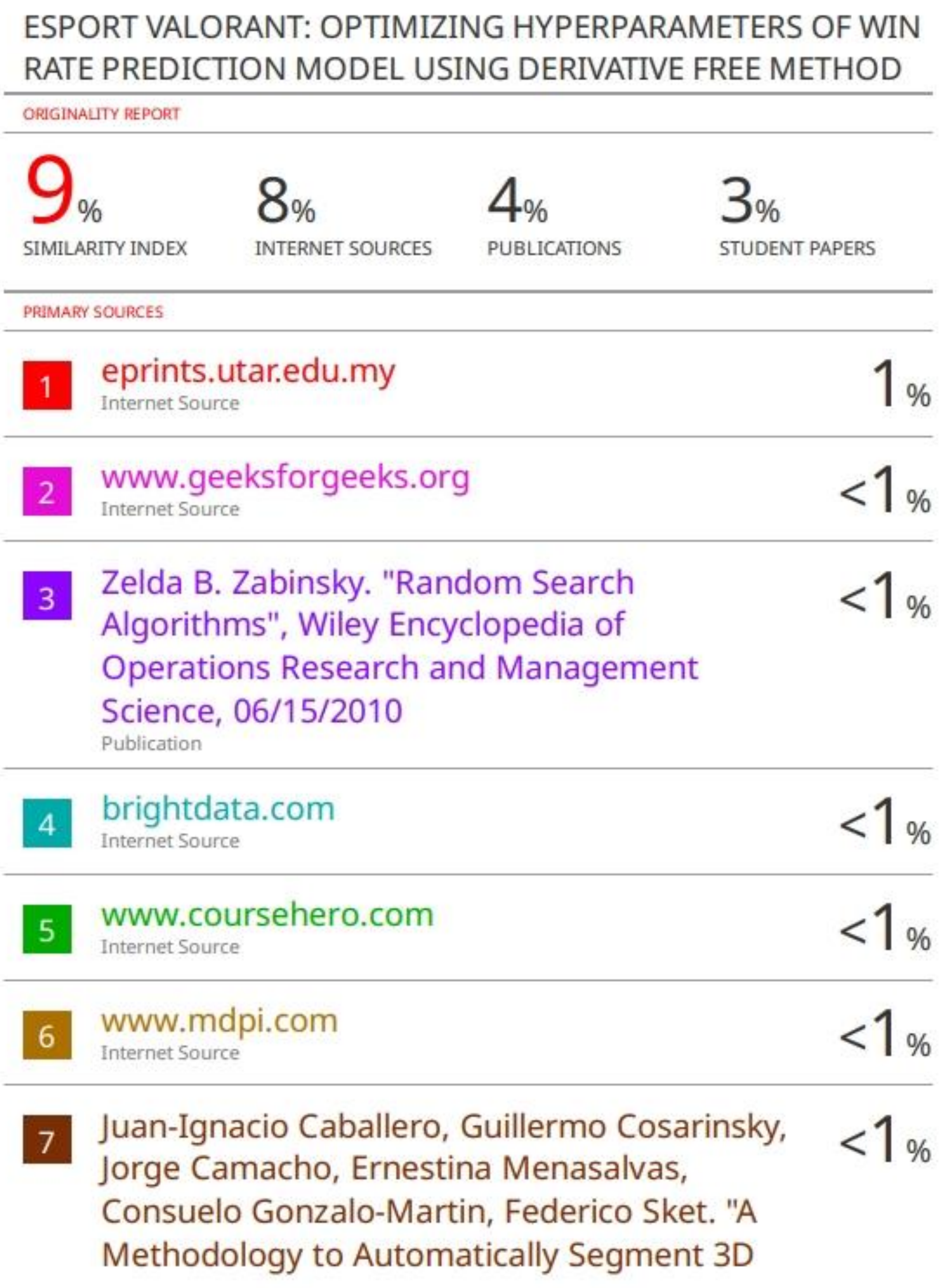
- optimize model using different fine-tuning method
- compare the performance of both model based on performance metrics
- repeat the steps with different model to determine which model is suitable to fine-tune with Bayesian Optimization

Conclusion

- Using Bayesian Optimization on complex model is better than exhaustive search method

Project Developer: Chang Yang   Project Supervisor: Ms. Tseu Kwan Lee

PLAGIARISM CHECK RESULT



ESPORT VALORANT: OPTIMIZING HYPERPARAMETERS OF WIN RATE PREDICTION MODEL USING DERIVATIVE FREE METHOD

ORIGINALITY REPORT

9%	8%	4%	3%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	eprints.utar.edu.my Internet Source	1%
2	www.geeksforgeeks.org Internet Source	<1%
3	Zelda B. Zabinsky. "Random Search Algorithms", Wiley Encyclopedia of Operations Research and Management Science, 06/15/2010 Publication	<1%
4	brightdata.com Internet Source	<1%
5	www.coursehero.com Internet Source	<1%
6	www.mdpi.com Internet Source	<1%
7	Juan-Ignacio Caballero, Guillermo Cosarinsky, Jorge Camacho, Ernestina Menasalvas, Consuelo Gonzalo-Martin, Federico Sket. "A Methodology to Automatically Segment 3D	<1%

Universiti Tunku Abdul Rahman			
Form Title : Supervisor's Comments on Originality Report Generated by Turnitin for Submission of Final Year Project Report (for Undergraduate Programmes)			
Form Number: FM-IAD-005	Rev No.: 0	Effective Date: 01/10/2013	Page No.: 1of 1

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

Full Name(s) of Candidate(s)	CHANG YANG
ID Number(s)	19ACB05073
Programme / Course	BACHELOR OF COMPUTER SCIENCE (HONOURS)
Title of Final Year Project	ESPORT VALORANT: OPTIMIZING HYPERPARAMETERS OF WIN RATE PREDICTION MODEL USING DERIVATIVE FREE METHOD

Similarity	**Supervisor's Comments (Compulsory if parameters of originality exceeds the limits approved by UTAR)**
Overall similarity index: 9 **%** **Similarity by source** Internet Sources: 8 % Publications: 4 % Student Papers: 3 %	
Number of individual sources listed of more than 3% similarity: **0**	
Parameters of originality required and limits approved by UTAR are as Follows: **(i) Overall similarity index is 20% and below, and** **(ii) Matching of individual sources listed must be less than 3% each, and** **(iii) Matching texts in continuous block must not exceed 8 words** ***Note: Parameters (i) – (ii) shall exclude quotes, bibliography and text matches which are less than 8 words.***	

Note Supervisor/Candidate(s) is/are required to provide softcopy of full set of the originality report to Faculty/Institute

Based on the above results, I hereby declare that I am satisfied with the originality of the Final Year Project Report submitted by my student(s) as named above.

[illegible]	
Signature of Supervisor	Signature of Co-Supervisor
Name: Tseu Kwan Lee	Name:
Date: 9/9/2024	Date:

UNIVERSITI TUNKU ABDUL RAHMAN

FACULTY OF INFORMATION & COMMUNICATION TECHNOLOGY
(KAMPAR CAMPUS)

CHECKLIST FOR FYP2 THESIS SUBMISSION

Student Id	19ACB05073
Student Name	CHANG YANG
Supervisor Name	MS. TSEU KWAN LEE

TICK (√)	**DOCUMENT ITEMS** Your report must include all the items below. Put a tick on the left column after you have checked your report with respect to the corresponding item.
√	Title Page
√	Signed Report Status Declaration Form
√	Signed FYP Thesis Submission Form
√	Signed form of the Declaration of Originality
√	Acknowledgement
√	Abstract
√	Table of Contents
√	List of Figures (if applicable)
√	List of Tables (if applicable)
√	List of Symbols (if applicable)
√	List of Abbreviations (if applicable)
√	Chapters / Content
√	Bibliography (or References)
√	All references in bibliography are cited in the thesis, especially in the chapter of literature review
√	Appendices (if applicable)
√	Weekly Log
√	Poster
√	Signed Turnitin Report (Plagiarism Check Result - Form Number: FM-IAD-005)
√	I agree 5 marks will be deducted due to incorrect format, declare wrongly the ticked of these items, and/or any dispute happening for these items in this report.

*Include this form (checklist) in the thesis (Bind together as the last page)

I, the author, have checked and confirmed all the items listed in the table are included in my report.

[illegible]

(Signature of Student)
Date: 10/9/2024