Bachelor thesis

Web Development of a Virtual DotA 2 Pre-Game Assistant using Statistical Intelligence

Submitted by: Sixiang Qiu

Department: Electrical Engineering and Computer Science

Degree program: Information Technology

First examiner: B.Sc. F. Anthony

Date handing out: 1st November 2020
Date handing in: 1st February 2021

(Professor Dr. Andreas Hanemann)
Head of Examination Board

Task description:

Description:

Gone are the days when E-sports was purely based on extra-ordinary skills and passion. Especially the so called Multiplayer Online Battle Arena (MOBA) games that often pitch teams of five against one-another require a level of strategic thinking and precise tactical execution that can only be achieved through continuous practice and the assistance of a subject matter expert, often a full time coach, to find the most critical issues to improve upon.

This thesis details the development of a Web Application that attempts to provide the reflective, pre-game assistance such a coach could offer at an intermediate level, by statistically analyzing previously recorded match data and, intelligently, make (a) strategic recommendation(s).

Tasks:

- Collection and/or gathering of DotA 2 (MOBA) match data
- Analyze the data with the help of statistical intelligence
- Develop a Web Application that visually highlights the meaningful components of the collected data set
- Offer a coach-like pre-game strategic recommendation

B.Sc. F. Anthony

Statement on the bachelor thesis

I assure you that I wrote the work independently, without outside help.

Only the indicated sources have been used in the preparation of the work.
The text or the sense of the text are marked as such.

I agree that my work may be published, in particular that the work may be submitted to third parties for inspection or that copies of the work may be made for forwarding to third parties.

28.01.2021
_____
Date

Qiu Sixiang
_____
Signature

Zusammenfassung der Arbeit
Abstract of Thesis

Fachbereich: Department:	Electrical Engineering and Computer Science
Studiengang: University course:	Information Technology
Thema: Subject:	Instructions for Writing a Thesis

Zusammenfassung:

Abstract:

In this thesis, a web application is developed integrating a website with model trained by machine learning algorithms which explores and then reveals relationship between hero combination and game result in a specific MOBA game, namely DOTA2. Two main functions are implemented on the website, the prediction function and the recommendation function. The first one predicts the game result when heroes are fully picked while the latter one recommends hero when heroes are not fully picked leaving space that there must be an optimal pick of hero. Model is trained by several machine learning algorithms which are considered suitable for the case. In the end, the application is being reviewed with respect to the part of website and the part of model.

Verfasser: Author:	Sixiang Qiu
Betreuender Professor/in: Attending Professor:	B.Sc. F. Anthony
WS / SS:	WS 20/21

Table of Contents

1 Introduction

1.1 Motivation

This chapter first explains the concept of DOTA2 such that the following thesis could be based upon. DOTA2 is a typical MOBA game, namely multiplayer online battle arena, which requires its players a higher level of strategic thinking than others to take the victory of a specific match. From the beginning of each match, ten players are split into two teams (five on five), picking heroes for themselves to fight against the other team. (See Figure 1.1)
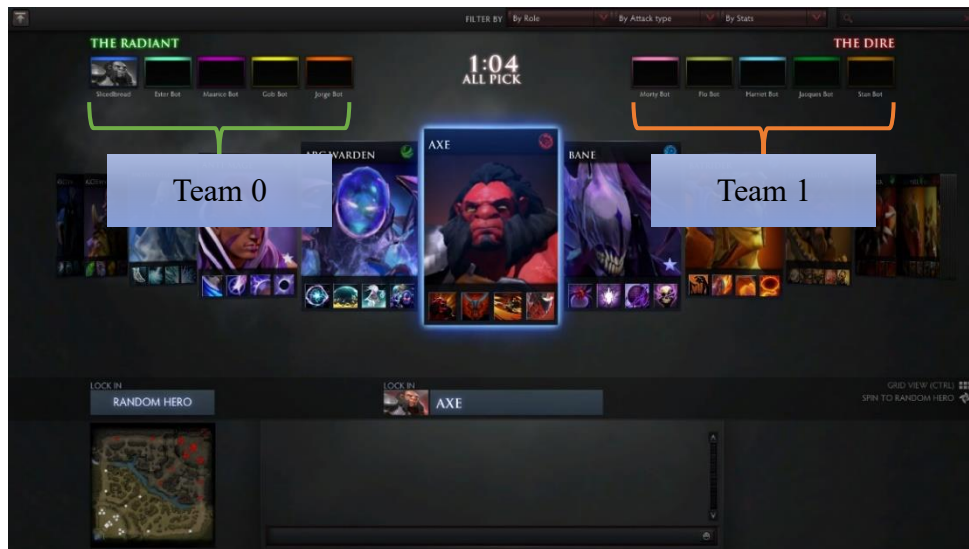
Figure 1.1 Ten players picking heroes

Among 119 heroes available in DOTA2 (until finishing this thesis), different heroes are assigned with different characteristics. Some might be more capable of dealing damage to the enemies while other might be a tank which could endure more damage from the enemies. The variety of characteristics yields different hero combination of two teams in which the team with better hero combination could have more probabilities to win once the match starts. Figure 1.2 shows the characteristics of hero named **AXE**.
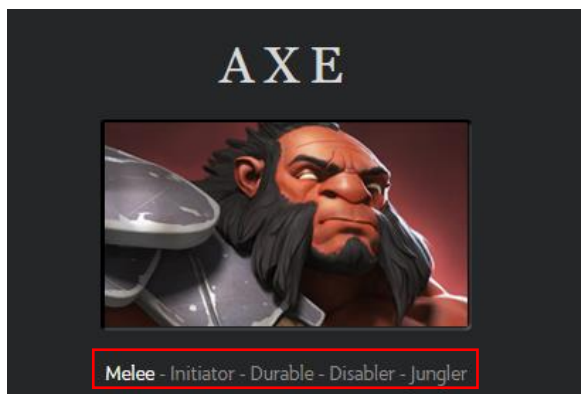
Figure 1.2 Characteristics of AXE

Again, among 119 heroes available in DOTA2, there could have tens of thousands of hero combination. Players who knew more about the hero combination would have more advantages against others. However, it is considered impossible for individuals to fully understand all of the hero combination together with their inner relationship. Professional teams provide one of the solutions. Data analysts are hired to learn

the underlying pattern so that coaches could be albe to determine the optimal hero combination for players on the spot. This works for professional team which strives for greater archeivement perhaps first title on the championship. Besides professional team, there are still millions of amateur players who cannot hire their personal back-up crew to look after their hero picking. In this thesis, with the knowledge of machine learning, it is considered feasible to develop an application which provides an opportunity for amateur players to enjoy the same assists as the professionals.

1.2   Goal

This thesis aims to explore and reveal the relationship between hero combination and the result of the game. Same as the professionals that coaches could not interfere players' in-game decision, this thesis is also regardless of players' in-game performance because DOTA2's authentication policy bans access from external links to in-game data (which all of the online games would do for security reasons).

In this thesis, previous match data is collected from open API website of DOTA2. After feature engineering, algorithms of machine learning are applied on the dataset to generate a reliable model representing the relationship between hero combination and the result of the game. In this thesis, a website is considered to be the container which finally carries out the model. The first usage of the model would be predicting the result of the game when heroes are fully picked beforehand while the second usage of it would be offering its users certain recommendation according to its knowledge learnt from machine learning algorithms when users are confused of which hero to pick.

1.3   Organization

The rest of the thesis is structured as follows:

- Chapter 2 states the technical background of working with machine learning algorithms and how to build up a web application. Before that, the architecture of the project, split stack development, is discussed. Then, the part of machine learning starts from feature engineering to Scikit-learn. The part of how to buid up a web application describes the web framework used in this project, namely Flask, comparing with its related work, namely Django.

- Chapter 3 focuses on the approaches and implementation details during the development of the actual project. Data collection module written in Python is provided which obtains the original data. The pre-processing and classification of the original data yields dataset to be passed into feature engineering trailing with model training. Model training would

be introduced from the degree of selection of algorithms. Finally, This chapter ends with describing the design and implementation of Flask application which carries out the model generated.

- Chapter 5 closes the whole thesis with a general conclusion and an outlook for future development of the application.

2 Technical background

2.1 Split stack development

Since this thesis aims to develop a web application which integrate the model generated by machine learning algorithms into a visualization on World Wide Web, it demands for a full stack developer who is capable of both the back-end coding language such as Python in this project, and the front-end coding language known as the HTML, CSS, and the JavaScript. Different from the traditional architecture of a full stack application, this project adopts split stack development architecture which has several advantages against the traditional one. Here first introduce the traditional architecture in comparison.
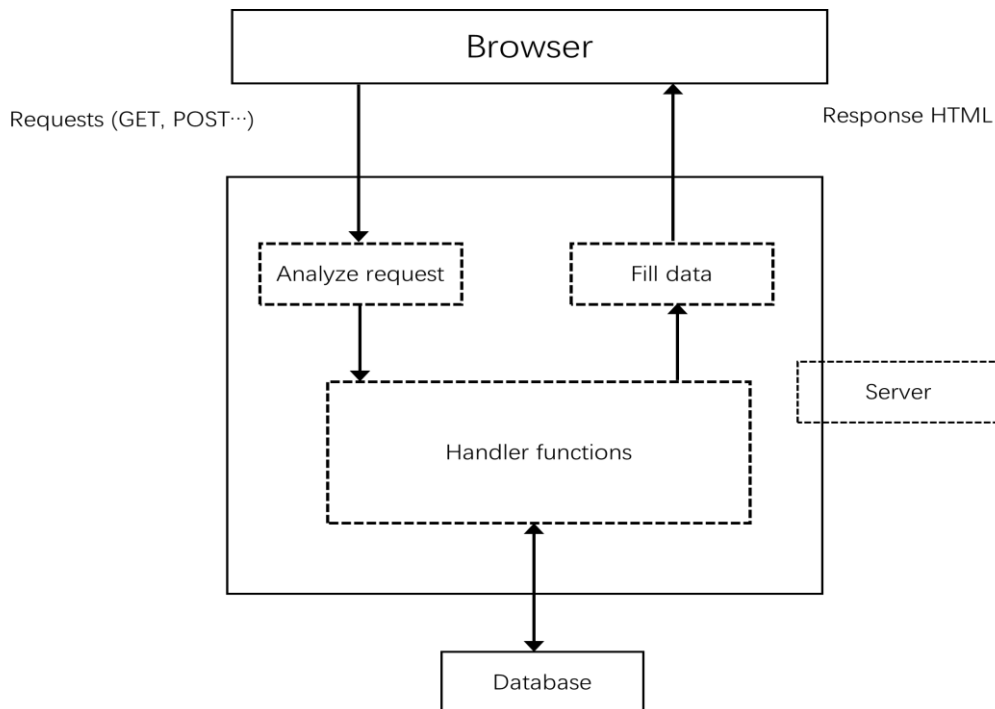
Figure 2.1 Full stack development

From Figure 2.1, such architecture yields inefficiency for the afterward codes management and project maintenance because the implementation of front-end and the implementation of back-end are all being put together. Providing that, in the future, the back-end part of the project is being improved, it is not expected to interrupt with the front-end part of the project in order to prevent errors.

For the sake of that, split stack development is adopted in which the front-end system and the back-end system are separated. According to split stack development, the architecture is updated in Figure 2.2.

Now, the front-end system and the back-end system are totally separated which gives a solution to the issue pointed out before. But split stack development is more than that. It provides other benefits as well.
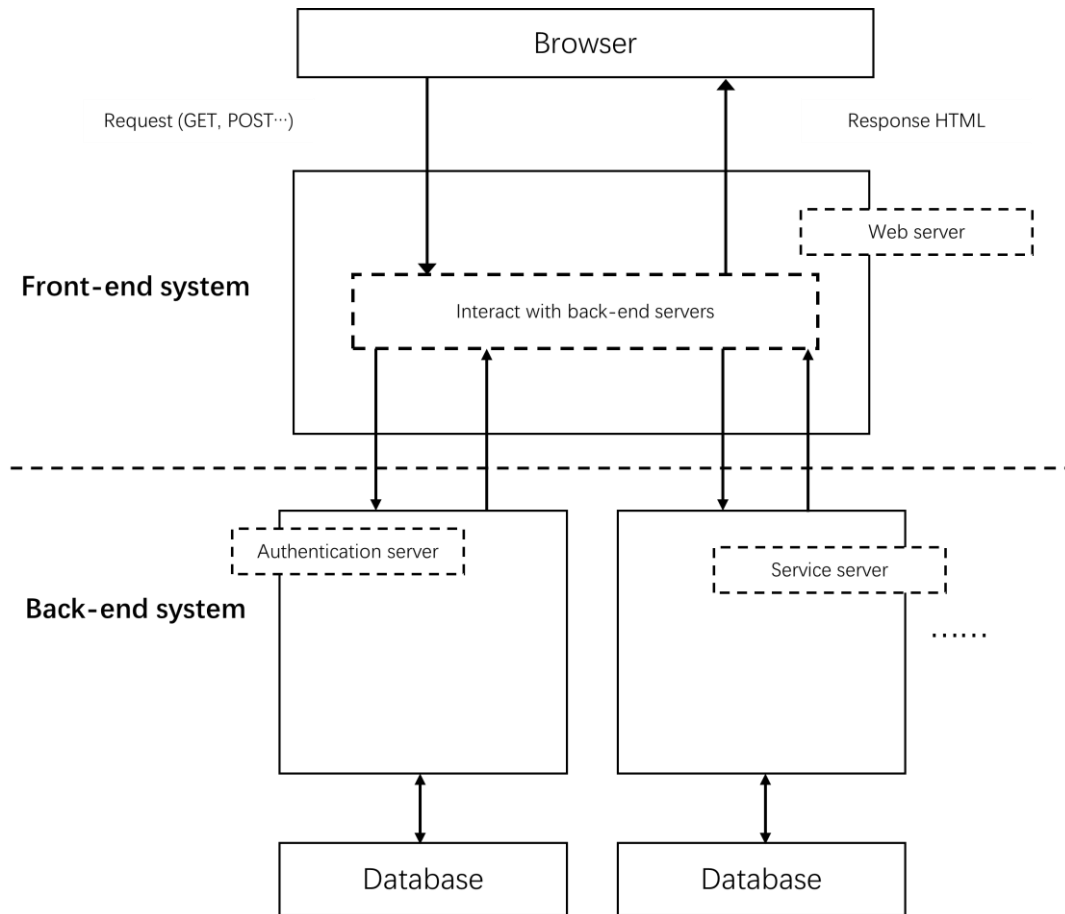
Figure 2.2 Split stack development

- Independent technology stacks: The separation of front-end system and back-end system helps to eliminate all the restraints on technology choices that each may have imposed on the other. This leads to the use of completely independent stacks, that could have been difficult or impossible to implement in a universal model. Such approach allows to choose the best technologies for the project.

- Simultaneous development and fast deployment: With front-end and back-end split, both layers can be developed independently and simultaneously. As neither stack is reliant on the other, the front-end code can be tested and deployed whenever it is done, with API endpoints brought together in the end. Besides, a new front-end part can be made and integrated with an old back-end part. It is able to upgrade each of them independently further on. [1]

2.2 Machine learning

Machine learning, as a phenomenal field these years, is considered too complicated to start explaining with. Instead, this thesis picks the title of statistical intelligence for descriptiveness. However, statistical intelligence as a subset of machine learning is always confused with the concept of statistics. Both of them seem to rely on theoretical algorithms to yield results.

The difference between them mainly lies on the point of emphasis of each. Statistics is theory-driven, depending on powerful mathematical theories to interpret the result. It focuses on parametric inference. However, statistical intelligence, which is actually machine learning, is data-driven, depending on big scale of data in order to predict the future. It focuses on model prediction.

Note that this project does not emphasize on interpreting the model generated. On contrary, this project concentrates on evaluating and improving the generated model by its performance in order to ensure the predictiveness of it.

2.2.1 Feature engineering
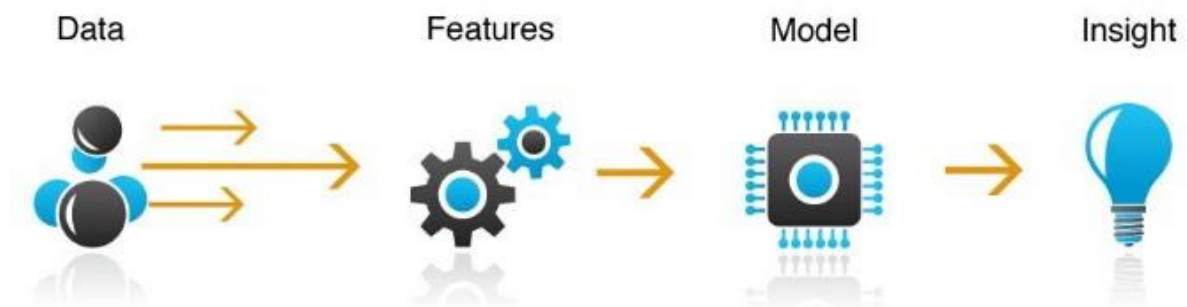
Figure 2.3 Typical machine learning procedure

From Figure 2.3 [2], within the process of machine learning, a typical project always starts with collecting data which is simple to understand. According to Figure 2.3, before applying algorithms to train the model with the data, data have to be converted into certain format which makes it possible to derive features beneath them. The derivation of features is called feature learning. With definition of appropriate features, it can improve the performance of the model generated by the algorithms.

Feature engineering uses **domain knowledge** to extract features from dataset via **data mining techniques**.

- Domain knowledge depends on the understanding of detailed project. For example, this project aims to explore and reveal the relationship between hero combination and the result of the game in which hero combination is exactly one of the features of the dataset. This

is obtained due to the domain knowledge of developer via the actual game, namely DOTA2 in this project.

- However, domain knowledge is not enough to put theory into practice. Data mining techniques are needed to analyze the features like hero combination. It is an interdisciplinary subfield of machine learning and statistics with an overall goal to extract information from a dataset and transform the information into a comprehensible structure for further use. [3]

2.2.2 Scikit-Learn

After feature engineering, data is ready to be passed into machine learning algorithms. Instead of implementing algorithms manually, Scikit-learn is introduced in this project. Scikit-learn is an open-source machine learning library largely written in Python because that Python provides powerful libraries such as Matplotlib for plotting, NumPy for array vectorization. These two libraries support the algorithms which are selected in this project, namely the decision tree and the SVM.

Including decision tree and SVM, Scikit-learn provides dozens of built-in machine learning algorithms and models. They are called estimators in Scikit-learn. For the subsequent chapter, term estimator is used substituting machine learning algorithms. After selection of estimators, data can automatically fit to the estimators using fit method in Scikit-learn. [4] An example of fitting the estimator is shown in Figure 2.4. Aftter fitting the estimator, it is fine to start training model. Chapter 3.3 reveals more details of training model.

```
>>> from sklearn.ensemble import RandomForestClassifier
>>> clf = RandomForestClassifier(random_state=0)
>>> X = [[ 1,  2,  3],  # 2 samples, 3 features
...      [11, 12, 13]]
>>> y = [0, 1]  # classes of each sample
>>> clf.fit(X, y)
RandomForestClassifier(random_state=0)
```

Figure 2.4 Fitting estimator in Scikit-learn

2.3 Web framework

Chapter 2.1 introduced the architecture of this project which helps for understanding how to build up a web application. As mentioned in Figure 2.2, front-end and back-end are separated but connected. In front-end system, URLs are defined to exchange data with other web pages or database, etc. It provides one of the solutions to realize the connection with back-end system as well. To handle URLs defined in front-end system, web framework is introduced which is able to bind the URLs with certain callback funcitons in back-end system. Added with web

framework, Figure 2.2 is updated. Call Figure 2.5 the architecture of this project.
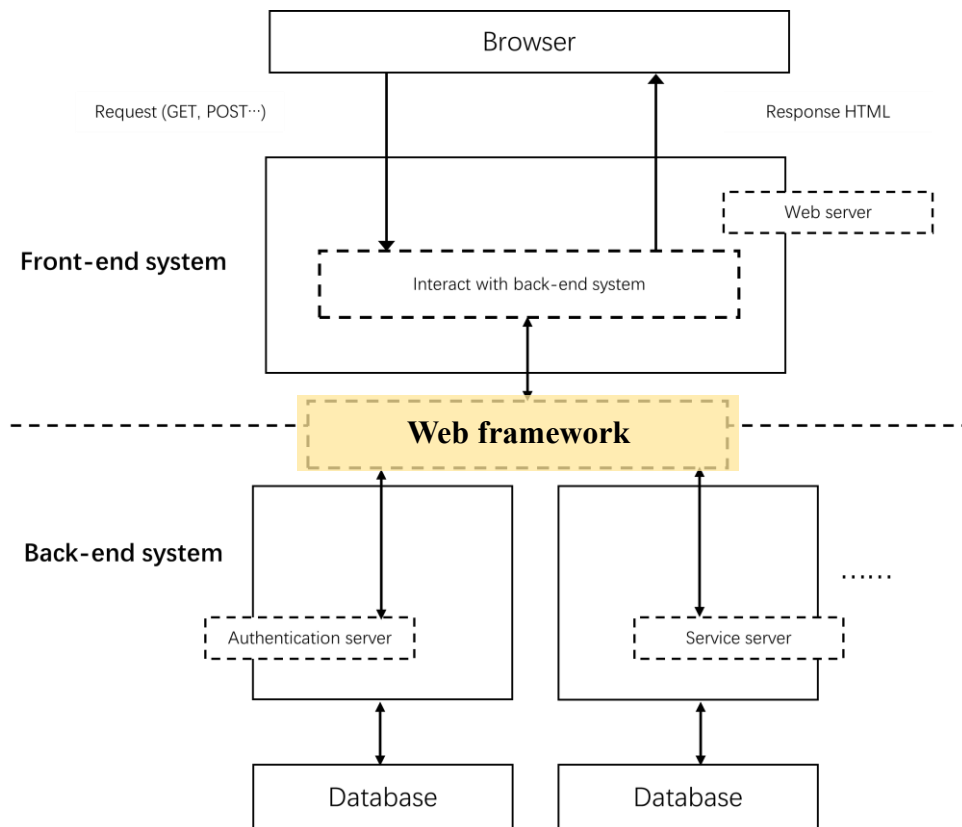
Figure 2.5 Architecture of this project 1.0

Instead of building up a self-made framework, this project decides to adopt ready-made framework in order to avoid unimportant issues and make it easier for others to practice on the project. Meanwhile, only frameworks written in Python are considered due to consistency with the back-end system especially with Scikit-learn. In this case, two of the most popular ones are introduced below which is Flask and Django.

Because that this project does not dig in deep into the aspect of web framework. Both the web frameworks are described shortly with their basic characteristics and their routing syntax which is the methodology of web framework to bind the URLs.

2.3.1 Flask

Flask is a micro web framework which aims to keep the core simple but extensible. Developers using Flask are able to make decisions by themselves instead of by the framework. Flask only makes decision of what templating engine it use, which is also easy to be changed according to requirement. Everything else is up to the developers, such as what database to use or other plug-in components if they are needed. [5]

Flask uses its 'decorator' syntax to route the front-end request with the back-end response on a pre-defined URL. A small example [6] is displayed below showing how easy it is to associate the client and the server with Flask's routing syntax. From Figure 2.8, it is shown that when the front-end requests to URL '/' which is by default the home page of a website, the flask will return 'Hello, World' to the client.

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello, World!'
```

Figure 2.6 Flask routing syntax [6]

2.3.2 Django

In comparison, Django is another framework written in Python which is even more popular among developers than Flask. Django was developed in a fast-paced newsroom environment which means that it was designed to make common Web-development tasks fast and easy. [7] Therefore, powerful utilities are integrated into Django in advance which makes it ready-to-use right after installation. For example, Django includes data validation algorithm and its original database handler both of which could be useful for this project.

Meanwhile, an example shows routing syntax of Django in Figure 2.9:

```
from django.urls import path

from . import views

urlpatterns = [
    path('articles/2003/', views.special_case_2003),
    path('articles/<int:year>/', views.year_archive),
    path('articles/<int:year>/<int:month>/', views.month_archive),
    path('articles/<int:year>/<int:month>/<slug:slug>/', views.article_detail),
]
```

Figure 2.7 Django routing syntax [8]

A request to '/articles/2005/03/' would match the third entry in the list. Django would call the function $views.month\_archive(request, year = 2005, month = 3)$.

'/articles/2003/' would match the first pattern in the list, not the second one, because the patterns are tested in order, and the first one is the first test to pass.

'/articles/2003/03/building-a-Django-site/' would match the final pattern. Django would call the function:

$views.article\_detail(request, year = 2003, month = 3, slug = "building - a - Django - site")$. [8]

2.3.3 **Discussion**

Finishing introducing them, it is also important to investigate the characteristics of this project to see which one best suits the needs.

The most useful features inborn with Django is the data validation module and the database module. However, since there is no login interface while only heroes' information in the game are needed to be tracked, data validation module would be quite small that implementing it manually is considered not quite hard. The database module is also considered unnecessary to be implemented by now due to heroes' information is fixed. Other than these two modules, Django seems a little too heavy for this project. For the future development's requirement, Flask takes advantage of its extensibility. It is also able to build up a database and implement it with Flask.

As for the routing syntax, since the project would probably be based only on one page with two of the main functions, predicting and recommending, the module for routing would not be quite large. In this case, the powerful routing syntax of Django is not needed whereas routing syntax of Flask should be sufficient. Django is more suitable for larger project where an integrative class of routing is better.

3   Approaches and Implementation

This chapter starts with the creation of dataset with Python programs which gather, pre-process and classify the original data. Then, hero combo is added as feature of the data trailing with the part of model training which explains the selection of estimators and the fusion of trained models in addition. This chapter also updates the architecture of this project with MTV architecture as well as the implementations details of Flask application on the basis of the architecture.

3.1   Dataset creation

To collect the original data from DOTA2, its operator company, Valve, provides web APIs for third party developers and researchers, among which API for public match data query is also available.

However, not many documents is provided online for the original web API. In order to get easy-to-use API, this project used API provided by another third-party developer, OpenDota. [9] The free version of public API from OpenDota is able to provide 50,000 requests per month, where each request can get data for up to 100 public DOTA2 matches, thus meeting the requirement for the original data collection.

3.1.1   Data collection module

To collect the original data from the public API, a raw data collection module was built by Python. The collection module is mainly composed of three parts, namely, the main module, the API request module and the data storage module.

The main module is responsible for overall data collection and scheduling work. It is implemented for firing other modules in the program. Running the main module firstly loads the pre-set API request URL, dispatching the API request module to make data requests. Then, the original data returns through the data storage module and is stored as a file.

The API request module is an auxiliary tool module, this module encapsulated the raw Web API provided by OpenDota and provide Python API interfaces for the main module to call. The encapsulated API has public match details request, hero information request etc.

The data storage module is also an accessibility module which is primarily responsible for tracking and storing two kinds of data:

(1) The raw data from the public API is stored in JSON format for subsequent operation;

(2) The match ids which have already been collected are stored preventing repeating operation.
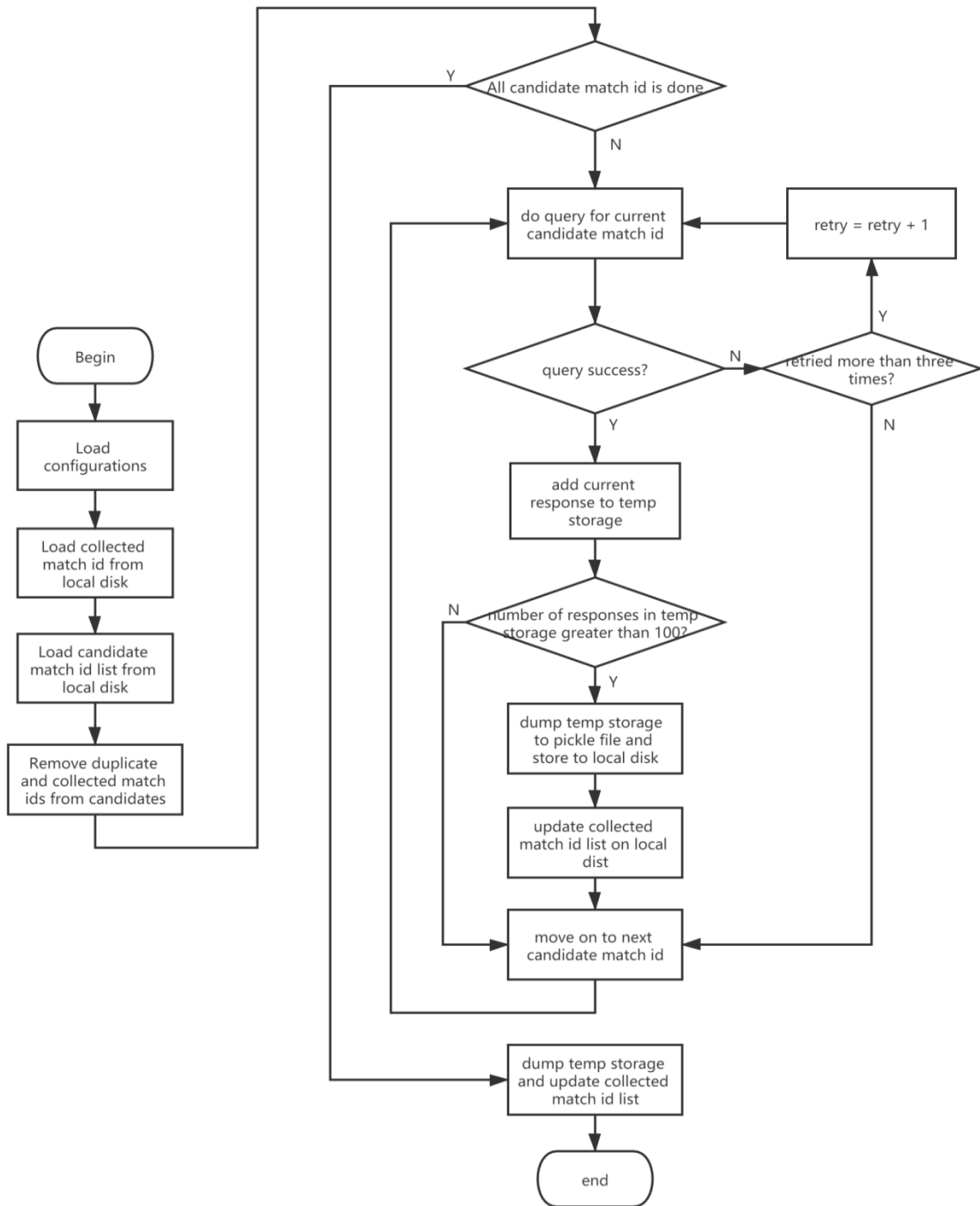

Figure 3.1 Flow chart of data collection module

3.1.2 **Raw data pre-processing**

Through the data collection module, approximately 150,000 DOTA2 match data were collected which are definitely enough for model training. Without pre-processing, each entry of the raw data has the format as List 3.1.

Elements	Data Type
Match id	Integer
Match version	Float
Match start time	Float
Game server	String
Duration	Float
Radiant win	Boolean
Skill level	Integer
Player_0 hero id	Integer
Player_1 hero id	Integer
Player_2 hero id	Integer
Player_3 hero id	Integer
Player_4 hero id	Integer
Player_5 hero id	Integer
Player_6 hero id	Integer
Player_7 hero id	Integer
Player_8 hero id	Integer
Player_9 hero id	Integer

List 3.1 Raw data format

Apparently, not all of the elements seems relevant to the result of the games while not all of the data seems to be reliable either. Hence, a pre-processing module is implemented realizing following functions:

(1) to serialize the raw data, retaining relevant elements whereas deleting irrelevant ones.

(2) to filter the raw data when some entries of the collected data are considered unreliable.

According to observation of the collected data, some entries are missing value of certain elements, such as empty hero ids. The filtering function is also responsible for verifying the integrity of the collected data. Entry lacking any of the information is filtered out.

Following paragraphs explain the logic of serializing and filtering the raw data, organized in the order of elements in the raw data:

Game server and Match start time

Among all of the elements in the raw data, the value of game server only tells where the match is arranged, for example in Chinese server or in German server which differs from lagging status. Meanwhile, the value of starting time of the game only tells the time when the match is arranged, for example at 9 p.m. These two elements are considered totally irrelevant to the result of the games, nor could they be used for filtering, thus deleting them at first place.
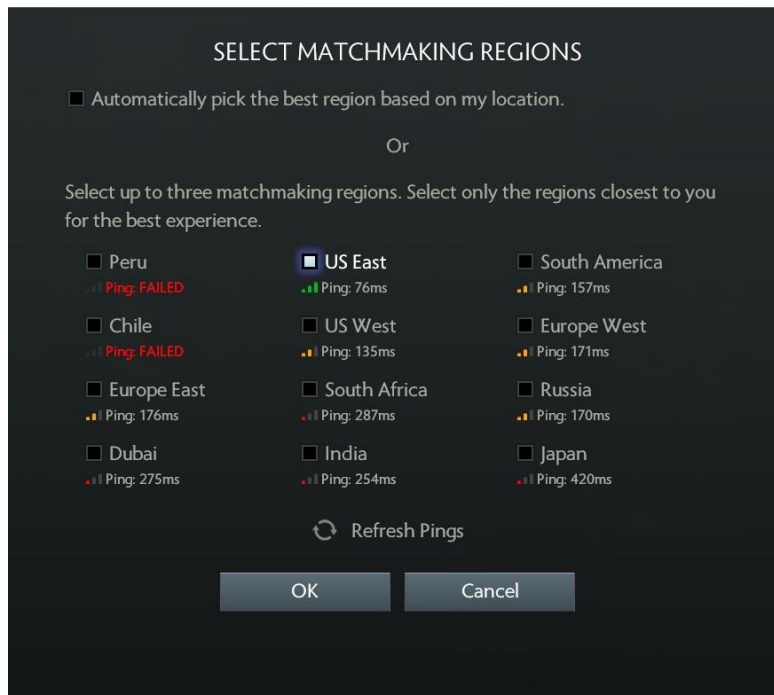

Figure 3.2 Selecting game server

Radiant win and Hero ids

Reversely, the value of hero id stands for which 10 heroes are picked by players in this match. Likewise, radiant is the name of the team in DOTA2 while the other team is named dire. The Boolean value of radiant win stands for the final result of the game. These two elements are exactly the input and the output variables for model training.

Remaining elements are evaluated below:

Match id

Element match id serves for indexing the match in data collection module which is introduced before. It is deleted since data with repeating match id was already filtered out which is the job done by the data collection module.

Match version

The version of the game is varying over time. In this project, only entries from the latest version is persisted in order to make the model generated up to date, other data are filtered out. For this reason, the model would have to be trained again once the version has updated. Then, it is deleted because that with collection of entries from the same version, it would not affect the result of the games.
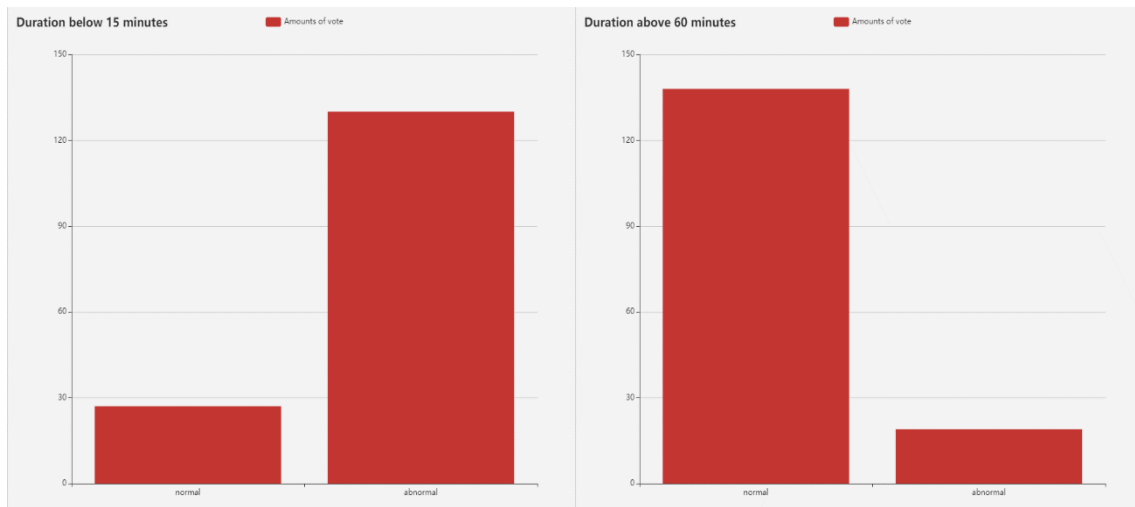
Duration

The duration of matches would not influence the result of them. In details, some hero combination might become more powerful over time while others might lose their influence when the match lasts too long. Hence, the element of duration is deleted by the pre-processor module.

Nevertheless, according to observation of the collected data, duration of a match varies from a wide range. Due to empirical judgement, an average DOTA2 match should range from 30 to 60minutes. Plot the data on a diagram where the x-Axis represents the duration, and the y-Axis represents the number of matches:
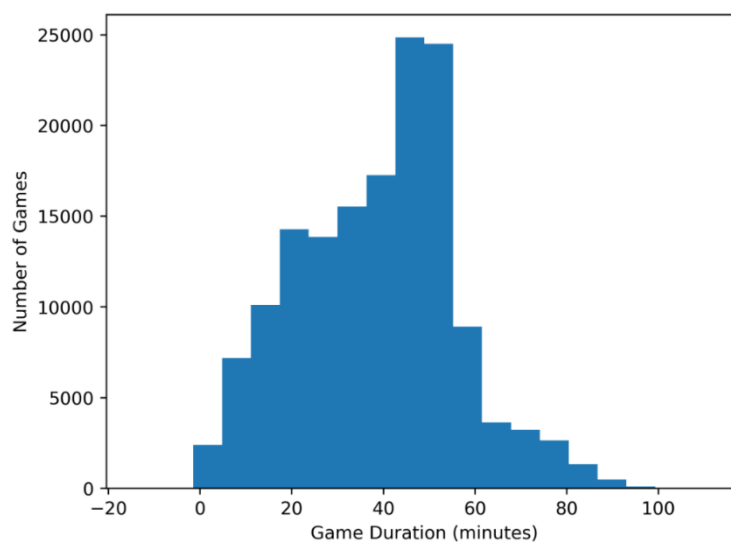
Figure 3.3 Duration vs. number of games

The result diagram indicates that, indeed, most of the matches end from 40 to 60 minutes, approximately matching the Bell curve. For matches that is too short or too long, investigation of potential reasons is needed to check if those matches are reliable. However, it is impossible to find out exactly why those matches happen. Instead, a brief questionnaire is handed out in players' community of DOTA2, namely Max+ which is a mobile phone application in China. 157 results is gained until the deadline of this thesis which are shown below where the left column means that the voter thinks the match duration is normal while the right column means that the voter thinks the match duration is abnormal. Since the questionnaire is merely dualistic, the result could be a bit inaccurate.

Figure 3.4 Players' opinion on game duration

But, in general, it is illustrated that extremely short matches are comparably much more unreliable than those which are extremely long.

In this case, matches which lasted below 15 minutes are filtered out while those lasted above 60 minutes are persisted. Specifically, the filter's threshold is selected as the 10th quantile of the matches, which is 14.81 minutes to be exact.

Skill level

For each public match, an overall skill level will be assigned, where skill level 'Very High' and 'High' represent matches with experienced and high-level players, and 'Normal' means junior-level matches. The players' skill level obviously impact on the result of the games. For instance, with exact same picks of heroes, the matches might not result the same with different skill levels of the matches. However, if the matching system of DOTA2 is trusted balanced, skill level would not impact an actual match which is similar to the element match version.

Therefore, the element skill level is deleted while the pre-processor module is designed to persist matches only assigned with skill level 'High' which is the intermediate one to fix the

effect of element skill level. In the future, it is considered feasible to classify the entries into three sets of data which generate three results of model to represent matches hold in all three skill levels.

List 3.2 illustrates the dataset format after pre-processing in which, the players' picks of heroes is re-named by adding their team name before. Team 0 stands for team of the radiant while team 1 stands for team of the dire. In the end, the value of element label represent the result of the game, replacing the Boolean element radiant win. Integer 0 means the victory of the radiant and integer 1 means the victory of the dire.

Elements	Data Type
Team0 hero id 0	Integer
Team0 hero id 1	Integer
Team0 hero id 2	Integer
Team0 hero id 3	Integer
Team0 hero id 4	Integer
Team1 hero id 0	Integer
Team1 hero id 1	Integer
Team1 hero id 2	Integer
Team1 hero id 3	Integer
Team1 hero id 4	Integer
Label	Integer

List 3.2 Dataset format

3.1.3 Dataset classification

After pre-processing, about 50,000 entries are persisted and serialized, entries are integrated into dataset which still needs classification into three different subsets for subsequent use. Subsets are composed of the training dataset, the validation dataset and the test dataset. The reason of classification is described as follow.

(1) Training dataset is the subset used to fit the estimators so that estimators could learn the underlying pattern directly from the dataset, and generating models trained.

(2) Validation dataset is used to evaluate a given estimator, and usually is used for fine-tuning the given estimator. Hence, the estimator occasionally observes the data, but it would never learn anything from the validation dataset.

(3) Test dataset is the subset used to provide an unbiased evaluation of the model generated. It is only used once the model is completely trained, which has already been passed through the training and validation dataset. By doing so, the test dataset can, in true sense, represent sample of data that is completely new and unfamiliar to the model. The intention of splitting test dataset is to perform the model on unexpected sample of data which is never faced by it in the process of training and validation and evaluate the performance of the model afterward.
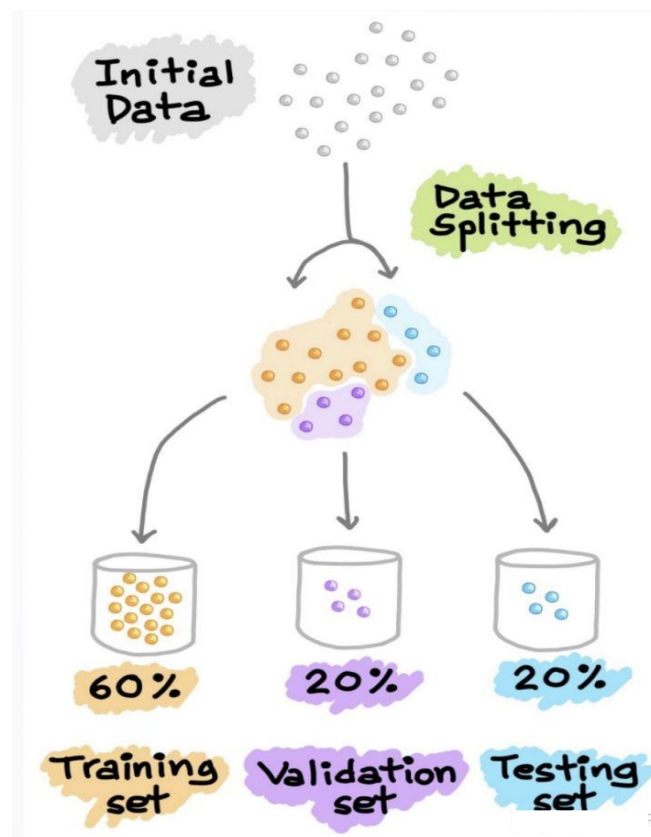

Figure 3.5 Dataset classification [17]

Because that only 50,000 entries is persisted, formatting into dataset in this project, splitting training dataset and validation dataset is operated on the basis of K-fold cross validation in order to maximize the use of sample.

In a word, K-fold cross validation is a resampling technique without replacement. Another advantage of a resampling technique is that each sample is used for training and validation exactly once. This yields a lower variance of the model's performance than the traditional method. The process of K-fold cross validation is:

1. Split the training dataset into K-fold (most often into 5 or 10 fold).

2. Out of the K-fold, (K-1) fold is used for training, the left one is used for validation.

3. The model is trained with training dataset (K-1 fold) and is validated with validation dataset as the left one. The performance of the model generated is recorded.

4. Step 3 is repeated until each of the k-fold is used for validation purpose. Hence, K models are generated with their performance recorded.

5. The mean and standard deviation of the model's performance is computed by accounting all of the performance of the models recorded in step 4.

6. Step 3 to Step 5 is repeated for fine-tuning the estimator.

7. Finally, the model is generated again on the training dataset with fine-tuned estimator and the performance of it is evaluated by calculating its performance on the test dataset.
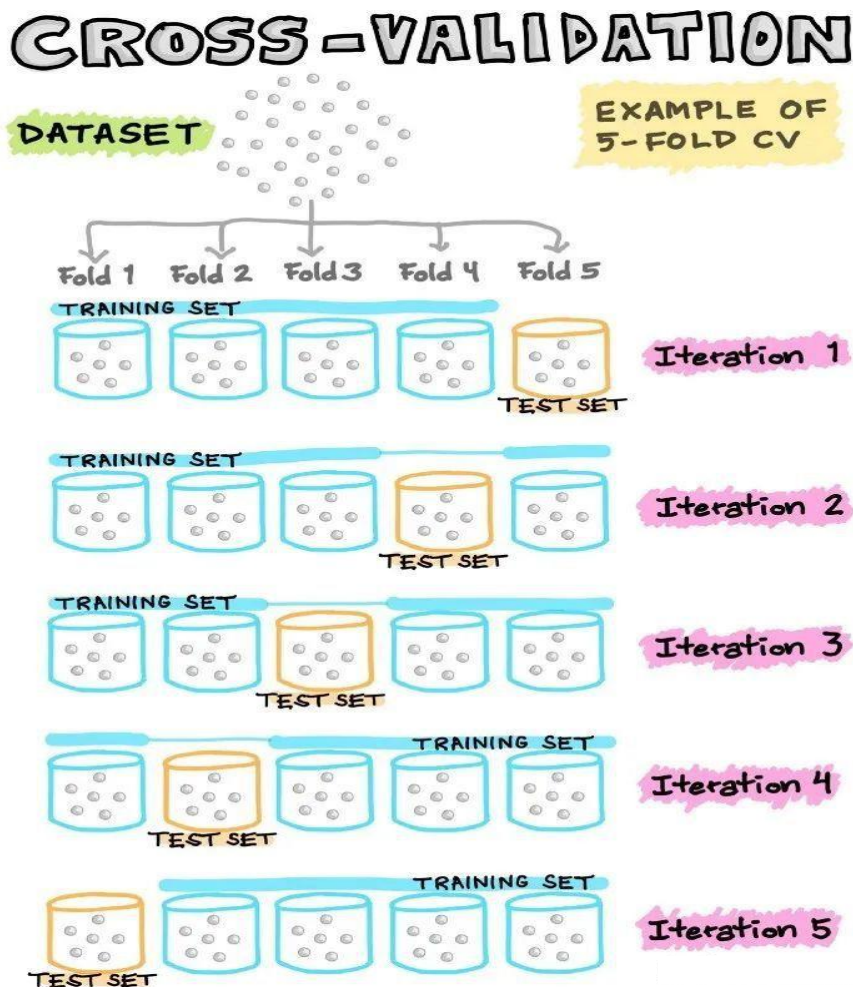

Figure 3.6 Example of 5-fold CV [17]

The entire dataset in this project is divided into training and test dataset at a scale of 80% and 20% since the size of it is comparably small. At the same time, the training dataset are divided into 10-fold randomly according to 10-fold cross validation process.

Fold 0	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Fold 6	Fold 7	Fold 8	Fold 9	Test set(20%)

Figure 3.7 Dataset classification with 10-fold CV

3.2   Feature engineering

Chapter 3.1 finishes dataset creation. However, Figure 2.3 indicates that data still needs to be transferred into features so that it could be handled by model estimators. The reason behind is that estimator cannot recognize hero id as features. Therefore, dataset is converted into feature matrix formatting like Figure 3.8.

$$X = \begin{bmatrix} x_{11} & x_{21} & \cdots & x_{n1} \\ x_{12} & x_{22} & \cdots & x_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ x_{1m} & x_{2m} & \cdots & x_{nm} \end{bmatrix}$$

Figure 3.8 Feature matrix

In the feature matrix, variable M stands for samples in the dataset while N stands for features. Note that, in real world, another column is added behind N representing the output result of matches. In this project, 119 heroes are pre-defined as features, multiplying by 2 teams that is to say N equals 238 which is given by: $119 \ heroes \ \times 2 \ \text{teams} = 238 \ \text{features}$. Now, it is able to describe feature of hero combination of two teams. Heroes picked in the sample will result in an integer 1 in the feature matrix while 0 if not picked.

Although, original feature is defined as hero combination in the dataset, predictive modeling of match results with inputting only hero combination often does not lead to an acceptable performance. More features should be added to the input dataset in order to improve the performance of the model generated.

According to DOTA2 domain knowledge, heroes are combined with each other due to their characteristics which is explained in Chapter 1.1. Different heroes with different characteristics may become powerful when combined together. Originally, it is considered as a combination of 5 heroes which is the scale of a whole team. But, the combined effect between lesser heroes may also influence the match to a great extent.

Take two heroes as an example, namely **Earth Shaker (ES)** with **Queen of Pain (QOP)** shown in Figure 3.9, ES is a hero which supports the team while dealing considerable damage (a Nuker does) to the enemies while QOP is a hero which carries the team (who needs support) while dealing considerable damage as well. With ES supporting QOP, they can together destroy the

enemy team with their overwhelming damage. Team with ES and QOP is considered having more chance to win the match. Hence, call the combination between two heroes a hero combo. Hero combo is added as a feature to the matrix as well.
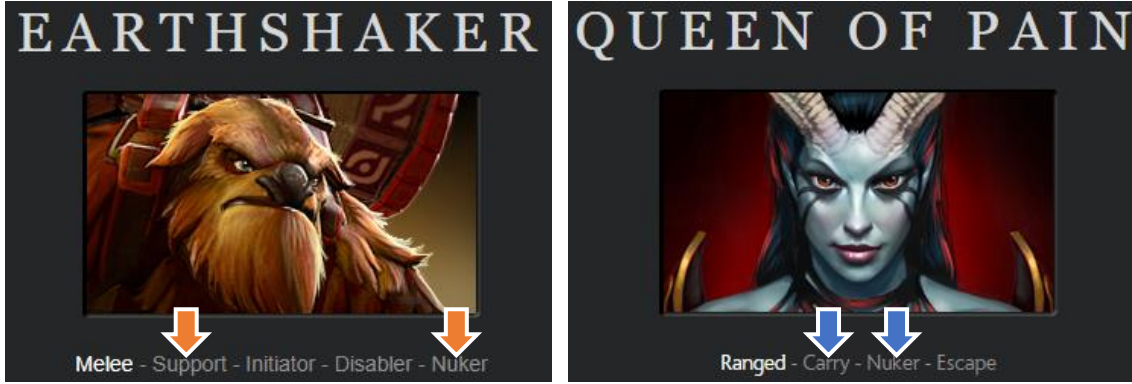

Figure 3.9 ES with QOP

By the time this thesis is written, DOTA2 has up to 119 heroes available in the game. If all hero combo are calculated as features, there will be a total of 7,021 features, which would possibly appear in team 0 and team 1 respectively. Added together, there would be a total of 14042-dimensional features. Such feature dimensionality is obviously excessive and can affect the generalization performance of the model, so it is necessary to reduce the dimension of this new feature. In order to degrade the dimension, dataset are processed through three different data mining techniques which are the stepwise feature selection, PCA and the statistical method. They would be introduced respectively below this chapter.

3.2.1 Stepwise selection

Stepwise selection was originally developed as a feature selection technique for linear regression estimators. It is combined with two different algorithms which are the forward selection and the backward elimination.

- Forward selection first initializes a subset of features as an empty set F, then calculates the improvement of accuracy of the model after gradually adding features to the empty set F. The algorithm selects the feature with the greatest increase in accuracy to be added to F, and loops above the operation until the accuracy is no longer improved.

- Backward Elimination, as opposed to the forward selection, initializes F as a collection of all features, selecting one feature at a time, and calculates the accuracy of the model when the feature is not included in the set F. If the accuracy is improved, the feature is eliminated from F until the accuracy no longer decreases.

In this project, the forward selection is adpoted since it is mentioned before that the initial features set F is considered excessive to be implemented.

3.2.2 Principal Component Analysis

PCA is a linear dimensionality reduction technique using singular value decomposition of the data. In order to apply PCA, features of hero combo together with the original hero combination of teams need to be converted into one-hot encoding. At the same time, stacking all the encoded data from the training dataset into a feature matrix, then could PCA be applied to reduce the feature dimensionality into a certain threshold range. In order to retain same amount of information in PCA as the other two techniques, the number of dimension reserved in PCA is set to 238, 338, 438 and 1238 respectively. These three threshold are calculated based on following formulas:

$$119 \ heroes \ \times 2 \ \text{teams} = 238 \ \text{dimensions (threshold 0)}$$

$$(119 \ heroes + 50 \ features) \times 2 \ \text{teams} = 338 \ \text{dimensions (threshold 50)}$$

$$(119 \ heroes + 100 \ features) \times 2 \ \text{teams} = 438 \ \text{dimensions (threshold 100)}$$

$$(119 \ heroes + 500 \ features) \times 2 \ \text{teams} = 1238 \ \text{dimensions (threshold 500)}$$

3.2.3 Statistical method

Unlike the techniques mentioned above, statistical method is an unsupervised process of learning the features. An unsupervised learning method stands for algorithms that reduce the column of output in the dataset while learning the input alone.

In statistical method, all of the features are first noted from the dataset. Then, the number of appearances for each of the features is counted. Threshold k (such as 50, 100 or 500) is set, sorting the features by their number of appearances, and the top k most frequently appeared features are selected. Interpretation behind this technique is that in actual game, players tend to pick the hero combo which they thought is more powerful thus considered more reliable as the feature in model training process.

3.2.4 Discussion

Since these three techniques all work for specifying the hero combo features upon the dataset, they are considered in competition with each other. To figure out which one of them is the optimal one for this project, all three techniques are applied on the dataset matrix for model training, specifically with logistic regression estimator which is easier to implement. Since it is also unnecessary to fine-tune and evaluate the model by this time, dataset is not classified beforehand. Call the model a roughly trained one which would be sufficient for comparing the data mining techniques with each other's performance. The performance is evaluated with the accuracy of the roughly trained model.

Threshold / Technique	0 (238 for PCA)	50 (338)	100 (438)	500 (1238)
Stepwise selection	0.5201	0.5425	0.5325	0.5322
PCA	0.5285	0.5401	0.5325	0.5190
Statistical method	0.5201	0.5430	0.5405	0.5385

Table 3.1 Accuracy for each technique

From Table 3.1, four thresholds are set in comparison with each other. Apparently, the optimal result happens in Statistical method when threshold is set 50. Based on the result, the statistical method is chosen for its better performance and top-50 hero combo is used for the rest of this project. The selected top-50 hero combo are listed in List 3.3.

After feature engineering, the feature matrix is expanded on its horizontal axis. 338 features is implemented compared to 238 features previously.

Hero id	Count	Hero id	Count	Hero id	Count	Hero id	Count	Hero id	Count
86-8	658	107-55	442	85-76	401	84-8	376	84-74	356
86-96	633	86-93	438	85-26	401	7-39	376	79-9	354
86-106	603	87-8	433	7-25	401	28-8	375	90-106	353
51-11	576	121-8	432	7-10	400	41-74	375	86-16	351
91-19	528	72-86	423	82-74	397	107-8	373	20-6	349
86-11	526	86-54	418	79-48	389	28-87	373	85-15	348
86-39	478	28-54	415	86-13	385	7-12	369	85-12	346
86-17	467	31-8	412	84-104	382	7-17	367	100-72	346
86-46	465	86-10	408	84-95	380	55-106	364	86-23	346
39-7	447	55-87	408	16-8	379	7-23	364	41-86	344

List 3.3 Top-50 hero combo

3.3   Model training

Estimators, generally speaking, are kinds of functions mapping the input data with the output ones. A basic premise is that there exists relationship between the input and the output, matching certain functions. Then, the process of training model can be described as the process of determining the parameters in the functions. For example, there is a function written in the form of:

$$f(x) = w_1 x_1 + w_2 x_2 + w_3 x_3$$

Training model is actually the process of determining the parameter *w* with corresponding input *x*. Despite automatically learning the parameter *w* by estimators, term hyper parameter is defined to control the learning process. Different model training estimators require different hyper parameters, some simple one requires none. Given these hyper parameters, the estimator can start to learn the parameters from the data. For instance, LASSO is an estimator that adds a regularization hyper parameter to ordinary least squares regression, which has to be set before estimating the parameters through the training dataset. [10]
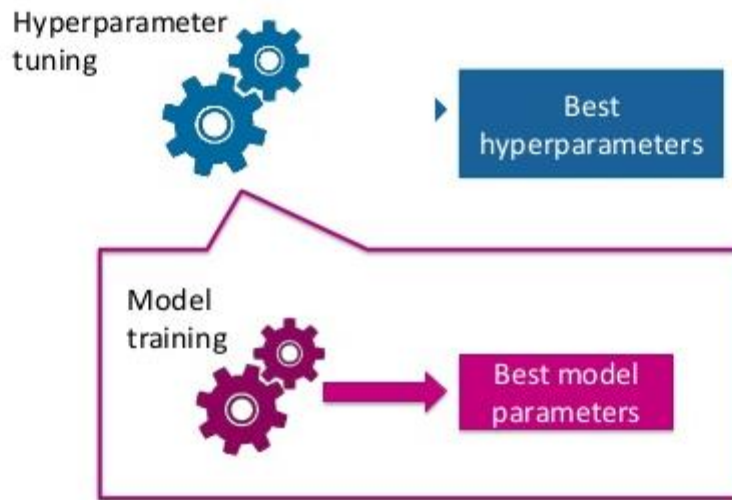

Figure 3.10 Hyper parameter [18]

In this project, three estimators, namely the logistic regression, the decision tree and the support vector machine, short for SVM are used to train the models respectively. In this chapter, it would be explained why to use these estimators, how they work like and finally the fine-tuning of hyper parameters of each of them. At last, models trained are fused together for better performance.

Since that K-fold CV is used to classify the dataset, training models and fine-tuning the hyper parameters of the estimator used are actually processed simantenously. From Chapter 3.1.3, specifically Step 6 in the inroduction of K-fold CV, mean accuracy is used to evaluate a trained model with pre-defined hyper parameters. With repeatation, different pairs of hyper parameters and the result mean accuracy are recorded. From multiple tries of hyper parameters, an optimal one would be selected which literally generates the optimal model. The exact work flow of passing estimator through training dataset and validation dataset with K-fold CV is illustrated in the following figure.
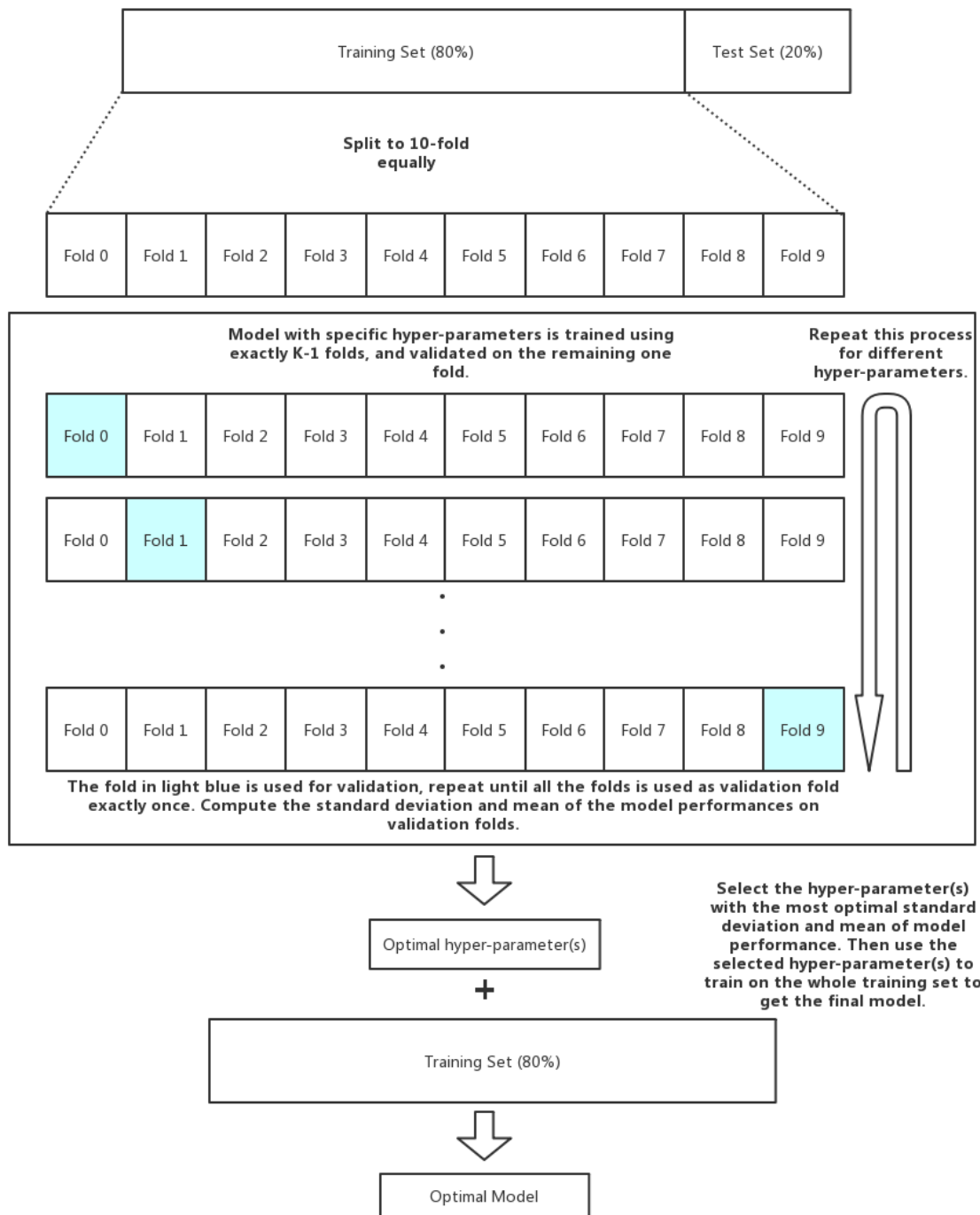
Figure 3.11 Training model with K-fold CV

3.3.1 **Logistic Regression**

Logistic regression, despite its name, is a linear estimator for classification rather than a regression one. Logistic regression is also known in the literature as logit regression, maximum-entropy classification or the log-linear classifier. In this estimator, the probabilities describing the possible outcomes of a single trial are modeled using a logistic function. The reason of applying logistic regression is that it is easier to train and implement as compared to other methods. Meanwhile, logistic regression works well for cases that is linearly separable. It is also much easier to interpret than other estimators which are rather more sophisticated since the weight parameters before each input features in logistic regression represents the importance and effectiveness of them in terms of the output prediction.

Weight Decay

When fitting data to the estimator in machine learning, a common problem is over-fitting, which means the trained model performs well on training dataset but does not generalize well on test dataset which usually means the estimator is too complex where the trained model tries too hard to fit the training dataset while in the case of test dataset. In this project, the model might over-fit probably because of too many input features.

To prevent the model from over-fitting, a widely used method in logistic regression is adding L2 regularization (also known as weight decay) to the loss function of logistic regression. The purpose of L2 regularization is to let the weight decay to a smaller value, to a certain extent to reduce the problem of model over-fitting. In practice, L2 regularization is realized by simply adding a regularization item after the original cost function.

After adding the L2 regularization item, the resulting problem of logistic regression is to minimize the following loss function, where w represents the weights of the features, C is the hyper parameter of weight decay:

$$min_{w,c}\frac{1}{2}w^Tw + C\Sigma_{i=1}^n log(exp(-y_i(X_i^Tw+c))+1)$$

When fine-tuning C, seven pairs of values are tried due to empirical exponential strategy which are listed below with the outcome accuracy of the model.

Value of C	$1e^{-4}$	$1e^{-3}$	0.01	0.1	1	10	100
Mean accuracy	0.5395	0.5459	0.5401	0.5235	0.5130	0.4823	0.4732

List 3.4 Fine-tuning Logistic regression

3.3.2 Decision Tree

Decision Tree is a graphing method of intuitively using probability analysis to evaluate project risk and judge its feasibility by forming decision tree. In machine learning, the decision tree is a predictive estimator that represents the mapping relationship between object properties and object values. The decision tree in this project is generated from algorithms ID3, C4.5 and C5.0. The process of generating a decision tree from given dataset is:

1. Calculate the entropy of every attribute a of the dataset S.

2. Partition ("split") the set S into subsets using the attribute for which the resulting entropy after splitting is minimized; or, equivalently, information gain is maximum.

3. Generate a decision tree node containing that attribute.

4. Recurse on subsets using the remaining attributes. [11]

The main challenge that a decision tree will face is to identify which feature to split upon. If the segment of dataset only contains one single class, it is considered pure. C5.0 uses the concept of entropy for measuring purity. The entropy of dataset indicates how mixed the class values are: the minimum value of 0 indicates that the sample is completely homogenous, while 1 indicates the maximum amount of disorder. The definition of entropy can be specified as:

$$\text{Entropy}(S) = \sum_{i=1}^{c} -p_i \log_2(p_i)$$

From the formula, for a given segment of dataset S, the term c refers to the number of different class levels, and p_i refers to the proportion of values falling into the class level i. For example, suppose a partition of data with two classes: red precenting 60% and white precenting 40%, its entropy can be calculated as:

$$-0.6 \times \log_2 0.6 - 0.4 \times \log_2 0.4 = 0.9709506$$

Given the measure of purity, the algorithm must still decide which feature to split upon. In order to figure out the answer, the algorithm uses entropy to calculate the change in homogeneity resulting from a split on each possible feature. This calculation is referred as information gain. The information gain for a feature F is calculated depending on the difference between entropy in the segment before the split (S_1), and the partitions resulting from the split (S_2). That is:

$$\text{InfoGain}(F) = \text{Entropy}(S_1) - \text{Entropy}(S_2)$$

One complication is that after a split, the dataset is divided into more one partition. Therefore, the function to calculate Entropy (S_2) needs to consider the total entropy across all of the partitions. In accomplishing this by weighing each partition's entropy by the proportion of records falling into that partition, which can be expressed by the following formula:

$$\text{Entropy}(S) = \sum_{i=1}^{n} w_i \text{Entropy}(P_i)$$

The higher the information gain, the better a feature is at creating homogenous groups after a split on that feature thus splitting upon it. [12]

Although Logic Regression is remarkably effective for linearly separable case, it is not the case for linearly inseparable case. Decision tree is inherently nonlinear, in other words, the decision tree can discover and learn those nonlinear patterns in the dataset. Therefore, it can do a good job of classifying nonlinear dataset. At the same time, the dataset preparation of the decision tree is quite simple. For example, it does not need data normalization, and it can also be applied to the categorical features in this project.
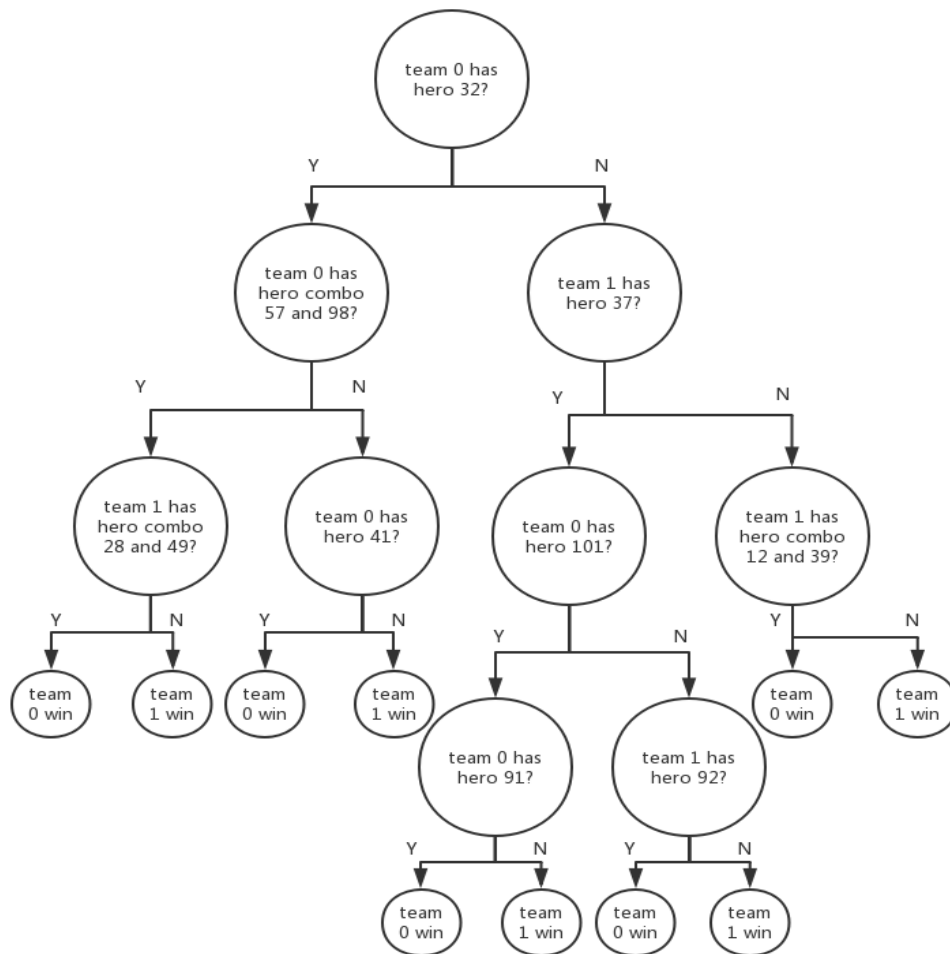
Figure 3.12 Example of decision tree with max depth 4

Decision trees tend to have more serious over-fitting problems, but the risk of it can be reduced by limiting the depth of the decision tree. As for the hyper parameter C of weight decay, the tree depth selection represents the hyper parameter in decision tree estimator. Scikit-learn API pre-defines the max depth for decision tree estimator. The value of the max depth is tried from 3 to 8 which the corresponding mean accuracy is listed below.

Max depth	3	4	5	6	7	8
Mean Accuracy	0.5435	0.5452	0.5473	0.5470	0.5351	0.5312

List 3.5 Fine-tuning Decision tree

3.3.3 Support Vector Machine

For more robust predictions, Support Vector Machine is added in addition to the previously mentioned logistic regression and decision tree. In general, Support Vector Machine is considered to be a classification approach, but it can be employed in both types of classification and regression problems. SVM is helpful when there is not much idea about the dataset. It could be used for data such as image, text, audio, etc. Moreover, it could be used for the data that is not regularly distributed or have unknown distribution.

SVM is able to handle multiple continuous and categorical variables easily. By first constructing a hyper-plane in multidimensional space, it separates different classes of the dataset. Meanwhile, optimal hyper-plane is generated in an iterative manner, which is used to minimize the errors. The core idea of SVM is to find a maximum marginal hyper-plane (MMH) that best divides the dataset into classes.
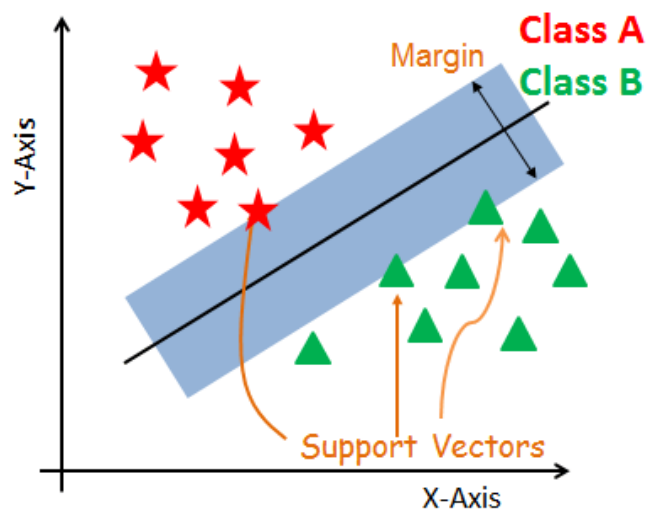

Figure 3.13 Example of SVM

Unlike the previous mentioned estimators, support vector machine generally does not suffer from condition of over-fitting and performs well when there is a clear indication of separation between classes. It could handle high dimensional data as well.

Mathematically, SVM solves the following optimization problem:

$$\min_{w,b,\zeta} \frac{1}{2}w^T w + C \sum_{i=1}^{n} \zeta_i$$
$$\text{subject to } y_i(w^T \phi(x_i) + b) \geq 1 - \zeta_i,$$
$$\zeta_i \geq 0, i = 1, ..., n$$

The SVM estimator is implemented in practice using a kernel. A kernel transforms an input data space into the required form. SVM uses a technique called the kernel trick. The kernel takes a low-dimensional input space and transforms it into a higher dimensional space. In other words, it converts non-separable problem to separable problems by adding more dimension to it. It is more useful in non-linear separation problem.

In this project, Radial Basis Function(RBF) kernel is used as shown in formula below. The Radial basis function kernel is a popular kernel function commonly used in support vector machine estimator. RBF can map an input space in infinite dimensional space.

$$K(x, x_i) = exp(-\gamma||x - x_i||^2)$$

Fine-tuning hyper parameter, namely the optimal penalty factor C, results in the SVM model with the lowest generalization error. It is similar to the case in logistic regression, the tries and mean accuracy is listed in List 3.6

| Value of C	$1e^{-4}$	$1e^{-3}$	0.01	0.1	1	10	100
Mean accuracy	0.5423	0.5435	0.5459	0.5472	0.5532	0.5235	0.5203

List 3.6 Fine-tuning SVM

3.3.4 Model fusion

From the last three chapters, among all of the estimators, over-fitting is quite common an issue occurred while training models. The fundamental reason is that the amount of data in training dataset is not enough to support complex estimators, resulting in model learning noise on the dataset. Hence the model is difficult to generalize because the model "considered" too one-sided.

Besides fine-tuning hyper parameters which is a method provdied by estimator itself to reduce over-fitting, if the results are averaged, the over-fitting phenomenon can be reduced to some extent. As shown in Figure 3.14, a single model produces a green decision boundary because

over-fitting. But, in fact, the black decision boundary has better results because it has better generalization capabilities. If multiple models were fit and averaged, the occurance of these noise points decreases because the results are evened, and the decision boundaries move slowly closer to the black line.
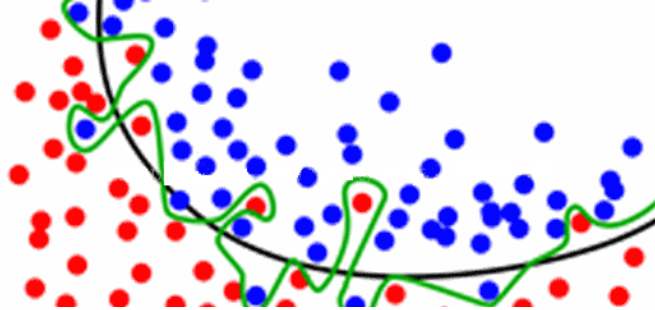
Figure 3.14 Example of over-fitting

One of the model averaging strategies is weighted average of different models. In weighted average method, each sub-model is assigned with a weight parameter to control how much it affects the result from the fused model. Different distributions of weight could have great influence on the final result of the fused model, and generally multiple weight values have to be tried to achieve the optimal multi-model fusion solution. It is like the fine-tuning of hyper parameters.

As could be obtained the probability output of the models trained respectively, model averaging method of this project is to simply distribute weights on probability outputs of the three models respectively, and predict the final result based on the averaged probabilities. Weights are also fine-tuned by performing the fused model on the final training dataset. Result with acceptably higher accuracy yields the weights of each sub-models.
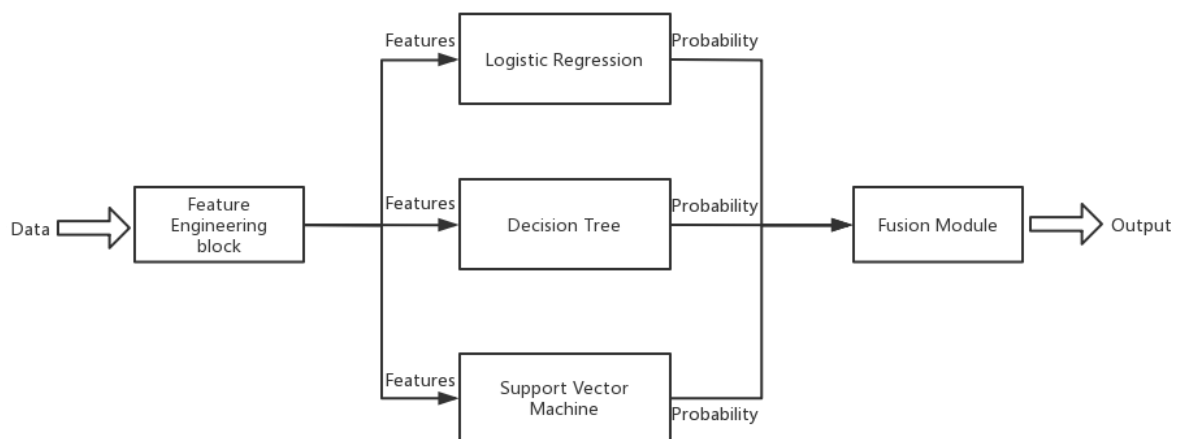
Figure 3.15 Model fusion

Unlike training models on given estimators, this time, Scikit-learn does not provide certain API for model fusion. In this project, the methodology is to implement a fusion module which

determine each weight parameter by multiple attempts on different distributions of it. Figure 3.16 shows a demo of the implementation in which three loop-functions are embedded with weights of each model stepping from 0 to 1 with an interval of 0.05.

```
for weight_lr = 0, weight_lr <= 1, weight_lr += 0.05
# logistic regression weight
  for weight_dt  = 0, weight_dt <= 1 - weight_lr, weight_dt += 0.05
  # decision tree weight
    for weight_svm  = 0, weight_svm <= 1 - weight_lr - weight_dt, weight_svm += 0.05
    # SVM weight
    fused_result = weight_lr * lr_result + weight_dt * dt_result + weight_svm * svm_result
    # fused result
```

Figure 3.16 Fusion module

Then the fused result is compared with the actual result on the final training dataset. Weight distribution which yields the highest accuracy is adopted as the final weight for each sub-model. The result is shown as follow.

	Logistic regression	Decision tree	SVM
Weight	0.35	0.25	0.4

Table 3.2 Weight for each model

3.4 Flask application

Now that model has finished training, it is possible to build up Flask application to make use of the trained model. This chapter would then explain the process of constructing Flask application. At the beginning, it is necessary to first get an overview on the architecture of a typical Flask application which is the MTV architecture. The whole application would be built up on the basis of it.

3.4.1 MTV vs. MVC architecture

Chapter 2 discusses a lot on the architecture of this project. In a typical Flask application, it is taken further by the MTV architecture which works well with object-oriented programming. Programs are split into three aspects and each of the aspects can be treated as an object thus improving reusability within an application.

However, the MTV architecture is actually evolved from the original MVC architecture in order to accommodate development environment of a web application. In this chapter, the original MVC architecture is introduced beforehand so that the concept of MTV architecture could be understood much easier.

MVC (Model-View-Controller)

MVC is first introduced as a software design pattern, commonly used for developing user interfaces that divides the related program logic into three interconnected elements. It has been widely adopted as a design for web applications as well. [13]
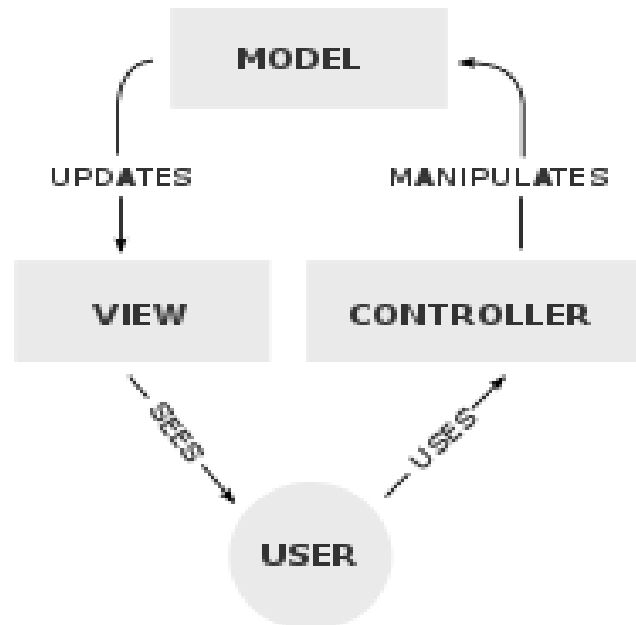

Figure 3.17 MVC architecture [13]

From Figure 3.17, Model section serves for central component of the architecture. It is the application's dynamic data structure, independent of the user interface. It directly manages the data, logic and rules of the application.

View section is primarily responsible for two jobs:

(1) One is collecting data transferred from Model and encapsulating them.

(2) One is generating web page displayed to the users. Any representation of information such as chart, diagram or table is generated in View section.

Controller section deals with requests and responses, taking care of the interaction between Model and View in the way that Controller responds to the users' input on the View pages and performs interactions on the data Model objects.

MTV (Model-Template-View)

While in the case of web frameworks, for example Flask, MVC architecture differs from their interpretations, mainly in the way that the MVC responsibilities are divided between the client and server. [13]
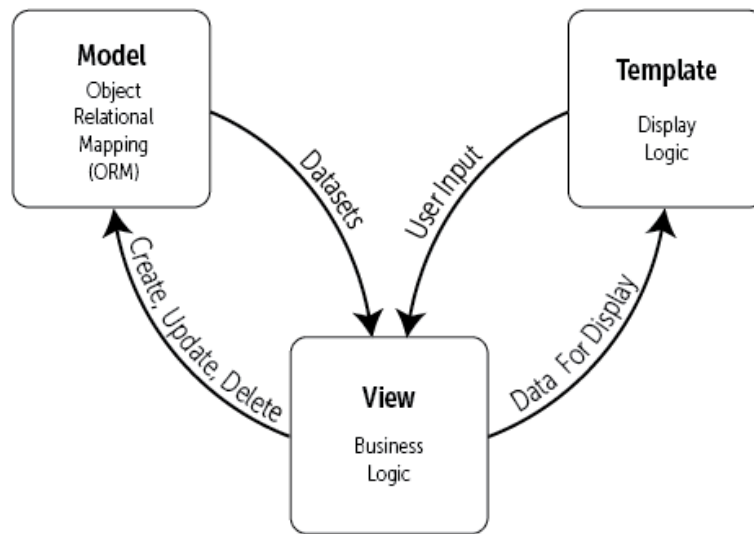
Figure 3.18 MTV architecture [19]

In MTV architecture, the definitions of Model still remains the same that is, Model section in MTV is responsible for handling data between database and View. Model always provides a definition of how the data formats as coming from the view so, it stores in the database and vice versa, i.e., the retrieving information from the database transfers to the View in the displayable format. [14]

Template inherits the functionality of (2) in the View section in MVC which is generating web pages for the users. It consists of all of the front-end system, including HTML, CSS and JavaScript parts.

Likewise, View inherits the functionality of (1) in the View section in MVC which is handling data transferred from Model. It serves for defining relationship between URLs and their corresponding callback functions. Each View stands for a simple Python function which is implemented with routing syntax of the web framework.

As for the Controller section in MVC architecture, its duty is actually done by the framework itself which involves the URL distributer integrated in the web frame work. The duty of URL distributer handles with the request, distributeing them to different URL routed in View section and gather the response return by Model section if necessary (for router that does not refer to data calling, the Model section would not be involved.).

With the knowledge of MTV architecture, the previous architecture of this project described in Chapter 2.3 can be updated as Figure 3.19. The following chapters would discuss the Flask application from its Database to its MTV parts respectively.
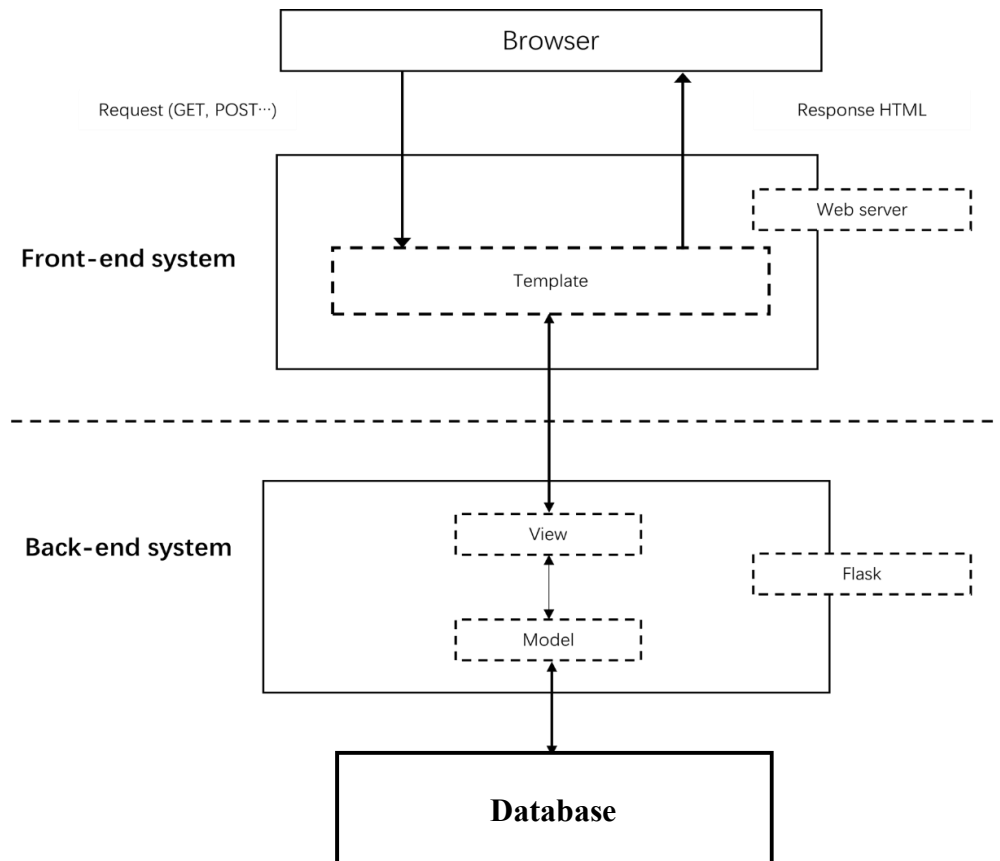
Figure 3.19 Architecture of this project 2.0

3.4.2 Substitution of Database

In Chapter 2.3.3, it is discussed that in this project, database is probably unnecessary to be implemented. Database mainly serves for connection with Model, providing data requested and receiving data responded. However, in this project, no data is received from Model to update the database. Sole responsibility for database is to provide prediction result for Model section. Therefore, it is considered sufficient to substitute it with static files.

One of the static files should encapsulate the model trained into certain form available to be referred to. Since the final model used is the fused version of sub-models generated respectively by three different estimators, a wrapper model is developed. To be specific, the wrapper model contains a 'predict' API that accepts original features and returns the prediction result by calling respective sub-models and adding weights before the results from sub-models. In order to deploy to Flask application, the wrapper model is instantiated and serialized to Python pickle format which is an easy-to-use format for all kinds of application.

However, only the wrapper model is not enough for the application since the features in this project are all collected from OpenDota in the form of hero id whereas hero id cannot remind the users of the corresponding hero. Thus, another static file is implemented for hero mapping from hero id to hero name and vice versa. Figure 3.20 is an example of hero mapping rule

obtained from OpenDota in JSON format where element id stands for the hero id and name stands for the hero name corresponding to its id.

```
1  {
2      'id': 1 ,
3      'name': Anti-Mage ,
4      'attack_type': Melee ,
5      'roles': ["Carry","Escape","Nuker"]
6  }
```

Figure 3.20 Example of hero mapping

Besides, the element attack type classify heroes by their attack range in the game. The value melee represents that this hero, namely Anti-Mage, fights against enemy in a close range while the value ranged represents a ranged hero. The roles of a hero stand for the official division of labor of the hero. Despite the fact that only id and name of hero is needed by now, each of the other elements in the mapping rule is considered useful as for the hero portrait. All of them might be useful in the future, for example, when screening function is implemented in the application. Thus, data of all of the 119 heroes is collected and stored.

3.4.3 **Model**

As mentioned previously, wrapper model can merely handle features as its input. Therefore, the first function of Model section is to convert input data to its feature. To reshape the input data, the hero id and top-50 hero combo is defined globally as list variables in Model section. The hero ids could also be obtained from hero mapping file but was considered unnecessary since more functions would have to be defined in that case.

```
combine_list = [(86, 8), (86, 96), (86, 106), (51, 11), (91, 19), (91, 72), (86, 11),
hero_ids = [1,2,3,4,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,25,26,27,28,29,
```

Figure 3.21 Global variables in Model

After that, a 'data_to_feature' function is implemented to convert the input data into its feature. The input data is considered in the form of ten hero ids. 'data_to_feature' module would reshape it into form of a sample out of the feature matrix described in Chapter 3.2 with additional hero combo feature.

To make sure the correct form of input, an extra data validation module is implemented to validate the input from different degrees:

(1) Whether the input is composed of ten integers instead of other data type.

(2) Whether the integers is included in the list of hero ids.

(3) Due to domain knowledge, input with duplicate integers would also return false because

DOTA2 does not allow duplicate heroes appearing in a single match.

Another part implemented in Model is the hero mapping, function is defined to map the relationship between hero id and hero name since other elements are not yet used in the application. Two variables are defined as dictionary type to store the data. The implementation is showed in Figure 3.22. The 'hero_mapping' returns hero id when given hero name and the 'inverse_hero_mapping' returns hero name when given hero id.

```python
def load_hero_mapping(mapping_file_folder):
    with open(mapping_file_folder, 'r') as fp:
        hero_meta = json.load(fp)
        hero_mapping = dict()
        inverse_hero_mapping = dict()
    for hero in hero_meta:
        hero_id = hero['id']
        hero_name = hero['name']
        hero_mapping[hero_name] = hero_id
        inverse_hero_mapping[hero_id] = hero_name
    return hero_mapping, inverse_hero_mapping
```

Figure 3.22 Hero mapping in Model

3.4.4 Template

Since this project simulates the hero picking process in the game and provides service based on that, the Template of the website is designed to be as similar to the actual hero picking interface in DOTA2 as possible which would be quite familiar for players of DOTA2. It makes the whole website more self-explainatory. Besides, two main functiosn which are the prediction and the recommendation funtions are put into two simple buttons in the center for users to check out after heroes picked.
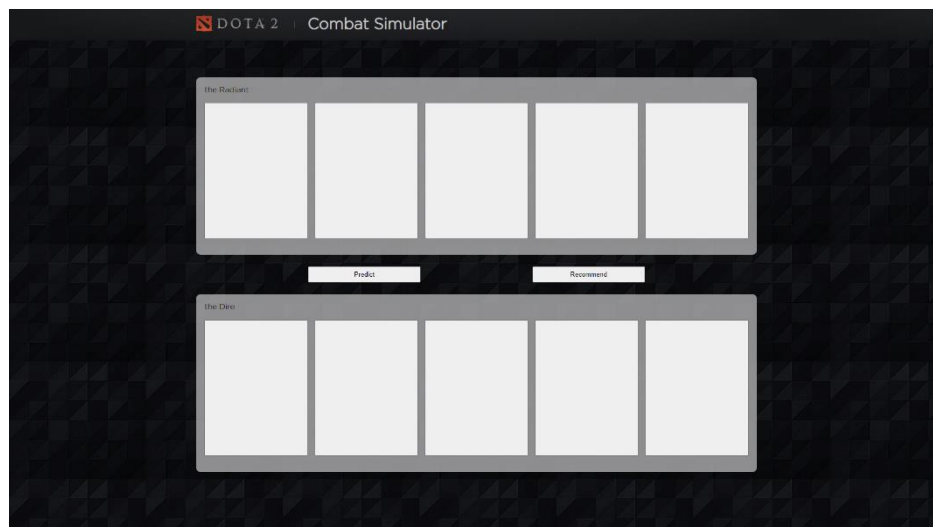
Figure 3.23 Template of the website

In comparison, the actual hero picking interface in DOTA2 is illustrated in Figure 3.24. The main characteristic of the interface is that two teams are split into two sides horizontally while ten players are put into ten small windows. Thanks to development of Graphic Users' interface, picking hero would directly change player's window with the icon of the hero. The Template of this project imitates the designing pattern of DOTA2 interface but needs more decoration to look more alike to it.

Figure 3.24 Hero picking interface in DOTA2

Figure 3.26 Drop down menu

To make picking heroes event available in terms of the browser, a drop-down menu is implemented in the Template as Figure 3.25. Once users single click on each player's window, the menu would be shown on the top of the website. Then, Clicking on any of the entries in the drop-down menu would lead to change on the player's window. Figure 3.26 shows an example that changes the player's window when user picks hero named **Chaos Knight (CK)**. Simantenously, the pick of hero is recorded by the Template for follow-up usages by the View section discussed afterward.

Figure 3.25 Picking CK

However, it is possible that users might have already picked ten heroes when they decided to reset one of them in order to check for the recommendation. For this reason, the player's window is made sensitive to double click event which represents the reset function. Single click of the player's window calls the drop-down menu while double click resets the hero if picked. Main challenge faced by the reset function is that the double click events always fire the single click events first rather than directly calling the reset function.

There are two ways to fix this. One of which is to implement new UIs responsible for resetting the picked heroes. It is considered making the website too complex adding more UIs to it. Thus, the second way is introduced which is placing a slight delay function between the single click and the double click events.

The implementations are briefly shown below. 200ms are close to the default double clicking speed on Windows OS. In this way, double clicking the player's window would never fire the single click event again.

```
switch (click_time) {
    case 1:
        click_timeout = setTimeout(function () {
            showMenu(pick_button)
        }, 200);
        break;
    case 2:
        resetHero(pick_button);
        break;
}
```

Figure 3.27 Implementation of double click event

3.4.5 View

View section is realized on the basis of routing syntax in Flask. This project possesses a quite simple routing module which mainly reacts with the two buttons in the Template. Figure 3.28 shows the router for predict button in the Template, URL is first defined in the decorator with the constraint of the requesting methods. This project select POST method for both two routers because that, among all of the requesting method, POST is considered the most secure method. Data sent by POST method are safer because it would not be disclosed as a part of the URL, nor would it be cached by the browser.

```
@app.route('/predict', methods=['POST'])
def predict():
    # do check to validate data input
    valid, res = valid_input(list(request.json))
    if not valid:
        return res
    else:
        feature = data_to_feature(res)
        prob = model.predict_proba(feature)[0]
        # prob: probabilities
        ret_val = dict()
        ret_val[0] = prob[0]
        ret_val[1] = prob[1]
        return ret_val
```

Figure 3.28 Implementation of predicit router

For easy debugging, the input data has already been converted from hero name to hero id in the Template. Therefore, the callback function in predict router only operates the data validation function on the input. Once the input is validated, it is sent into 'data_to_feature' function and the wrapper model would return the probabilities of two teams. Figure 3.29 shows an example of the output probabilities from the wrapper model. Each of the float variables stands for probabilities of corresponding team.

`{0: 0.49092923354313245, 1: 0.5090707664568676}`

Figure 3.29 Output of prediciton

To be more striking, the containers of two teams on Template would also change their background-color accordingly to the result. It is shown in Figure 3.30.
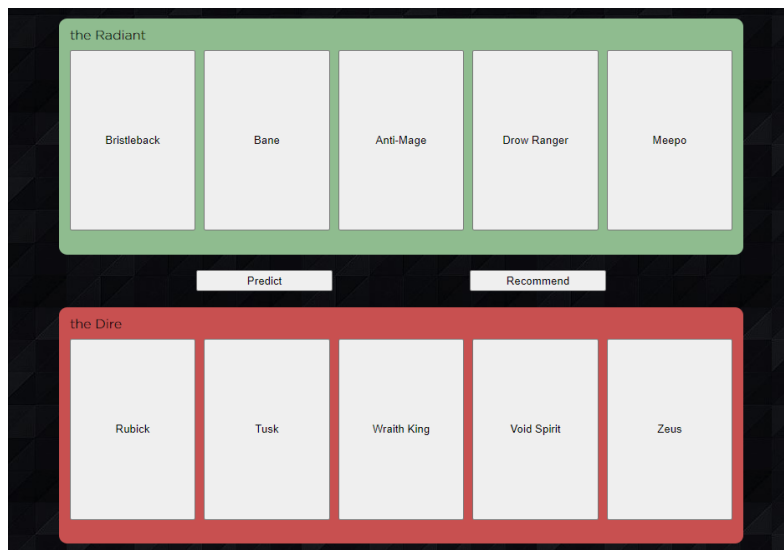
Figure 3.30 Output of prediction on Template

For the case of recommend router, due to the limitation in wrapper model which demands ten picked heroes to predict the result, by now, it is only available to recommend the last hero not picked. In the callback function, input data is operated on an additional data validation process which is written to assure the input pertaining an unpicked hero whose value is set to be default -1. The implementation is shown in Figure 3.31.

```python
def recommend():
    idx = -1
    raw_data = list(request.json)
    for i, id_str in enumerate(list(request.json)):
        if id_str == -1:
            idx = i
            break
    if idx == -1:
        return "ERROR: illegal input."
```

Figure 3.31 Data validation for recommend

After that, the input is filled with hero id that does not occur in the input in replace. Then, the new input is sent into the wrapper model. By traversing through all of the hero ids which have not yet occurred in the input while comparing their probabilities returned, an optimal pick of hero is computed representing the recommendation. Figure 3.32 illustrates the detailed implementation of the traversal function.

```
predict_side = 0 if idx < 5 else 1
hero_2_prob = dict()
max_prob = 0
recommended_hero_id = -1
for hero_id in hero_ids:
    raw_data[idx] = str(hero_id)
    valid, current_data = valid_input(raw_data)
    if not valid:
        continue
    feature = data_to_feature(current_data)
    prob = model.predict_proba(feature)[0,predict_side]
    hero_2_prob[hero_id] = prob
    if prob > max_prob:
        recommended_hero_id = hero_id
        max_prob = prob
ret_val = dict()
ret_val['hero_id'] = recommended_hero_id
ret_val['hero_name'] = inverse_hero_mapping[recommended_hero_id]
return ret_val
```

Figure 3.32 Implementation of traversal function

Console the name of the recommended hero on the website yeilds outcome in Figure 3.33 in which Spectre is one of the hero name in DOTA2. The outcome in front-end system is by now only displayed on an alert window thus obmitted in the thesis.

```
Spectre                                                          main.js:98
```

Figure 3.33 Output of recommendation

4 Evaluation

The previous chapter clarifies the detailed approaches of training model and designing Flask application. To examine whether each part of them works well on practice, the first part of this chapter would be using accuracy, ROC/AUC to evaluate model on the test dataset. For the second part which is the Flask application, white-box test is arranged on Model section while usability is tested on the outcoming website.

4.1 Performance analysis

Throughout the whole thesis, term performance is repeatedly used to describe the ability of the model to predict the output from the input. In machine learning, definition of performance varies from different degrees. General metrics includes evaluating model with its accuracy, precision and recall ratio etc. In this project, accuracy is used for describing the performance of the model trained. One of the main reasons is that the concept of accuracy is quite simple to understand.

To compute the accuracy of a model, confusion matrix is introduced which comparing the predicted results from the model with the actual results from the test dataset in the form of a matrix:

Figure 4.1 Confusion matrix [20]

As is illustrated, in this project, a dualistic matrix is sufficient in describing the status of both predicted results and actual ones with 0 meaning failure of radiant while 1 meaning victory. Then, it is defined in the confusion matrix that:

(1) True Negative (TN) stands for predicted negative with failure of match.

(2) True Positive (TP) stands for predicted positive with victory of match.

(3) False Negative (FN) stands for predicted positive with failure of match.

(4) False Positive (FP) stands for predicted negative with victory of match.

With these four variables, accuracy of a model is calculated by:

$$Ac = \frac{TP + TN}{TP + TN + FP + FN}$$

Apparently, accuracy means 'accuracy' of model predicting true results accounting both the 'True Positive' and 'True Negative'. But, error would happen in circumstance where the test dataset is not well curated. For example, the test dataset is extremely filled with 90% percent of failure samples. Then, it would be 90% accuracy if the model is set to only give negative prediction. Therefore, the test dataset must be balanced before the start of evaluation. In this project, test dataset is curated as shown below. It is considered fairly balanced.
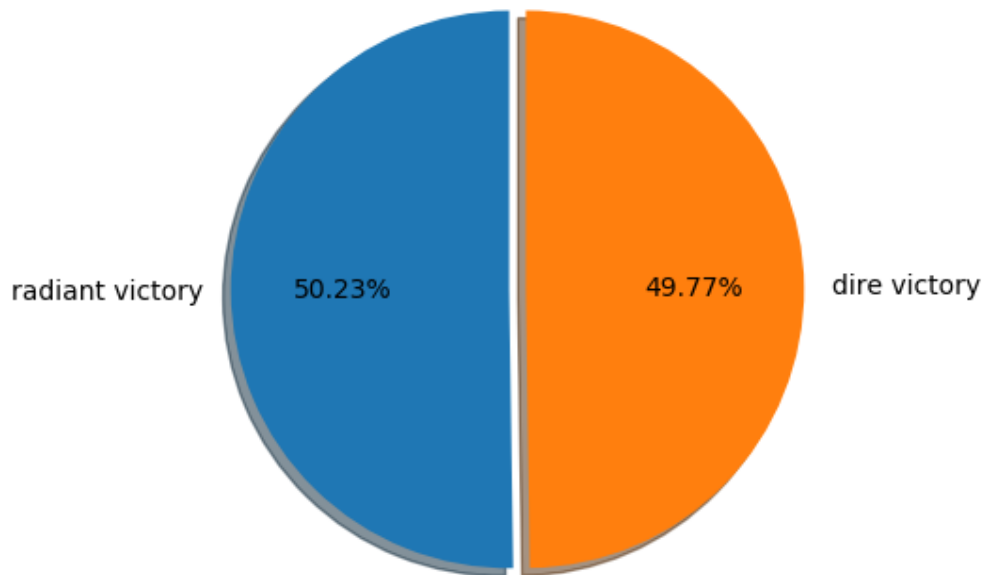
Figure 4.2 Balance of test dataset

Besides accuracy, ROC/AUC is introduced to fix the problem of unbalanced test dataset. A ROC curve is constructed by plotting pairs of the true positive rate (TPR) against the false positive rate (FPR). Both of the TPR and the FPR are calculated by variables from the confusion matrix as well:

$$TPR = \frac{TP}{TP + FN}$$

$$FPR = \frac{FP}{FP + TN}$$
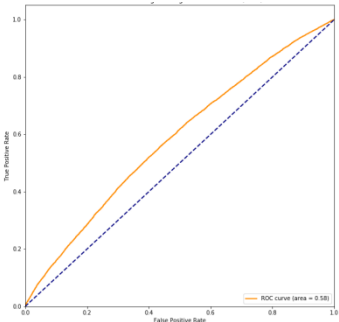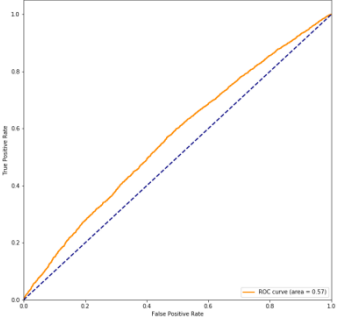
The denominator of fraction in these two formulas indicates that the TPR and FPR are based respectively on the actual results, which means both of them evaluate the performance of the model from the actual positive and negative samples in the test dataset. This is the reason that, no matter how unbalanced the samples of test dataset are, the result of TPR and FPR would not be influenced anyway.
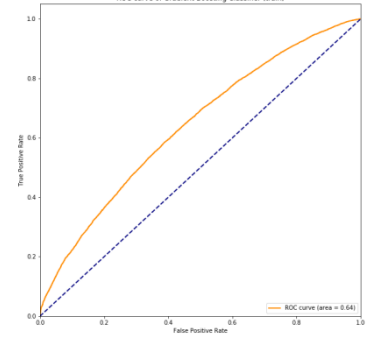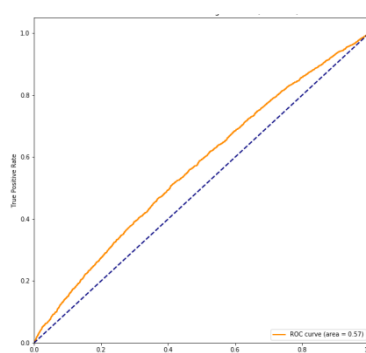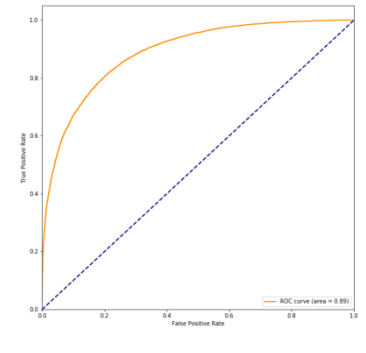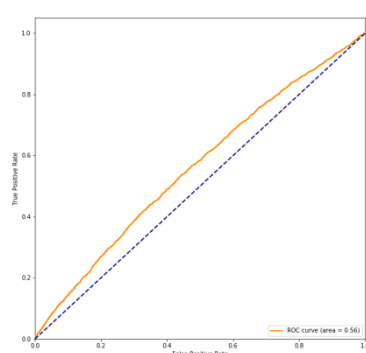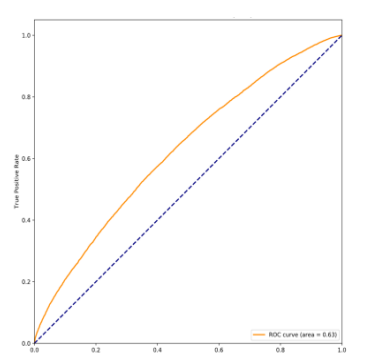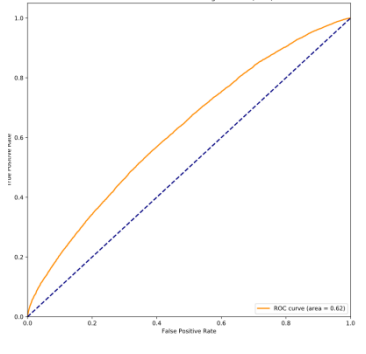
Still, as for a dualistic confusion matrix, threshold is set to classify the predicted probabilities. For example, given a threshold 0.6, probabilities predicted by logistic regression that is larger than 0.6 would be classified to positive class and vice versa. Since the threshold ranges from 0 to 1, for each value of threshold, there is a pair of TPR and FPR. The ROC curve is plotted by traversing through all thresholds.

Not only logistic regression returns with values that could be classified by threshold, all three estimators used in this project return values which stand for the probabilities or win rate in other words. Therefore, ROC curve is a common measure for evaluating the performance of model in this project,

Back to the definition of TPR and FPR, TPR defines the ratio of TP over all positive samples. FPR, on the other hand, defines the ratio of FN over all negative samples. With TPR valuing as higher as possible and FPR valuing as lower as possible, the performance of model would be the best resulting in a steepest ROC curve. Plot the diagonal curve on the diagram, the area under ROC curve is defined as AUC. Apparently, the values of AUC represent the performance just like the gradient of ROC but it is easier to compute.

As mentioned in Chapter 3.3, model trained would be applied again on the whole training dataset to yield the generated model. Meanwhile, it would be applied on the test dataset to see how it works in unfamiliar environment. During both process, the performance of models are recorded for comparison in List 4.1.

Model	Accuracy	Train ROC & AUC	Test ROC & AUC
Logistic Regression	0.5473		

Decision Tree	0.5502		
SVM	0.5549		
Fused Model	0.5815		

List 4.1 Evaluation based on accuracy and ROC/AUC

From the list, it reveals that all of the three models reach the accuracy higher than 50% which means the performance of them is fine to be used in real world environment. Among all, SVM has the best performance supported by both the accuracy and the ROC/AUC. However, all of the models trained perform bad on the test dataset with remarkable decrease of AUC. Likewise, SVM has the worst generalization ability since it decreases the most. By model fusion, this problem is solved to a great extent. The fused model performs well both on the training dataset and the test dataset with its AUC values approximately 0.6 which is quite promising in the circumstance of predicting the result of a game match. Also, the accuracy is improved to about 0.58 after model fusion.

4.2   White-box test

White-box test is a method of software testing that tests logical structure of an application, as opposed to its functionality (i.e., black-box testing). [15] As for the situation of the Flask application in this project, white-box test is arranged for the Model section where most of the functions are defined. Talking of black-box test, it is not arranged because the goal of setting up a black-box test is to evaluate the functionality of an application. As for the situation of a web application, functionality is considered to be tested in the part of usability test which would be discussed in the next chapter.

In Model section, white-box test would be applied to verify the data validation module to check whether it works correctly. The implementation of data validation module is illustrated as Figure 4.3. The logical structure flows in 4 steps, thus 4 groups of use case are designed.

```python
def valid_input(raw_data):
    try:
        raw_data = [int(hero_id) for hero_id in raw_data]
    except:
        return False #'ERROR: hero id must be digit.'
    if not len(set(raw_data)) in [10, 20]:
        return False #ERROR: duplicate hero id.'
    for h in raw_data:
        if not h in hero_ids:
            return False #'ERROR: invalid hero id.'
    return True
```

Figure 4.3 Data validaiton

	Use case 1	Use case 2	Use case 3	Use case 4
Try	false	true	true	true
If		false	true	true
If			false	true
return				true

Table 4.1 Design of use case

Implementation of the test unit is in Figure 4.4 with the use case tailored for firing certain logic branch in the data validation function:

(1) Use case 1 contains element with type of string.

(2) Use case 2 contains element with duplicate value.

(3) Use case 3 contains element 5 which are not recorded in hero mapping function.

(4) Use case 4 is the correct version of input which should pass through all of the logic branches and return true.

```python
def test_data():
    test_cases = [
        ['1', 1, 2, 3, 4, 10, 6, 7, 8, 9], # use case 1
        [1, 1, 1, 1, 1, 1, 1, 1, 1, 1], # use case 2
        [1, 2, 3, 4, 5, 6, 7, 8, 9, 10], # use case 3
        [1, 2, 3, 4, 6, 7, 8, 9, 10, 11] # use case 4
    ]

    expected_results = [False, False, False, True]

    for i in range (len(test_cases)):
        print (valid_input(test_cases[i]))
```

Figure 4.4 Implementation of test unit

The output of the test unit is as follow. Apparently, the output matches the expected results of input with all 4 groups of use case, indicating that the logic structure of data validation function when requestion prediction is fine.

```
False
False
False
True
```

Figure 4.5 Output of test unit

4.3   Usability test

Usability test is a technique used in user-centered interaction design to evaluate a product by testing it on users. This can be seen as an irreplaceable usability practice, since it gives direct input on how real users use the system. It is more concerned with the design intuitiveness of the product and tested with users who have no prior exposure to it.

The timing of arranging such usability test is divided by:

(1) Exploring stage: at the very beginning of designing the product, to test the enforceability of thoughts and the acceptance of users towards the product. Test might be arranged on the static designing draft which demands low fidelity.

(2) Evaluation stage: when the comparably integrative prototype is already implemented, test is arranged to check if users could easily reach the goal or accomplish the task. The testing device should be prototype of the product claiming low UIs' integrity but thorough functionality.

(3) Validation stage: As soon as the product is almost finished but before production, to test if its quality matches marketing standard and reaches users' expectation. Test device demands a productive archetype.

(4) Comparison stage: For identical function, several approaches have been tried however not determined which to use. At this point, test is arranged on the product to find the solution that seemingly the optimal one.

(5) Improvement stage: During iteration, test is held to check the efficiency of users' finishing the task. Otherwise, it is to check if product needs to be modified according to marketing requirements. By this time, test is held on the final product which is market oriented. [16]

From developer's point of view, this project is still on the stage of evaluation at present where only functional prototype is ready to be tested. Therefore, usability arranged aims to check if users are able to reach the goal of this project.

Since usability test is arranged to test the website interface, the goal of this project is described in a different way which is to use this website for predicting match result of DOTA2 and recommending pick of hero. For the tested users, three scenarios are planned:

(1) Scenario 1: You are an amateur player of DOTA2. By the time you open this website, you wish to pick some of the heroes just like you do in the game.

(2) Scenario 2: With the knowledge of how to pick heroes on this website, now you would like to simulate a combat and see which team would win against another team.

(3) Scenario 3: At this time, you are playing DOTA2 on your own. It is time for you to pick your hero. However, you do not know exactly which hero to pick. Now you want recommendation of which hero to pick.

	Purpose	Pay attention to
Scenario 1	1. User is able to pick heroes.	1. Click the picking button. 2. Select hero from the drop-down menu.
Scenario 2	1. User is able to pick ten heroes. 2. User is able to check prediction.	1. Pick ten heroes. 2. If guided to validate input.. 3. Reaction on the prediction result.

Scenario 3	1. User is able to pick nine heroes. 2. User is able to check recommendation.	1. Pick nine heroes. 2. If guided to validate input. 3. Reaction on the recommendation.

Table 4.2 Design of scenario

Based on this, a questionnaire is handed over among target users of this website. Since the website is still on the stage of evaluation, not too detailed users' portrait is defined. But in order to make the result more representative, only users who are previously experienced in DOTA2 are selected to evaluate the website through pre-defined scenarios. Accounting that this project is completed with one man's effort. It is extremely hard to arrange usability test on a large scale of target users. All of the ten target users are personally invited from people surrounding. The answer in the questionnaire is split into 7 gradients ranging from strongly satisfied to strongly unsatisfied with the accomplishment process through each scenario. The result is shown in Table 4.3.

	1	2	3	4	5	6	7
Scenario 1		3	5	2			
Scenario 2		2	4	3	1		
Scenario 3	2	5	2	1			

Table 4.3 Result of usability test over scenario

As a prototypical website, the result is fairly acceptable since most of the users vote on gradient 3 or 4 for the scenarios. For insight into the result of the usability test, investigations are conducted to each of the user.

The condition for scenario 1 is that eight out of ten users reflect on the Template of this website with the hero picking interface in DOTA2. Therefore, most of the users know exactly how to pick a hero. However, the low gradient voted is mainly due to inconvenience of the drop-down menu. It is considered lacking sort of filtering functions. It is previously mentioned in Chapter 3.4.2 that there are numbers of ways to filter heroes in DOTA2 such as the attack range or the role of them. Users reflecting on the website suddenly got disappointed when seeing only heroes' name in the drop-down menu instead of other message like their icons. They felt annoying to the present drop-down menu. It needs to be improved by implementing heroes' icons and a filter function with any of their features.

The condition for scenario 2 is that most of the users knew that they should pick ten heroes. For those who did not know, the validation function guide them well. Thus, the result for this scenario is fine. Users only got upset due to the simplicity of prediction result. Barely telling which team would win is too lacking. Users thought that there should at least exist the win rate of two teams which is actually existed in the back-end system, it should be displayed on the website in the future.

Most of the problems occur in scenario 3. Users tend to think that the website should recommend a hero to them no matter how many heroes are already picked. They did not know in advance that the website would only provide recommendation when nine heroes are picked. Meanwhile the validation function did not guide them as well. It only tells that the input is invalid without giving users hints to validate their input or reset the hero picked. On contrarry, the output of recommendation seems fine. Users are satisfied with the final pick of hero recommended by the website. For the two parts which should be improved, it is easier to start with the validation function. More tailored functions should be implemented to guide the user of how to check the recommendation. In addition, it is necessary to add hint messages to the website in the future which would also solves the problem to some extent. The tough one is how to improve the recommendation function itself. Introduced in Chapter 3.4.5, the recommendation function by now is actually a traversal function which merely runs the wrapper model for dozens of times. It is unfeasible to recommend hero when heroes picked are less than nine. To improve that, the whole logic structures of both the recommendation function and the model training algorithm would have to be scrapped and started all over again. This is considered almost impossible for the time being.

5 Conclusion and Outlook

Development of machine learning technology makes it possible for prediction on certain events such as the win rate of soccer, NBA matches. This thesis put into practice to see if it is also feasible to apply the technology on gaming, specifically on DOTA2 game matches. If it is also feasible, then, amateur players of DOTA2 could have a chance to enjoy the same treatments as professional players. Machine learning technology could take the responsibilities as a coach to the players.

In this thesis, data is obtained through open API website of DOTA2, trailing with pre-processing, classification and feature engineering in order for model generalization. Three estimators are applied using Scikit-learn to generate models after training, validating and testing.

- **Logistic regression** estimator is in charge of linear separable features of the dataset, providing fine prediction as an outcome.

- On the contrary, **decision tree** estimator is implemented in case the input features pertain nonlinear pattern to be learnt.

- For robustness of output, **SVM** estimator is added in the end which is capable of discovering underlying pattern when there is not much clue about the feature of dataset.

Meanwhile, this thesis tried model fusion as a solution to model over-fitting issue which is proved effective. Fused of model is realized with manual implementation of a traversal function to determine the weight distribution of the three sub-models. The performance of models are evaluated with their accuracy, ROC/AUC value.

To make use of the generated model, Flask application is developed in order to visualize the model in World Wide Web. The developing process follows architecture of MTV to split front-end system with back-end system. Though the outcome website remains at the stage of evaluation which means it is still prototypical, its usability is tested to make sure the functionality of the website works fine.

For future development, improvement could be made from following aspects.

- Most importantly, more data from the API could be collected for two main usages.

  (1) With more data, players' skill level could be concerned, resulting in classification of matches held in different skill levels.

(2) Since DOTA2 has different game modes besides merely picking ten heroes for the match, more data collected yields more game modes concerned as well.

- Meanwhile, the performance of the model could be enhanced. Although the performance now is acceptable, higher performance is better as it may. To enhance the performance of the model, more features could be added. Currently, only combination of two teams and the hero combo are concerned. In the future, it is able to take hero counter relationship into account as well. Some of the hero is extremely powerful when versus another due to domain knowledge of DOTA2.

- For now, users' characteristics are not considered. Yet in the future, by classifying users' portrait, recommendation could be made more precisely to meet the requirements of different users. For example, user might be good at certain heroes while bad at others. The recommendation could be made based on that.

- Finally, for production, the appearance of the website could be improved. Website up to now provides ready-to-use functionality but is lack of UIs and Users' guide. Style of UIs is yet to be determined while guideline of the website is also necessary to implement. The website would need several iterations to be production-ready.

Appendix A – List of figures

Appendix B – List of listings

Appendix C – List of tables

Bibliography

[1]   W. Dunkley, "Split Stack Development: A Model For Modern Applications," 7 Jun 2016. [Online]. Available: https://medium.com/@Future__Friendly/split-stack-development-a-model-for-modern-applications-d7b9abb47bd5. [Accessed 15 Jan 2021].

[2]   P. JOSHI, "Automated Feature Engineering | Feature Tools Python," 22 Aug 2018. [Online]. Available: https://www.analyticsvidhya.com/blog/2018/08/guide-automated-feature-engineering-featuretools-python/. [Accessed 15 Jan 2021].

[3]   Wikipedia, "Data mining - Wikipedia," [Online]. Available: https://en.wikipedia.org/wiki/Data_mining. [Accessed 15 Jan 2021].

[4]   Scikit-learn, "Getting Started — scikit-learn 0.24.1 documentation," [Online]. Available: https://scikit-learn.org/stable/getting_started.html. [Accessed 15 Jan 2021].

[5]   Flask, "Foreword — Flask Documentation (1.1.x)," [Online]. Available: https://flask.palletsprojects.com/en/1.1.x/foreword/. [Accessed 15 Jan 2021].

[6]   Flask, "Quickstart — Flask Documentation (1.1.x)," [Online]. Available: https://flask.palletsprojects.com/en/1.1.x/quickstart/. [Accessed 15 Jan 2021].

[7]   Django, "Django at a glance | Django documentation | Django," [Online]. Available: https://docs.djangoproject.com/en/3.1/intro/overview/. [Accessed 15 Jan 2021].

[8]   Django, "URL dispatcher | Django documentation | Django," [Online]. Available: https://docs.djangoproject.com/en/3.1/topics/http/urls/. [Accessed 15 Jan 2021].

[9]   OpenDota, "OpenDota - Dota 2 Statistics," [Online]. Available: https://www.opendota.com/. [Accessed 19 Oct 2020].

[10]   Wikipedia, "Hyperparameter (machine learning) - Wikipedia," [Online]. Available: https://en.wikipedia.org/wiki/Hyperparameter_(machine_learning). [Accessed 18 Jan 2021].

[11]   Wikipedia, "ID3 algorithm - Wikipedia," [Online]. Available: https://en.wikipedia.org/wiki/ID3_algorithm. [Accessed 18 Jan 2021].

[12]   C. Yobero, "RPubs - C5.0 Decision Tree Algorithm," 9 Jul 2018. [Online]. Available:

https://rpubs.com/cyobero/C50. [Accessed 18 Jan 2021].

[13] Wikipedia, "Model–view–controller - Wikipedia," [Online]. Available: https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller. [Accessed 18 Jan 2021].

[14] R. Gour, "Working Structure of Django MTV Architecture," 16 Apr 2019. [Online]. Available: https://towardsdatascience.com/working-structure-of-django-mtv-architecture-a741c8c64082. [Accessed 18 Jan 2021].

[15] Wikipedia, "White-box testing - Wikipedia," [Online]. Available: https://en.wikipedia.org/wiki/White-box_testing. [Accessed 20 Jan 2021].

[16] 小曼 D, "如何从 0 到 1 进行可用性测试(Usability Test)," 知乎, 25 Aug 2020. [Online]. Available: https://zhuanlan.zhihu.com/p/159871425?utm_source=wechat_session. [Accessed 22 Jan 2021].

[17] Dataprofessor, "Building the Machine Learning Model," 12 Jan 2020. [Online]. Available: https://www.youtube.com/watch?v=BOk1hlCPW0c. [Accessed 17 Jan 2021].

[18] Prabhu, "Understanding Hyperparameters and its Optimisation techniques," 3 Jul 2018. [Online]. Available: https://towardsdatascience.com/understanding-hyperparameters-and-its-optimisation-techniques-f0debba07568. [Accessed 18 Jan 2020].

[19] DjangoBook, "Django's Structure – A Heretic's Eye View - Python Django," [Online]. Available: https://djangobook.com/mdj2-django-structure/. [Accessed 18 Jan 2021].

[20] Packt, "Confusion Matrix - Applied Deep Learning with Keras," [Online]. Available: https://subscription.packtpub.com/book/big_data_and_business_intelligence/97818385 55078/6/ch06lvl1sec34/confusion-matrix. [Accessed 20 Jan 2021].