### Introduction

### Motivation

E-sports, unlike the old days, is grabbing more and more audience these years due to the rise of live broadcasting, streaming and other factors. The rapid development not only raises the rewards of winning first title in the championship but also enhances players’ cognition of how to win the game. Gone are the days when pure skills and passion are enough to take the victory, especially when it comes to Multiplayer Online Battle Arena, namely the MOBA games. MOBA games often start with a team with five players picking their choice of heroes against another team.

Among hundreds of heroes, there could have tens of thousands of combinations in which some heroes combining together might be more powerful while some might be the exact counter pick of other. This is one of the reasons which makes MOBA games rather dependent on strategic thinking and precise tactical execution.

To practice this sort of ability, he who strives for achievement needs an assistant outside of the game, searching, analyzing data and developing tactics, most likely a full-time coach. A team manager is able to employ a coach for the team while there are millions of players who cannot employ their personal coach to look after their gameplays. Apparently, for those who only play the game for fun but want to improve their performance as well, it is necessary to develop an application which aims to provide the same assists for them.

### Goal

This thesis selected DOTA2 which is considered the most typical MOBA game in the market due to its diversity of heroes’ combination and in-game strategies. Since DOTA2’s authentication policy bans access of external links to in-game data (which all of the online games would do), this thesis put emphasis on exploring and revealing the relationship between heroes’ combination and the result of the game.

In this thesis, previous match data is collected from the open API website of DOTA2 [1]. After feature engineering [2], algorithms of machine learning are applied on the dataset to generate a reliable model representing the relationship between heroes’ combination and the result of the game. In this thesis, a web-based application is considered to be the container to finally carry out the model. The first usage of the model is to predict the result of the game when the 10 picked heroes are given beforehand while the second usage of it is to offer its users certain recommendation according to its knowledge learnt from machine learning algorithms when users are confused of which hero picking.

### Organization

The rest of the thesis is structured as follows. Chapter 2 states the technical background of machine learning and web framework. 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 web framework describes the framework used in this project, namely Flask, comparing with its related work, namely Django.

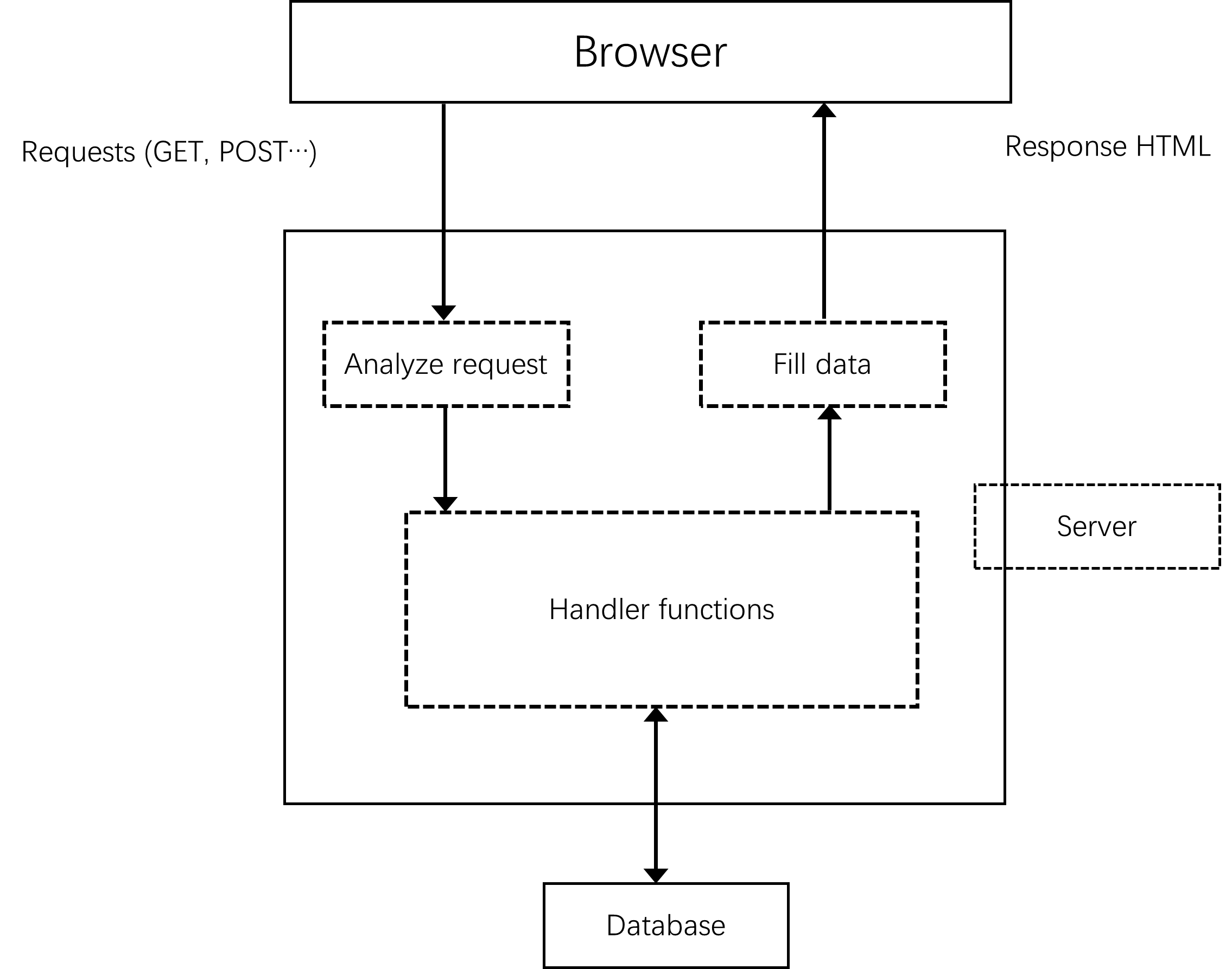
Chapter 3 focuses on the approaches and implementation details during the development of the actual project. Data collection module written in Python is provided which obtains the original data. The pre-processing and classification of the original data yields dataset to be passed into feature engineering trailing with model training. After that, the encapsulation of generated model is described. This chapter ends with revealing the design of Flask application.

Chapter 4 evaluates the performance of the generated model as well as the Flask application. The evaluation of the model is arranged in terms of ROC&AOC to check the precision of each algorithms applied. The part of Flask application is divided into two parts which are the Python programs and the outcome of final website. White box test is arranged on Python programs while the website is tested by the usability of it.

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

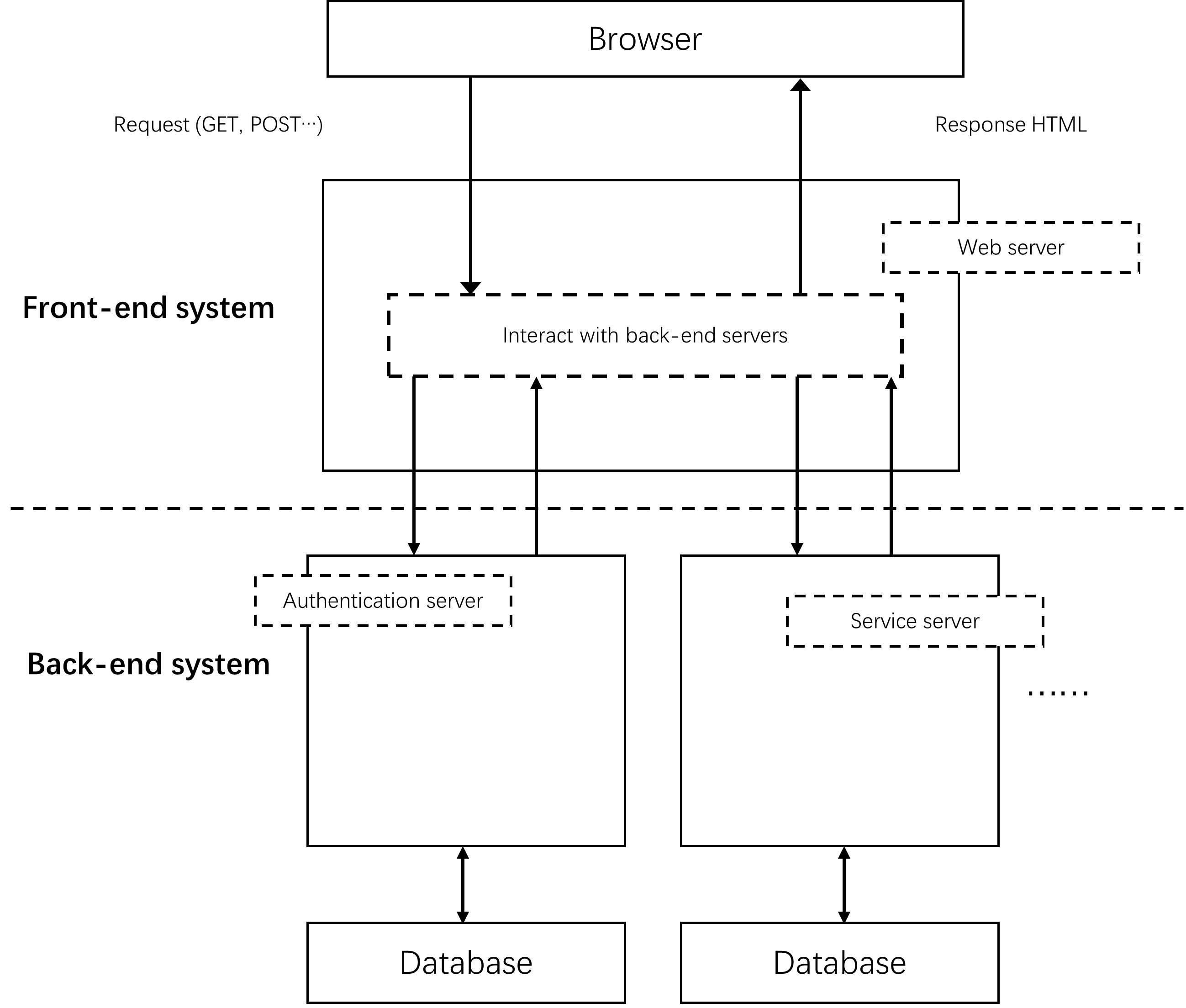
### Background and Related work

### Split stack development

Since this thesis aims to develop a web-based application which integrate the model generated by machine learning methods 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. With a full stack developer, the final website might work like this.

It is actually quite harmful for the afterward codes management and project maintenance because the part of front-end and the part of back-end are all putting together. Organizations which design systems are constrained to produce designs which are copies of the communication structures of these organizations [3]. 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 this project in which the front-end system and the back-end system are separated. According to split stack development, the final website would work in a fresh new flow like this.

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.

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, integrating it with an old back-end part, and upgrade each independently further on.

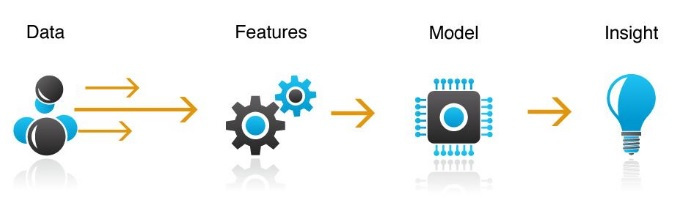
### Machine learning

Machine learning ,as a phenomenal field these years, is too complex to start explaining with. In substitution, this thesis pick 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 methods to yield results.

The difference lies on respective point of emphasis of them. 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 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 precision in order to ensure the predictiveness of it.

### Feature engineering

Before applying machine learning algorithms to generate the model, feature learning is the very beginning step of processing the dataset. With definition of appropriate features, it can improve the performance of the model generated by certain algorithms applied.

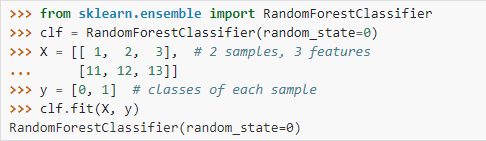
Feature engineering uses domain knowledge to extract features from dataset via data mining techniques.

Domain knowledge depends on detailed project. For example, this project aims to explore and reveal the relationship between heroes’ combination and the result of the game in which heroes’ 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 heroes’ combination. Data mining 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. Detailed data mining techniques used would be introduced exhaustively later in Chapter 3.2.

### Scikit-Learn

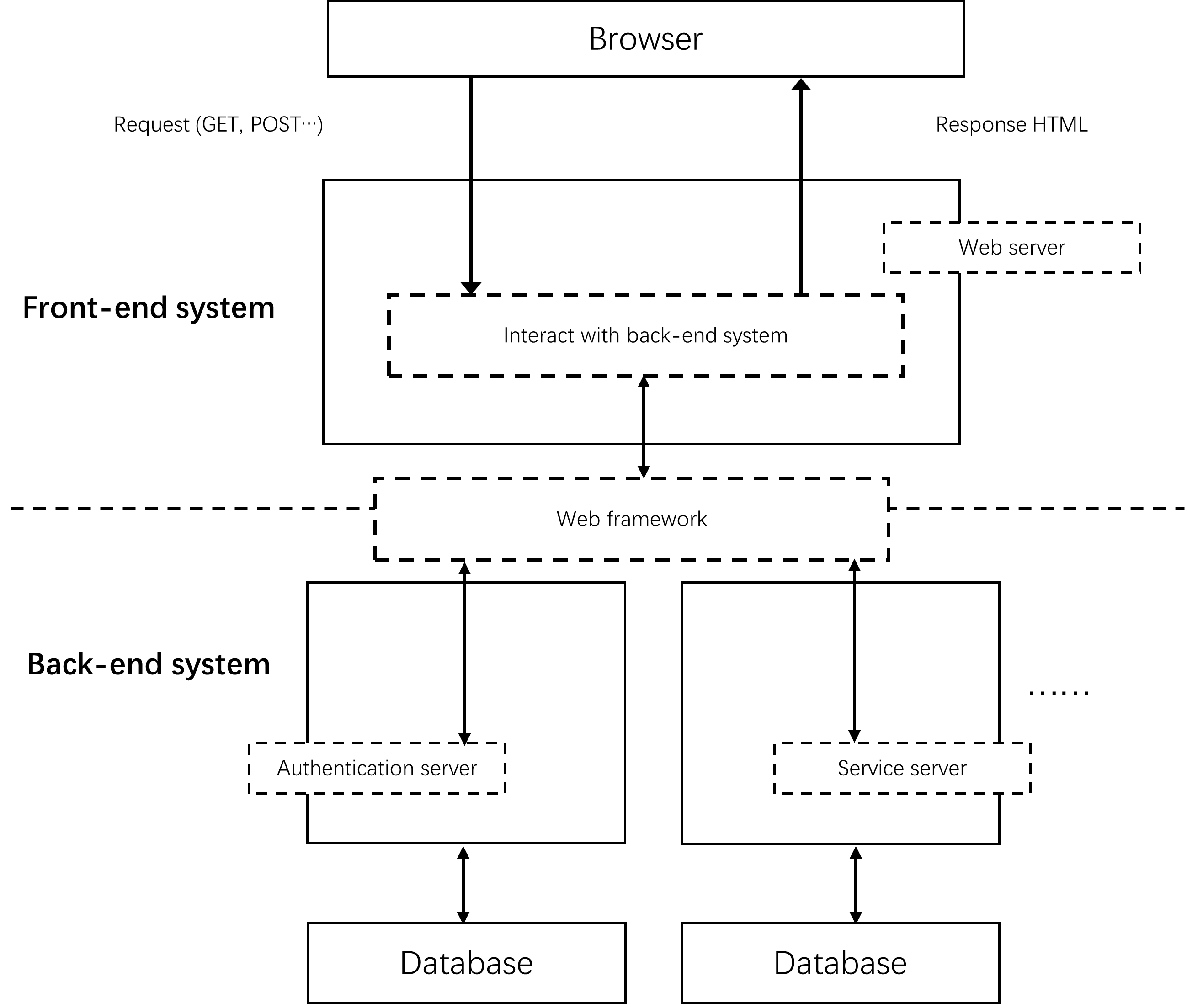
After feature engineering, the dataset is ready to be passed into machine learning algorithms. Instead of implementing algorithms manually, Scikit-learn is introduced in this project. Scikit-learn is an open-source machine learning library largely written in Python because that Python provides powerful 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.

Scikit-learn provides dozens of built-in machine learning algorithms and models. They are called the estimators in Scikit-learn. For the subsequent chapter, term estimator is used substituting machine learning algorithms and models. After selection of estimators, data can automatically fit to the estimators using fit method in Scikit-learn. An example is shown below:

Once the estimator is fitted, it can be used for predicting target values of new data. This process is actually a miniature of the generation of model. By iteration, estimator is being fine-tuned thus yielding more reliable model with better performance. The detailed selection of estimators would be discussed in Chapter 3.3.

### Web framework

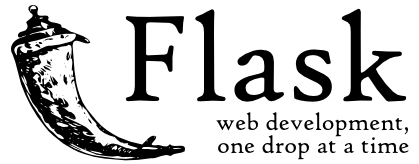
As mentioned in the figure of split stack development in Chapter 2.1, one of the components in the front-end system works for interacting with the back-end system. The front-end system deals with interactions by creating URLs and send requests to them. In circumstances that the front-end system interacts with the back-end system, each URL is the path to access the corresponding functionality.

Front-end system uses methods like Ajax to create URLs and requests. In terms of back-end system, a web framework is needed to do the job of binding responses with URLs. The mechanism called routing decides which functionality to call based on the provided URL. For subsequent quotes, the figure of split stack development in Chapter 2.1 is updated like below. Call it the architecture of this project.

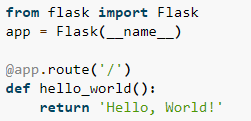
Instead of building up a self-made framework, this project decides to adopt ready-made framework in order to avoid unimportant issues and make it easier for others to practice on the project. Meanwhile, only frameworks written in Python are considered due to consistency with the back-end system which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

### Flask

Flask is a micro [web framework](https://en.wikipedia.org/wiki/Web_framework) written in [Python](https://en.wikipedia.org/wiki/Python_(programming_language)). Micro does not mean that the whole web application has to fit into a single Python file, nor does it mean that Flask is lacking in functionality.

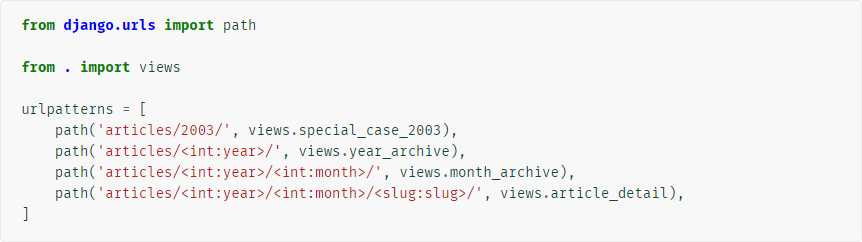
However, micro in micro framework means that Flask aims to keep the core simple but extensible. Developers using Flask are able to make decisions by themselves instead of by the framework. Flask only makes decision of what templating engine it use, which is also easy to be changed according to requirement.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

Flask uses its ‘decorator’ syntax to route the front-end request with the back-end response on a pre-defined URL. A small demo is displayed below showing how easy it is to associate the client and the server with Flask’s routing syntax.

From the figure, it is shown that when the front-end requests URL ‘/’ which is by default the home page of a website, the flask will return ‘Hello, World’ to the client.

### 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. 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, routing in Django looks like the figure below.

A request to ‘/articles/2005/03/’ would match the third entry in the list. Django would call the function.

‘/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

.

### Conclusion

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 weighs the highest in this project. 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, the front-end coding is able to do the job of data validation. And since there is no login interface while only heroes’ information in the game are needed to be tracked, database is considered unnecessary to be implemented by now. 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 the connection of it with Flask.

Also, since the project would probably be based only on one page with one or two functions requesting data from back-end system (let’s say predict function and recommend function), the module for routing would not be quite large. In this case, the powerful routing syntax of Django is fairly 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.

### Approaches and Implementation

This chapter starts with the creation of dataset. Then, heroes’ 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. This chapter also specifies the process of packaging the model generated so that it could be available for Flask application to access. The end of this chapter update the architecture of this project with MTV architecture as well as the implementations details of Flask application on the basis of architecture.

### Dataset creation

Given in Chapter 1.2, DOTA2 is selected to be the target MOBA game in this thesis because that its operator company, Valve, provides web API for third party developers and researchers, including API for public match data query.

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

### Data collection module

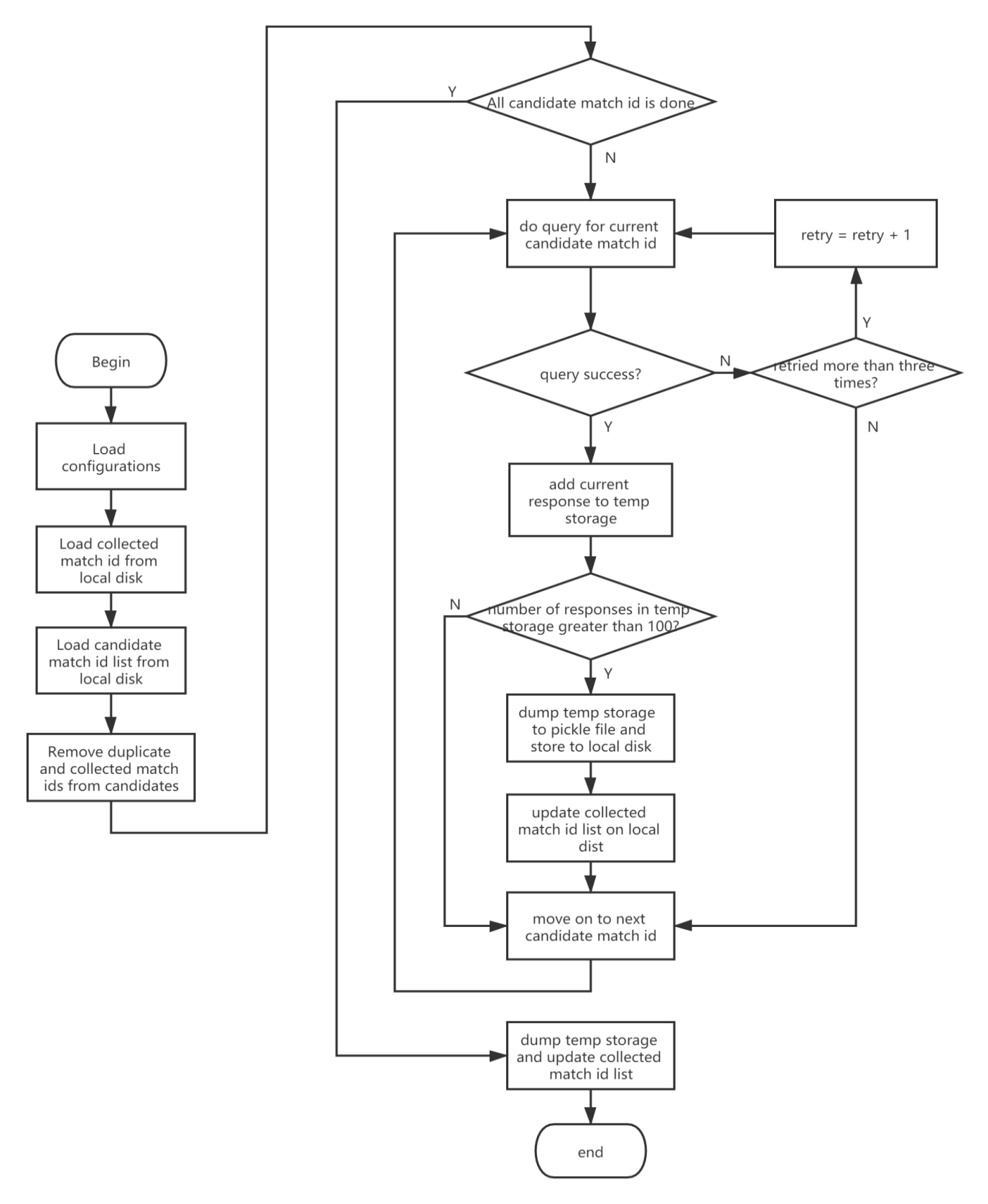
In order to first obtain the data from the public API, a raw data collection module was built by Python. The collection module is mainly composed of three parts, namely, the main module, the API request module and the data storage module.

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 Open Dota 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.

### 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 piece of the raw data has the format as follow.

|  |  |
| --- | --- |
| 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 |

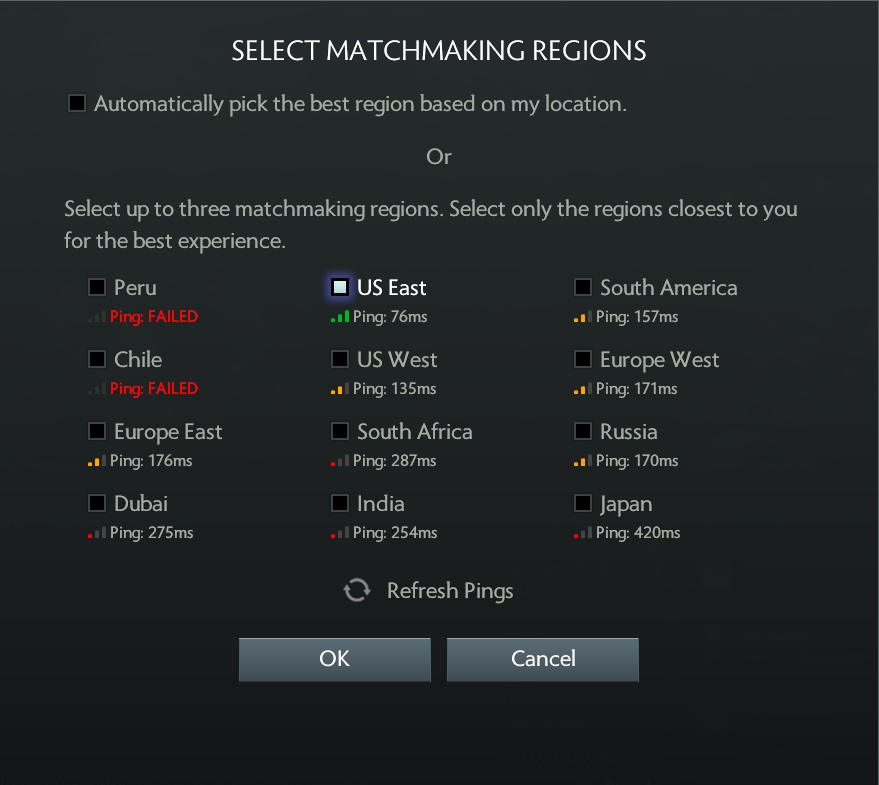
Apparently, not all of the elements seems relevant to the result of the games while not all of the data seems to be reliable either. Hence, a pre-processing module is implemented realizing following functions:

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 responsible for verifying the integrity of the collected data as well. Data lacking any of the information is believed unreliable thus being filtered out.

Following paragraphs explain the logic of serializing and filtering the raw data, arranged 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.

**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**

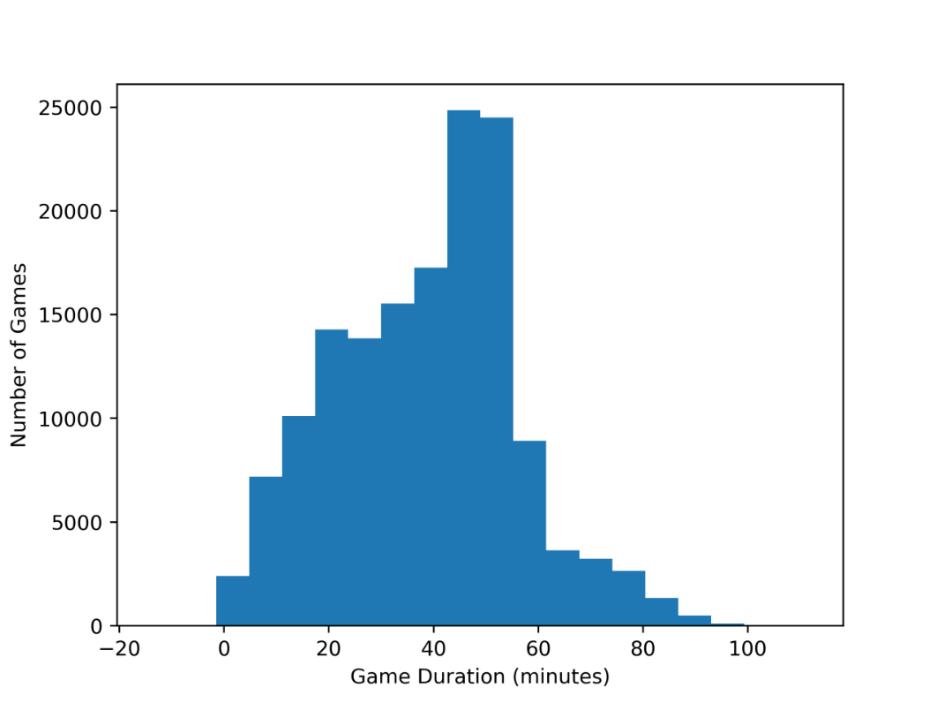
Element match id serves for indexing the match in data collection module which is introduced before. It is deleted since data with repeating match id was already filtered out which is the job done by the data collection module.

**Match version**

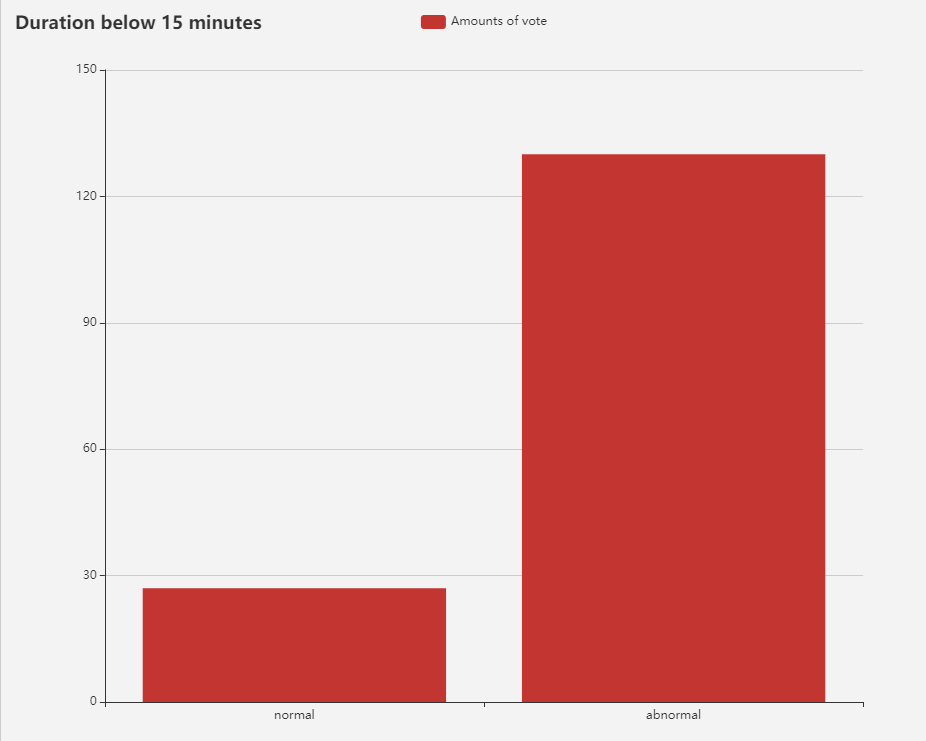
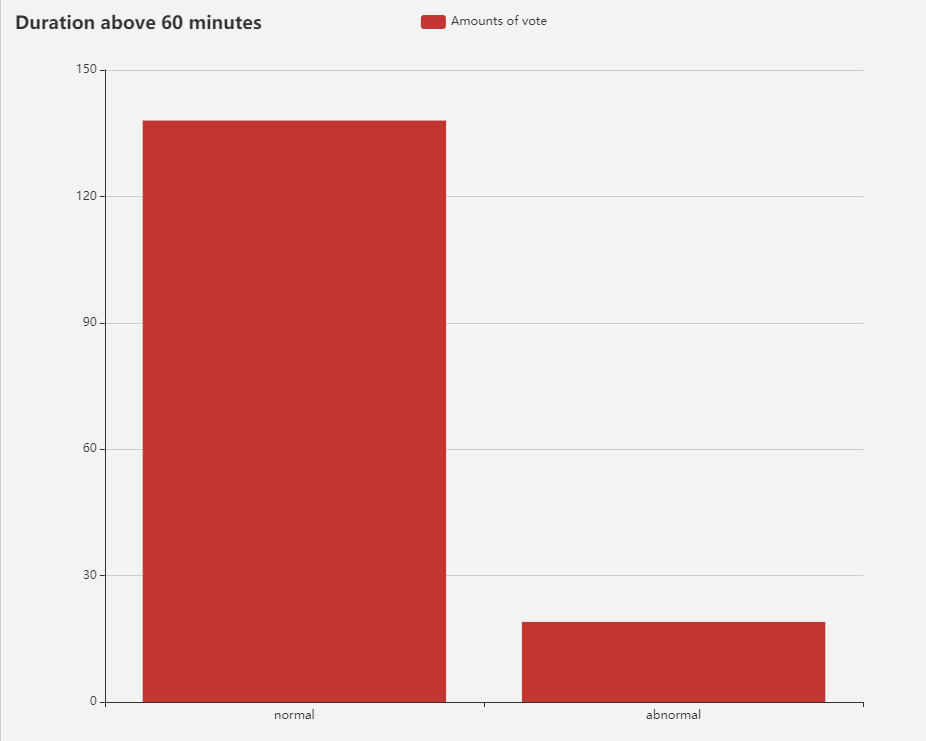
The version of the game is varying over time. In this project, only entries from the latest version is persisted in order to make the model generated up to date, other data are filtered out. For this reason, the model would have to be trained again once the version has updated. Then, it is deleted because 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 heroes’ 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:

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:

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.

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.

After pre-processing, about 50,000 entries are persisted serialized into dataset like:

|  |  |
| --- | --- |
| 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 |

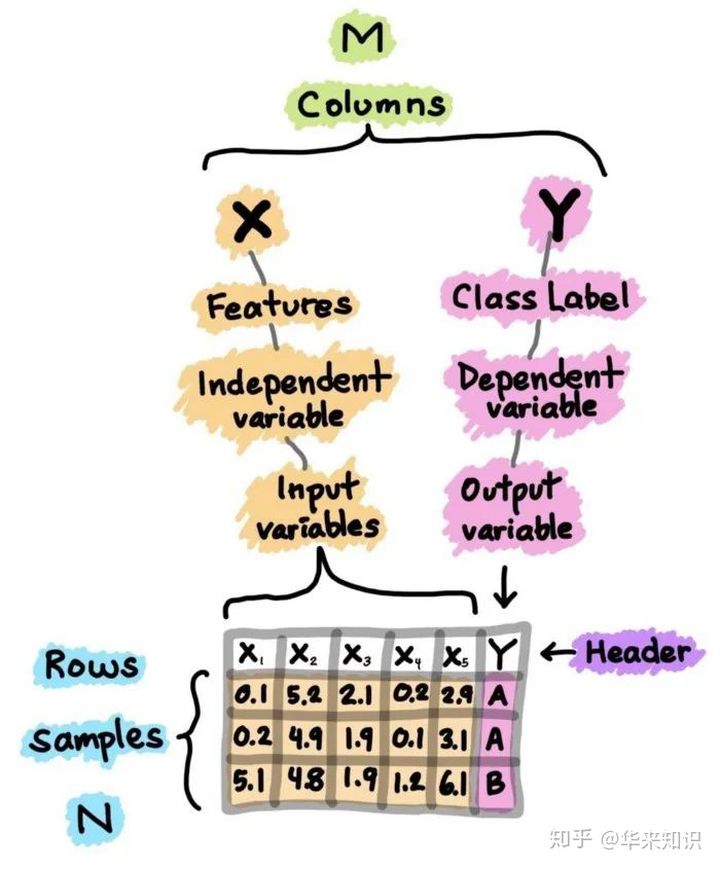
In the table, the players’ picks of heroes is re-named by adding their team name before. Team 0 stands for team of the radiant while team 1 stands for team of the dire. In the end, the value of element label represent the result of the game, replacing the Boolean element radiant win. Integer 0 means the victory of the radiant and Integer 1 means the victory of the dire.

### Dataset classification

After pre-processing, the dataset gained can be described as a matrix where M is the columns of variable and N is the rows of samples. In machine learning, variable of the dataset consists of two components *X* and *Y* which, respectively, stands for the input and output. Figure below illustrates the matrix and shows some of the substitutive terms of the input and output:

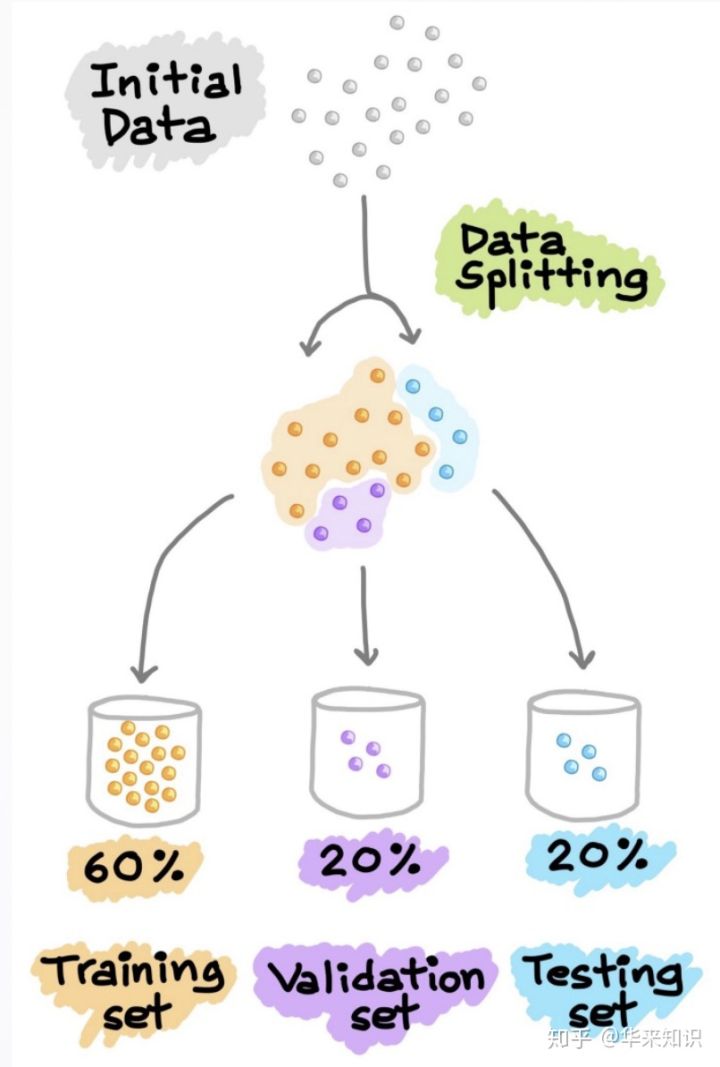
In this project, apparently, X stands for the heroes’ combination and Y is the result of the game. Before applying estimator on dataset to train the model, the dataset still needs classification into three different subsets for subsequent usability. Subsets are composed of the training dataset, the validation dataset and the test dataset. The reason of doing classification is to improve the performance of the model trained.

Detailed regulation of classification is described as follow:



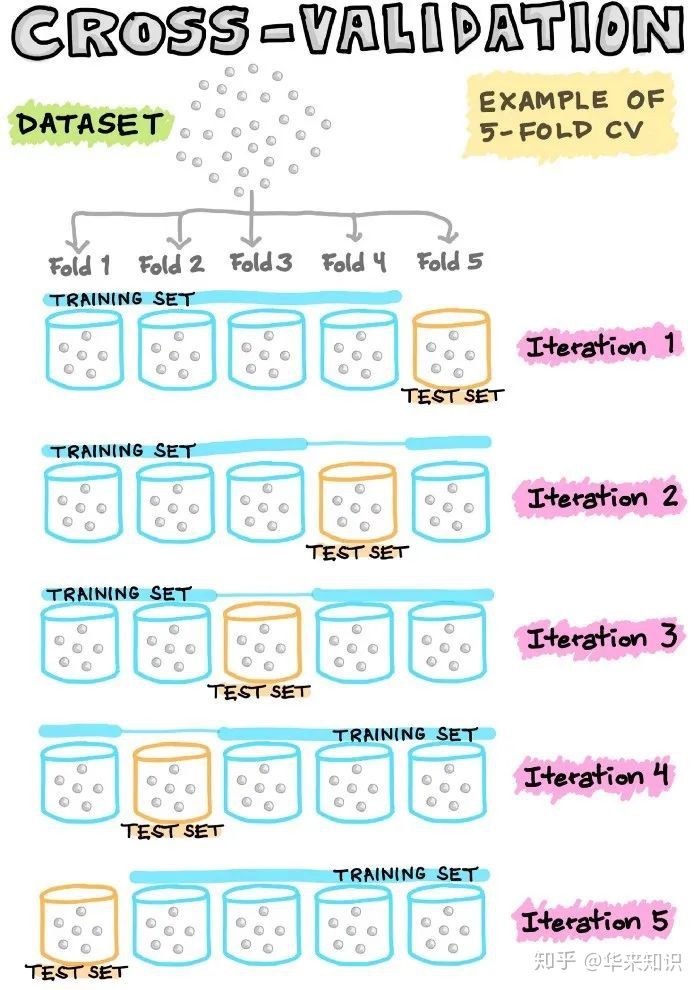
1. Training dataset is the subset used to fit the estimators, which means the estimators could observe the dataset and would try to learn the underlying pattern and extract knowledge from it, thus generating models trained.
2. Validation dataset is used to evaluate a given estimator, and usually is used 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.

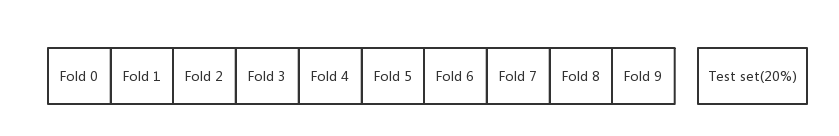
Because that only 50,000 entries is persisted, formatting into dataset in this project, splitting training dataset and validation dataset is operated on the basis of K-fold cross validation in order to maximize the use of sample.



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-folds (most often into 5 or 10 folds).
2. Out of the K-folds, (K-1) folds is used for training, the left one is used for validation
3. The model is trained with training dataset (K-1 folds) 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-folds 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 considering all of the performance of the models recorded in step 4.
6. Step 3 to Step 5 is repeated for fine-tuning the estimator. The approaches of fine-tuning the estimator would be introduced in Chapter 3.
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

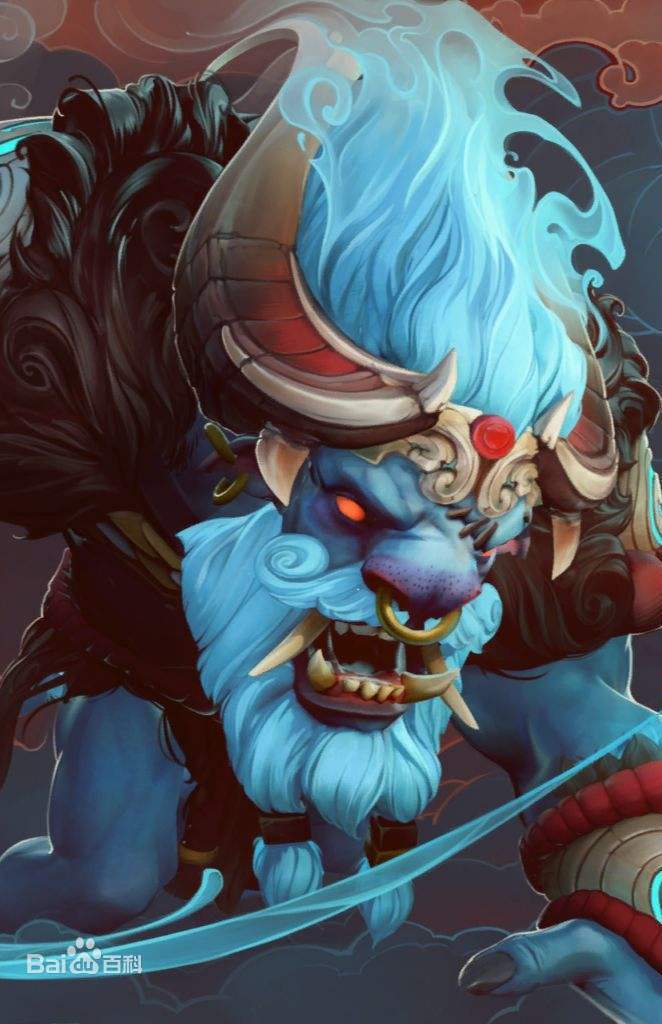


The entire dataset in this project is divided into training and test dataset at a scale of 80% and 20% since the size of it is comparably small. At the same time, the training dataset are divided into 10-folds randomly according to 10-fold cross validation process.

### Feature engineering

Although, the original feature is defined as heroes’ combination in the dataset which has a dimension of 10, predictive modeling of game results with inputting only heroes’ combination often does not lead to an acceptable performance. As is introduced in Chapter 2.2.1, more features should be added in order to improve the performance of the model generated.

According to DOTA2 domain knowledge, in addition to heroes’ combination as a whole team, the combined effect between lesser heroes can also have a great impact on the outcome of the game. Take two of the heroes, namely Spirit Breaker and Bounty Hunter for example, are known as a killer combo in the game. Teams with these two heroes tend to have great advantages.



By the time this thesis is written, DOTA2 has up to 119 heroes available in the game. Let’s define heroes’ combo which consists of two heroes as a new feature to the model. If all heroes’ 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 (if one-hot encoding were used). 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 be able to degrade the dimension, dataset are processed through three different data mining techniques which are the stepwise feature selection, PCA and the statistical method.

### 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.

### PCA (Principal Component Analysis)

PCA is a linear dimensionality reduction technique using singular value decomposition of the data. In order to apply PCA, features of heroes’ combo together with the original heroes’ 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.

### Statistical method

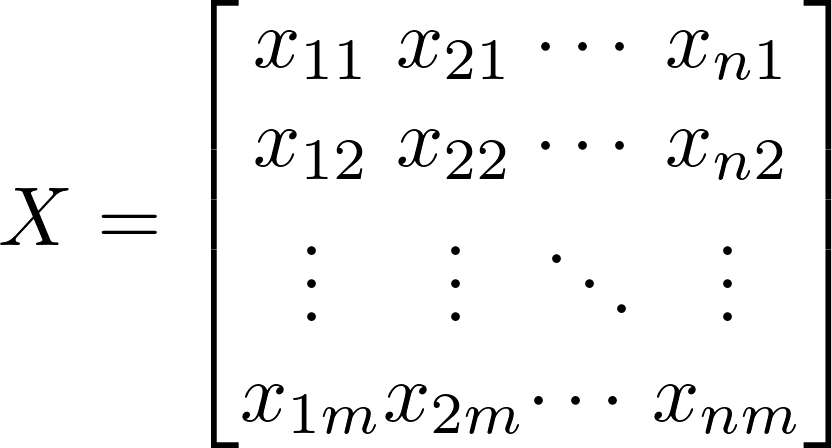
Unlike the techniques mentioned above, statistical method is an unsupervised process of learning the features. An unsupervised learning stands for algorithms that reduce the column of output in the dataset while learning the input automatically.

Statistical method is an intuitive technique which is easy to operate with. 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 as features of the model.

Interpretation behind this technique is that in actual game, players tend to pick the heroes’ combo which they thought is more powerful thus considered more reliable as the feature in model training process.

### Conclusion

All these three dimensionality reduction techniques were tried among which the statistical technique gave out the best result. Hence the approaches and implementations are based on statistical method described above.

After feature engineering, dataset is converted into a 238-dimensional matrix(taking 50 heroes' combo as an example). The details are as follows where *m* equals to 238, and *n* represents the number of samples in the dataset.

### Selection of estimators

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:

Training model is actually the process of determining the parameter *w* with corresponding input *x*. To generate reliable model, the core of model training begins with the selection of certain functions which are the estimators in Scikit-learn.

In this case, hyper parameters are introduced which are variables pre-defined bound with the estimators. Each estimator would have different number of hyper parameters. The process of selecting estimators is actually the process of fine-tuning the hyper parameters of estimators to see if the parameter *w* derived could best describe the mapping relationship between the input and the output, resulting in the performance of model generated.

In this project, three estimators, namely the logistic regression with weight decay, the decision tree and the support vector machine, short for SVM are introduced as follow.

### Logistic Regression with Weight Decay

**Logistic Regression**

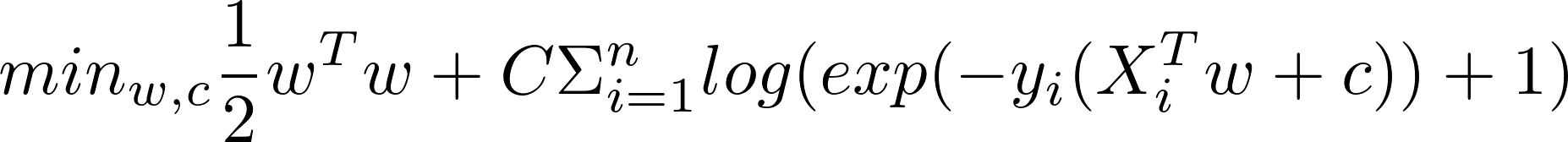
Logistic regression, despite its name, is a linear estimator for classification rather than a regression one. Logistic regression is also known in the literature as logit regression, maximum-entropy classification or the log-linear classifier. In this estimator, the probabilities describing the possible outcomes of a single trial are modeled using a [logistic function](https://en.wikipedia.org/wiki/Logistic_function).

Logistic regression is applied for 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 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. In this project, the model might over-fit probably because of too many input features.

In order to prevent the model from over-fitting, a widely used method 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 qt_temp represents the weights of the features, C is weight decay parameter:

When training the model, 10-fold cross validation is used to fine-tune the weight decay parameter C which is the hyper parameter of this estimator.

### Decision Tree

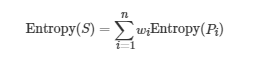
Decision Tree is a graphing method of intuitively using probability analysis to evaluate project risk and judge its feasibility by forming decision tree. In machine learning, the decision tree is a predictive estimator that represents the mapping relationship between object properties and object values. 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.

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:

From the equation, for a given segment of dataset S, the term *c* refers to the number of different class levels, and *pi* 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:

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 (S1), and the partitions resulting from the split (S2). That is:

One complication is that after a split, the dataset is divided into more one partition. Therefore, the function to calculate Entropy (S2) 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:

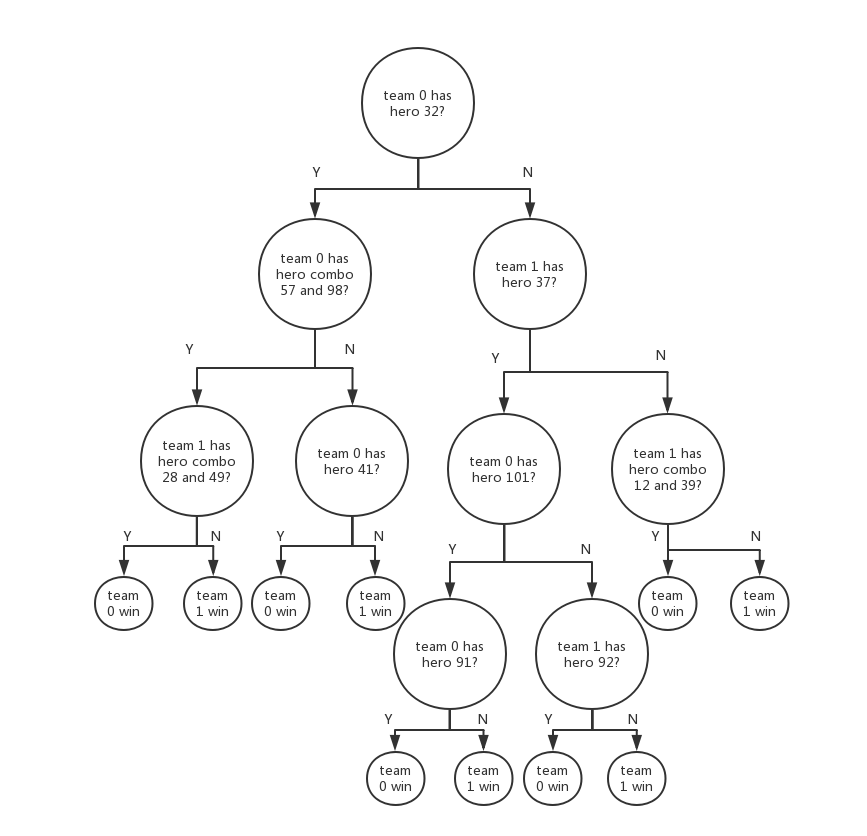
The higher the information gain, the better a feature is at creating homogenous groups after a split on that feature thus splitting upon it.

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 feature.

Although decision trees tend to have more serious over-fitting problems, the risk of over-fitting can be reduced by limiting the depth of the decision tree. As with the coefficient of weight decay selected in the above logistic regression, the tree depth selection, in this project, is made through cross validation.

Note, although decision tree could accept categorical feature as input, it is still necessary to use the one-hot encoding feature for consistency with other selected models.

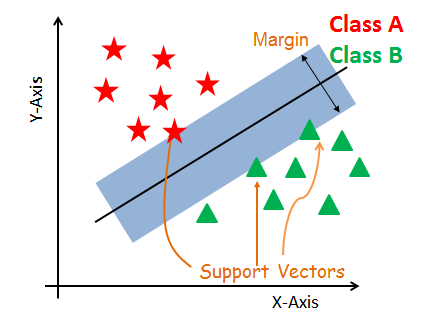
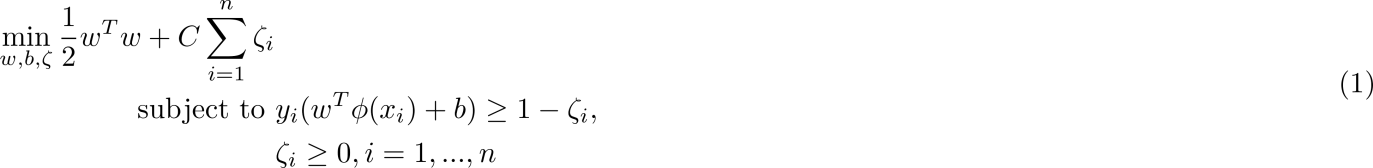
Finally, a simple decision tree example with max depth equals to 4 is shown in figure below describing approach of decision tree estimator in this project.

### Support Vector Machine

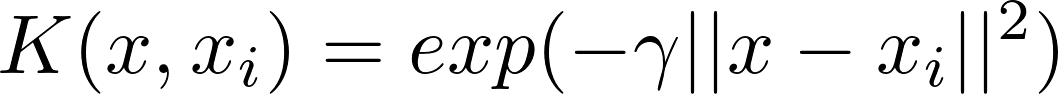
For more robust predictions, Support Vector Machine is added to the list of estimators used in addition to the previously mentioned logistic regression and decision tree. In general, Support Vector Machine is considered to be a classification approach, but it can be employed in both types of classification and regression problems.

SVM is 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.

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. Unlike the previous mentioned estimators, support vector machine generally does not suffer from condition of overfitting 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:

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 equation below.

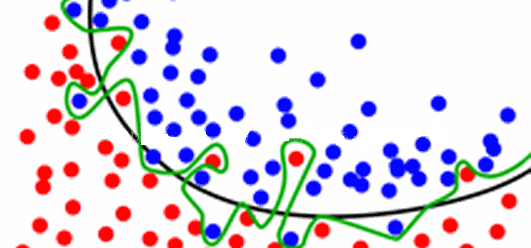
The Radial basis function kernel is a popular kernel function commonly used in support vector machine estimator. RBF can map an input space in infinite dimensional space.

Similarly, 10-fold cross validation is used to determine the hyper parameter, namely the optimal penalty factor C, resulting in the SVM model with the lowest generalization error.

### Fusion

In machine learning, overfitting problem is quite common as already mentioned in above sections. The fundamental problem 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.

However, if the results are averaged, the overfitting phenomenon can be reduced to some extent.

As shown in the figure, 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 consideration of these noise points decreases because the results are evened, and the decision boundaries move slowly closer to the black line.

One of the model averaging strategies is weighted average of different models. In weighted average method, a weight parameter is added to the result method to control how much each model affects the fusion result. Different weight combinations have a great influence on the final results of the fusion model, and generally multiple weight values have to be tried to achieve the optimal multi-model fusion solution.

As could be obtained the probability output of the models used respectively, model averaging strategy of this project is to simply apply weighted average on probability outputs of the three models respectively, and predict the final results based on the averaged probability. Weights are also fine-tuned by performing the averaged model on test dataset. Result with acceptably higher predictive accuracy yields the weights of each sub-models.

### qt_tempModel encapsulation

As described in the above chapters，the final model used in the subsequent application is the fused version of models generated by logistic regression, decision tree and SVM. The fused model has to be encapsulated for reference by the subsequent application. Therefore, a wrapper model is developed.

To be specific, the wrapper model has an “predict” API that accepts original features which is the heroes’ combination of two teams, the picked hero id for both teams. It returns the predicted probabilities from fused model. Inside the “predict” API, feature engineering method is called for generating features, and the “predict” methods of the sub-models (logistic regression, decision tree, SVM) are called and the probabilities are retrieved. The wrapper model then conducts weighted average of the returned results and gives the final result.

In order to deploy the model to web application, the wrapper model is instantiated and serialized to python pickle format, which is an easy-to-use format for all kinds of applications.

### Flask application

Given the packaged model of sci-kit learning result, it is possible to design Flask application which plays the role of web framework described in chapter 2.3. This chapter would then explain the implementation details of modules in Flask application. But, at the beginning of this chapter, it is necessary to first get an overview on the architecture of a typical Flask application which is the MTV architecture.

### MTV architecture

MTV architecture shares common concept with MVC architecture which is more familiar to developers. Both of them are introduced since each of them works well with object-oriented programming. Programs are split into three aspects and each of the aspects can be treated as objects thus improving reusability within an application. Besides similarity, MTV architecture varies from MVC architecture, which is quite apparent since the keyword in MTV, Template, replaces Controller section in MVC. This chapter would explain the MTV architecture in comparison with the MVC architecture for easy understanding.

**MVC (Model-View-Controller)**

MVC architecture 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.

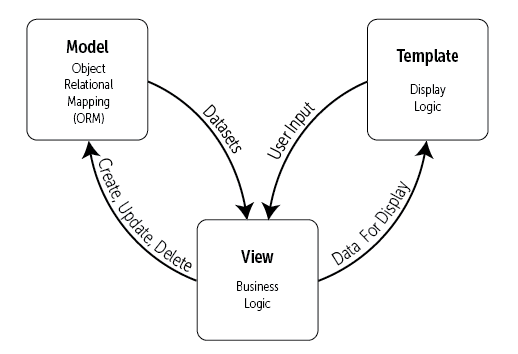
From the figure, Model section serves for central component of the architecture. It is the application’s dynamic data structure, independent of the user interface. It directly manages the data, logic and rules of the application.

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)**

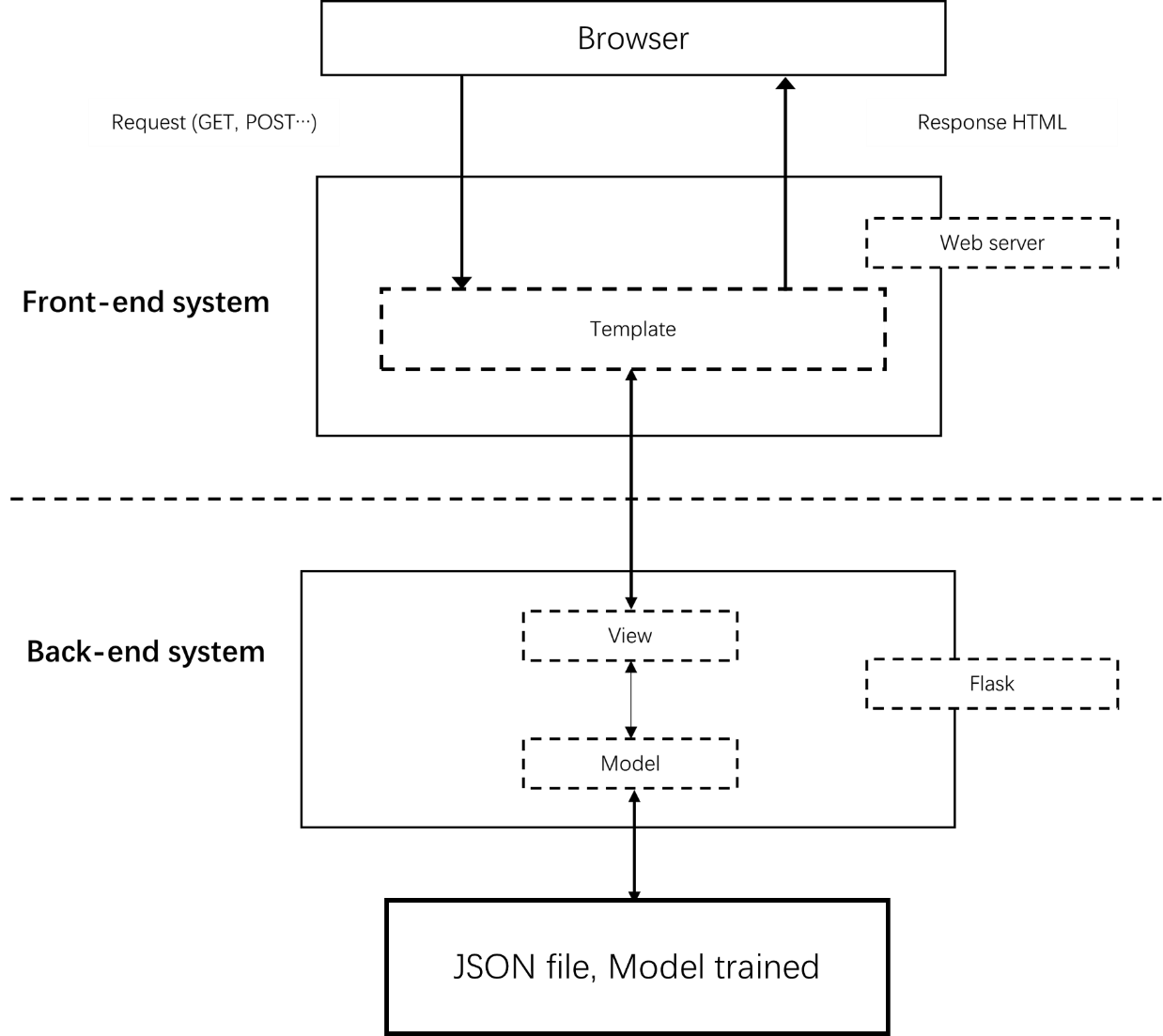
While in the case of web frameworks, for example Flask, MVC architecture differs from their interpretations, mainly in the way that the MVC responsibilities are divided between the client and server.

In MTV architecture, the definition of Model still remains the same that is, Model section in MTV is responsible for handling data between database and View. Model 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.

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.

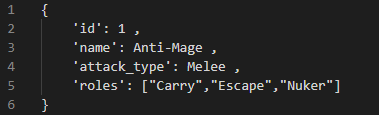
As for the Controller section in MVC architecture, its duty is actually done by the framework itself.

With the knowledge of MTV architecture, the previous architecture of this project can be updated as follow.

### Implementation of JSON file

As is illustrated from the architecture of this project, it can be seen that the part of substitution of database in this project still lacks one component which is the JSON file. Before starting to explain the implantation details of each MTV sections, the implementation of JSON file would be described in this chapter.

So far as the model is trained and encapsulated, database of this project remains incomplete, missing data to build up the communication system between the database with View section. That is the mapping from plain hero ids, which is collected by the data collection module mentioned before, to the name of the hero according to its id. In fact, it is unnecessary and ought not to show plain hero ids to the users.

Therefore, the corresponding relationship between them is defined in JSON format which is easy to organize and use. The mapping rule is copied from the official website of DOTA2 whose attributes can be illustrated as follow:

The element id stands for the hero id which is previously collected by the back-end system. More importantly, the corresponding name of the hero is recorded in it which would definitely been used in the front-end system. 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.

Unlike the back-end system where data need to be pre-processed, each of the elements in the mapping rule is considered useful as for the hero portrait. All of them might be useful ,for example, when screening function is implemented in the front-end system. Thus, data of all of the 120 heroes is collected and stored.

### Model

Model, repeatedly speaking, is responsible for handling data communication events which are held between View section and the database. One of the communication processes can mainly be described as:

1. Inputting from front-end system with 10 picked heroes’ names;
2. Outputting from database with the probabilities of two teams.

For functionality of (1), Model section begins with mapping the picked heroes’ names with their corresponding hero ids generating an input array. After that, a data validation function would be performed to the array of hero ids to verify the input.

For functionality of (2), input is passed to data-to-feature function after validation. Then, Model would open the trained model by pickle function, trailing with features processed through the trained model generating results of probabilities to be returned.

To be specific, the handling process mentioned above is actually based on prediction event. On behalf of recommendation event, it is a bit more complicated.

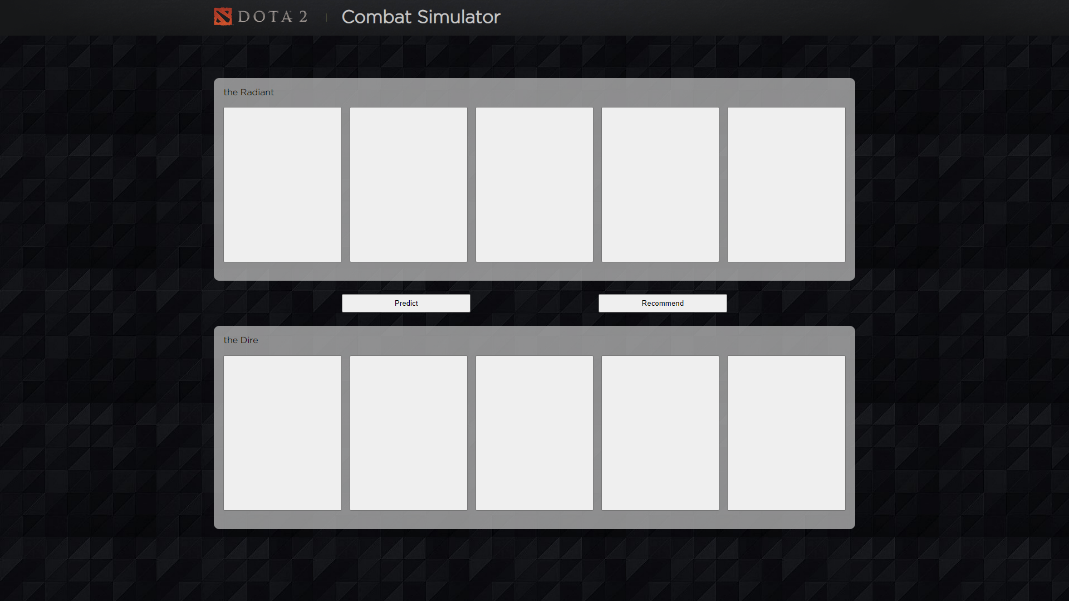
For the input, different validation rules are performed to assure that the input only consists of 9 picked heroes.

For the output, the back-end system would perform a traversal function adding each of the available heroes to the input array and comparing the result probabilities respectively. In the end, the hero added to the input array that result in best improvement of probabilities is returned. Of course, the returning data would be in hero id format which needs to be transferred into hero name format.

In conclusion, Model section fulfils a list of functions listed below:

1. Hero mapping function which maps hero ids with hero names and vice versa;
2. Data validation function which validate the input data;
3. Data-to-feature function which transfer the input data into form which can be then processed through the model trained;
4. Pickle function which calls embedded functions in the model trained.

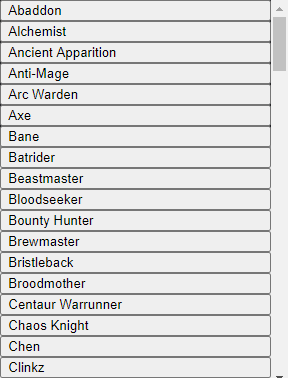
### Template

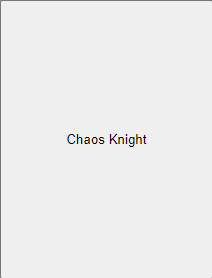
Since this project simulates the hero picking process in the game and provides service based on that, the home page of the website is designed similar to the actual hero picking interface in DOTA2 which would be quite familiar for players of DOTA2.

The actual hero picking interface in DOTA2 is illustrated as follow:

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 the window of player with the icon of the hero.

The home page of this project imitates the designing pattern of DOTA2 interface but needs more decoration to look more alike to it.

Also, to make picking of heroes available on the browser, a drop-down menu is implemented in the home page of the website which presently looks like:

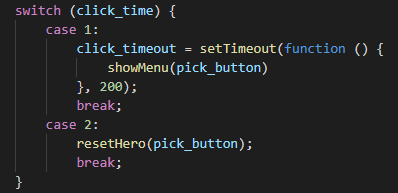
Clicking on any one of the entries would lead to change of the player’s window and update of the input array which is defined as a global variable in JavaScript. Following figure is an example showing the change of the player’s window when picking hero named Chaos Knight.

For save of resources in the back-end system, JSON file of hero mapping is transferred into the front-end system as well. The name of the hero is connected with its id as soon as Template has been rendered by the browser. By operating in JavaScript, the final input from the front-end system would be an array with its length of 10 containing the hero ids which input by users:

This type of input works for prediction events held in View. For recommendation events, the default value of the array is set -1 for data validation in the back-end system because the value would be empty if not set which is not easy to validate.

However, chance is 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 kind of 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.

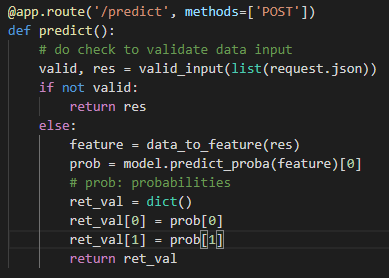
There are two ways to fix this. One of which is to implement new UIs responsible for resetting the picked heroes. It is considered making the home page too complex adding more UIs to it. Thus, the second way is introduced which is placing a slight delay function between the single click and the double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

### View

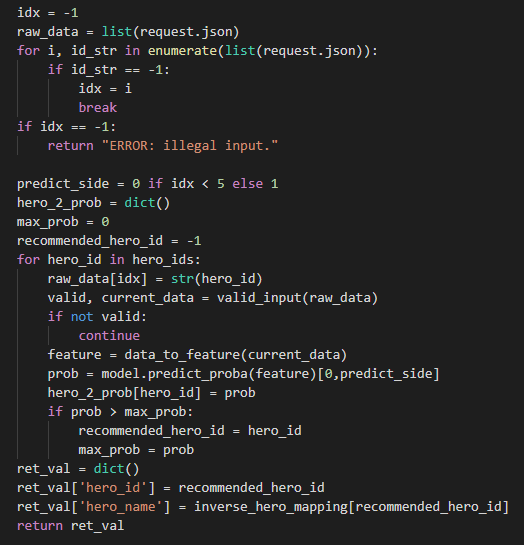
View section is realized on the basis of routing syntax in Flask. Foreseen in Chapter 2.3.3, this project possesses a quite simple routing module which mainly realizes function of prediction and function of recommendation.

The function of prediction is realized as:

In the figure, URL is first defined in the decorator of Flask application trailing with the constraint of requesting method in which only requests sent by POST method would be responded. Among all of the requesting method, POST is considered more standard way of modifying and recording data. Besides, data sending by POST would be more secured since it would not be a part of the URL, nor would it be cached by the browser.

It is mentioned above that Template would directly pass array in which hero names are already mapped to hero ids. Therefore, validation function called from Model is performed once View has obtained the data. If validated, Model would apply machine learning functions on it as described in Chapter 3.5.3 and return a dictionary with probabilities of two teams. Consoling the outcome on the browser yields:

Each of the floats variable stands for probabilities of corresponding team.

The function of recommendation is only described by its callback functions which differ from those of the function of prediction.

Data validation function is written to assure the input array pertaining an element of -1. After that, the input array is passed through a traversal function to generate recommendation via probabilities. The hero id is converted into hero’s name before returning. Therefore, consoling the outcome on the browser yields:

In case of misunderstanding, Spectre is one of the heroes in DOTA2.

### 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 apply ROC curve &AOC value on the test dataset which are introduced in Chapter 3.1.3 to evaluate the performance of each estimator used to train the model. For the second part which is the Flask application, white box test is arranged on Python programs while usability is tested on the outcoming website.

### ROC&AOC

The evaluation criteria to measure the generalization ability of the model are also needed. Evaluation metrics are an important part of machine learning tasks. Different machine learning tasks have different evaluation indicators, while the same machine learning tasks also have different evaluation indicators, each indicator has a different focus. Such as classification, regression, sorting, clustering, popular topic modeling, recommendation, and so on. And many metrics can evaluate many different machine learning models, such as precision-recall, which can be used in classification, recommendation, sorting, and so on. Like classification, regression, sorting are supervised machine learning.

In this thesis, accuracy, ROC curve and AUC is used to evaluate the trained model.

Accuracy

Accuracy is defined as the ratio of samples that the model gives the predicted result same as the true label

A ROC curve is constructed by plotting the true positive rate (TPR) against the false positive rate (FPR). The true positive rate is the proportion of observations that were correctly predicted to be positive out of all positive observations (TP/(TP + FN)). Similarly, the false positive rate is the proportion of observations that are incorrectly predicted to be positive out of all negative observations (FP/(TN + FP)). For example, in medical testing, the true positive rate is the rate in which people are correctly identified to test positive for the disease in question. A discrete classifier that returns only the predicted class gives a single point on the ROC space. But for probabilistic classifiers, which give a probability or score that reflects the degree to which an instance belongs to one class rather than another, it is able to create a curve by varying the threshold for the score. Note that many discrete classifiers can be converted to a scoring classifier by ‘looking inside’ their instance statistics. For example, a decision tree determines the class of a leaf node from the proportion of instances at the node.

To compare different classifiers, it can be useful to summarize the performance of each classifier into a single measure. One common approach is to calculate the area under the ROC curve, which is abbreviated to AUC. It is equivalent to the probability that a randomly chosen positive instance is ranked higher than a randomly chosen negative instance, i.e., it is equivalent to the two sample Wilcoxon rank-sum statistic. A classifier with high AUC can occasionally score worse in a specific region than another classifier with lower AUC. But in practice, the AUC performs well as a general measure of predictive accuracy.

The evaluation results of the three above-mentioned model are listed in the following table. According to the evaluation results, support vector machine has the best performance across all of the three proposed models.

|  |  |  |  |
| --- | --- | --- | --- |
| Model | Validation Accuracy | Train ROC & AUC | Validation ROC & AUC |
| Logistic Regression | 0.5473 | Logistic Regression Classifier (train) | Logistic Regression Classifier (test) |
| Decision Tree | 0.5502 | Gradient Boosting Classifier (train) | Gradient Boosting Classifier (validation) |
| Support Vector Machine | 0.5549 | XGBoost Classifier (train) | XGBoost Classifier (validation) |

### White box test

### Usability test

### Conclusion and Outlook