### 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 character against another team. Among hundreds of characters, there could have tens of thousands of combinations in which some characters 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 characters’ 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 characters’ 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], methods of Sci-kit learning [3] are applied on the formulated data set to generate a reliable model representing the relationship between characters’ combination and the result of the game. The model is used in the process of web-based visualization. The first usage of it is predicting the result of the game when the 10 picked characters are given beforehand while the second one is providing certain recommendation according to its knowledge via characters’ combination when users are confused of which character to pick.

### Organization

The rest of the thesis is structured as follows. Chapter 2 states the technical background of machine learning and website visualizing. The part of machine learning is introduced from feature engineering to Sci-kit learning. The part of website visualizing mainly talks of how to associate the front-end and the back-end system with micro framework Flask.

Chapter 3 focuses on the approaches and implementation details during the development of the project. Web crawler in Python is provided which obtains and formulates the original data. Then the machine learning methods applied are discussed. After that, the process of packaging and importing interface from model generated is described. Finally, this chapter ends with explaining the data communication flow between the front-end and the back-end system.

Chapter 4 evaluates the performance of the model generated and the usability of the website. The evaluation of the model generated is arranged in terms of ROC curve to check the precision of each methods being applied respectively. The usability of the website is tested by white box and black box tests to analyze the functionality of the website. A questionnaire is also included in the end of this chapter to see whether the website is actually useful for different type of users.

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

### Approaches and Implementation Details

### Dataset Preparation

Valve provided web API for third party developers and researchers, including API for public match data query. However, there is not many document available online for the original web API. In order to get easy-to-use API, the thesis used API provided by another third-party developer, OpenDota. The free version of the public API can provide 50,000 requests per month, and each request can get data for up to 100 public Dota2 games, thus meeting the needs of the original data collection.

##### Data collection module

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

**Main Module**

The main module is responsible for the overall data collection and scheduling work, is responsible for loading the pre-set API request URL, dispatching the API request module to make data requests, and the original data returned through the data storage module is stored as a file.

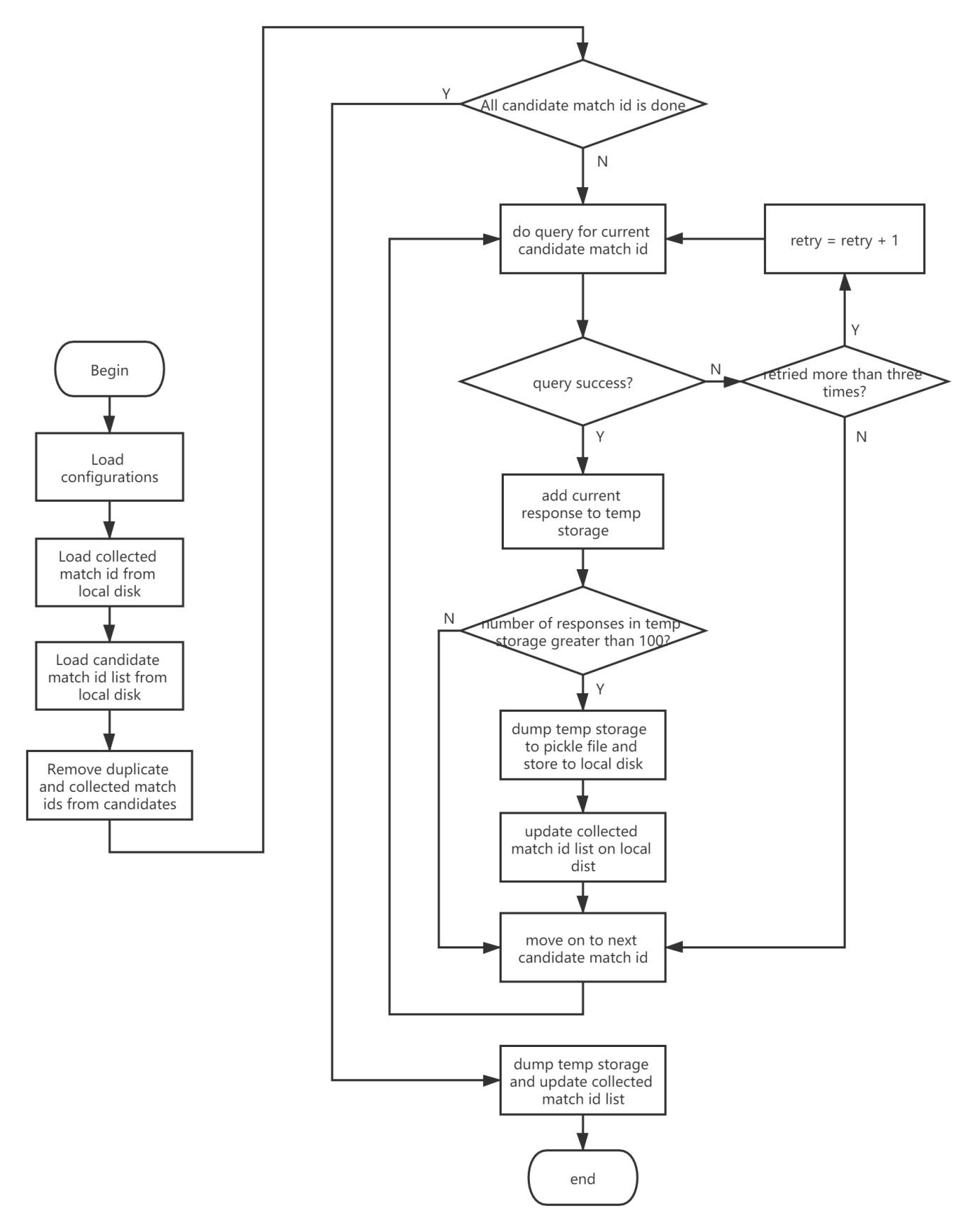
**API Request Module**

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

**Data Storage Module**

The data storage module is also an accessibility module, which is primarily responsible for tracking and storing two kinds of data: (1) maintaining and storing the match id that has been collected; (2) The raw data collected is stored in a pickle format in a file.

The flow chart of data collection module is shown in figure ?



##### Raw Data Cleansing and Pre-processing

Through the data collection module, about 150,000 open dota2 match data was collected, and the data collection module did not verify the integrity and correctness of the collected data, so an additional data cleaning module is needed to clean the data, while the original data in the johnson format is organized and output into Pandas Dataframe format for subsequent use.

We know that a complete Dota2 match needs 10 players to play, divided into two camps, and the choice of heroes is one of the factors that determine the final result of the game, there are many other factors that affect the result of the game, including but not limited to the skill level of the players, the version of the game (different versions will be differed from each other in some aspects) and so on. Therefore, the following cleaning and screening processes are performed for the collected public dota2 match data:

**Data integrity**

According to observations of the collected data, there exist data missing information for some games, such as empty hero ids, so we write filters to filter out games with fewer than 10 heroes.

**Reliability of match results**

Based on a statistical analysis of the total duration of the 150,000 open dota2 matches collected, it is found that there are games with shorter game durations, such as games with a duration of about 10 minutes. According to empirical analysis, the average Dota2 game duration will be concentrated in 30 minutes to 60 minutes, so for games that are too short, we do not think it is available, because the short time is often due to some players negative games or network reasons, etc. It has a decisive effect on the outcome of the game, so this part of the game data needs to be filtered out.

**Effect of the players’ skill level**

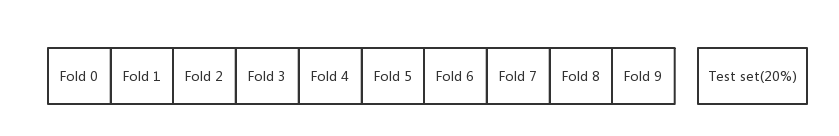
The players’ skill level has an impact on the results of the matches, and for games with more junior and entry-level players, we believe that the results of those matches are less reliable, so we built a filter to filter out matches with overall skill level as “normal”.

When we finished these filtering and data cleaning operations, we got the cleaner raw data, after the statistics, a total of about 50,000 games, and then by writing a data pre-processing script, the original data in jason format into the Original data in Pandas Dataframe format, the specific column fields are as follows. Among them, Label indicates the comparison results, where 0 indicates team 0 won, and 1 indicates team 1 won：

|  |  |
| --- | --- |
| Column field name | Data type |
| index | Integer |
| 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 |

##### Dataset Creation

In machine learning, data sets need to be divided into training datasets, validation datasets, and test datasets. Since the sample size is not very large, we divide the entire dataset into training and test datasets at a scale of 80% and 20%. At the same time, 80% of the training dataset is divided into 10-folds randomly according to 10-fold cross-verification process, which is then used as datasets for follow-up training and verification. In subsequent model training, the model super-parameters are selected by the k-fold cross-validation method.



##### Feature Engineering

Predictive modeling of game results using only heroic lineups as feature inputs often does not lead to good results. According to dota2 domain knowledge, we can see that in addition to the choice of heroes, the combined effect between heroes can often have a great impact on the outcome of the game. Storm Spirit and Life Stealer, for example, can have strong skill combinations, and teams with these two heroes tend to have some advantages.

By the time data is collected for this project, Dota2 has 119 heroes, and common hero combination consists of two heroes. If all hero combinations are calculated as generated features, there will be a total of 7,021 combination features, which will appear in team 0 and team 1, respectively, so there will be a total of 14042 dimensional combination features(if one-hot encoding was used). Such feature dimensionality is obviously excessive and can affect the generalization performance of the model, so we need to reduce the dimension the combined features.

In order to be able to degrade the feature, we process the feature in two directions: statistical method and PCA.

**Statistical Method**

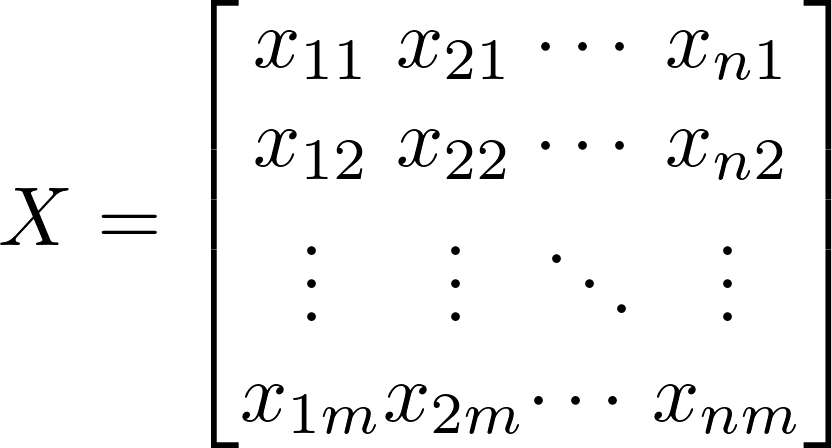
Statistical method is an intuitive and easy method. We first calculate the hero combinations that exist in all training data sets, then count the combinations that appear. We then set a threshold k (such as 50, 100 or 500), sort the hero combinations by the number of appearances, and select the top-k hero combinations as hero combination features. An intuitive explanation behind this method is that in the actual game, the more times the hero combination appears, the often the players prefer to choose, hence the more powerful hero combination.

**Principle Component Analysis**

Another useful tool for dimension reduction is PCA(principal component analysis). In short, PCA is a linear dimensionality reduction method using Singular Value Decomposition of the data（你这里增加一些PCA的介绍，引用）. In order to apply PCA to our dataset, we need to convert the original feature, the hero selection, and the combined features into one-hot encoding. At the same time stack all the encoded data from the training set into a feature matrix, then apply PCA to reduce the feature dimensionality to a certain threshold range.

Both dimensionality reduction methods were tried, and the statistical method gave out better results than PCA method. Hence the approaches and implementations were based on statistical method described above.

After the feature engineering, each sample is converted into a 238-dimensional vector(take 50 hero combinations as an example). The details are as follows:



Where m equals to 238, and n represents the number of samples in the dataset.

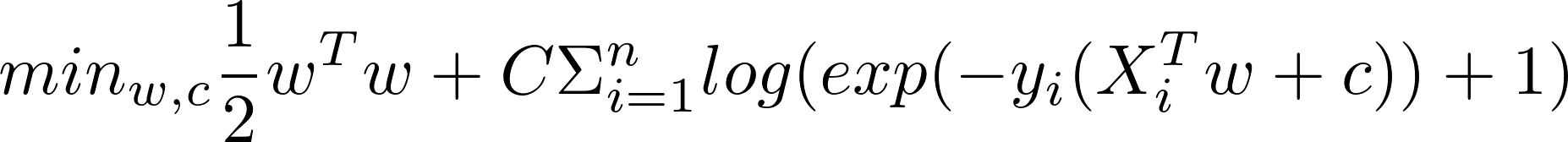
##### Model Training

Three machine learning models, namely, logistic regression with weight decay, decision tree and support vector machine were used in this thesis.

**Logistic Regression with weight decay**

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

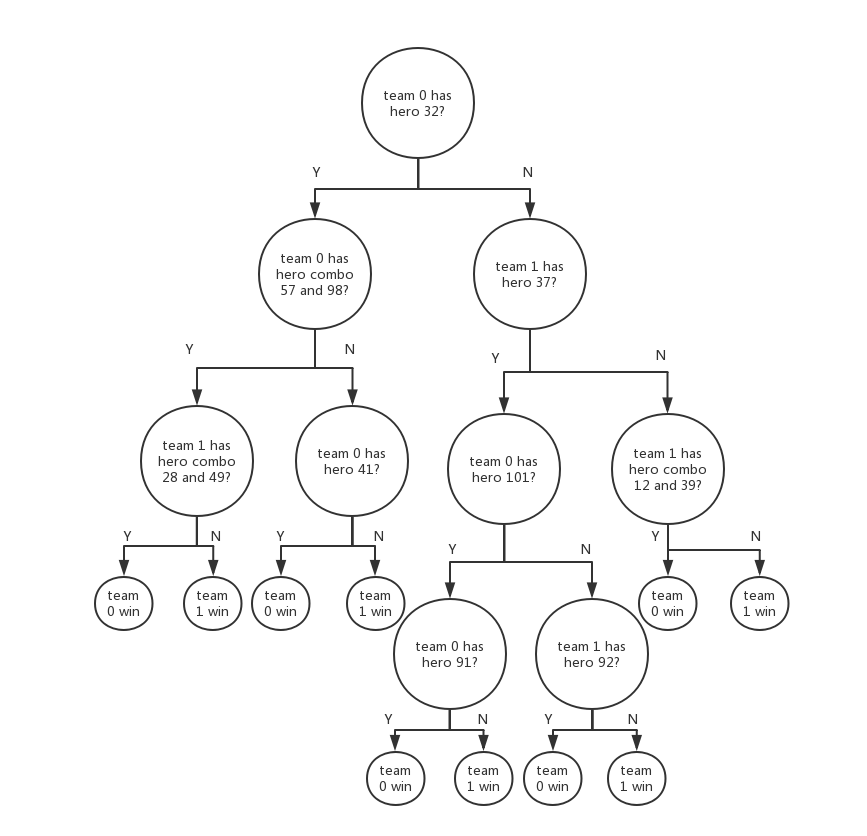
In order to prevent the model from over-fitting, a widely used method is adding weight decay (also known as L2-norm penalty) to the loss function of logistic regression. Hence the resulting problem is to minimize the following loss function:



When training the model, we used the 10-fold cross-validation to choose the weight decay parameter C.

**Decision Tree**

Decision Tree is a kind of graph method of intuitively using probability analysis to evaluate project risk and judge its feasibility by forming decision tree to obtain the probability of NSI being greater than or equal to zero. In machine learning, the decision tree is a predictive model that represents a mapping relationship between object properties and object values. Entropy - The degree of clutter in the system, using algorithms ID3, C4.5, and C5.0 to generate tree algorithms using entropy. This measure is based on the concept of entropy in information theory. Classification tree (decision tree) is a very common classification method. It is a kind of supervised learning, so-called supervised learning is given a bunch of samples, each sample has a set of attributes and a category, these categories are predetermined, then by learning to get a classifier, this classifier can give the correct classification of emerging objects. A simple decision tree example with max depth equals to 4 is shown in figure ?.

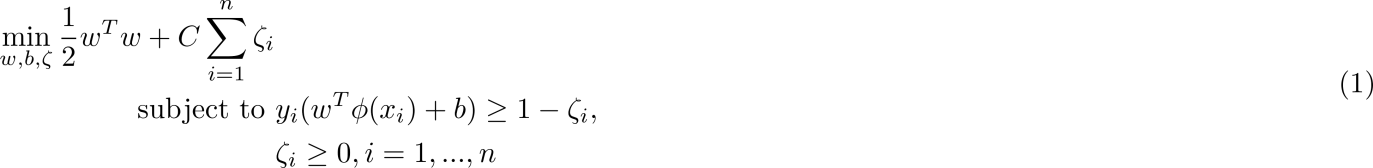


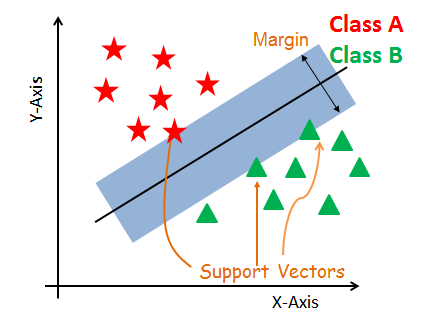
Although decision tree could accept categorical feature as input, we still use the one-hot encoding feature for consistency.

**Support Vector Machine**

Generally, Support Vector Machines is considered to be a classification approach, it but can be employed in both types of classification and regression problems. It can easily handle multiple continuous and categorical variables. SVM constructs a hyper-plane in multidimensional space to separate different classes. SVM generates optimal hyper-plane in an iterative manner, which is used to minimize an error. The core idea of SVM is to find a maximum marginal hyper-plane(MMH) that best divides the dataset into classes.

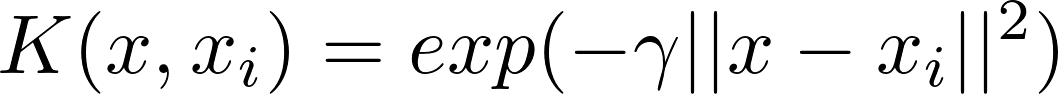
Mathematically, SVM solves the following optimization problem:





The SVM algorithm 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 most useful in non-linear separation problem.

In our case, we use Radial Basis Function(RBF) kernel as shown in equation ?. The Radial basis function kernel is a popular kernel function commonly used in support vector machine classification. RBF can map an input space in infinite dimensional space.



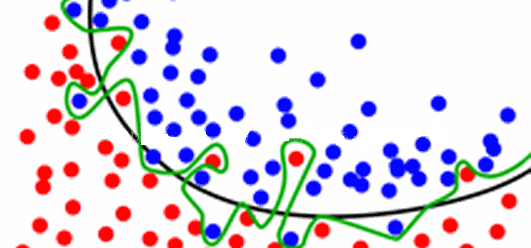
Similarly, we use 10-fold cross-validation to determine the optimal penalty factor C, resulting in the SVM model with the lowest generalization error.

##### Model Fusion

**Model Averaging**

In the application of machine learning, overfit is very common, the fundamental problem is that the amount of training data is not enough to support complex models, resulting in the 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 overfit phenomenon can be reduced to some extent. As shown in the figure, a single model produces a green decision boundary because it is overfitted, 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.



An intuitive enhancement of the model averaging strategy 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 we could obtain the probability output of the models used in the thesis easily, our strategy is simply apply weighted average on probability outputs of the two models, and predict the final results based on the averaged probability. We also searched the weights to get the optimal weights using the prediction accuracy on test dataset. The diagram of the fused model are shown in figure ?.

