



Figure 2.1.1 An example of the decision tree algorithm

The parameters of the Decision Tree algorithm used in the project are shown below (Table 2.1.1). Other parameters which are not shown in the table is determined by default.

Parameters	Value
criterion	gini
splitter	random
max_depth	15
min_samples_split	2

Table 2.1.1 Parameters of the Decision Tree algorithm

2.2 Support Vector Machine

A Support Vector Machine (Figure 2.2.1) is a supervised learning algorithm that can be used for binary classification or regression. The Support Vector Machine transform input data to a high-dimensional feature space and then it constructs an optimal hyperplane as a decision surface such that the margin of separation between the two classes in the data is maximized. There are many kinds of kernel of the transformation step, which will affect the result of the prediction. Once the hyperplane is determined, it will use this plane to classify the unlabeled data.

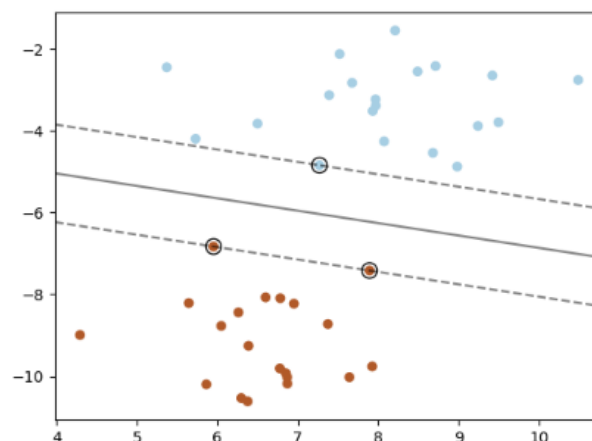


Figure 2.2.1 An example of the support vector machine algorithm

The parameters of the Support Vector Machine algorithm used in the project are shown below

(Table 2.2.1). Other parameters which are not shown in the table is determined by default.

Parameters	Value
kernel	rbf
gamma	auto

Table 2.2.1 Parameters of the Support Vector Machine algorithm

2.3 Multi-Layer Perceptron

Multi-Layer Perceptron (Figure 2.3.1) is a forward structured artificial neural network consists of input layer, hidden layers and output layer. It can be treated as a directed graph with many nodes in one layer and each layer is connected to the next layer. A backpropagation method is used to improve the weight of each link between two layers which is the process of training the whole network.

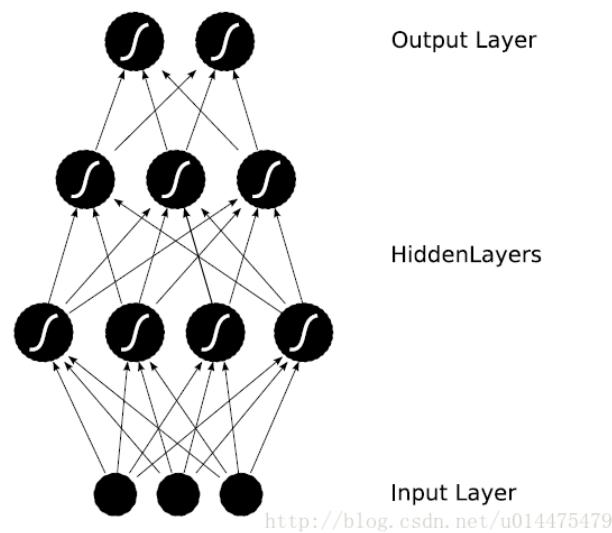


Figure 2.3.1 A module of the Multi-Layer Perceptron algorithm

The parameters of the Multi-Layer Perceptron algorithm used in the project are shown below (Table 2.3.1). Other parameters which are not shown in the table is determined by default.

Parameters	Value
hidden_layer_sizes	(10,10)
activation	relu
batch_size	auto
learning_rate_init	0.001

max_iter	1500
tol	0.00001
early_stopping	Ture
n_iter_no_change	20

Table 2.3.1 Parameters of the Multi-Layer Perceptron algorithm

2.4 K Nearest Neighbor

K Nearest Neighbor (Figure 2.4.1) is one of the simplest algorithms in machine learning. It uses the distance between the unlabeled data and samples to classify the input. The distance means the similarity of the samples and the unlabeled data. There are many kinds of ways to calculate the distance, such as the Manhattan distance, the Euclidean distance. It will choose the nearest K samples and according to the majority rule to classify the input.

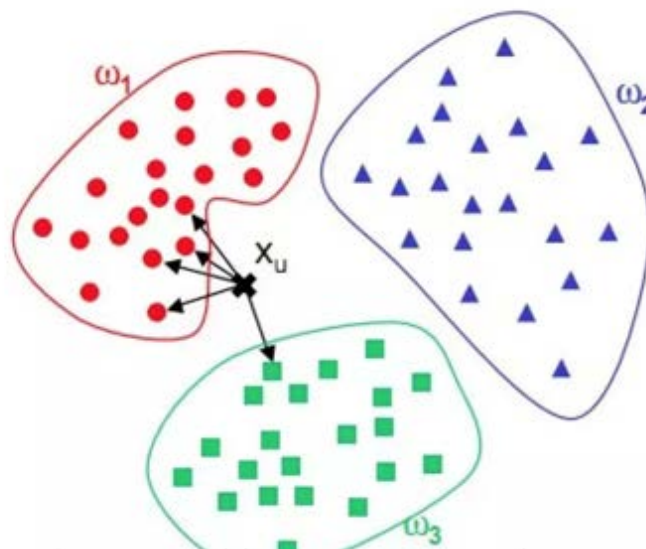


Figure 2.4.1 An example of the K Nearest Neighbor algorithm

The parameters of the K Nearest Neighbor algorithm used in the project are shown below (Table 2.4.1). Other parameters which are not shown in the table is determined by default.

Parameters	Value
n_neighbors	5
weights	uniform
algorithm	auto
leaf_size	16

p	2
---	---

Table 2.4.1 Parameters of the K Nearest Neighbor algorithm

2.5 Voting Ensemble

The Voting Ensemble method is a combination of all the other algorithms mentioned above. It considers all the predicted outcome of each classifier and according to the majority rule to choose the label for the input.

The parameters of the Voting Ensemble algorithm used in the project are shown below (Table 2.5.1) . Other parameters which are not shown in the table is determined by default.

Parameters	Value
estimators	[('dt',clsDT),('svm',clsSVM) ,('knn',clsKNN) ,('mlp',clsMLP)]
voting	hard

Table 2.5.1 Parameters of the Voting Ensemble algorithm

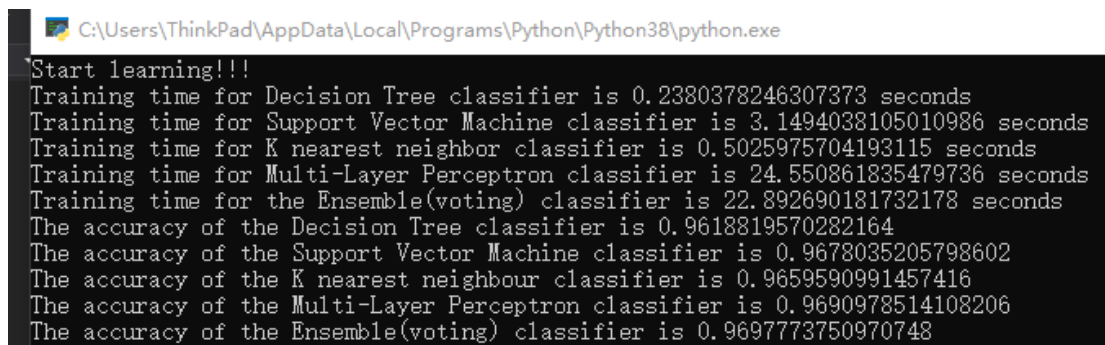
3.Experimental requirements

In the project, python is used to realize the function of predicting the winner. Some important packages help a lot when determining the classifier and training it. The version of the integrated development environment (IDE), interpreter and the main packages used in the experiment are shown in the following table (Table 3.1).

	<i>Visual Studio</i>	<i>Python</i>	<i>sklearn</i>	<i>pandas</i>	<i>numpy</i>
<i>version</i>	2019	3.8.2	0.23.1	1.0.5	1.19.0

Table 3.1 the version of IDE interpreter and packages

4.Experimental results

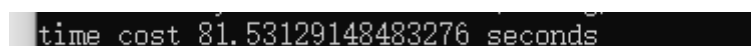


```

C:\Users\ThinkPad\AppData\Local\Programs\Python\Python38\python.exe
Start learning!!!
Training time for Decision Tree classifier is 0.2380378246307373 seconds
Training time for Support Vector Machine classifier is 3.1494038105010986 seconds
Training time for K nearest neighbor classifier is 0.5025975704193115 seconds
Training time for Multi-Layer Perceptron classifier is 24.550861835479736 seconds
Training time for the Ensemble(voting) classifier is 22.892690181732178 seconds
The accuracy of the Decision Tree classifier is 0.9618819570282164
The accuracy of the Support Vector Machine classifier is 0.9678035205798602
The accuracy of the K nearest neighbour classifier is 0.9659590991457416
The accuracy of the Multi-Layer Perceptron classifier is 0.9690978514108206
The accuracy of the Ensemble(voting) classifier is 0.9697773750970748

```

Figure 4.1 The training time and accuracy of each algorithm



```

time cost 81.53129148483276 seconds

```

Figure 4.2 The total cost of time to run the code

From the screen capture, we can obtain the training time and the accuracy of each algorithm, we summarize it in the table below (Table 4.1).

	Training time/s	Accuracy
Decision Tree	0.2380378	0.961882
Support Vector Machine	3.1494038	0.967804
Multi-Layer Perceptron	24.550862	0.969098
K Nearest Neighbor	0.5025976	0.965959
Voting Ensemble	22.892690	0.969778

5.Comparison and discussion

5.1 Comparison

From the experimental results, we can find that the Decision Tree algorithm and K Nearest Neighbor algorithm takes the less time and the Multi-Layer Perceptron algorithm and Voting Ensemble algorithm take the much more time. The accuracy of each algorithm is nearly the same but we can roughly say that the longer it takes the more accurate it predicts.

When I was improving the parameters of these algorithms, I found that some parameters change a bit the training time will increase a lot. For example, the hidden layer of the Multi-Layer Perceptron algorithm. Since the accuracy is approximately equal, I think Support Vector Machine algorithm performs better with short running time and high accuracy.

5.2 Discussion

5.2.1 Dataset processing

In the project, we selected the fields after the “winner” to be the feature of a match. The dataset of fields “gameID”, “creationTime”, “gameDuration” and “seasonID” are delete before we train the classifiers. I think these four fields are almost irrelevant to the winner which could increase the complexity of the dataset. To decrease the training time, these four fields should not be considered. However, there are some data with distinct value of the features will cause a mistake.

For example, some matches only last for a few minutes and then one team surrender.

In this situation, the features like “firstblood” or other features chosen for training can be all 0, but it still has a winner which means that in this situation the features are irrelevant to the result. This kind of data is a noise data, which will cause the mistake when predicting.

If more time allowed, we should delete those noise data before use the dataset for training and predicting.

5.2.2 Algorithm

5.2.2.1 Decision Tree

We need to control the max depth to avoid overfitting. Using an ensemble method is also effective to prevent this.

5.2.2.2 Multi-Layer Perceptron

The amount of the hidden layer should be approximately equal to the features of the dataset.

If there are much more hidden layer, the running time will increase since it need to do more calculation and the accuracy will decrease because of overfitting.