Report about Classification of Unlabeled LoL Match Records with “Win/Loss”

Mao Rui Shien-ming Wu School of Intelligent Engineering 201930362347

1. Introduction

League of Legends (LoL) is one of the most played eSports in the world at the moment. Its appeal lies in the fact that users can always match up with teammates and opponents of similar skill level, making the outcome of the game unpredictable.

The goal of this project is to use publicly available game statistics to train appropriate classifiers to predict game results. The training set and the test set have been given. Both of these two sets contain fields that are not related to the results, such as game id, creation time, game duration, season id, and also contain fields that are related to the results such as the attribution of first Baron, dragon, tower, blood, inhibitor, Rift Herald, and the number of tower, inhibitor, Baron, dragon and Rift Herald kills each team has as well. Most importantly, the label, called “winner”. The training set is used to train the classifiers. The accuracy of the classifiers can be obtained by comparing the predicted labels from the test set after applying the classifiers with the actual labels of the test set records.

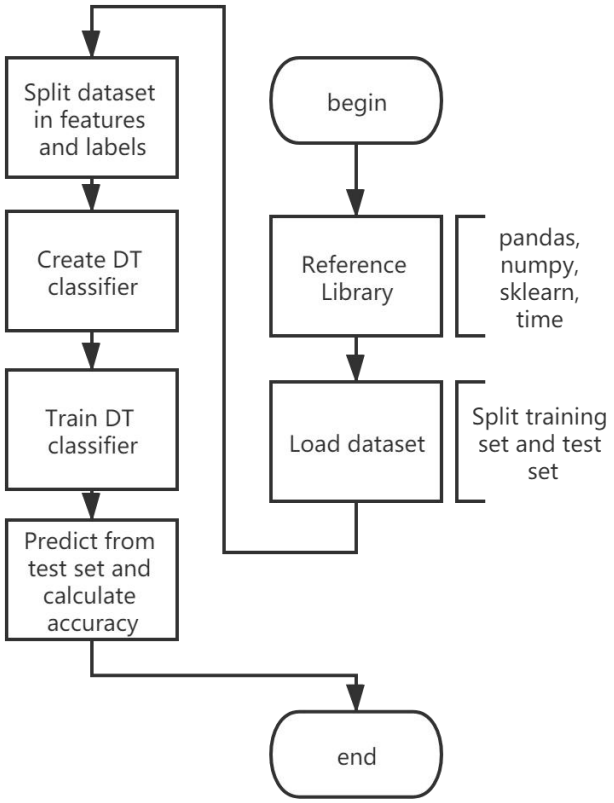
Two methods, DT (Decision Tree) and ANN (Artificial neural network) are applied in the training set and obtain high accuracy of 0.9612 and 0.9696 respectively, comparing with the test set available labels.

2. Algorithm

2.1. Decision Tree Classifier

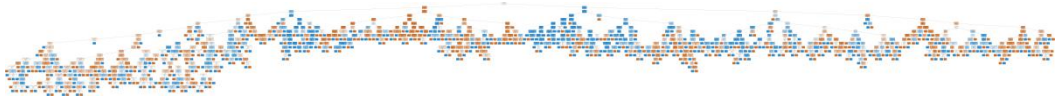
Decision Tree Classifier is a non-parametric supervised learning method used for classification. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. One of its advantages is using a white box model to make the explanation for the condition is easily explained by boolean logic. By contrast, in a black box model (e.g., in an artificial neural network), results may be more difficult to interpret.

In this project, the whole process of applying the decision tree classifier includes the following parts.



Visualizing Decision Tree

The Scikit-learn's *export_graphviz* function is used for display the tree. (Extra requirements: graphviz, pydotplus)



(As this picture is too big to see clearly, it will be attached to the report in the

form of an attachment.)

2.2. Artificial Neural Network

An artificial neural network (ANN) is the piece of a computing system designed to simulate the way the human brain analyzes and processes information. It is the foundation of artificial intelligence (AI) and solves problems that would prove impossible or difficult by human or statistical standards. ANNs have self-learning capabilities that enable them to produce better results as more data becomes available.

In this project, the whole process of applying the artificial neural network is similar to the DT classifier.

One difference is that the ANN model has a binomial classification of 0 and 1, therefore, we need to use an extra mapping function to map 1 and 2 in 'winner' to 0 and 1.

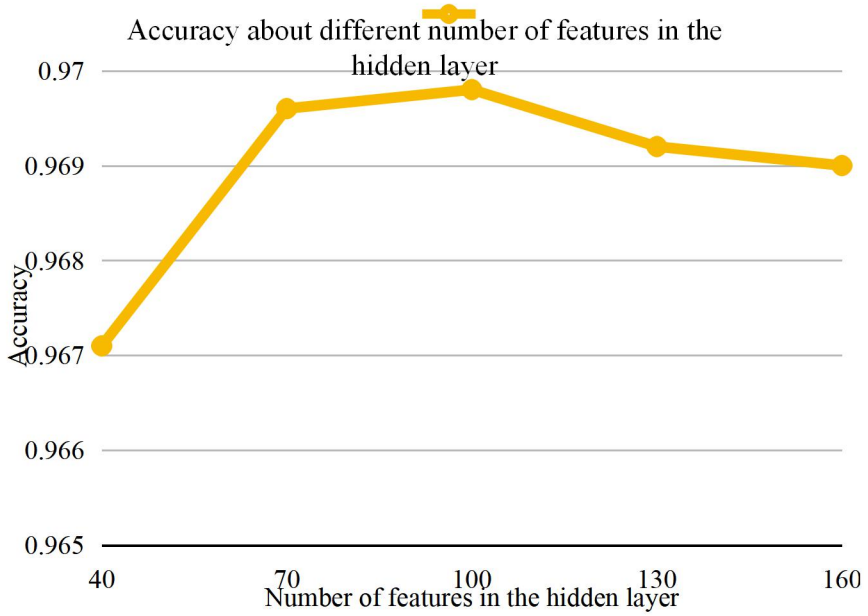
```
#Remap values from the winner column to numbers 0, 1.
mappings = {
      1:0,
      2:1
}
ts['winner'] = ts['winner'].apply(lambda x: mappings[x])
nd['winner'] = nd['winner'].apply(lambda x: mappings[x])
```

Due to the complexity of artificial neural network, more parameters need to be selected to improve the accuracy of the model. The selection process for the number of features in the hidden layer and epochs is as follows.

As for the architecture of the model, it will be very simple. We will construct a 3-layer MLP: 1. Fully Connected Hidden Layer (16 input features , X output features (arbitrary)) 2. Output Layer (X input features (number of output features from the previous layer), 3. output features (in this project, there are 2 distinct classes). This is the most simple MLP which includes only 1 hidden layer. Besides that, we’ll use Sigmoid for our activation function.

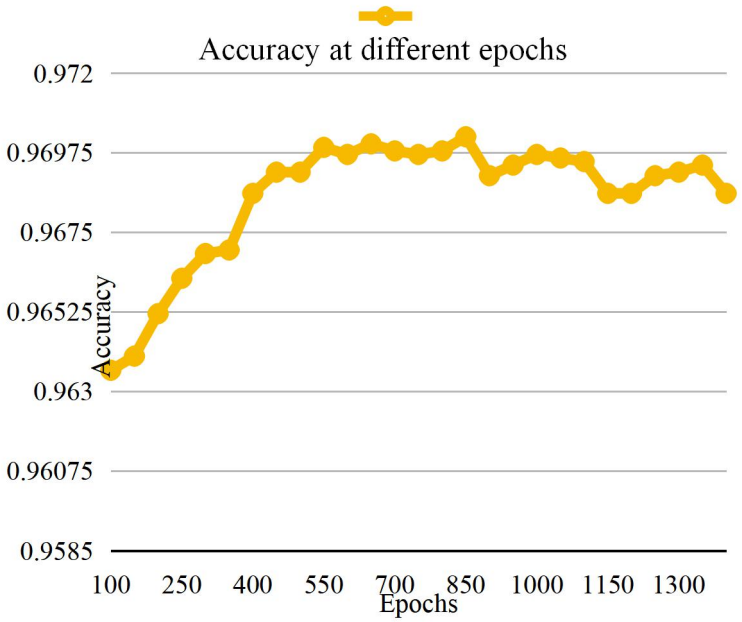
Selection of the number(X) of features in the hidden layer

Firstly, 5 experiment of different X are taken to decide the best X in this model. The testing results are as follows.



To keep the accuracy as high as possible, the number of features in the hidden layer was chose to be 100.

Selection of Epochs



```
Epochs: 100
Accuracy: 0.963532228837691
Runing time: 1.5597729682922363 seconds

Epochs: 150
Accuracy: 0.9662179653119337
Runing time: 2.2307238578796387 seconds

Epochs: 200
Accuracy: 0.9695508672016567
Runing time: 3.0859158039093018 seconds

Epochs: 250
Accuracy: 0.9695832254724307
Runing time: 3.8251547813415527 seconds

Epochs: 300
Accuracy: 0.9690978514108206
Runing time: 4.353606700897217 seconds

Epochs: 350
Accuracy: 0.9685477608076625
Runing time: 5.0218329429626465 seconds

Epochs: 400
Accuracy: 0.9685801190784364
Runing time: 5.765467643737793 seconds

Epochs: 450
Accuracy: 0.9682565363706963
Runing time: 6.648603677749634 seconds

Epochs: 500
Accuracy: 0.9682565363706963
Runing time: 7.699291706085205 seconds

Epochs: 550
Accuracy: 0.9684506859953405
Runing time: 8.3989737033844 seconds

Epochs: 600
Accuracy: 0.9681918198291484
Runing time: 9.411727666854858 seconds

Epochs: 650
Accuracy: 0.9700038829924928
Runing time: 9.36698579788208 seconds

Epochs: 700
Accuracy: 0.9698420916386228
Runing time: 10.891464948654175 seconds

Epochs: 750
Accuracy: 0.9697450168263008
Runing time: 11.713629245758057 seconds

Epochs: 800
Accuracy: 0.9698420916386228
Runing time: 11.84023404121399 seconds

Epochs: 850
Accuracy: 0.9701656743463629
Runing time: 12.652789115905762 seconds

Epochs: 900
Accuracy: 0.9690654931400466
Runing time: 13.565281867980957 seconds

Epochs: 950
Accuracy: 0.9693567175770127
Runing time: 14.431846857070923 seconds

Epochs: 1000
Accuracy: 0.9697450168263008
Runing time: 15.023342847824097 seconds
```

Consider both running time and accuracy, the best performance of this ANN model is approximately at epochs of 550. Higher epochs will lead to more time-consuming and overfitting.

2.3. Support Vector machines

Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and outliers detection. It has advantages of effective in high dimensional spaces and different kernel functions can be specified for the decision function.

In this project, the whole process of applying the support vector machine classifier (SVC) is similar to the DT classifier.

Extra tests were taken to decide the best kernel function and corresponding gamma for the classifier. The result is as follows.

Kernel = ‘rbf’			Kernel = ‘poly’			Kernel = ‘sigmoid’		
gamma	Accuracy	Running time	gamma	Accuracy	Running time	gamma	Accuracy	Running time
0.01	0.9649	2.187	0.01	0.9616	2.0493	0.01	0.905	2.2713
0.02	0.9663	1.9285	0.02	0.9628	2.3051	0.02	0.8733	3.8142
0.03	0.9672	2.0503	0.03	0.964	2.607	0.03	0.8609	4.2777
0.04	0.9676	2.0465	0.04	0.9642	3.3241	0.04	0.8569	4.7684
0.05	0.9679	2.0527	0.05	0.9642	4.4123	0.05	0.8548	4.9434
0.06	0.9679	2.3349	0.06	0.9642	5.5215	0.06	0.8536	5.1962
0.07	0.9679	2.2208	0.07	0.9641	6.791	0.07	0.8467	5.3686
0.08	0.9703	2.251	0.08	0.9644	10.5404	0.08	0.8391	5.7923
0.09	0.97	2.3592	0.09	0.9643	16.0989	0.09	0.8218	6.5661
0.1	0.9701	2.4623	0.1	0.9644	19.9634	0.1	0.803	7.3773

By observing the experiments’ results in the table, we found that the classifier had the highest accuracy when kernel=poly and gamma=0.08.

3. Requirements:

NumPy: NumPy is an open source numerical computation extension of Python. This tool can be used to store and process large matrices efficiently, supports a large number of dimensional arrays and matrix operations, and also provides a large library of mathematical functions for array operations.

Pandas: Pandas is a NumPy based tool created to solve data analysis tasks. Pandas incorporates a number of libraries and some standard data models, providing the tools needed to manipulate large data sets efficiently. In this project, we use it for read csv files.

Time: Time is a built-in python module for dealing with time issues, providing a series of functions for manipulating time. The function *time.time()* is used to get the timestamp, then calculate the running time of the program by simple addition and subtraction.

Scikit-learn: Sklearn is a free software machine learning library for the Python programming language. In this project, we use the *DecisionTreeClassifier*, *SVC*, *accuracy_score* and *export_graphviz* function.

Pytorch: PyTorch is an open source Python machine learning library based on The Torch for applications such as natural language processing. In this project, we use it to create an ANN model.

4. Results

4.1. Decision Tree

The running result of Decision tree is 0.9607 of accuracy and 0.0959 of running time on average.

The results of ten experiments of decision tree

n	Accuracy	Running time
1	0.9609	0.09986
2	0.9613	0.09537
3	0.9602	0.11744
4	0.9605	0.09755
5	0.9613	0.09409
6	0.9607	0.09668
7	0.96	0.08639
8	0.9607	0.08763
9	0.9606	0.09217
10	0.9606	0.09211
Average	**0.96068**	**0.095929**

```
Accuracy: 0.9608775847005145
Runing time: 0.09986305236816406 seconds
Accuracy: 0.9613306151506327
Runing time: 0.09537410736083984 seconds
Accuracy: 0.9602303983432029
Runing time: 0.11743712425231934 seconds
Accuracy: 0.9604892728861275
Runing time: 0.09755325317382812 seconds
Accuracy: 0.9613306151506327
Runing time: 0.0940859317779541 seconds
Accuracy: 0.9607481474290522
Runing time: 0.09668302536010742 seconds
Accuracy: 0.9600038831181439
Runing time: 0.08638787269592285 seconds
Accuracy: 0.9607157881111866
Runing time: 0.08762693405151367 seconds
Accuracy: 0.9606187101575899
Runing time: 0.09217190742492676 seconds
Accuracy: 0.9606187101575899
Runing time: 0.09211087226867676 seconds
```

4.2. ANN

After repeated experiments for ten times, the average accuracy of the ANN model (epochs = 550, hidden layer's feature = 100) is 0.9695, with an average running time of 8.5527 seconds.

The results of ten experiments of ANN

n	Accuracy	Running time
1	0.9696	8.9715
2	0.9691	8.3095
3	0.9696	8.4633
4	0.9696	8.4411
5	0.9692	8.5016
6	0.9693	8.565
7	0.9701	8.5605
8	0.9696	8.6292
9	0.9693	8.398
10	0.9693	8.6874
Average	0.96947	8.55271

```
Accuracy: 0.9695832254724307
Runing time: 8.97154188156128 seconds

Accuracy: 0.9690978514108206
Runing time: 8.309518098831177 seconds

Accuracy: 0.9696479420139787
Runing time: 8.463318109512329 seconds

Accuracy: 0.9695832254724307
Runing time: 8.441112041473389 seconds

Accuracy: 0.9691625679523687
Runing time: 8.50165605545044 seconds

Accuracy: 0.9693243593062387
Runing time: 8.56497597694397 seconds

Accuracy: 0.9700685995340409
Runing time: 8.560473680496216 seconds

Accuracy: 0.9695832254724307
Runing time: 8.62923288345337 seconds

Accuracy: 0.9693243593062387
Runing time: 8.3979971408844 seconds

Accuracy: 0.9692920010354646
Runing time: 8.687448024749756 seconds
```

4.3. SVC

After repeated experiments for ten times, the average accuracy of the ANN model (epochs = 550, hidden layer's feature = 100) is 0.9695, with an average running time of 8.5527 seconds.

n	Accuracy	Running time
1	0.9703	2.4487
2	0.9703	2.559
3	0.9703	2.355
4	0.9703	2.3213
5	0.9703	2.3144
6	0.9703	2.5699
7	0.9703	2.4734
8	0.9703	2.3367

9	0.9703	2.284
10	0.9703	2.3413
Average	0.9703	2.40037

```
kernal: rbf
Accuracy: 0.9702617868815325
Runing time: 2.448702812194824 seconds
kernal: rbf
Accuracy: 0.9702617868815325
Runing time: 2.558972120285034 seconds
kernal: rbf
Accuracy: 0.9702617868815325
Runing time: 2.355029821395874 seconds
kernal: rbf
Accuracy: 0.9702617868815325
Runing time: 2.3213250637054443 seconds
kernal: rbf
Accuracy: 0.9702617868815325
Runing time: 2.3144237995147705 seconds
kernal: rbf
Accuracy: 0.9702617868815325
Runing time: 2.5698790550231934 seconds
kernal: rbf
Accuracy: 0.9702617868815325
Runing time: 2.473356008529663 seconds
kernal: rbf
Accuracy: 0.9702617868815325
Runing time: 2.33670711517334 seconds
kernal: rbf
Accuracy: 0.9702617868815325
Runing time: 2.2839419841766357 seconds
kernal: rbf
Accuracy: 0.9702617868815325
Runing time: 2.3412506580352783 seconds
```

5. Comparison and discussion

5.1. Comparison

	Accuracy	Running time
DT	0.9607	0.0959
ANN	0.9695	8.5527
SVC	0.9703	2.4004

The accuracy of ANN and SVC model is less than 1% higher than DT classifier, but the ANN’s running time is more than 800% longer than DT. By comparison, SVC can obtain more accurate classification results in a relatively short time. Generally speaking, SVC performs better.

5.2. Discussion

These three classifiers all have good accuracy, but There are still many parameters that I haven't compared. If I have more time, I hope to try more parameters to achieve faster and more accurate classification result.

At the same time, I have heard that ANN also has some visualization methods, and I hope to have the opportunity to apply them to my classifier through learning. In this case, I can not only predict the result from a dataset but also know how the correlation between the given features and the final result are built.