**Project 1**

**Classification of unlabeled LoL match records with “Win/Loss”**

Name: 叶迪锴

Student ID: 201930650154

**Introduction**

As we are in a digital society, we live in the era of big data, how to let machines use big data to learn, classify and perform big data analysis tasks is particularly important. We can train classifiers to use existing data to predict and make reasonable plans. In this project, we are going to predict which team win the LOL game. In this project, I will have access to about 3 Million match records of solo gamers as training set. Each record comprises of all publicly available game statistics of a match played by some gamer, including an important field called "winner". If "winner" is “1”, the team 1 won the match, or vice versa. In sum, the fields include: Game ID, Creation Time (in Epoch format), Game Duration (in seconds), Season ID, Winner (1 = team1, 2 = team2), First Baron, dragon, tower, blood, inhibitor and Rift Herald (1 = team1, 2 = team2, 0 = none), The number of tower, inhibitor, Baron, dragon and Rift Herald kills each team has. My task is to create one or more classifiers that take as inputs from any fields from above (except “winner) such match record, and labels this record as a "1" or a "2". The test set comprises of ~2 Million of such records.

**Methodology**

In this project, we are going to predict which team win the LOL game which means the accuracy is the most important results and we have to find the best solution. In order to find the most effective classifier, we need to train each classifier with the same dataset. Then, we should predict a value of 1 or 2 and calculate the accuracy. In class, we have discussed the different accuracy of many classifiers such as Decision tree, SVM, K-NN, MLP and so on. In my opinion, Decision tree is a better choice because it can make feasible and good results for large data sources in a relatively short period of time. To be innovative, I want to see the prediction of Regression and compare with other classifications. Therefore, I choose Decision tree Classifier, K nearest neighbour Classifier, Multi-Layer Perceptron and Logistic Regression Classifier to compare their advantages or disadvantages in terms of this dataset, by using the Accuracy and Training time.

**Algorithms**

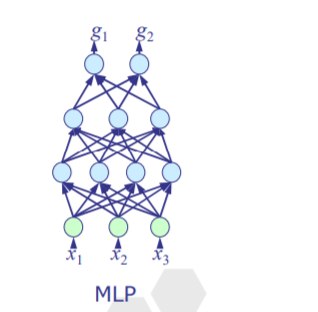
Decision Tree: The algorithm of decision tree learning is usually a process of recursively selecting the optimal feature and segmenting the training data according to the feature, so that each sub data set has a best classification process. It includes feature selection, decision tree generation and pruning process. We use parameters ‘criterion’ as ‘entropy’ which is one of the feature selections and ‘random\_state=85’ to determine the randomness of feature selection.

1. criterion{“gini”, “entropy”}, default=”gini”. The function to measure the quality of a split. Supported criteria are “gini” for the Gini impurity and “entropy” for the information gain.
2. splitter{“best”, “random”}, default=”best”. The strategy used to choose the split at each node. Supported strategies are “best” to choose the best split and “random” to choose the best random split.
3. max\_depthint, default=None. The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min\_samples\_split samples.
4. random\_stateint, RandomState instance, default=None. Controls the randomness of the estimator. The features are always randomly permuted at each split, even if splitter is set to "best". When max\_features < n\_features, the algorithm will select max\_features at random at each split before finding the best split among them. But the best found split may vary across different runs, even if max\_features=n\_features. That is the case, if the improvement of the criterion is identical for several splits and one split has to be selected at random. To obtain a deterministic behaviour during fitting, random\_state has to be fixed to an integer.

Logistic Regression: Logical regression is such a process: in the face of a regression or classification problem, the cost function is established, and then the optimal model parameters are iteratively solved by optimization method, and then the quality of our solved model is verified by testing. We use parameter ‘solver’ as ‘lbfgs’, which will deal with multiple losses for multi class problems.

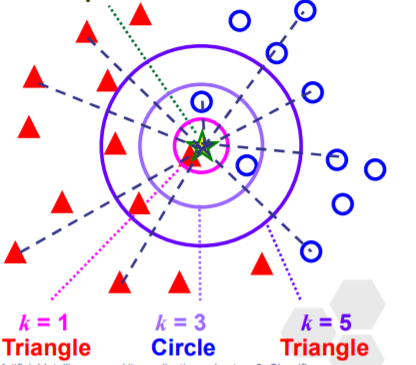
Multi-Layer Perceptron: The multilayer perceptron is based on the feed forward artificial neural network. The multi-layer sensor contains multi-layer nodes, and each layer node is fully connected with the next layer node of the network. The nodes in the input layer represent the input data, and the nodes in other layers can obtain the output of the layer by linearly combining the input data with the weight W and deviation B of the nodes on the layer and applying an activation function.

1. activation: str, activation function for the hidden layer, “relu” by default.
2. hidden\_layer\_sizes: tuple, the ith number represents the number of neurons in the ith hidden layer.
3. batch\_size: int, decide how many samples are used for one update, min(200, n\_samples) by default.



K Nearest Neighbors: The core idea of the KNN algorithm is that if most of the k nearest samples in the feature space belong to a certain category, then the sample also belongs to this category and has the characteristics of the samples in this category. In the decision-making of classification, the method only determines the category of the sample to be classified according to the category of the nearest one or several samples.  We use parameter ‘n\_neighbors = 3’ which means k=3 as mentioned above.

1. n\_neighbors: Number of neighbors to use for kneighbors queries.



**Requirements**

We will use some prerequisite packages such as pandas, sklearn and time.

Pandas is a powerful tool set for analyzing structured data. It is based on numpy (providing high-performance matrix operation) and used for data mining and data analysis, and also provides data cleaning function.

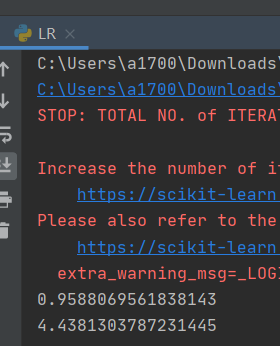
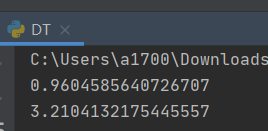
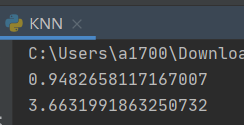
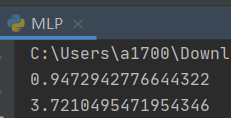
Scikit learn is also known as sklearn, which is one of the most famous Python modules in the field of machine learning. Sklearn includes a number of machine learning ways: Classification, Regression, Clustering, Dimensionality reduction, Model Selection and Preprocessing.

Time provides the function of getting the system time and formatting the output. The system level precise timing function is provided for program performance analysis.

**Results**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | LogisticRegression | DecisionTree | KNeighbors | MLP |
| **Accuracy** | **0.9588069561838143** | **0.9604585640726707** | **0.9482658117167007** | **0.9472942776644322** |
| **Time** | **4.4381303787231445** | **3.2104132175445557** | **3.6631991863250732** | **3.7210495471954346** |
| **γ** | **0.2160384834** | **0.2991697638** | **0.2588627491** | **0.2545771739** |

γ=Accuracy/Time

**Comparison and discussion**

We first load the game statistics of match records and we need to create a matrix containing independent variables and a vector of dependent variables. Then we create target columns such as 'gameId', 'creationTime', 'gameDuration', 'seasonId', 'winner' and so on and check target incidence whether the data is missing or the checking the winner team variable is 1 or 2. Since the factors of determining the winner are independent of game ID, creation time and so on, so we have to delete these data. Next, we split training data into train and test datasets, train the Decision tree, Logistic Regression, Multi-Layer Perceptron, K Nearest Neighbors classifiers. After that, we input the test data and predict, finally compare the result. It is of importance to select appropriate parameters for they decide the accuracy of the prediction. We have to try different parameters to determine which one fits well. What’s more, the running time of a program also counts a lot and we should consider it. We expect the higher accuracy and lower time consume as much as possible, so I set a function as γ=Accuracy/Time. The higher γ is, the better model and parameters are. After dozens of experiments, I find out the most suitable parameters and draw the table containing the results which have shown above.

As we can see, the highest value of γ is Decision Tree Classifier, which means it is high-accuracy and less time-consume classifier. However, if more time is allowed, there is beyond all doubt that we can improve the accuracy of program. Because we may not consider the factor of time consume and only focus on the accuracy. It is much easier to improve a program when we only consider one factor, for example, we can change KNN parameter ‘n\_neighbors = 3’ to ‘n\_neighbors = 5’ or ‘n\_neighbors = 7’ which performs better in accuracy but poorer in running time. Moreover, we can consider using ensemble such as boosting or bagging to deal with problem

**Conclusion**

In conclusion we may note that Decision tree works better than other classifiers like Logistic Regression, Multi-Layer Perceptron and K Nearest Neighbors in this classification task. When dealing with the similar problem, we should not only consider one factor but also others. Different dataset and different task may need different solution and we should consider comprehensively. Sometimes changing parameters or using ensemble instead of single classifier will lead to better outcome. Just like an old saying goes, real knowledge comes from practice.