**Project 1**

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

He Qiuhong 201930650079

**Problem introduction**

1. Overview

League of Legends (LoL) is one of the most played eSports in the world at the moment. Arguably, the fun aspect of games or sport lies within the uncertainty of their outcomes. There is nothing more boring than playing or watching a game, be it soccer, basketball, or any video games, with a predictable ending. To make the game more fun to watch, the creators of LoL have tried their best to match players with teammates and opponents of as similar skill level as possible. In this project, I will have access to about 30,000 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 several fields called *“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”*.

1. Objectives, Requirements and Methodology

My objective is to create several classifiers using the fields from the training set (except “winner”) as inputs, and use the well-learned classifiers to label the test dataset as a "1" or a "2". Generally speaking, the goal is to identify which team wins the match (“1” or “2”) in the given test samples using the previous training dataset. Also, the trained classifier is the intermediate between the 2 datasets, and making it achieve a high accuracy is one of my objectives. Creating more classifiers initially can help to select the best classifier. The requirement of the project may involve implementing at least one classification method and successfully evaluate the test dataset using a python program. The evaluation result should at least be higher than 50%. Python and libraries like *Sklearn* and *Pytorch* are available to be used to realize the implementation. As for the methodology, in brief, is using the machine learning models (classifiers) in *Sklearn* to learn and classify the processed training dataset, and using the trained classifiers to label the test dataset. Also, the output of the classifiers’ accuracies and training times are included in the program.

**Algorithms**

The classification methods used in this project are DT, KNN and SVC.

1. Decision Tree

Classification decision tree model is a tree structure and composed of nodes and directed edges. There are two types of nodes: internal nodes and leaf internal nodes represent a feature or attribute, and leaf node represents a class. The internal nodes are the attributes in the dataset and the leaf are the basis of distinguishing the data and finally arrive the result. This algorithm uses the characteristics of the training set to form a tree diagram, and the structure is as mentioned above. Decision tree algorithms mainly include ID3, C4.5 and CART, and attribute selection is carried out according to the following feature selection methods: Entropy and Gini Index.

Important parameters:

max\_depth: The maximum depth of the decision tree. If this para is too small, the accuracy will be lower. Besides, the bigger the max\_depth is, more time is needed to train the classifier. The default value of this param is none, which means there is no limitation.

Criterion: It can be gini or entropy. This is to measure the quality of the split. gini is related to the Gini impurity and entropy for the information gain.

1. K-NN:

The whole name of the algorithm is K Nearest Neighbors. The principle of KNN is to determine which category x belongs to when predicting a new value x according to the category of the nearest K points. The value of K is very important. Generally speaking, the undetermined point is as same as the kind which has the most points in the scope in the nearest K points.

Important parameters:

n\_neighbors: It determines the number of the neighbors to use. Values that are too large or too small will affect accuracy. Thus, this param is the most important one of K-NN algorithm.

algorithm: It is used to choose the tree type as it can be ‘auto’, ‘ball\_tree’, ‘kd\_tree’, ‘brute’. Generally, you just need to set it as auto if you needn’t specify the type.

1. SVM:

The whole name is Support Vector Machine. It is a generalized linear classifier for binary classification of data. The decision boundary is the maximum margin hyperplane for solving the learning sample. Popularly speaking, we can input the X = {X1, X2, X3…Xn} and y = {y1, y2, y3...yn} and each sample contains multiple features and from that a feature space is formed. And we can know the all samples are distributed in a space. We can draw a line (linear or nonlinear) between the sample points. And then split the all data into two (or more) classes. The line is called hyperplane. If the feature space where the input data is located exists as the hyperplane of the decision boundary, the learning target is separated according to the positive class and the negative class, and the distance between the point and the plane of any sample is greater than or equal to 1, It is said that the classification problem is linearly separable. The decision boundary that meets this condition actually constructs two parallel hyperplanes as the separation boundary to determine the classification of the sample. The full line is decision boundary and the dotted-line is interval boundary. The distance between two interval boundaries is 2/||w||. The positive and negative samples located on the boundary of the interval are support vectors. Thus, that’s why it’s called SVM.

Important parameters:

kenel: It is a string. It is used to specifies the kernel type. It can be ‘linear’, ‘poly’, ‘rbf’, ‘sigmoid’ or ‘percomputed’. If none is given, ‘rbf’ will be used. If a callable is given it is used to pre-compute the kernel matrix from data matrices.

gamma: Kernel coefficient for “rbf”, “poly” and “sigmoid”. It can be ‘scale’ and ‘auto’. If gamma='scale' (default) is passed then it uses 1 / (n\_features \* X.var()) as value of gamma, if gamma=‘auto’, uses 1 / n\_features.

**Experimental Requirements**

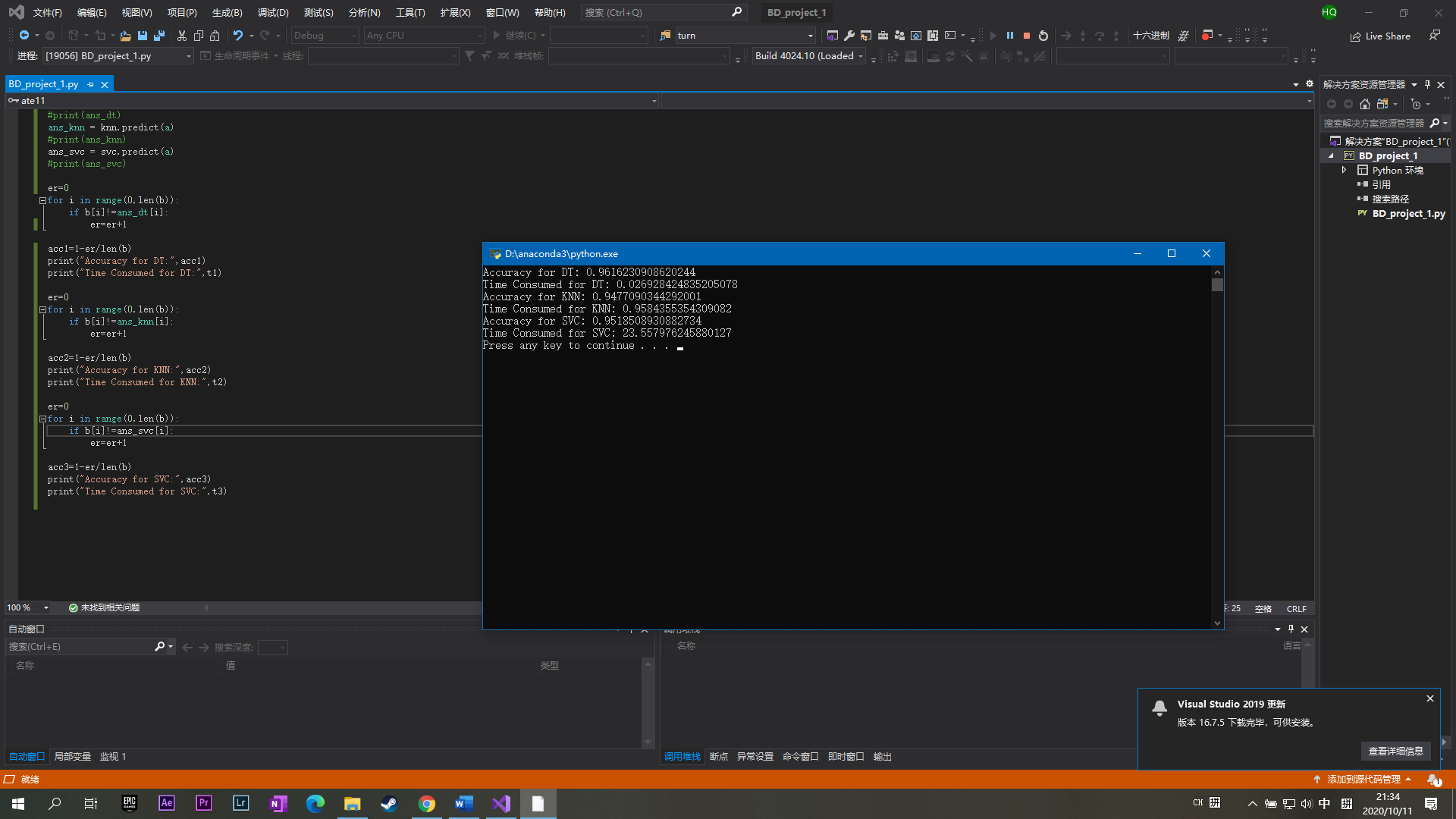
The prerequisite packages in the code are:

1. sklearn (including sklearn.model\_selection, sklearn.tree, sklearn.neighbors and sklearn.svm)
2. numpy
3. pandas
4. time

**Experimental Results**

The results when using the classifiers to classify the test dataset:

|  |  |  |
| --- | --- | --- |
| Classifiers | Accuracy | Time(s) |
| DT | 0.9616230908620244 | 0.026928424835205078 |
| KNN | 0.9477090344292001 | 0.9584355354309082 |
| SVC | 0.9518508930882734 | 23.557976245880127 |



**Comparison and Discussion**

1. The Parameters of the Classifiers

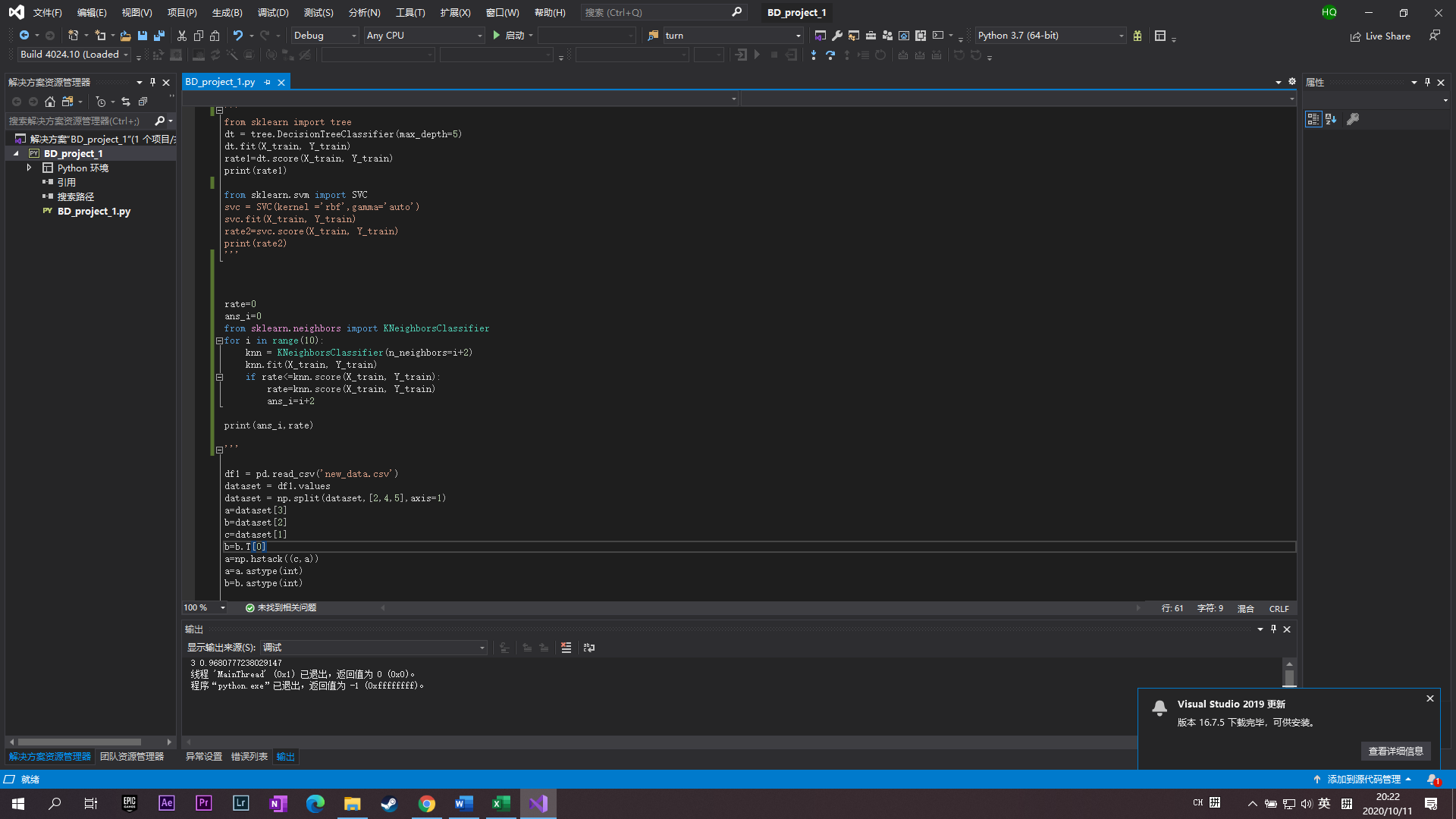
About the selection of the parameters for classifiers, is to change each parameter gradually in a for loop, then select the parameter which runs out the highest accuracy. Since the *train\_test\_split()* function splits the data randomly, it is needed to be tested several times to minimize the error. Here are some tests:

1. Decision Tree:

As the *max\_depth* increases, the accuracy increases rapidly initially. However, in order to avoid overfitting, here choose 13 as the value of parameter *max\_depth*.

1. KNN:

As the test and results showed in the following, after 5 tests, the result for *n\_neighbors* are all 3, which achieve the highest accuracy. Thus choose 3 as the value of parameter *n\_neighbors*.



|  |  |
| --- | --- |
| n\_neighbors (ans\_i) | Accuracy (rate) |
| 3 | 0.9692574600971547 |
| 3 | 0.9685634975711311 |
| 3 | 0.9692574600971547 |
| 3 | 0.9680777238029147 |
| 3 | 0.9684941013185288 |

1. SVC:

Here, use default parameters.

1. The Pretreatment of the Training Dataset

In this program, the training dataset is in a big .csv file, so after inputting the data using function *pandas.read\_csv(),* it is important to pretreat it.

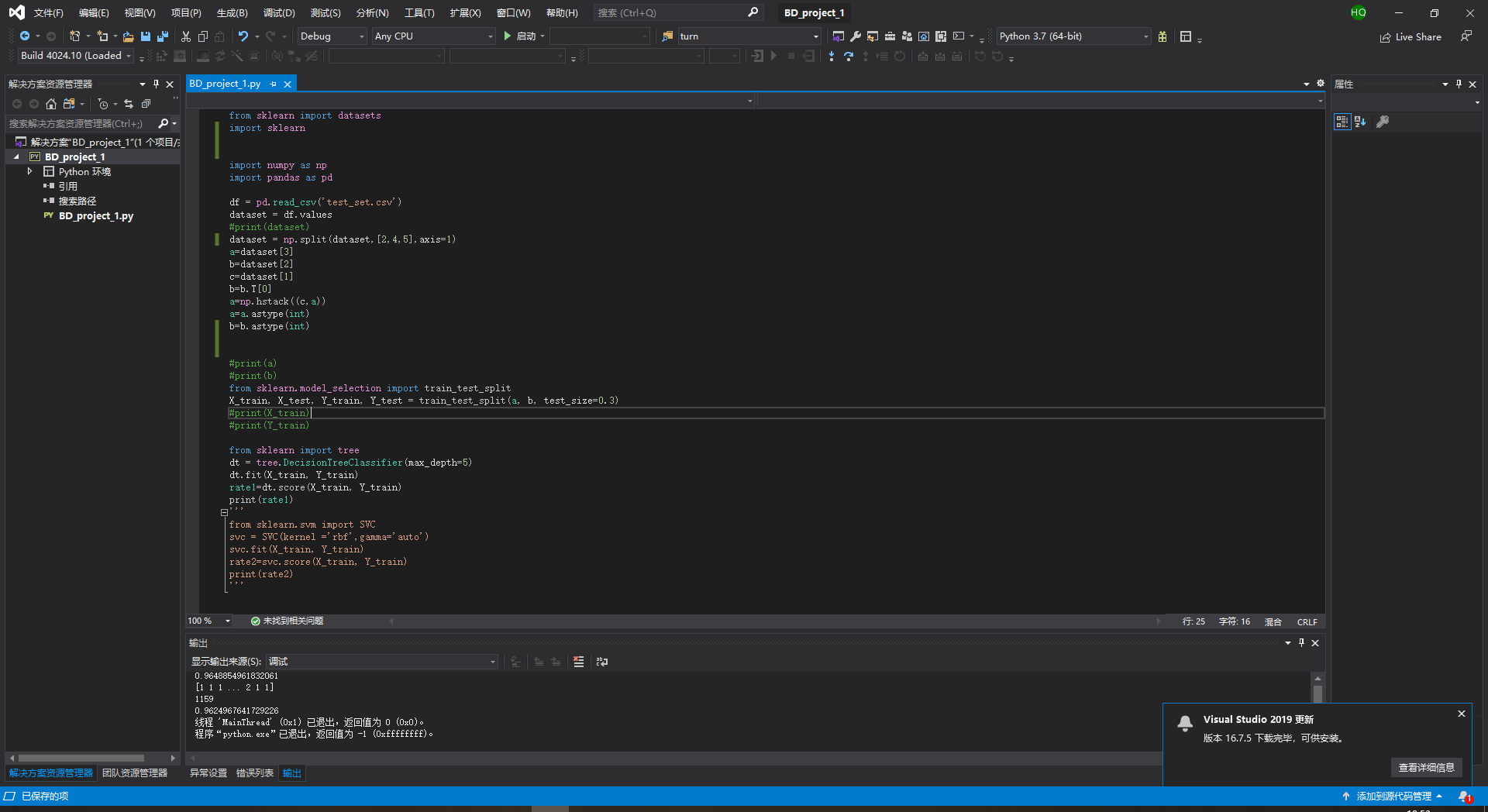
1. While observing the format of the storage of the data, I found all the attributes are arranged in columns and the label representing the winner is in the 5th column. The Pandas library reads the entire data, while the learning of the classifiers from data needs the attributes and label separately, thus the function *np.split()* is needed to splits the data in column.
2. Analyzing the content of the data, I found the data in the first 2 columns are in float type, which are the “Game id” and “Creation Time”. Because the precision of float is bigger than int, the whole array is stored in float type. Looking in the attributes, “Game id” and “Creation Time” are independent of other attributes, and so do the label, so these 2 columns can not affect the outcomes. In the program, the first 2 float type columns are removed and the other attributes are transformed to integer before learning. Since the original data in these attributes are int type, so this transformation will not cost information loss, and integer is convenient for classifiers to learn and also saves the computing resources.
3. The following figure shows the pretreatment of the training data.

*dataset*: the original data then the data after splitting.

*a*: the 6th to the last column then the new attributes combined with c

*b*: the label in the 5th column then transformed to a row

*c*: the 2nd and 3rd column



1. The Comparison Between Classifiers

According to the results, among the three classifiers, DT has the shortest training time and the highest accuracy, KNN has the lowest accuracy and SVM has the longest training time. If only consider a single classifier, in this case, the DT classifier is the best, since it cost the lowest time to achieve the highest accuracy, which shows a high efficiency.

1. The Experience Gained
2. Being familiar with the use of these various packages.
3. Being familiar with the use of cutting the array and reshape it.
4. Learned the principles of each classifier, as well as the advantages and disadvantages.
5. Know what types of data are suitable for classifier correspondingly.
6. Being familiar with the test of accuracy and parameters.
7. After this learning, can have the ability to find data on the Internet and classify it according to needs.
8. Further Improvement
9. Introduce more classification methods to the program, like ANN, to increase the accuracy. Also, can use all the 3 classifiers to make decision. For example, suppose each classifier have accuracy of 0.9, and the error rate is 0.1. Then (0.13+3\*0.9\*0.12) is the error rate of the three classifiers, which is 0.028, smaller than 0.1. With further information, it can even give higher weight to classifier with the higher accuracy, instead of averaging them.
10. Can make a program to predict the win rate in real time. This may need more data: the changes of each attributes with the time of the game increasing, in order to predict the rate of winning at each single time of the game.

**Summary**

During the project, *Classification of unlabeled LoL match records with “Win/Loss”,* I learn a lot of techniques of python and gain a better understanding of machine learning and classification methods. More importantly, is to maintain an attitude and enthusiasm for learning.