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

Introduction

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. Similarly, the creators of LoL have tried their best to match players with teammates and opponents of as similar skill level as possible, so that it can make the result more fascinating.

To be able to accurately predict the outcome of the game, this project has encapsulated four different classifier models for training the training set to predict the result of games, which are the decision tree model, artificial neural networks model, k nearest neighbor model and multi-layer perceptron model. By comparing these four models with the accuracy and the efficiency of training, the optimal one can be used in practice.

Noting that some requirements are contained to complete the project. It has access to enough match records of solo gamers as training set and test set. Each record comprises of all publicly available game statistics of a match played by some gamer. All the program must be support by python.

Algorithms

Dataset preprocessing is necessary in the program before it defines classifier models. Python has prefect data capture model and is convenient to call from the library named pandas. The function, called pandas.read_csv (“file path”, header=None, names=col_names), can capture the dataset from the file path specified with its names of columns defined. Extracting the data set by column names, the feather dataset and the label dataset are packaged accurately. The other useful built-in model called sklearn.model contains a function named train_teat_split(feather set, label set,

test_size, random state), whose parameters mean it will split the set into a training set and a test one separately by a defined random state and test size. In this project, the test size is 0.3 and the random state is 1, with which the dataset is split randomly and be capable to test the model after training before using to predict the default test set. All the dataset converts to the value type for use.

With the datasets are already, the classifier models are time to define. The decision tree model depends on the sklearn.tree model, containing the decision tree classifier definition, the interface function for passing data to the model, the prediction function for the result of training model and so on. The project defines the decision tree classifier with the parameters of entropy criterion and 4 of maximum depth.

Moreover, the next model, artificial neural networks model, depends on pytorch library. The dataset of value type in this model must be converted to float type by numpy model function called .astype(float) and transformed to tensor format by torch.FloatTensor(dataset). The artificial neural networks classifier contains three layer and with fully connected. Pytorch uses object-orientated way of declaring models, by which it defines the input feather nodes as 16, which is the number of the extracted feathers before, one hidden layer only with 264 nodes which is enough to train the model in this project when taking account of accuracy and efficiency and the output nodes as three in the constructor. The forword() method defines a forward pass. With declaring the parameter named criterion as cross entropy loss and using the optimization algorithm and Adam with a learning rate of 0.01 to make an optimizer, the model can make an instance of the model and match its architecture as declared. In the project, the artificial neural networks model will train the model for 100 epochs, keeping track of time and loss. Every 2 epochs will be print to the console the current status to indicate the current epoch and loss. The classifier result should be got by applying the softmax function, whose dimension equals 1.

As for the multi-layer perceptron model, it depends on the sklearn.neural_network model. In this classifier model, the algorithm get help from the grid search method to find the best parameters for the project from sklearn.model_selection model. After training and making the parameters visual, the multi-layer perceptron model selects

parameters that activation function of Rectified Linear Unit, hidden layer sizes of 100, learning rate of invscaling algorithm and maximum iteration of 100. While the maximum iteration is too small to make the optimization converge, the accuracy of the training is high and the small iteration is more efficiency. It's not that the more complex the neural network, the better. And it seems that the model is not unfitted or overfitted. Hence, the maximum iteration selects 100 in this project.

The last but not lease one is the k nearest neighbor model, which depends on the sklearn.neighbors model. It also gets help from the grid search method to find the best parameters for the project from sklearn.model_selection model. After training the k nearest neighbor classifier and making the parameters visual, the model selects parameters that the weight of uniform, 5 of neighbors, and distance measure of 5. It makes trade-offs among the algorithms of brute, kd_tree and ball_tree, and choose the best algorithm that fits best.

All of the models must be recorded the training time by time function from time model. And print the training time in the console. Otherwise, the training accuracy and the parameters of the models are also outputted in the console for monitoring the models visual.

After training the classifiers with the best parameters, there are four functions packaging the prediction method by each classifier model. Call these functions by incoming the default test dataset. The prediction functions make use of the accuracy score function named accuracy_score(testset label, prediction label by decision tree model), to evaluate the accuracy. While the prediction function of the artificial neural networks model predicts the labels by the method of torch. Others predict the labels by .predict(default test date set) function. All of the prediction functions above will print the prediction results and the accuracy of the test data set in the console.

Requirements

Model	Function	Parameters(main)	Description

time	time()	null	record the time
pandas	Read_csv()	file path, header, names	load the data set
sklearn.model_selection	train_set_split()	feather set, test_size, random_state	split the dataset into training set and testing one
sklearn.metrics	accuracy_score()	label set, prediction set	calculate the accuracy
sklearn.tree	DecisionTreeClassifier()	criterion, max_depth	decision tree classifier
sklearn.neural_network	MLPClassifier()	activation, hidden_layer_sizes, learning_rate, max_iter	multi-layer perceptron classifier
sklearn.model_selection	GridSearchCV()	classifier, parameters grid	grid search assignment
sklearn.neighbors	KNeighborsClassifier()	weights, n_neighbors, p	k nearest neighbors classifier
torch	FloatTensor()	feather data set	convert the float type data to tensor

			format
Torch	optim.Adam()	model.parameters	define the optimizer
torch.nn	Linear()	in_feathers, out_feathers	define the numbers of the feather nodes
torch.nn	CrossEntropyLoss()	null	the loss function of cross entropy loss
torch.nn.functional	softmax()	dimension	output the probability of category

Results

The results of the decision tree model:

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='entropy',
                       max_depth=4, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
Training accuracy of the DT classifier : 0.9512629533678757
Training time of the DT classifier: 0.18944454193115234 Seconds

Testing accuracy of the DT classifier : 0.9501359047372508
```

The results of the artificial neural networks model:

```
ANN(
  (fc1): Linear(in_features=16, out_features=264, bias=True)
  (output): Linear(in_features=264, out_features=3, bias=True)
)
Training accuracy of the ANN classifier : 0.9648639896373057
Training time of the ANN classifier: 6.841701984405518 Seconds
Testing accuracy of the ANN classifier :  0.9623996893606006
```

The results of the multi-layers perceptron model:

```
Epoch: 0 Loss: 1.1403785943984985
Epoch: 2 Loss: 0.7895199060440063
Epoch: 4 Loss: 0.6649330854415894
Epoch: 6 Loss: 0.6242551207542419
Epoch: 8 Loss: 0.607668399810791
Epoch: 10 Loss: 0.6013743877410889
Epoch: 12 Loss: 0.5978124141693115
Epoch: 14 Loss: 0.5955575704574585
Epoch: 16 Loss: 0.5943725109100342
Epoch: 18 Loss: 0.5934596061706543
Epoch: 20 Loss: 0.5926176309585571
Epoch: 22 Loss: 0.5920199751853943
Epoch: 24 Loss: 0.5916004776954651
Epoch: 26 Loss: 0.5911799669265747
Epoch: 28 Loss: 0.5907723307609558
Epoch: 30 Loss: 0.5904595851898193
Epoch: 32 Loss: 0.5901973843574524
Epoch: 34 Loss: 0.5899263620376587
Epoch: 36 Loss: 0.5896714329719543
Epoch: 38 Loss: 0.5894623398780823
Epoch: 40 Loss: 0.5892752408981323
Epoch: 42 Loss: 0.5890858173370361
Epoch: 44 Loss: 0.5889163017272949
Epoch: 46 Loss: 0.5887718200683594
Epoch: 48 Loss: 0.5886316895484924
Epoch: 50 Loss: 0.5884966254234314
```

```
Epoch: 52 Loss: 0.5883802771568298
Epoch: 54 Loss: 0.5882694125175476
Epoch: 56 Loss: 0.5881597399711609
Epoch: 58 Loss: 0.5880606770515442
Epoch: 60 Loss: 0.5879653692245483
Epoch: 62 Loss: 0.587874710559845
Epoch: 64 Loss: 0.5877875685691833
Epoch: 66 Loss: 0.5877053141593933
Epoch: 68 Loss: 0.5876243114471436
Epoch: 70 Loss: 0.5875474810600281
Epoch: 72 Loss: 0.5874736309051514
Epoch: 74 Loss: 0.587403416633606
Epoch: 76 Loss: 0.5873377323150635
Epoch: 78 Loss: 0.5872731804847717
Epoch: 80 Loss: 0.5872107744216919
Epoch: 82 Loss: 0.5871515274047852
Epoch: 84 Loss: 0.5870943069458008
Epoch: 86 Loss: 0.5870391130447388
Epoch: 88 Loss: 0.586985170841217
Epoch: 90 Loss: 0.586932897567749
Epoch: 92 Loss: 0.5868824124336243
Epoch: 94 Loss: 0.5868313312530518
Epoch: 96 Loss: 0.5867824554443359
Epoch: 98 Loss: 0.5867334008216858
```

```
GridSearchCV(cv=None, error_score=nan,
             estimator=MLPClassifier(activation='relu', alpha=0.0001,
                                     batch_size='auto', beta_1=0.9,
                                     beta_2=0.999, early_stopping=False,
                                     epsilon=1e-08, hidden_layer_sizes=(100,),
                                     learning_rate='constant',
                                     learning_rate_init=0.001, max_fun=15000,
                                     max_iter=200, momentum=0.9,
                                     n_iter_no_change=10,
                                     nesterovs_momentum=True, power_t=0.5,
                                     random_state=None, shuffle=True,
                                     solver='adam', tol=0.0001,
                                     validation_fraction=0.1, verbose=False,
                                     warm_start=False),
             iid='deprecated', n_jobs=None,
             param_grid={'activation': ['relu'], 'hidden_layer_sizes': [(100,)],
                         'learning_rate': ['invscaling'], 'max_iter': [100]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring=None, verbose=0)
```

```
Training accuracy of the MLP classifier : 0.9718264248704663
Training time of the MLP classifier: 75.78854513168335 Seconds
Testing accuracy of the MLP classifier : 0.9683859694537924
```

The results of the k nearest neighbors model:

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                     weights='uniform')
KNNaccuracy:  0.9638924870466321
Training time of the KNN classifier: 5.1631903648376465 Seconds
Testing accuracy of the KNN classifier : 0.963952886357753
```

Comparison and discussion

The decision tree can process nominal and numerical data at the same time, so that it is more suitable for processing samples with missing attributes and be able to deal with irrelevant features. When testing the data set, the running speed is relatively fast, because it can be able to make feasible and effective results for large data sources in a relatively short time. By comparing the training time of the four models, it is obviously the decision tree model requires the minimal time for training model. However, overfitting easily occurs and it is easy to ignore the correlation of attributes in the data set. For those data with inconsistent sample sizes in each category, in the decision tree, when dividing attributes, different judgment criteria will bring different attribute selection tendencies; the information gain criterion has a preference for attributes with a larger number of desirable, and the gain rate criterion CART has a preference for a smaller number of desirable attributes, but CART no longer simply uses the gain rate to divide the attributes directly, but uses a heuristic rule. As long as the information gain is used, it has this disadvantage. The improvement measures can be as follow. Prune the decision tree using cross-validation and regularization methods. Or using combination algorithms based on decision trees, such as bagging algorithm and random forest algorithm, can solve the problem of overfitting.

The artificial neural networks model has high accuracy. Unlikely the decision tree model, it is suitable for noisy data sets. However, it trains slow because there are too many parameters to adjust. For avoiding the mode being overfitting, regularization and dropout can help without simply reducing the number of layers and the number of neurons in each layer.

The training time complexity of the k nearest neighbor is low and insensitive to abnormal points. Since the KNN method mainly relies on the surrounding limited nearby samples, rather than the method of discriminating the class domain to determine the category, the KNN method is better than the other sample sets to be divided for the cross or overlap of the class domain. Method is more suitable. Otherwise, this algorithm is more suitable for the automatic classification of class domains with a relatively large sample size, and those with a small sample size are

more likely to be misclassified using this algorithm. While there are also some of disadvantages should be considered. The amount of calculation of it is large, especially when the number of features is very large. When the sample is unbalanced, the prediction accuracy of the rare category is low. Using lazy learning methods, basically not learning, resulting in slow predictions. Compared with the decision tree model, the KNN model is less interpretable. Although k nearest neighbors algorithm has some of shortcoming, condensing technology and editing technology can greatly reduce the training sample size while maintaining classification accuracy. Comprehensively considering the advantages and disadvantages of each classifier, the project adjusts the parameters of each classifier model carefully and gains high accuracies, which are all more than 0.95 in not only the training process and the testing processing. But for the shortcomings of each classifier, there are still many areas to be improved in the future.

Summary

The project has basically completed its goal of accurately predicting the result of the game. It also analyzes the efficiency and the accuracy of the decision tree classifier model, the multi-layer perceptron model, the artificial neural networks model and the k nearest neighbors model. The accuracy of four models are all more than 0.95. Especially, the decision tree model has a high efficiency. Through continuous improvement of the model, I believe it can be applied to practice sooner or later.