AU 332 Artificial Intelligence: Principles and Techniques

By: Xiaocheng Wang(517030910326) Shixuan Gu(517030910314)

HW#: 5

December 15, 2019

I. INTRODUCTION

In this project, we will apply some machine learning algorithms to solving the forest categories classification problem with the help of scikit-learn.

II. PROBLEM DESCRIPTION & FORMULATION

The forests can be classified into seven categories based on some observed features. The given data includes 54 feature fields and one label field. This is a typical classification problem that can be sovled by several well-established machine learning methods, such as logistic regression, support vector machine, KNN, decision tree and so on. Therefore, we compare different algorithms and attempt to tune the hyper-parameters for higher accuracy and lower computation consumption.

III. EXPERIMENT & RESULTS

A. Comparing algorithms

1. Testing k-fold validation without preprocessing data

Since we are comletely unaware of which algorithms perform better, we first make a rough estimate simply using the defaulting parameters of different algorithms on the original dataset. We test different numbers of k-fold as well. TABLE I shows the mean accuracy for each model and each choice of k-fold.

Algorithm \ k-fold	3	4	6	7	8	9	11	12	13	14
Logistic Regression	0.4815	0.4787	0.4794	0.4851	0.4825	0.4815	0.4833	0.4778	0.4804	0.4830
Gaussian Naive Bayes	0.5925	0.5912	0.5943	0.5921	0.5918	0.5927	0.5931	0.5928	0.5926	0.5916
C-Support Vector Classification	0.1404	0.1382	0.1384	0.1357	0.1351	0.1337	0.1348	0.1308	0.1327	0.1318
Multi-layer Perceptron	0.6369	0.6162	0.6022	0.6279	0.5652	0.6083	0.6015	0.6156	0.5958	0.6150
Decision Tree	0.7767	0.7831	0.7881	0.7919	0.7936	0.7966	0.7958	0.7931	0.7937	0.7915
K-nearest neighbors(n_neighbors=4)	0.7974	0.8092	0.8142	0.8152	0.8181	0.8196	0.8213	0.8205	0.8218	0.8219
Gradient boosting	0.7951	0.7953	0.7976	0.7973	0.7964	0.7960	0.7991	0.7971	0.7966	0.7976
Random Forest(n_estimators=110)	0.8539	0.8581	0.8606	0.8634	0.8623	0.8654	0.8658	0.8652	0.8668	0.8650

TABLE I: Comparing models with default parameters (except KNN and Random Forest)

We can infer that Decision Tree, KNN, Gradient boosting and Random forest have better capability to predicting accurate results.

Also, we find that the train-test-spliting of data makes difference. If we split dataset into k consecutive folds (the same order of original dataset), since the distribution of different categories may not be balanced, the model could be underfitted or overfitted. Therefore, if we use KFold of sklearn, the shuffle parameter should be set True, thus splitting the data randomly.

2. Preprocessing data

The dataset may need preproccessed since the various fields of features. Therefore we use normalization, standardation and binarization to preprocess features respectively. TABLE II demonstrates the result.

Algorithm \ k-fold	5				10			
	original	normalized	standardized	binarized	original	normalized	standardized	binarized
Logistic Regression	0.4804	0.3752	0.7068	0.6153	0.4830	0.3797	0.7076	0.6162
Gaussian Naive Bayes	0.5908	0.5490	0.4640	0.4521	0.5934	0.5488	0.4605	0.4533
C-Support Vector Classification	0.1359	0.2521	0.7310	0.6085	0.1320	0.2678	0.7342	0.6083
Multi-layer Perceptron	0.6278	0.6296	0.8134	0.6207	0.5302	0.6172	0.8210	0.6190
Decision Tree	0.7844	0.7884	0.7838	0.6200	0.7938	0.7964	0.7946	0.6209
K-nearest neighbors(n_neighbors=4)	0.8125	0.7134	0.7825	0.4526	0.8190	0.7254	0.7882	0.4451
Gradient boosting	0.7972	0.7843	0.7961	0.6173	0.7962	0.7874	0.7963	0.6184
Random Forest(n_estimators=110)	0.8628	0.8626	0.8607	0.6208	0.8660	0.8679	0.8650	0.6209

TABLE II: Different data-preprocessing methods

We can see that standardization can dramatically improve the logistic regression, C-Support vector classification, and Multi-layer perceptron models.

B. Tuning Hyper-parameters

Then we choose the algorithms which have more than 0.8 accuracy and tune their hyper-parameters. According to the results above, multi-layer perceptron, KNN, gradient boosting and random forest algorithm satisfy the requirement.

1. Multi-layer Perceptron

The most influential parameter is the size of hidden layers, therefore we start from the defaulting value to some larger values. We can see that when taking 600, the accuracy achieves 0.85, which has almost caught up with the best ones in previous results. We also use different activation functions like tanh, but the performance is worse than the defaulting relu.

However, the training cost of this model is quite expensive. Training a single model using the given data takes several minutes.

hidden-layer-size	accuracy
(100,)	0.8287
(200,)	0.8416
(300,)	0.8437
(400,)	0.8438
(500,)	0.8473
(600,)	0.8510
(700,)	0.8501
(800,)	0.8484
(900,)	0.8498

TABLE III: Tuning hidden layer size of Multi-layer Perceptron(k-fold=8)

2. KNN

Usually, KNN is also time-consuming since when predicting each an input datum, it needs to calculate the distance between every datum in training set. However, since the training dataset size is only 10k and the dimension of features are 54 in this problem, the computation cost is affordable, and considering the predicting accuracy, KNN is an acceptable choice. Actually, compared with multi-layer perceptron and random forest, KNN is the fastest on this dataset.

n-neighbors \ weight	accuracy	
	uniform	distance
1	0.8478	0.8478
2	0.8249	0.8478
3	0.8281	0.8390
4	0.8181	0.8386
5	0.8107	0.8310
6	0.8038	0.8293
7	0.7972	0.8219

TABLE IV: Tuning n neighbors of KNN(k-fold=8)

We observe that the best n-neighbors takes only 1, which is surprising. Even though we use distance-varied weights of neighbors instead of uniform, the result remains unchanged.

3. Gradient Boosting

Gradient boosting is based on regression trees, the continuous version of decision tree, so we should mainly decide how many trees it has. Since our dataset is not huge, the defaulting parameters of tree structure is good enough. This method is quite expensive and takes about one minute for a single model training.

n-neighbors	accuracy
80	0.7885
90	0.7927
100	0.7964
110	0.8012
120	0.8038
130	0.8071
140	0.8067
150	0.8069

TABLE V: Tuning n estimators of Gradient Boosting(k-fold=8)

4. Random Forest

Random forest model consists of several decision trees, similar to gradient boosting, but much faster. Therefore, the number of decision trees is a major parameter we want to adjust. We also try to tune the parameters of decision trees, but since the dataset is small and not complex, the default setting performs well, so we don't pay much attention to this part. TABLE VI is the tuning process on n-estimator.

n-estimators	accuracy
10	0.8350
20	0.8505
30	0.8576
40	0.8574
50	0.8624
60	0.8603
70	0.8641
80	0.8644
90	0.8621
100	0.8625
110	0.8648
120	0.8647
130	0.8632
140	0.8648

TABLE VI: Tuning n estimators of Random Forest(k-fold=8)

The variation is not significant after 50, so it's unwise to continue increasing the value which leads to unnecessary computation consumption.

IV. FINAL CHOICE

Considering the strengths and weaknesses of discussed algorithms, we finally choose random forest to do the forest categories classification.