Multi-Classification Tasks by Using Machine Learning Algorithms

Abstract

In this report, I used four machine learning algorithms including Decision Tree, SVM (Support Vector Machine), Random Forest, and Adaboost, to solve multi-classification tasks based on different datasets. First I applied Python's Pandas package to preprocess data. Then, I implemented the experiment code by Scikit Learn package of Python. During the model learning step, I used the grid search method to tune each model's hyperparameters and visualized some of the tuning processes. Finally, combined with the information saved from the learning process, the report gives some analysis and conclusions.

Introduction

The datasets [1] I chose in this experiment have different types. The first one is "Algerian Forest Fire", whose features' values are all continuous. It has 2 classes, 245 samples with 14 features representing the level of temperature, humidity, wind speed, and FWI (Fire Weather Index) components during measurement. The second one is "Red Wine Quality", which has 1600 samples and 11 chemical indicators with continuous values. The differences between the two datasets are "Red Wine Quality" has 6 quality levels (3 to 8) as classes and a larger sample size. The third one is "Credit", which consists of 690 instances among two classes, most of its attributes have nominal values. All its attributes are changed to meaningless symbols to protect the confidentiality of respondents. All three datasets are downloaded from the UCI machine learning repository.

Both Decision Tree and SVM are non-parametric supervised learning algorithms. Random Forest and Adaboost all belong to the ensemble learning algorithm and can make use of Decision Tree and SVM as the weak classifier.

Literature Review

The article "Hyperparameter Optimization Techniques to Improve Your Machine Learning Model's Performance" [2] written by Davis David, 2020, has introduced some techniques like grid search and random search to conduct hyperparameter optimization. Also, the author recommended some tools and set some usage examples. Scikit Learn is a simple and efficient tool for predictive data analysis. It allows us to solve various machine learning problems.

Methodology

Data preprocessing

To start with, I use Pandas, a Python package, to remove or replace missing values. After that, I converted discrete values to numerical types in turn. (Scikit Learn's Decision Tree package uses the CART tree to treat data as continuous values by default, so this step is inevitable.) Meanwhile, the features that are irrelevant to the classification outcomes are removed. Plus, if we use SVM, it is important to standardize or normalize each feature. In this report, I use standardization during data preprocessing since this method is less affected by noise samples. Before training, each dataset is split as a train set and test set, which account for 70% and 30%, respectively. To eliminate the effect of dataset splitting, I always used the same random seed in this process.

Grid Search

During hyperparameters' optimization, I used the grid search technique, which is to try training with different combinations of parameters and find the best one. I used train sets to conduct grid search, and ten-fold cross validation was used for comparing each combination of parameters. For Decision Tree, I set the criterion as "Gini" or "entropy", max-depth range from 1 to 10, min-samples leaf range from 1 to 50 with an interval of 5, and min-impurity-decrease range from 0 to 0.5 divided by 20 intervals. For SVM, I tried four kernels -- linear, gaussian, polynomial, and sigmoid functions and set proper parameters. For example, the tolerance of misclassified samples is set as 11 numbers with the order of magnitudes ranging from -3 to 2. I also set different kernel coefficients, degrees of kernel functions, and separate ideas (One against all and Pairwise) for multi-classification tasks. In the grid search of the polynomial and sigmoid functions, I set the max number of iterations as 1000 because these two types of SVM may rapidly converge. My report's Random Forest and Adaboost used Decision Tree as the weak classifier. First, an ablation experiment was done by varying the number of estimators and leaving the rest of the parameters default. Similarly, I set different parameters of Decision Trees in these two models' grid search. In addition, Adaboost's learning rate was fine-tuned.

Results

Dataset	Decision Tree	SVM	Voting Classifier (Decision Tree + SVM)	Random Forest	Adaboost
1st	0.988	0.947	0.988	0.980	0.972
2nd	0.812	0.824	0.808	0.852	0.840
3rd	0.539	0.529	0.550	0.578	0.505
4th	0.864	0.837	0.826	0.852	0.864

Table 1. The mean ten-fold cross validation accuracy scores by using the opimal parameters. Serial number 1st to 4th represents Algerian Forest Fire, Algerian Forest Fire dataset without FWI system indicators, Red Wine Quality, and Credit, respectively.

Figure 1. An example of the Decision Tree tuning process of “Algerian Forest Fire” dataset without FWI system indicators

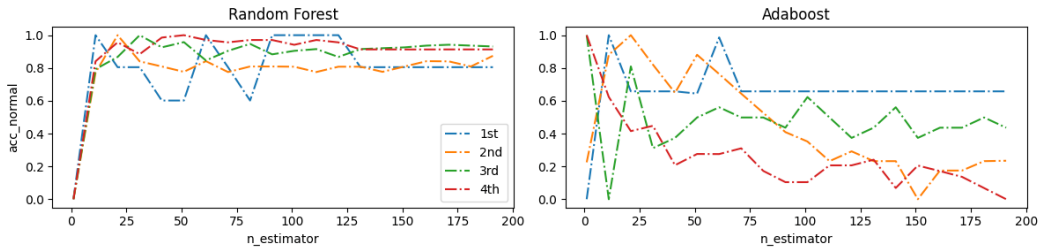

Figure 2. The number of estimators' influence on Random Forest and Adaboost. The four colored lines symbolize the four datasets mentioned in Table 1. To better visualize the variation in cross-validation accuracy, I normalized this value for each dataset.

Discussion

As it is shown in Table 1, the first dataset was precisely classified because the FWI system features could accurately reflect the risk of forest fire. This could be demonstrated by the features' information gain-ratio shown in Decision Tree and Random Forest. For "Algerian Forest Fire without FWI", ensemble algorithms could learn better representations. The third dataset is not well partitioned because the number of classes is not evenly distributed (quality levels 6, 7, and 8 accounts for 95% of the samples), and the samples in these classes are difficult to distinguish. Totally, for all four datasets, especially for the datasets that are not classified well with weak classifiers, the ensemble algorithms perform better.

Figure 1 shows the Decision Tree's structure before and after tuning. Simpler structures could improve Decision Tree's generalization performance.

During hyperparameter optimization, I observed the number of Decision Trees' influence on Random Forest and Adaboost are dissimilar. In Random Forest, the accuracy index reached a relatively stable plateau after the rapid rise, but in Adaboost, drops in fluctuations overall. It indicates that Adaboost is prone to overfitting on these four datasets.

Conclusion

In this report, I used multiple datasets and machine learning algorithms to conduct classification tasks. To improve the models' performance, I utilized the grid search technique and analyzed each hyperparameter's effect. Finally, my experiment achieved good results.

Appendices

References

[1]: Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[2]: Davis David, "Hyperparameter Optimization Techniques to Improve Your Machine Learning Model's Performance", Oct, 20th , 2020