Student Number: 236789

1. Introduction

In this report, I carried out a binary classification task based on lightgbm method. Specifically, I first preprocessed the original data, including filling of missing data, feature selection, confidence processing, and data standardization. Then input the processed data into the lightgbm method for training and verification, and finally make predictions on the test set, and the experimental results were analyzed and discussed.

2. Data preprocessing

The task provides two training sets, the first training set contains 615 samples, and the second training set contains 300 samples with missing data. Each sample has 4608 features, including 4096 CNN features and 512 gist features. In addition, the task provides a test set for prediction which has 2167 samples, and it also contains data missing.

2.1. Filling of missing data

I counted and visualized the number of missing values under each feature in the training set and test set. The number of missing values under each feature is kept at about 600. A popular approach to fill the missing data is to use a model to predict the missing values. The k-nearest neighbor (KNN) algorithm[8] has proven to be generally effective. Each sample's missing values are imputed using the mean value from $n_{neighbors}$ nearest neighbors found in the training set. Two samples are close if the features that neither is missing are close. Therefore, I introduced $KNNImputer$ by $scikit-learn$ to impute missing values in both data sets.

2.2. Confidence processing

Regarding the confidence features in the training data, I think there will be a small number of mislabeled cases, so I used a toolkit named $cleanlab$ to clean the data. $cleanlab$ first estimates the joint distribution of the noise label and the real label, and then finds and filters out the wrong samples[5]. According to the program running results, I deleted 918 incorrectly labeled data. Finally, there are 2697 training data samples remaining.

2.3. Feature selection

There are 4608 features in the data set. I thought there will be a few irrelevant features that do not have any effect on the classification results. Therefore, feature selection is required for the data set. I used the random forest[4] method to perform feature selection on the data set, and ranked the feature importance, and drew a histogram of the first 12 features.

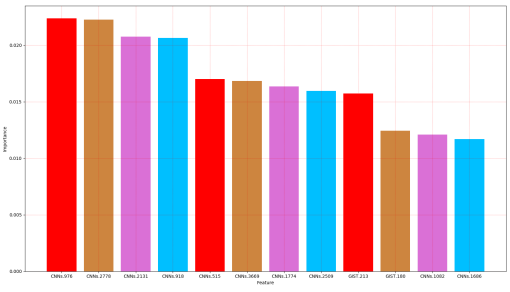

Figure 1. Feature importance

It can be seen from Figure 1 that CNN features occupy most of the important features. I deleted the features whose importance is less than 0.0001. Among the remaining 1944 features, CNN occupies 1633, and gist features occupies 318. In general, CNN features have a greater impact on the classification results than gist features. After completing the feature selection, I use $StandardScaler$ by $scikit-learn$ to standardize the data. So far, the data preprocessing has been completed.

3. LightGBM method

LightGBM[2] is a high-performance gradient boosting framework based on decision tree algorithm. It can be used in sorting, classification, regression and many other machine learning tasks. Unlike XGboost[1], lightGBM directly selects the node with the greatest profit to expand, and selects the required decision tree at a smaller computational cost. Besides, it uses the histogram algorithm to divide the eigenvalues into many small tubes, and then searches for split points on the tubes, which reduces the calculation cost and obtains better performance. The lightGBM implements biunary classifications mainly in the following steps:

(1). Load the processed data, and separate the classification target data and feature data from the processed data.

(2). Set the parameters of lightGBM, and use the 5-fold cross-training method to train the model and plot the training results.

(3). Calculate and visualize the importance of each feature in the model.

(4). Load the test set and make predictions.

4. Results and discussion

4.1. Training results

I used the $sklearn.model\_selection$ module for training, especially using k-fold cross-validation[7]. It divides

the original data into k groups, makes a validation set for each subset data, and uses the remaining $k-1$ subset data as the training set, so that k models will be obtained. The k models are evaluated in the verification set respectively, and MSE is added and averaged to obtain the cross-validation error. k-fold cross-validation is often used for model selection. I set the k value to 5 and perform model training. Due to space limitations, the training loss and auc curves of the 3rd fold is shown in the figure below.

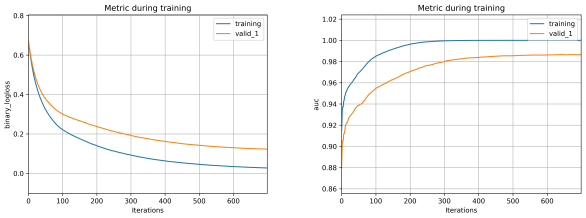

Figure 2. Training loss and auc curve of the 3rd fold

As can be seen in the Figure 2 , the training loss value decreases rapidly and becomes stable, and the accuracy of the classification is also very high. In addition, I visualized the importance of data features and plotted the top ten important features.

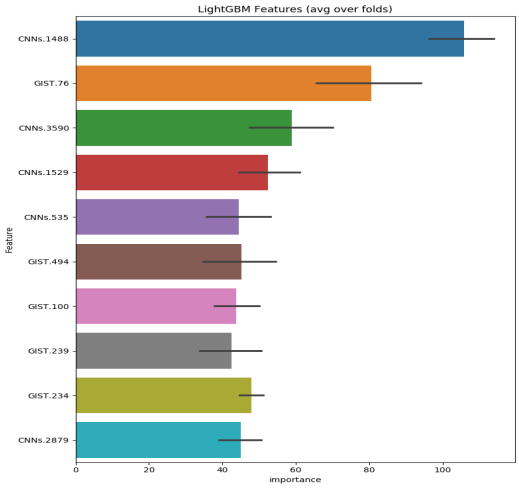

Figure 3. Top ten important features

4.2. Parameters sensitivity

To optimize the training performance of the model, I introduced $GridSearchCV$ by $sklearn.model\_selection$ to determine lightGBM parameters[6].

It can be seen that different parameters will affect the model training performance, and finally the following important parameter values are determined: ['bagging_fraction': 1], ['bagging_freq': 1], ['max_depth': 8], ['num_leaves': 40], ['min_child_samples': 18], ['min_child_weight': 0.001], ['reg_alpha': 0.001], ['reg_lambda': 1]

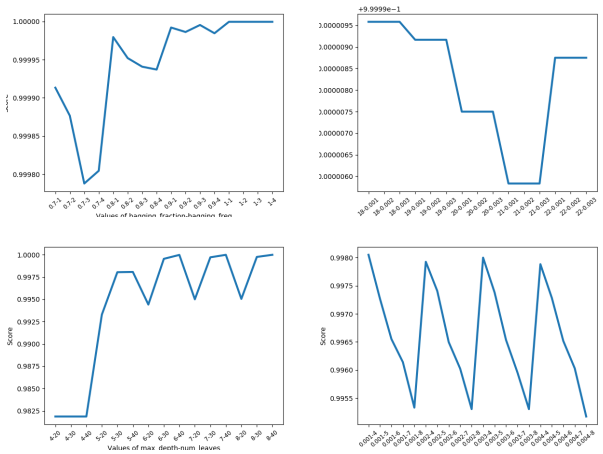

Figure 4. Training loss and auc curve of 1st fold

4.3. Different data sets and training performance

In order to reflect the effectiveness of data preprocessing, I input the data set before and after processing into the model, and compare their training performance under the premise of ensuring the uniformity of the parameters. The comparison results are shown in the Figure 5 below.

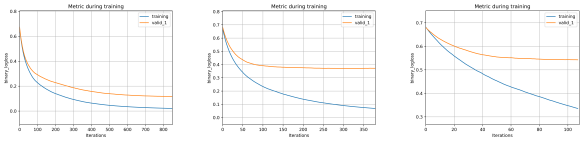

Figure 5. Training loss of different data sets. The left is the loss curve of the preprocessed data, the middle is the loss curve of the data filled with missing values; the right is the loss curve of the unprocessed data

It can be clearly seen that the training effect without data preprocessing is the worst, and the loss curve of the validation set does not even converge. The training performance of the data set after filling missing data has been improved, but it is not as good as the training effect of the data set with complete preprocessing. These three figures reflect the effectiveness of data preprocessing.

4.4. Discussion

This task gave me a deeper understanding of what I learned and also exercised my programming ability. I think the PCA[3] method can be used to further reduce the dimensionality of the data, which can speed up the training speed of the model, and also help the model better capture the data features.

References

[1] T. Chen, T. He, M. Benesty, V. Khotilovich, Y. Tang, H. Cho, et al. Xgboost: extreme gradient boosting. *R package version 0.4-2*, 1(4), 2015. 1

[2] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30:3146–3154, 2017. 1

[3] A. M. Martinez and A. C. Kak. Pca versus lda. *IEEE transactions on pattern analysis and machine intelligence*, 23(2):228–233, 2001. 2

[4] B. H. Menze, B. M. Kelm, R. Masuch, U. Himmelreich, P. Bachert, W. Petrich, and F. A. Hamprecht. A comparison of random forest and its gini importance with standard chemometric methods for the feature selection and classification of spectral data. *BMC bioinformatics*, 10(1):1–16, 2009. 1

[5] C. G. Northcutt, L. Jiang, and I. L. Chuang. Confident learning: Estimating uncertainty in dataset labels. *Journal of Artificial Intelligence Research*, 2021. 1

[6] D. Paper and D. Paper. Scikit-learn classifier tuning from complex training sets. *Hands-on Scikit-Learn for Machine Learning Applications: Data Science Fundamentals with Python*, pages 165–188, 2020. 2

[7] J. D. Rodriguez, A. Perez, and J. A. Lozano. Sensitivity analysis of k-fold cross validation in prediction error estimation. *IEEE transactions on pattern analysis and machine intelligence*, 32(3):569–575, 2009. 1

[8] S. Zhang. Nearest neighbor selection for iteratively knn imputation. *Journal of Systems and Software*, 85(11):2541–2552, 2012. 1