Supervised Learning

Caleb Thian Jia Le
Miin Wu School of Computing
National Cheng Kung University
Tainan
NN6114035@gs.ncku.edu.tw.

***Abstract*—This report is about the discussion of different kind of supervised learning, including the supervised learning by scratch and by packages.**

Keywords—Supervised Learning, Naïve Bayes Classifier, Random Forest Classifier, XGBoost, CatBoost, LightGBM

I. Introduction

This report would like to discuss several kind of supervised learning. The datasets used is *train.csv* of Car Insurance Claim provided by Kaggle[1], with 44 features and 43 features respectively after preprocessing. The approaches to be discussed is listed below:

1. Naïve Bayes Classifier by scratch
2. Random Forest Classifier by scratch [1]
3. Random Forest by Scikit-learning
4. XGBoost
5. CatBoost
6. LightGBM

These approaches are compared by comparing their performance. Due to the imbalance data, the performance is evaluated by F1-score. The code is on the github[2].

II. PROBLEM SETTING

A. AIM

The dataset has 58592 records, with 42 attributes and 1 label column marked as "is_claim". This dataset has serious imbalance problem as there is 54844 records with is_claim = 0, but only 3748 records with is_claim=1, which the ratio is about 93.6% : 6.4% . Analyzed is made to compare these classifiers' performance on these datasets.

B. Terminology

X is used to represent the features for classification, *Y* will be the ground truth, and *pred* will be the prediction made by linear classifier. *F1-score* is the F1-score of responding classifier on the dataset.

III. Methodological framework

Before talking about the classifier, data preprocessing is required. Data preprocessing will first to be discussed. After that the implementation of classifier by scratch will be discussed. Then will be the classifier by third-party packages.

A. Data Preprocessings

The dataset is preprocessed by the following steps:

1. Column casting: Cast all the column data types to float or int, according.
2. Label encoding: Encode all the categorical features by label encoding, for example, a column with *a* and *b* is represented by *0* and *1*.
3. Remove unused column: Discard any column that cannot use for classification. In this example, only policy_id is dropped.
4. Split into train datasets and test datasets: Because randomly shuffle the data is necessary, *randint* from *random* is used. The size ratio is train:test = 9:1.

These preprocessing methods is implemented in data.py.

B. Naïve Bayes Classifier

In this classifier, some assumptions are made, that is all the features are independent to each other, and the continuous features follow normal distribution. To classify a row data, in naïve bayes classifier, the target is to find the class c so that $arg\max_{c} P(is\ claim = c) \prod_{i=1}^{n} P(F_i = f_i | is\ claim = c)$.

Here *n* is the total number of features, F_i is the i^{th} feature column and f_i represent the value of the i^{th} feature of the row data.

The definition of the probability is

$$P(F_i = f_i | is\ claim = c) = \begin{cases} \frac{count(f_i,c)}{count(y)}, \text{if } F_i \text{ is categorical features} \\ \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(f_i-\mu_c)^2}{2\sigma_c^2}}, \text{if } F_i \text{ is continuous features} \end{cases}$$

. These probabilities are used to classify the row data.

C. Random Forest Classifier by scratch

The implementation of my random forest classifier is referred to the github.[3] The random forest classifier is using bootstrapping the data and bagging the trees approaches. About the creation of the tree, first, the node is selected by comparing the information gain under specific criteria. Starting from building the forest, tree needs to be built on bootstrapping samples. 10 trees is built, thus 10 bootstrapping samples with each size equals to the size of train dataset is obtained with corresponding out-of-bag observations. The out-of-bag observations are used to evaluate the performance. Then each tree is built by iterates bootstrapping the features and finding the split point. The number of features bootstrapping is 6, which is the round off of square root of total number of features, same as the default of random forest classifier of scikit-learning. Then the prediction is made by bagging among the trees. Because the bootstrapping approach involves randomness, *randint* and *choices* are imported. Beside, calculating information gain needs log function, thus *log* from *math* is imported.

[1] https://www.kaggle.com/datasets/ifteshanajnin/carinsuranceclaimprediction-classification?select=train.csv

[2] https://github.com/CalebThian/superviesed_learning

[3] https://carbonati.github.io/posts/random-forests-from-scratch/

D. *Random Forest Classifier by scikit-learning*

This is much simple, the difficulties is to find suitable parameters. Because of the data imbalance, the performance is hard to be improved by just modifying parameters. The parameters used at the end is n_estimators = 10 and max_depth=10, notice that the random forest classifier by scratch is also using this parameters.

E. *XGBoost, CatBoost and LightGBM*

These classifiers are directly used. Notice that the is_unbalance parameter of LightGBM is set to True.

F. *k-Fold Cross Validation*

k-fold cross validation is also done on these all classifiers to verify the stability of each classifiers. The performance of each classifiers is done by taking the average of the score of the predicted results from k classifiers. This increases the complexity according to k, because the higher the k, the more classifiers need to be trained, even though with smaller train dataset. The performance will be discussed in next session.

IV. EXPERIMENTS RESULTS WITH DISCUSSION

A. *F1-score*

The F1-score of each method is showed below:

	without cross-validation	3-fold cross-validation	5-fold cross-validation	10-fold cross-validation
Naïve Bayes Classifier	12.49%	12.13%	12.13%	12.13%
Random Forest Classifier by scratch	0.00%	0.00%	0.00%	0.00%
Random Forest Classifier by scikit-learning	0.00%	0.13%	0.20%	0.07%
XGBoost	0.26%	0.00%	0.27%	0.20%
Catboost	0.00%	0.13%	0.13%	0.13%
LightGBM	15.44%	16.36%	16.05%	16.16%

Table 1： F1-score of different approaches

From the table, LightGBM always get the best performance, and the performance is stable from the results of cross-validation, including the result of 5-fold cross validation. LightGBM has a good performance because it can handle imbalance data. Naïve Bayes classifier also have a better performance compares to other because it don't seriously affected by imbalance data. The random forest classifier neither have a good performance nor stability. Its stability will be affected by the number of estimators, the more the estimators, the more stable the classifier. Its performance cannot be improved even increase the maximum depth of the tree. XGBoost and Catboost cannot handle the imbalance data.

REFERENCES

[1] "Random Forests From Scratch," 14 12 2016. [Online]. Available: https://carbonati.github.io/posts/random-forests-from-scratch/.