Experiments on Text Classification using Statistical Machine Learning Models

Runyao Yu
KPMG
Munich, Germany
Email: runyaoyu@kpmg.com

***Abstract*—This report showed the comparison of using different statistical classification models, i.e., Support Vector Machine (SVM), Random Forest (RF), Logistic Regression (LR), and XGBoost (XGB), with stratified splitting, hyperparameter search and cross validation.**

I. NFL Theorem

When the performance of all optimization techniques is averaged across all feasible problems, No Free Launch (NFL) theorem states that they all perform equally well. It suggests that there is no one ideal machine learning algorithm. This is the reason why we need to always try different models with a new dataset.

II. Previous Work

The baseline as a reference was completed using Random Forest and its default setting, i.e., n_estimators=100, max_depth=None. The cross validation was not applied, the data was randomly split into 80% training set and 20% test set.

III. Updated Work

1. The data is highly imbalanced. I split the dataset into 80% training set and 20% test set utilizing stratified splitting to maintain its original distribution. Another option is to use random oversampler to balance data. However, this will increase the number of dataset significantly. Therefore, it is infeasible in this case to train multiple models with too huge dataset and limited computational power.

2. In order to add robustness to models, I utilized cross validation, i.e., CV=10. Adding this trick will mainly lead to two issues. One is the increase of computational power. The another problem is that when training the ”company code”, the error appeared: The least populated class in y has only x members, which is less than n_splits=10. Thus, I decided not to apply cross validation for ”company code”.

3. I utilized grid search to find its optimal combination of hyperparameters. Due to this, the duration for training was accordingly increased. Training each model took approximately 1.5-8 hours on CPU (SK-Learn doesn't support GPU). Random Forest is the fastest, while XGBoost is the slowest.

IV. Experiment and Discussion

The experiment results can be seen from Table I (next page). There are several important things that need to be noticed:

1. The Random Forest (base) from Table I is the default one using n_estimators=100, max_depth=None. Random Forest (mine) added hyperparameter search and cross validation. In some cases, the accuracy of Random Forest (mine) is lower than the default one, one possible reason might be that the default one didn't add cross validation and thus, the accuracy was inflated (randomly good).

2. Not recommend to use SVM in any case, since its performance is relatively bad for all classification problems. Moreover, SVM took very long time with hyperparameter search to converge. Additional information: SVC is usually combined with PCA to use, however, adding PCA to the grid search will again exponentially increase the running time. Therefore, it is not recommended to use in our case.

3. Not use Logistic Regression or SVM for ”company code”, since the solver needs samples of at least 2 classes in the data, but the data contains only one class: '2340141000'. The error raised due to its solver function.

4. Not use Logistic Regression for ”gl_vendor_id”, it simply didn't converge after the maximum allowed iteration.

The recommended usage of models with their optimal hyperparameter combination can be found in the Table II.

All the models were trained on the CPU. There are also several conclusions:

1. The duration for each task was different. Considering the first classification task ”gl_aacount_id”, the duration for RF, SVM, LR, and XGB took around 1.5 hours, 5 hours, 2 hours, and 8 hours, respectively, to finish the model training. The multiple can be applied for other tasks. Without hyperparameter search or cross validation, the duration can be dramatically reduced.

2. It should be noticed that for ”gl_legal_entity_id”, the performance of Logistic Regression and XGBoost was almost equally great. However, considering the time consumption, Logistic Regression was a bit faster than XGBoost to converge. Therefore, Logistic Regression is also a great choice for this class.

3. When training for ”company code”, it can be seen that Random Forest and XGBoost are almost equally great.

TABLE I
TEST ACCURACY COMPARISON

Model Name	**gl_accounts_id**	**gl_legal_entity_id**	**gl_tax_code_id**	**gl_vendor_id**	**company code 0037, 0065, 0301, 0303, 0330, 0362, ...**
Random Forest (base)	78.0%	97.4%	91.0%	97.9%	94.6%, 71.6%, 72.8%, 61.7%, 76.9%, 81.4%, ...
Random Forest (mine)	77.4%	97.4%	88.9%	**97.4%**	**98.2%, 82.7%, 81.5%, 72.4%, 80.7%, 85.7%, ...**
Support Vector Machine	75.9%	95.1%	90.0%	96.5%	94.5%, 81.7%, 73.1%, 46.5%, 67.1%, 80.0%, ...
Logistic Regression	**79.6%**	98.5%	**92.2%**	/	94.5%, 78.3%, 68.1%, 43.0%, 64.1%, 80.0%, ...
XGBoost	75.9%	**98.6%**	91.9%	96.6%	98.2%, 86.0%, 79.7%, 76.8%, 82.1%, 83.8%, ...

TABLE II
OPTIMAL SELECTION OF MODEL

Class	**Optimal Model**	**Hyperparameter**	**Cross Validation**
gl_accounts_id	Logistic Regression	{'C': 1, 'penalty': None, 'solver': 'saga'}	True
gl_legal_entity_id	XGBoost	{'n_estimators': 100, 'max_depth': 5}	True
gl_tax_code_id	Logistic Regression	{'C': 1, 'penalty': None, 'solver': 'saga'}	True
gl_vendor_id	Random Forest	{'n_estimators': 120, 'max_depth': None}	True
company code	Random Forest	{'n_estimators': 120, 'max_depth': None}	False

However, Random Forest is much faster (almost 10 times) than XGBoost. As there are lots of class to train, the multiple of duration for XGBoost will be accordingly accumulated. Thus, here I selected Random Forest. However, if ignoring computational consumption, XGBoost is also a great choice.