5 Kaggle competition 2:Otto Group Product Classification

This competition [36] has a higher-level difficulty (Kaggle category: Featured) than the first one. I wanted to check if the pipeline stemming from the first one is also applicative for harder competitions, so this competition was chosen. Also, it has a proper amount of data that a personal computer can manage.

At first I will describe the main goal of this competition, the dataset given by the competition host and evaluation metric. Then I will describe in chronological order the various approaches I have taken to solve this problem. Lastly, a conclusion and lessons learned from this competition will be given.

5.1 Problem Description

The Otto Group is a large e-commerce company. They posed this competition to ask researchers to classify products into one from nine categories, aiming at achieving better product analysis by improving the ability to accurately classify similar products.

5.2 Dataset

This competition provides a training dataset including 61,878 instances used for training models, and a test set containing 144,368 instances to be classified. Each instance represents a single product. 93 obfuscated features are provided for both datasets, which represent counts of different events.

The randomly selected products are to be classified into nine categories. Each target category corresponds to one of the most important product categories (like fashion, electronics, etc.).

All features are listed in Table 11.

Table 11: Description of features for competition Otto group products classification [37]

Feature	Type	description
id	numeric	anonymous id unique to a product
feat_1, feat_2, ..., feat_93	numeric	various features of a product
target	nominal	the class of a product

5.3 Evaluation metric

The evaluation metric uses the multi-class logarithmic loss (logloss). This evaluation metric is being used to penalize heavily in case a wrong prediction comes up. Each product has been defined with one true category. For each product, we are asked to submit a set of predicted probabilities (one for every category). The formula of logloss is as follows:

$$logloss = -1 * \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij}\log(p_{ij})$$

Where N represents the total number of products in the test set, M is the number of categories, log is the natural logarithm, y_{ij} equals 1 if product i is in category j and 0 otherwise, and p_{ij} is the predicted probability that product i belongs to category j. p_{ij} lies in [0,1], to prevent infinity ($log(0)$) the submitted predicted probabilities are substituted with:

$$\max(\min(p, 1 - 10^{-15}), 10^{-15})$$

The logloss is always greater than 0. The lower value a logloss has, the better the prediction is.

For this competition the leaderboard is evaluated on approximately 70% of the test data and the final results will be based on the other 30%, so the final standings may be different from the initial leaderboard ranking.

5.4 Approach and progress

5.4.1 Data exploration

The features are completely obfuscated; the meaning behind the 93 features and what the 9 categories are remains unknown. We only know the features are integer counts and contain many zeros. Figure 12 shows how the 9 product categories are distributed.

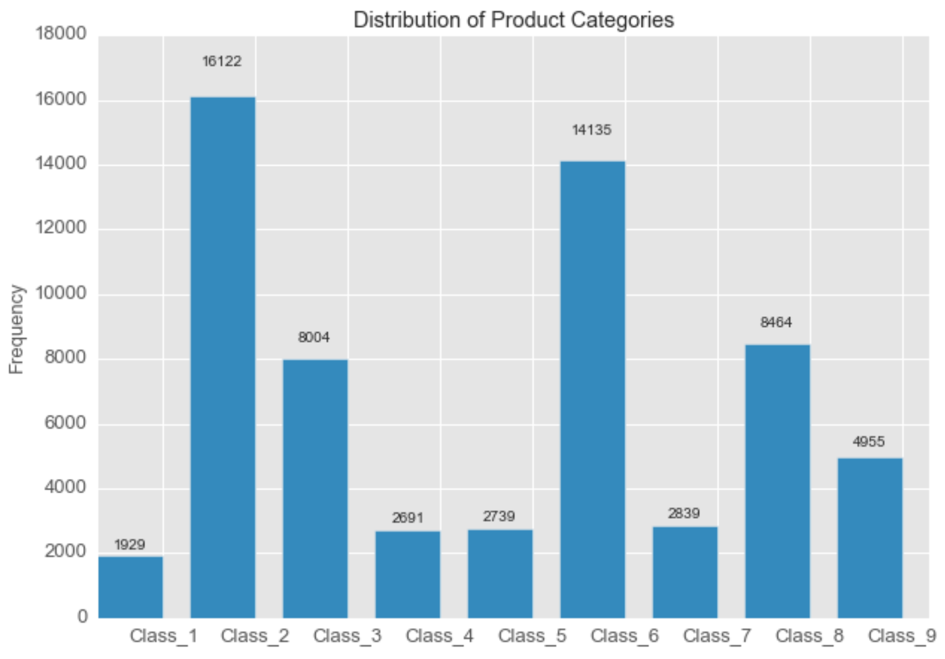

Figure 12: Distribution of Product Categories

There are many features, in Figure 13 I plotted the correlation matrix only for the 15 most important features selected by Random Forest to see if they are closely correlated. We can see most of them do not have high correlation, only *feat_14* and *feat_25*'s correlation is higher than 0.5.

The data seems tidy and clean and there is no necessity to preprocess the data.

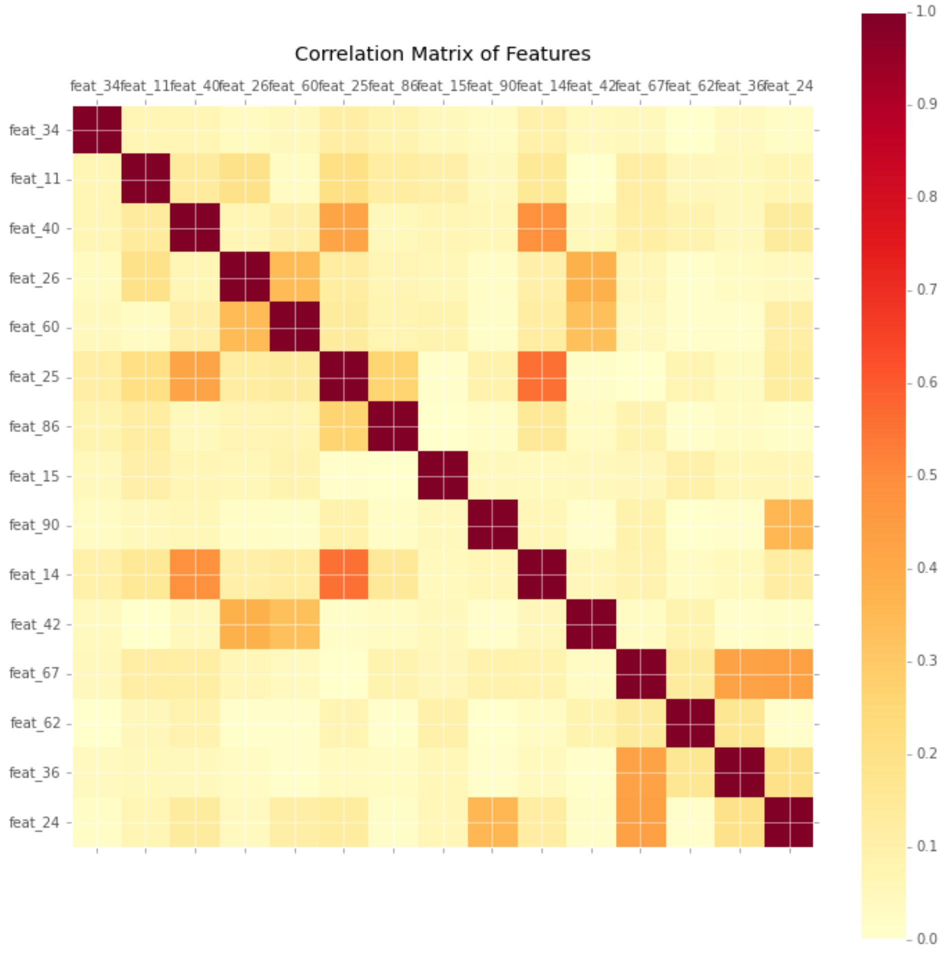

Figure 13: Correlation Matrix of 15 most important features

5.4.2 Beat the benchmark

The competition administrators have provided a benchmark using a Random Forest model with 10 trees, which resulted in a score of 1.50241 on the leaderboard. To get started on the competition and to have more insights into the subsequent procedure, I decided to run and submit the benchmark code, but setting the Random Forest model with 100 trees. This submission got a score of 0.58653, which has beat 40% of all competitors.

5.4.3 Feature construction

Since all features have been obfuscated, it is impossible to construct new features by learning from their meanings. I added only three new features: mean value, standard deviation, and count of non-zero values of every instance with the exception of id and target attributes. The standard deviation is calculated as:

$$standard\ deviation = \sqrt[2]{\frac{\sum_{i=1}^{N}(x_i - mean_x)^2}{N}}$$

But the new features proved ineffective. With the new features I got a score of 0.59432 on the leaderboard.

5.4.4 Feature selection

At first I tried a recursive feature elimination method with a linear Support Vector Classification [38] model to do feature selection, but there was no significant evidence to discard a given feature.

Then as in the forest cover type competition I tried to run local validation recursively with top-n (n from 20 to 93) features based on the feature importance computed by Random Forest, the final result did not show an obvious improvement compared to the result of applying all features. Even so, I submitted one prediction with 90 features, which saw a slight advantage, and this submission achieved a score of 0.58840. Even after feature selection the score was lower than the prediction directly on raw data, so I used the raw dataset in the following steps.

5.4.5 Model selection

Two models are selected in this step: Random Forest and GBRT. Random forest got a score of 0.58653 on the leaderboard and GBRT scored 0.59601.

5.4.6 Parameter tuning for random forest

I ran a grid search to find the optimum parameters for Random Forest and GBRT. The parameters tuned are same as in Section 4.4.9. The tuned Random Forest achieved a score of 0.53472. The hyperparameters tuned for GBRT were number of trees and the learning rate. The tuned GBRT model scored 0.50902.

5.4.7 Probability Calibration for random forest

Random Forest averages a number of decision tree classifiers on various sub-samples of the dataset, according to [39] the high variance in the underlying based trees will bias probability estimation away from the values which should be near 0 or 1, model like Random Forest can have difficulty making predictions close to 0 and 1. That results in a greater logloss error because of increased -1**log(p_{ij})* in the logloss formula which is introduced in Section 5.3.

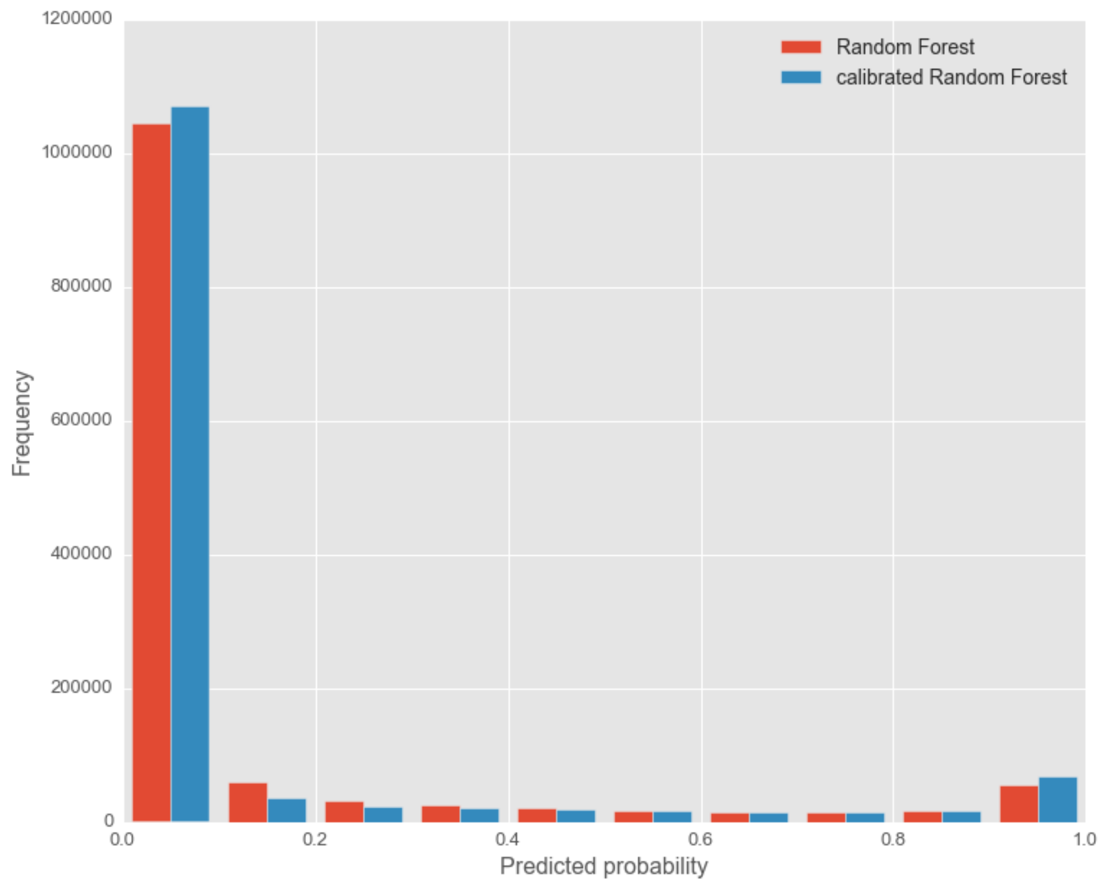

Figure 14: Probability distribution after calibration

Getting inspiration from the Kaggle competition forums, I used probability calibration methods to improve Random Forest. Probability calibration is a post-processing technique that can improve the probability estimation or the error distribution of an existing model. It performs a cross validation on the training instances and test instances. An average of predicted probabilities for each fold is then calculated for the final result to reduce variance. It has been tested by [39] that the calibration technique works for the Random Forest model once the calibration sets are large enough. The calibrated Random Forest model scored 0.48998 on the leaderboard. Figure 14 shows the change in distribution of the probability predictions after calibration. The percentage of predicted probabilities near 0 or 1 increased, which tends to decrease logloss error.

5.4.8 Using Xgboost

Referencing some advice in the Kaggle forum I tried Xgboost—Extreme Gradient Boosting. It is a very efficient Gradient Boosting package, which is widely used in Kaggle competitions. The default Xgboost scored 0.47409 on the leaderboard.

5.4.9 Tuning Xgboost

Xgboost has more parameters to be tuned, and the exhaustive grid searching required is expensive and very time-consuming. Detailed information regarding the parameters can be found in [40]. I ran a randomized search that sampled parameter settings 70 times to yield a superior model. This approach resulted in a score of 0.43237.

5.4.10 Mean stacking

From the 70 random searches of good parameters I encountered 7 outputs that hit a validation score between 0.43 and 0.44. I used a very simple ensemble method to get the average output produced by 7 Xgboost models with different parameter settings. The idea behind it is that the errors are also “averaged” when getting the mean. This little trick scored 0.42932 on the leaderboard.

5.4.11 Final result

The public leaderboard was calculated on approximately 70% of the submissions before the end of the competition and the final results would be based on all test datasets. So the final standings may be different after validation.

According to the rules of the competition, every participant was permitted 2 submissions and the best one would be my final leaderboard score. Since the public leaderboard appeared to be based on a large enough set (70% of test set), I selected the 2 submissions with the best public leaderboard scores. When the competition was concluded, my final private score was 0.43002, this is the score of the mean stacking of 7 Xgboost models described in section 5.4.9. Compared to the public score 0.42932, although the private score went down, the final ranking saw slight improvement, from 354th to 347th, out of 3,514 teams. A brief overview of the leaderboard is shown in Table 12.

Table 12: Private leaderboard overview of second competition

Ranking	Team Name	Private Score
1	Gilberto Titericz & Stanislav Semenov	0.38243
2	¯_(ツ)_/¯	0.38656
3	i dont know	0.38667
...	...	...
345	SagaU	0.42978
346	y	0.42997
347	**Ying Dong**	**0.43002**
348	TeamNeuland	0.43007
349	lvchen1989	0.43007
...	...	...
3512	jcis	33.49091
3513	stan	33.82664
3514	Pierre-Loic Doulcet	34.53878

5.5 Conclusion

In this competition I started with a Random Forest model and got a promising result. Feature engineering and selection did not contribute to the prediction, so I just used raw data in the later processes. The Random Forest and GBRT model saw a large improvement after tuning the parameters. The Random Forest model surpassed 0.5 on the public leaderboard, which was very encouraging. The Xgboost attempt greatly helped. The default Xgboost had already achieved a better score than the calibrated Random Forest. After using a random search of the parameters for Xgboost, and averaging the 7 best outputs, I reached the top 10% on the leaderboard. Figure 15 shows how the different approaches performed throughout the entire process.

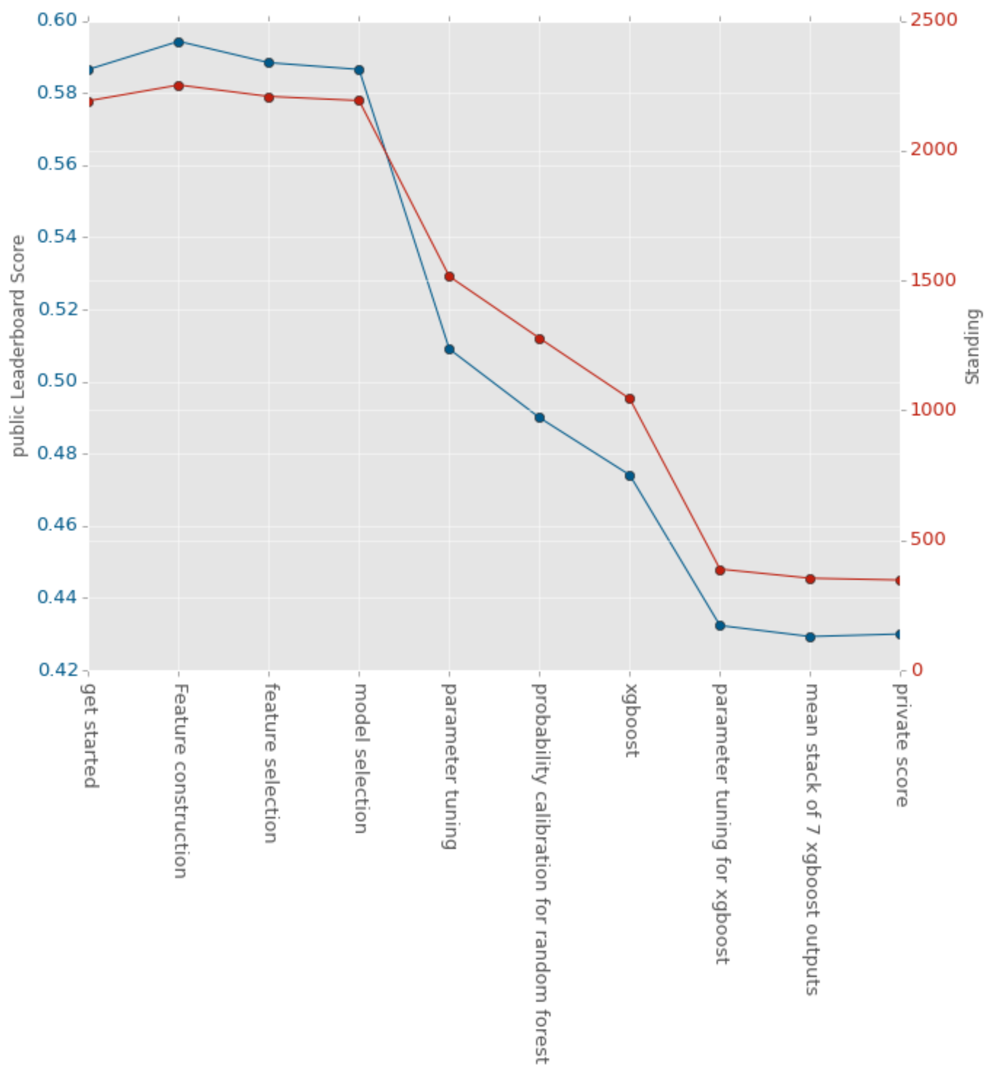

Figure 15: Score vs. Standing for competition Otto group products

5.6 Lessons learned

1. As more and more new and efficient machine learning models like Xgboost appear, we should always stay hungry for knowledge.

2. Some steps of the pipeline are not effective for the competition, but we cannot know whether they are useful without trying them.