Ensemble Learning theory to application
Text Classification with Rakuten France Product Data

YANG Shaung
Master in DSBA
shuang.yang@student-cs.fr

ZHANG Chaoran
Master in DSBA
chaoran.zhang@student-cs.fr

PENG Ruixue
Master in DSBA
ruixue.peng@student-cs.fr

1 PROBLEM DEFINITION

The project focuses on the topic of large-scale product type code text classification where the goal is to predict each product's type code as defined in the catalog of Rakuten France. This project is derived from a data challenge proposed by Rakuten Institute of Technology, Paris. Here, our main purpose is to execute ensemble algorithms introduced in the class. Hence, we just use text data and focus on compare different approaches.

2 DATASET DESCRIPTION

The original dataset consists of product designations, product descriptions, product images and their corresponding product type code, around 99K product listings in CSV format, including the train (84,916) and test set (13,812). In our project, we only use the product designations and their corresponding type. Below is the plot of class distribution in our dataset.

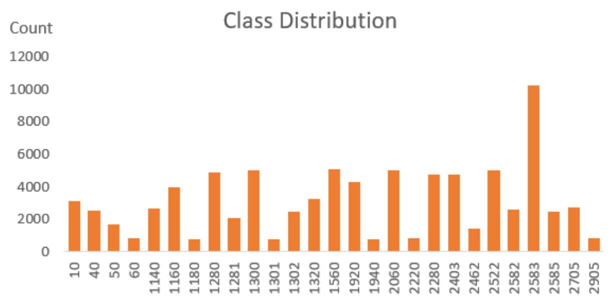

Figure 1: Class Distribution

To convert product designations, which is in the form of text, into feature sets that is more understandable for algorithms, we use Tf-idf Vectorizor. Prior to this, we preprocess the text into a single canonical form through text normalization (including lower case transformation and accented characters handling), tokenization, and stopword and punctuation removal. Upon vectorization, there are 79419 features in total. However, such vectorized data is too sparse and thus needs tremendous memory especially when applying bagging and boosting methods. At the same time, most terms/features of the vocabulary/feature space are useless because of the power law. To adapt to the configuration and capability of our computers, we cutoff 1000 features with highest frequency across the corpus and randomly sample 50% of the training set for primary model comparison and 20% for parameter tuning.

3 MODELS USED

3.1 Base Classifier

3.1.1 Decision Tree. Decision Trees are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.It is simple to understand and to interpret, trees can be visualised. It requires little data preparation. Other techniques often require data normalisation, dummy variables need to be created and blank values to be removed. The cost of using the tree is logarithmic in the number of data points used to train the tree.

3.2 Bagging Based Methods

Bagging is one of ensemble learning methods. By aggregating (eg. taking the average) multiple weak learners, of which decision tree is an example, the expectation of the aggregated prediction is unbiased whereas the variance is reduced. Therefore, an ideal bagging method should improve prediction robustness.

3.2.1 Bagging. The basic bagging algorithm is introduced based on two key points, boostrap and aggregation. Given the intuition mentioned above, one prerequisite for reducing the variance is that initial base learners are independent. One way is to train base learners on disjoint subsets of the training set. Since the training set is finite, however, each base learner will have little observation to train and thus realize poor performance in this way. Bootstrap is a sampling method that solves the problem to some extent - it ensures the number of observations for each base learner and maintain the independence among base learners by randomly drawing observations (with replacement) from the training set.

3.2.2 Random Forest. The basic weak classifier part of Random is decision tree. To tackle with a machine learning task, it will train bunch of individual tree and has it final result using voting, to synthesize the result of each tree. To overcome the default of bagging sampling method which might increase variance, the strategy to sample is bootstrap, which will lower the correlation between trees. As the same thought, during the tree growing process, a random choice of the input feature will be used to spilt the nodes.This method is available to tackle with missing value and atypical samples, thus is easy to use in practical.

3.2.3 Extremely Randomized Forest. As it is named, extremely randomized forest is a method that develops the randomization from random forest. At each split of the base learner, not only the variables/features are randomly sampled, the split point for those sample variables are selected at random. Besides, unlike random forest which draws a bootstrap for each base learner, extremely randomized forest takes the whole training set for growing each base learner. As a result, compared to random forest, extremely randomized forest should have lower bias. Meanwhile, the cost of increased variance thanks to the full draw can be compensated by the randomized split selections.

3.3 Boosting Based Methods

In bagging based methods, base learners are built in parallel. In contrast, in boosting based methods, base learners are built in an iterative way, i.e. sequentially. Although bagging based methods reduce the prediction variance by aggregating independent weak learners, they do not tackle the large bias and high training error. Similar to the idea of gradient descent, boosting can adapt to solve such problems through sequential ensemble of base learners.

3.3.1 Gradient Boosting Tree. Decision tree are used as weak learners in Gradient Boosting Algorithm.Specifically regression tree are used to output real values for splits and whose output can be aggregated. This allow subsequent model can be modified according to the residual of previous predictions.Trees are constructed in a greedy strategy, which will choose the best spilt points with a minimal purity score such as Gini, so as to minimize the loss.To avoid overfitting, hyper parameter such as number of trees,depth,maximum number of splits should be constrained.

3.3.2 Adaboost. The weak learners in AdaBoost are decision trees with a single split, called decision stumps for their shortness.AdaBoost works by weighting the observations, putting more weight on difficult to classify instances and less on those already handled well. New weak learners are added sequentially that focus their training on the more difficult patterns.Predictions are made by majority vote of the weak learners' predictions, weighted by their individual accuracy. Hence, a new hyper parameter, learning rate, is introduced here. It is the step size in renew of objective function using gradient descent.

3.3.3 Xgboost. Xgboost is a state-of-art boosting method developed based on gradient boosting. Basically, it adds regularized objective, shrinkage and column/feature subsampling to prevent overfitting. Additionally, it is more efficient compared to traditional gradient boosting. It supports both exact greedy algorithm and approximate algorithm for split finding. The latter maps features into buckets and finds the best solution among proposals based on the aggregated statistics of buckets, which lowers the complexity of split finding. It is also able to tackle sparse data by classifying an example into a default direction trained on non-missing entries when the feature needed for the split is missing, which is proved much more efficient than naive algorithms. Besides, Xgboost is famous for its scalability to handle large dataset thanks to its effective cache-aware block structure for out-of-core tree learning.

3.3.4 LightGBM. As a Gradient Boosting framework, LightGBM remains the original training method. The improvement LightGBM have is reducing complexity of histgram building by down sampling data and feature using GOSS and EFB. GOSS (Gradient Based One Side Sampling) is a novel sampling method which down samples the instances on basis of gradients. In a nutshell GOSS retains instances with large gradients while performing random sampling on instances with small gradients.LightGBM down sample the feature by bundling features together.And, it safely identifies features which are mutually exclusive and bundles them into a single feature to reduce the complexity, which is the main idea of EFB(Exclusive Feature Bundling).

3.3.5 Catboost. Catboost is an improved Gradient Boosting framework based on oblivious trees.Instead of immediately building a complex model, it build many small models in turn. Catboost introduces two critical algorithm advances- the implementation of or orderd boosting, a permutation-driven alternative to the classic algorithm, and an innovative algorithm for preprocessing categorical features.These method are designed to fight prediction shift caused by a special kind of target leakage present in all existing implementation Gradient Boosting framework. In ordered mode boosting, it perform a random permutation of the training example, and maintain n different supporting models, such that the model M_i is trained using only the first i samples in the permutation.

4 EXPERIMENTAL STRATEGY

4.1 Training-validation split

In the original dataset, we do not have the label information in test set. Hence, you can just spilt training set into two parts, which contains training and valid set, to evaluate the performance of each approach.The ratio of valid set was set to 0.3.

4.2 Grid Search

Grid Search is a hyper-parameter tuning method, which adjust specific parameter in a given range and step, calculating the loss function thus to find an optimal parameter combination in the situation. This method is simply an exhaustive searching through a manually specified subset of the hyper-parameter space of a learning algorithm.

4.3 Random Search

Random Search replaces the exhaustive enumeration of all combinations by selecting them randomly. This can be simply applied to the discrete setting described above, but also generalizes to continuous and mixed spaces. It can outperform Grid search, especially when only a small number of hyperparameters affects the final performance of the machine learning algorithm. In this case, the optimization problem is said to have a low intrinsic dimensionality. Random Search is also embarrassingly parallel, and additionally allows the inclusion of prior knowledge by specifying the distribution from which to sample.

4.4 K-fold cross-validation

Cross validation technical is a validation techniques for assessing how the results of a statistical analysis will generalize to an independent data set. In k-fold cross-validation, the original sample is randomly partitioned into k equal sized subsets. Of the k subsets, a single subsets is retained as the validation data for testing the model, and the remaining $k1$ subsets are used as training data.The advantage of this method is that all observations are used for both training and validation, and each observation is used for validation exactly once. It help us to observe whether overfitting comes.

5 COMPARISON AND ANALYSIS OF RESULTS

5.1 Model Comparison Before Tuning

In the first experiment, we want to implement each approach with it default hyper-parameters so as to find some intuition for next specific tuning step.

Firstly, we use 50% training samples to compare the $F1$ score of each algorithm. It can be seen that ExtraTree and LightGBM out stand among all methods regarding this predictive task. Otherwise, we noticed that compared boosting algorithms always spent more training time, may because of their gradient descent optimization strategy. Whats'more, boosting approach tended to perform better than bagging. What we want to research more is the reason AdaBoost got a relative low accuracy even lower than basic DecisionTree.

To summary the reason of different performance, it is not difficult to find that ensemble methods, which combines several decision trees to produce better predictive performance than utilizing a single decision tree. The main principle behind the ensemble model is that a group of weak learners come together to form a strong learner. On top of that, Bagging is to create several subsets of data from training sample chosen randomly with replacement. Now, each collection of subset data is used to train their decision trees. As a result, we end up with an ensemble of different models. Average of all the predictions from different trees are used which is more robust than a single decision tree. Boosting is another ensemble technique to create a collection of predictors. In this technique, learners are learned sequentially with early learners fitting simple models to the data and then analyzing data for errors. In other words, we fit consecutive trees (random sample) and at every step, the goal is to solve for net error from the prior tree.

Table 1: Model Comparison With Default parameters

TrainingSize=0.5			
Algo	$F1$	Time(s)	Type
DesicionTree	0.6514	23.85	Basic
Bagging	0.6816	1212.63	Bagging
RandomForest	0.6960	62.4	Bagging
ExtraTree	**0.7049**	**121.49**	Bagging
AdaBoost	0.2906	34.88	Boosting
Xgboost	0.6756	1319.27	Boosting
LightGBM	**0.7065**	**79.46**	Boosting
CatBoost	0.6836	2791.49	Boosting

5.2 Parameter Tuning

According to performances above, we further select random forest, extratree, adaboost, xgboost and lightgbm to compare their performance upon tuning. However, restricted by computer configuration and capability, here we experiment on 20% of the training data instead of 50%.

5.2.1 Random Forest. For the basic ensemble model Random Forest,we choose two basic hyper-parameters,including n_estimators and max_depth, hoped to have some intuition for latter experiments on another algorithms. Grid Research is used in the tuning process.150 for estimators and 200 for depth had the optimal f1 score. Thinking about using bagging in the sampling processing, we try to set parameter oob_score=True, this trial alsl lead an increase in accuracy/

5.2.2 ExtraTree. For ExtraTree, we adjusted 4 parameters, n_estimators, max_depth, min_samples_leaf and bootstrap. Based on the result of randomized parameter search, the best parameters should be bootstrap=False, min_samples_split=2, min_samples_leaf=1,

5.2.3 LightGBM. For LightGBM, since the offical document adviced that max_depth, num_leaves,learning_rate are relative important parameters, we first did some rectify on them. For max_depth, searched on the range of 5-20, the optimal value. After 15, the classifier was tend to be over-fitting. In expreience, num_leaves can approximately equal to $2^(max_depth)$, but this will also lead overfitting, so actually we choose a number lower than that as 40, it got better accuracy than depth-wise.Then we did some arrangment on learning_rate =0.1,feature_fraction = 0.6,and n_estimator=150. We found that a defualt learning_rate is most suitable here, values lower than 0.1 would cause accuracy decrease. Also, default setting of feature_fraction was insteaded due to a better accuracy. Otherwise, the number of n_estimators actually had little affection to the result,but we still choose a relative large value since we just use 20% data to train the classier, the same value as we test on Random Forest.

5.2.4 AdaBoost. The important parameters to vary in an AdaBoost Classifier are learning_rate and n_estimators. As with the previous algorithms, we perform a randomized parameter search to find the best scores that the algorithm can do. Another point is that the base estimator from which the boosted ensemble is built. If it is defined as None, the default is DecisionTreeClassifier(max_depth=1), which may not be suitable for current data-set, thus the performance of AdaBoost before tuning is awful. To adjust it efficiently, we perform a randomized parameter search to find the best parameters of Decision Tree firstly, then apply them to AdaBoost while tuning it. After randomized search, the f1_score is improved to 0.57.

5.2.5 Xgboost. For Xgboost, we tune separately on max_depth, min_child_weight, subsample, colsample_bytree and n_estimator comparing performances on the test set in a greedy way. max_depth determines how deeply each learner is allowed to grow. Increasing this value can lower the training bias but will also result in higher complexity of the model and add risk of overfitting. In experiments, the score do not increase a lot since max_depth=7, thus having max_depth=7 may be an optimal choice. min_child_weight corresponds to the minimum number of instances required in each node. The higher min_child_weight is, the more conservative the model will be, which is proved in experiments that increasing min_child_weight results in worse prediction performance. subsample is how much observations each learner takes as training data and colsample_bytree is how much features taken for training. Lower value help to randomize each learner and reduce the prediction variance but, on the other hand, can lead to higher bias. Experiments show the elbow near subsample=0.7 and colsample_bytree=0.4 that after these value the performances are not significantly enhanced. Finally, n determines the number of base learners used for training, for which 150 is found optimal. After tuning, the prediction performance on the test set is improved from 0.6190 to 0.6685.

Table 2: Tuned Parameters

Classifier	Parameter Values

RandomForest	n_estimators=150
	max_depth=200

ExtraTree	min_samples_split=2
	min_samples_leaf=1
	bootstrap=False

AdaBoost	learning_rate=0.1
	n_estimators=60
	base_estimator=
	DecisionTreeClassifier(
	splitter='best',
	min_samples_split=12,
	min_samples_leaf=18,
	criterion='gini',
	max_features=9,max_depth=30)

Xgboost	max_depth = 7
	min_child_weight = 1
	colsample_bytree = 0.4
	subsample = 0.7
	n_estimators = 150

LightGBM	max_depth=15
	num_leaves=40
	min_child_samples=15
	learning_rate =0.1
	feature_fraction = 0.6
	n_estimator=150

Table 3: Model Comparison Upon Tuning

TrainingSize=0.2		
Algo	$F1$ before	$F1$ after
RandomForest	0.6455	0.6712
ExtraTree	0.6589	0.6722
AdaBoost	0.2584	0.5728
Xgboost	0.6190	0.6685
LightGBM	0.6604	0.6675

In general, we can see that after tuning, accuracy of every individual approach have been improved in some extent with ExtraTree still having the best performance. The performance of AdaBoost boosts a lot through tuning, but it is still lower than other method. By inference, the best f1-score can be enhanced higher than 0.7 if it is able to tune on 50% or more of the training set.

6 CONCLUSION

In this project, we aimed to properly predict the label of every merchandise by machine learning approach among multiple ensemble methods. We attempted all methods which have discussed in class time, in addition, some advance ensemble methods such as LightGBM and Catboost are also considered. After established baseline model with their default hyper-parameters, different tuning technique has been used to search optimal parameter group for specific algorithms chose according to their performance and consuming time. Thinking over these two factors and we should make a compromise on calculating cost. Hence, ExtraTree and LightGBM are found most appropriate for this data set. They both had high accuracy and low running time.

Finally, the contribution of each member is listed as follows, we distributed workload equally.

Table 4: Contribution of group members

Member	Contribution
YANG Shuang	researching, coding, writing report
ZHANG Chaoran	researching, coding, writing report
PENG Ruixue	researching, coding, writing report