Project Report - Text Classification with Rakuten France Product Data
CentraleSupélec
ENSEMBLE LEARNING FROM THEORY TO PRACTICE

Pravallika Mavilla
CentraleSupélec
MSc DSBA
mavilla.pravallika@student-cs.fr

Shahmir Kazi
CentraleSupélec
MSc DSBA
shahmir.kazi@student-cs.fr

Veerendarnath Naladala
CentraleSupélec
MSc DSBA
Veerendarnath.Naladala@student-cs.fr

1 PROBLEM DEFINITION

The goal of this data challenge is large-scale multimodal (text and image) product data classification into product type codes.

For example, in Rakuten France catalog, a product with a French designation or title Klarstein Presentoir 2 Montres Optique Fibre associated with an image and sometimes with an additional description. This product is categorized under the 1500 product type code.

There are other products with different titles, images and with possible descriptions, which are under the same product type code. Given these information on the products, like the example above, this challenge proposes to model a classifier to classify the products into its corresponding product type code.

We will train different ensemble models on 30 percent of the data and compare the F1 scores of the models to select a model that performs better on the data set.

2 DATA DESCRIPTION

The text data contains 84916 rows, which consists of product designations, product descriptions, product images and their corresponding product type code.

(1) An integer ID- This ID is used to associate the product with its corresponding product type code
(2) designation - The product title, a short text summarizing the product. There are no missing values for this column.
(3) description - A more detailed text describing the product. However, it has missing/null values
(4) productid - An unique ID for the product.
(5) imageid - An unique ID for the image associated with the product.

There are a total of 27 classes to be classified into. The 84916 data points are not uniformly distributed into these classes. As seen in the figure1, most of the classes have around 5000 data points each representing them. six of the classes have only around 800 data points representing them and hence may be under represented but not too low to make a significant impact on the prediction, hence no need to oversample these classes. The class '2583' has the highest representation of 10209. Hence we need to under sample this class by ignoring some values belonging to this class

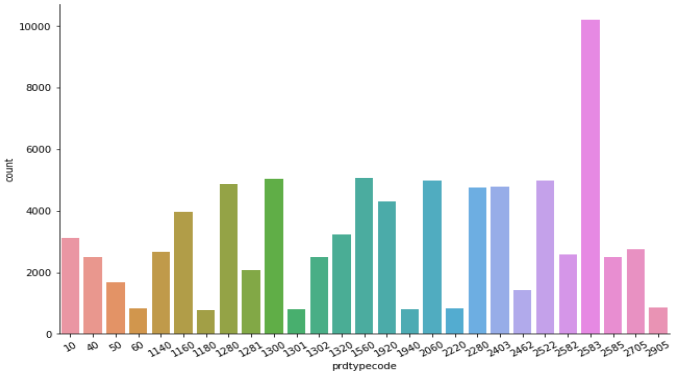

Figure 1: Distribution of data to different classes

3 MODELS USED

3.1 Non Ensemble methods

3.1.1 *MultinomialNB*.

A Naive Bayes classifier is a probabilistic machine learning model that's used for classification task. The crux of the classifier is based on the Bayes theorem.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Using Bayes theorem, we can find the probability of A happening, given that B has occurred. Here, B is the evidence and A is the hypothesis. The assumption made here is that the predictors/features are independent. That is presence of one particular feature does not affect the other. Hence it is called naive.
This is mostly used for document classification problem, i.e whether a document belongs to the category of sports, politics, technology etc. The features/predictors used by the classifier are the frequency of the words present in the document.

3.1.2 *ComplementNB*.

Complement naive Bayes is an naive Bayes variant that tends to work better than the vanilla version when the classes in the training

set are imbalanced. In short, it estimates feature probabilities for each class y based on the complement of y, i.e. on all other classes' samples, instead of on the training samples of class y itself.

3.1.3 *BernoulliNB*.

Like MultinomialNB, Bernoulli Naive Bayes classifier is suitable for discrete data. The difference is that while MultinomialNB works with occurrence counts, BernoulliNB is designed for binary/boolean features.

3.2 Ensemble methods

Below we will explore multiple ensemble methods learned during the course. Ensemble methods and bagging, boosting and stacking will be discussed in each of the methods individually, rather than generally right now.

For coherence, we my understand that ensemble methods can:

(1) Decrease variance using bagging (bootstrap aggregation, combining multiple learners)
(2) Decrease bias using boosting (a set of weak classifiers to create a highly accurate classifier). AdaBoost, gradient Tree Boosting and XGBoost are the most popular algorithms.
(3) Improve predictions using stacking (also referring to as blending, it combines multiple base classification model predictions into a new data set).

3.2.1 *GradientBoostingClassifier*.

Gradient boosting is a type of machine learning boosting technique. It relies on the intuition that the best possible next model, when combined with previous models, minimizes the overall prediction error. The key idea is to set the target outcomes for this next model in order to minimize the error.

- If a small change in the prediction for a case causes a large drop in error, then next target outcome of the case is a high value. Predictions from the new model that are close to its targets will reduce the error.
- If a small change in the prediction for a case causes no change in error, then next target outcome of the case is zero. Changing this prediction does not decrease the error

Target outcomes for each case are set based on the gradient of the error with respect to the prediction. Each new model takes a step in the direction that minimizes prediction error, in the space of possible predictions for each training case. At any instance t, the model outcomes are weighed based on the outcomes of the previous instance t-1. The correct predictions are given a lower weight than the wrongly classified predictions.

Hence, a weak model is built, conclusions drawn about feature importance and parameters, and then using this knowledge a stronger model is built, using the misclassification error of the previous model. Boosting, statistically, is a minimization of a convex loss function over a convex set of functions. The same boosting framework (GBM) will be used in later Boosting techniques as well.

3.2.2 *RandomForestClassifier*.

Random forest, like its name implies, consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction.

Decision tree is a type of supervised learning algorithm (having a pre-defined target variable) that is mostly used in classification problems. It works for both categorical and continuous input and output variables. It follows a flowchart-like structure in which each internal node represents a "test" on an attribute (e.g. whether a coin flip comes up heads or tails), each branch represents the outcome of the test, and each leaf node represents a class label (decision taken after computing all attributes). The paths from root to leaf represent classification rules.

The fundamental concept behind random forest is a simple but powerful one — the wisdom of crowds.A large number of relatively uncorrelated models (trees) operating as a committee will outperform any of the individual constituent models. The reason for this wonderful effect is that the trees protect each other from their individual errors (as long as they don't constantly all err in the same direction). While some trees may be wrong, many other trees will be right, so as a group the trees are able to move in the correct direction.

3.2.3 *XGBoostClassifier* .

Considered amongst "state of the art' since its inception, XG Boost (Extreme Gradient Boosting) works on a principle of Boosting, but has usually proved to be faster and effective than other similar Boosting techniques. The core is based on GBM framework and the algorithm is parallelizable and has parameters for cross validation, regularization, varied objective functions, ability to handle missing values and various tree parameters.

For hyper parameter definition, we will use the same methodology as all models, however, some definitions are required. The learning_rate is the step size shrinkage used to prevent overfitting. The max_depth shows how deep each tree is allowed to grow, subsample_bytree needs to be optimum to prevent underfitting as it is the percentage of samples used per tree. The same applies for colsample_bytree which takes into account percentage of features to be used per tree. n_estimators is the number of trees we build and objective function here will be chosen as per this data challenge.

The regularization parameters are a core feature of XGBoost for penalizing models from becoming complex. This allows efficient application on large datasets. Gamma controls if a node will be split based on expected loss reduction after the split. Hence, a higher value leads to lower splits. Alpha with a large value will lead to greater regularization. This is L1 regularization base don leaf weights. Lambda is L2 regularization on leaf weights but is smoother than L1.

3.2.4 *ExtraTreesClassifier*.

Named for Extremely Randomized Trees, Extra Trees Classifier

provides a low variance solution using Decision Trees as the base classifier. It randomizes certain decisions and subsets of data to minimize overlearning and overfitting.

Decision Tree may be classified as high variance, Random Forest is medium variance and Extra Trees as low variance. The way this achiveed is based on two key differences compared to Random Forest.

(1) It does not bootstrap observations - it samples without replacement, Bootstrap = false by default.
(2) Nodes are split on random split, not the best split among a random subset of features selected at every node.

Hence randomness doesn't come from bootstrapping, rather comes from random splits of all observations. GridSearchCv allows us to correctly choose the bias/variance tradeoff and the degree of "randomness" for this method.

3.2.5 *AdaBoostClassifier*.

Short for of the boosted classifier, Adaptive Boosting combines weak learners into a weighted sum that represent the final output of the boosted classifier. It is adaptive in its nature as subsequent weak learners are are changed in favour of instances misclassified by previous instances. Also, this algorithm recudes the "curse of dimensionality" by choosing only those features that improve the prediction power of the model, unlike SVM and neural networks for example.

In terms of weighting, at each iteration of the training process, a weight is assigned to each sample equal to the current error. This is the same as explained above and are used to improve the model e.g - decision trees can be grown to favour splitting sets of samples with high weights. Hence the most accurate classifier will get the higher weight. This iterative approach continues until error is reduced or maximum number of estimators is used, This is further explained in the methodology section of this report.

In terms of parameters, base_estimator, n_estimators and learning rate are the most important ones. We used DecisionTreeClassifier as the weak learner. For number of estimaors and learning rate we use GridSearchCV, as with all other approaches. Althoughnot prone to overfitting, it is sensitive to noisy data and hence our dimenstionality reduction will improve results. However, it is also slower on out dataset compared to XGBoost.

4 METHODOLOGY

Initially, to check which model performs better for the data set, we split the train_x and train_y data into 30 percent train and the rest as validation set since the data was huge. This is in line with the online discussion held with our course instructors.

Hence the set that we trained the models for comparison had 25,474 data points. We decided on focusing on the 'designation' column of the data as it is mentioned to be the primary discriminant for the classification. There were no null values in this column.

We performed preprocessing on this column:

(1) Changing all the letters to lower case
(2) Removing all the stop words available on the french package of Spacy.
(3) By changing all the accent letters into normal letters
(4) remove any html script

Then, we got the count matrix of this column and got the TF-IDF matrix of it such that each item's description is considered as a document and the train set is considered the set of documents.

After TFIDF, we had 25,474 rows × 38,520 columns, which was too huge to be processed on our laptop processors. So we had to run on a GPU backed Google Colab.

Since the matrix was too sparse, we wanted to do a dimension reduction. But the traditional PCA would require a lot of RAM as it processes the entire matrix as a whole. So instead, **we used randomized PCA which gets the SVD** by projecting the matrix on to a random matrix and hence uses less RAM at a time.

With Randomized PCA, we found that the first 3 columns itself explain >99 percent of the variance. So we tried all the models with this reduced data set as well.

4.1 EXPERIMENTAL STRATEGY

We trained the aforementioned models by performing a hyper parameter grid search on the models by a 5-fold cross validation. This ensures that the model doesn't train on the actual test data labels and also reduces chance of overfitting.

With 5 fold Cross Validation using GridSearchCV, we perform hyper parameter tuning for **all models** mentioned above, This is t ensure a fair comparison and deduce the best model for Rakuten.

```
#XGBoost on tfidf

#import xgboost as xgb
from xgboost import XGBClassifier

param_grid = {}


#Define Classifier with no parameters
clf = XGBClassifier(tree_method='gpu_hist')

#Run GridSearchCV
clf = GridSearchCV(clf, param_grid,scoring = 'f1_micro')

#Fit the training sets
clf.fit(tfidf_reduced_svd,ytr['prdtypecode'])

Deduce best parameters
print('Best parameters:')
print(clf.best_params_)
print()

#Deduce best score with 5 fold CV and chosen parameters
print('Score:')
print(clf.best_score_)

Score:
0.6785742120750113
```

Figure 2: XG Boost algorithm for hyper parameter tuning with 5 Fold CV

We create a hyper parameter dictionary named **params:** which holds all hyper parameters with our given limits/values as key-value pairs. Hence, optimal hyper parameters are obtained for each of our model with a robust 5 fold CV.

4.1.1 *Scoring methodology - F1 Micro*

. The problematic at hand is a multi-class classification problem with class imbalance, as seen in the earlier section. Hence we use the F1 Score as a metric to evaluate prediction performance. This is passed as an argument in the GrdiSearchCv function for all approaches used.

Mathematically, it accounts for errors and provides a true representation as :

$$F1 = 2\frac{(Precision * Recall)}{(Precision + Recall)}$$

where

$$Recall = \frac{TruePositive}{(TruePositive + FalseNegative)}$$

and

$$Precision = \frac{TruePositive}{(TruePositive + FalsePositive)}$$

Now that we have selected this as the metric for evaluation, we need to choose from **Weighted, Macro or Micro.**

The **Weighted** method calculates score for each class independently but when aggregating it uses a weight dependent on the number of true labels in each class, therefore favouring the majority class. **Macro** calculates it by class separately but not using weights fir aggregation. This results in bigger penalization when the model doesn't perform well for minority classes. We are not using techniques to down sample the majority or up sampling the minority hence we move the last approach. **Micro** uses the global number of True Positive, False Negative and False Positive, calculating the F1 score directly. As no class is favoured in particular, we use this **F1_Micro** for our evaluation.

5 COMPARISON AND ANALYSIS OF RESULTS

The below table shows a complete overview of all the 8 models run for the problematic at hand. Comparison between Ensemble and Non Ensemble techniques show that Ensemble performed better.

Ensemble techniques are also faster to run and more effective and efficient on large datasets such as the current one. Hence running the model chosen for running on the complete dataset must be an Ensemble method, as they also have higher F1 scores.

The description, advantages and disadvantages of each model was discussed in Section 3 - Models Used. Hence, we request that Section 3 be viewed in conjunction with this section while evaluating for models and choice of model, both .

Evaluation of models using F1-Micro scoring on 30% of X_train dataset using 5 fold CV				
		With entire TF IDF	**With PCA**	**With SVD**
Non-Ensemble Methods	Multi Nomial NB	0.726	0.125	*NA due to negative values*
	Compliment NB	0.729	0.679	
	Bernouli NB	0.699	0.119	0.599
Ensemble Methods	Gradient Boosting Classifier	*RAM limited to 32 GB*	0.673	0.612
	Random Forest Classifier	0.734	0.660	0.603
	XG Boost Classifier	*RAM limited to 32 GB*	**0.718**	**0.679**
	Extra Trees Classifier	0.742	0.642	0.608
	Ada Boost Classifier	0.603	0.619	0.395

Figure 3: All models with their respective F1 scores. SVD with 70 percent and PCA with 99 percent variance

The optimal model for the current dataset is XG Boost Classifier with an accuracy of 71.8 percent with PCA and 67.9 percent with SVD. It is also efficient and scalable on larger sets and hence has been chosen for more reasons than just the F1 score.

XG Boost, due to its algorithm structure and ability to be scaled up on larger datasets through threads and parallelized methodology, is the ideal choice as it also has a faster run time and provides relatively much higher accuracy even if the explainable variance in the TF - IDF drops due to using SVD.

We hence run the model on the 70 percent complete set using SVD and achieve an accuracy of 72 percent with the optimal hyper parameters deduced from GridSearchCV.

Overall, as we witnessed through extensive testing and using multiple approaches for dimensionality reduction that ensemble methods are more robust, more reliable on larger sets, more efficient and more accurate. Non ensemble methods have inherent shortcomings and the reason for comparison and running them was also to highlight the benefits of ensemble learning through real world application.

Within ensemble methods, Gradient Boosting has a lower accuracy than XG Boost due to the algorithm itself and considering the details and comparison provided in Section 3, the results also show these differences.

Learnings gained from Random Forest and Extra Trees show that though the "random" element plays a role in increasing accuracy of the results, XG Boost is still more robust and more reliable. Adaptive Boosting is less reliable than XG Boost as results fluctuated upon reruns and accuracy dipped when the explainable variance in the TF - IDF dropped due SVD.

The team faced constraints in the form of RAM limitations and also lost time running models on TF-IDF reduced via PCA. Solutions proposed are using a paid cloud server with the complete TF-IDF or using PCA - as it covers more variance in the dataset compared to the SVD TF - IDF with 4000 columns only.

6 CONCLUSION

References

1. XGBoost: A Scalable Tree Boosting System. Tianqi Chen, Carlos Guestrin.

2. Extremely randomized Trees. Geurts Pierre, Ernst Damien and Wehenkel Louis.

3. Multiple Classifier Systems. G. Goos, J. Hartmanis and J. van Leeuwen.

Github link: https://github.com/veerunaladala/Ensemble-Learning.git

	Pravallika Mavilla	Shahmir Kazi	Veerendarnath Naladala
Code	Preprocessing, PCA and Classifications on PCA reduced data	Classifiers on data without dimensionality reduction	SVD and classifications on SVD reduced data
Report	Work shared equally	Work shared equally	Work Shared equally

Figure 4: Contribution of Each Member