Predicting Return Quantity for a Fashion Distributor via Gradient Boosting and Ensemble Learning

Minjie Fan

July 5, 2016

Abstract

In this paper, we present our approach in the Data Mining Cup (DMC) 2016, where the task is to predict return quantity for a fashion distributor. The order data and the related return data were recorded over a two-year period, which are used to build a model enabling a good prediction of return quantity. From the raw data, we extracted a number of features with a strong predictive power and good interpretability. We transformed the data such that the problem is reduced to binary classification, and thus the computational cost is significantly lowered. Several state-of-the-art predictive models are fitted to the data, including regularized logistic regression, random forest, gradient boosting and deep learning, where gradient boosting gives the best predictive accuracy. To further improve the predictive accuracy and leverage the strength of each model, we applied three-layer stacked generalization to combine the the predictions from these models. Our team ranked the first out of 120 teams.

1 Introduction

Returns have been a major cost driver for online shops for many years. This is particularly the case for assortments in the fashion industry. Winkler (2016) reported that nearly one third of all online purchases are returned, and the news is even worse for specific ecommerce categories, such as apparel, soft goods and expensive items. Whether an ordered item would be returned or not is highly related to the characteristics of the item itself and customer shopping behaviour. Reliably predicting return quantity based on the information of items and customers can effectively reduce the cost of returning for online shops.

The task of DMC 2016 is the prediction of return quantity for a real anonymized fashion distributor, which sells articles of particular sizes and colors to its customers. The prediction period is from October 2015 to the end of the year, and the prediction is based historical sales and related return data from January 2014 to September 2015. The goal is to use the data and methods of data mining to build a model which enables a good prediction of return quantity. The training data consist of about 2.33 million observations, and 14 predictors including 10 categorical variables, and 4 numerical variables. Example observations are shown in Appendix A.1, and a table of original variables is available in Appendix A.2. We found several difficulties in the exploratory analysis phase, and sought different methods and strategies to resolve the issues. This paper highlights the key points of our modeling procedures and strategies.

2 Exploratory Analysis and Difficulties

In the exploratory analysis, we encountered several difficulties in the data. Exploratory plots and tables are shown in Appendix A.4.

1. The response, *return quantity*, is a count bounded by the predictor *quantity* for each observation, and it is highly unbalanced, where more than 99.8% of the responses are 0 or 1. Thus, the first problem is to decide which approach we should take to model the data: regression or classification. Given that the response is discrete, it is more natural to treat the problem as classification. The benefit of treating it as regression is that it respects the ordinal nature of the response.

2. Similarly, imbalance can be found in other categorical variables, most of which also have huge number of levels. To better use the information of the predictors, we need to dig into the data and derive useful features instead of using them directly.

3. There are also many new articles, colors and product groups that appear only in the test data, indicating new articles entering the market. We need to carefully match them with the existing colors and product groups and impute these variables.

4. Finally, we need to propose an appropriate and reliable criterion to evaluate our models with the available training data given their time series nature.

3 Feature Engineering

We approach the feature engineering problem from several different perspectives. The most straightforward method is to derive features by aggregating. First, we group data by certain variables, such as orderID, customerID, articleID and orderDate. For these variables, we can group the data by only one of them or by several of them. Then, we apply aggregate functions to each group of the data. The aggregate functions can be min, max, mean, median, std, sum, mode, number of elements, number of unique elements, etc. Last, we expand the summarized data by inserting them into each row. Here are some examples: total quantity per order, total number of orders per customer and mean recommended retail price per article.

Among these 14 predictors, we found that some of them have implicit meanings. ColorCode is represented by four-digit numbers (some colorCodes in the test data can be converted into four-digit numbers by dropping the second digit, which is always zero), and actually each digit has its own meaning, such as color, shade, pattern and product type. Thus, it is more reasonable to use each digit of colorCode instead of the whole number as features. We also found that sizeCode is represented in different units for different productGroup. Thus, we converted them into a uniform unit. Rrp (recommended retail price) is not just a price. It also reflects the price level of the items, such as cheap, regular and luxury. PaymentMethod is represented by German abbreviations. We speculated that they have the translations listed in Appendix A.3. These payment methods can be manually classified into two groups: invoice and right-away.

It is very important to derive features from customer behavior. Some customer analysis data reveal that, for some retailers, certain customers are 15x more likely to return items compared to other customers. For the customers who frequently return items, we can classify them into three categories: Wardrober, "Try it on" customer and fitting roomer. The last one is of most interest to us, who buy different sizes and colors of the same or similar item to eventually pick their favorite. We created features based on this motivation. First, we proposed some criteria to measure similarity between two items in a single order. For example, items with the same articleID, with the same articleID and colorCode, with the same productGroup and colorCode, etc. Then, for each item in a single order, we count the total quantity of items that are similar to it (including itself). Intuitively, the larger the value is, the higher probability of returning this item. Apart

from that, we also created features that reflect customer preferences since they tend to pick their preferred items among similar items. For example, the percentage of each possible color of a specific product group bought by a customer. The idea of similar items in the same order can be extended to similar items across orders. If a customer bought two similar items within a short time, it is very likely that the customer will return the former. If a customer has bought similar items many times, the customer is very familiar with this type of items and has a higher probability of keeping it. We created relevant features based on these conjectures. The above mentioned features mainly characterize fitting roomers. In order to take into consideration the first two types of returners, we created a feature called likelihood of returning, which is the ratio between the total quantity of returned items and the total quantity of ordered items for each customer. We added some random noise to the likelihood in order to avoid overfitting. Otherwise, the classifier would rely too much on the likelihood feature, and thus has high generalization error. Moreover, to avoid the potential data leakage issue, which is the creation of unexpected additional information in the training data, we created the feature by cross-validation, where the responses in the validation set are always assumed to be unknown although they are available.

The new product groups that appear only in the test data are manually imputed by matching them with the existing product groups. We can not validate this extrapolation using the available data. To be conservative, the final prediction is a weighted average of the predictions where the imputation is conducted in different ways, and the weights are assigned according to our personal beliefs on how reliable these imputations are.

Figure 3 shows the importance plot of the top 20 features. The features related to price have the strongest predictive power, which is consistent with the empirical findings in Winkler (2016) that expensive items tend to be returned more frequently.

4 Modeling Strategy

We reformulated the prediction task into a classification problem by expanding the data with respect to the predictor *quantity*, so that the new response on each line is either 0 or 1. The predicted probabilities are summed accordingly and rounded in the end to revert back to the original scale. Doing so reduces the multi-class classification problem to binary classification, where the computational cost of the latter is much lesser since there are fewer parameters to be estimated in the model. Although the evaluation criterion of the task is mean absolute error (MAE), we used logarithmic loss (log-loss) as the objective function to be optimized. Compared with MAE, which is non-smooth, log-loss is more stable and a better criterion to estimate class probabilities.

With the derived features, we applied stacked generalization (Wolpert, 1992): a multi-layer modeling approach to combine the predictions of several base learners to improve the predictive power through leveraging the strength of each base model and to avoid overfitting. The outline (see Figure 4) is as follows:

1. Train base learners with the help of cross-validation, and predict for both training and test data. We used regularized logistic regression (glmnet) (Friedman et al., 2010), random forest (parRF) (Breiman, 2001), deep learning (h2o) (Arora et al., 2015) and gradient boosting (xgboost) (Chen and He, 2015) models, where the R packages we used are indicated in parenthesis.

2. Treat the predicted probabilities of return as new features. Combine them with the top 100 important features, and feed them into xgboost and deep learning to generate second layer predictions.

3. Bagging the second layer predictions to form the final prediction. In practice, we only used the boosting predictions due to their very high correlation with the deep learning predictions.

It is very important to have a robust performance evaluation criterion for feature selection and model comparison. In time series predictor evaluation, a blocked form of cross-validation (Bergmeir and Benítez, 2012) is more suitable than the traditional cross-validation since the former respects the temporal dependence. However, it suffers from the problem of predicting the past based on the future. Another common practice is to reserve a part from the end of each time series for testing, and to use the rest of the series for training. This strategy avoids predicting the past, but it does not make full use of the data, and thus is less stable than cross-validation and prone to be over-fitting to the test data. To validate the modeling strategy, we applied both methods. Specifically, we divided the training data into 7 cross-validation folds with about 3 months in each fold, and treated the last fold (called holdout set) as a pseudo test set.

5 Results and Discussion

The following table shows our progress in prediction measured by mean absolute error in different stages of modeling. In the baseline stage, we used the boosting model with the raw features. The "selective feature" stage used all the engineered features in the boosting model. We confirmed our expectation that the improvement becomes more and more difficult as the modeling procedure moves forward.

MAE	CV prediction	holdout prediction
baseline	0.3642	0.3733
selective feature	0.3129	0.3191
stacking model	0.3023	0.3105

Looking back to the modeling procedure, we experienced leaps of improvements in feature engineering, parameter tuning as well as stacked generalization. Each of the tasks is tricky and can be a research topic itself, so it requires lots of time for heavy computation in a large number of attempts and building complicated ensembles. We also learned that intuition/interpretability is extremely important in feature engineering, and it is worth spending quite some time digging into and understanding the data.

Statement of Contributions

I did literature reading, wrote slides and reports, participated in group meetings and conducted most data analyses.

Acknowlegement

There are five other teammates, without whom the competition is impossible to be finished: Jilei Yang, Hao Ji, Qi Gao, Nana Wang and Chunzhe Zhang. We would like to express our special thanks to Prof. Hao Chen, who provided us very insightful advice and practical guidance. We are also thankful for the support we get from the STA260 class and Prof. Wang, and also for Prudsys AG that held and sponsored this interesting competition.

We appreciate the opportunity of participating in the competition, during which all of us learned a lot about data mining techniques, modeling strategies and most importantly about teamwork.

References

Arora, A., Candel, A., Lanford, J., LeDell, E., and Parmar, V. (2015) *Deep Learning with H2O*.

Bergmeir, C. and Benítez, J. M. (2012) On the use of cross-validation for time series predictor evaluation. *Information Sciences*, **191**, 192–213.

Breiman, L. (2001) Random forests. *Machine learning*, **45**, 5–32.

Chen, T. and He, T. (2015) xgboost: extreme gradient boosting. *R package version 0.4-2*.

Friedman, J., Hastie, T. and Tibshirani, R. (2010) Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, **33**, 1.

Winkler, N. (2016) How to Reduce Your Return Rate & Predict Exactly What Customers Want. https://www.shopify.com/enterprise/102947142-how-to-reduce-your-return-rate-predict-exactly-what-customers-want.

Wolpert, D. H. (1992) Stacked generalization. *Neural networks*, **5**, 241–259.

Appendix

A.1 Example Observations

orderID	orderDate	articleID	colorCode	sizeCode	productGroup	quantity	price	rrp	voucherID	voucherAmount	customerID	deviceID	paymentMethod	returnQuantity
a1000010	2014-01-01	i1002243	3978	34	13	1	20.00	49.99	0	0	c1089299	2	BPRG	1
a1000010	2014-01-01	i1003000	3975	I	17	1	15.00	25.99	0	0	c1089299	2	BPRG	1
a1000011	2014-01-01	i1000577	3953	44	3	1	25.00	29.99	0	0	c1065839	2	BPRG	0
a1000011	2014-01-01	i1000605	1927	42	3	1	49.99	49.99	0	0	c1065839	2	BPRG	1
a1000011	2014-01-01	i1001987	3001	44	8	1	25.00	39.99	0	0	c1065839	2	BPRG	0
a1000011	2014-01-01	i1001998	3992	44	8	1	39.99	39.99	0	0	c1065839	2	BPRG	0

Figure 1: Example observations in the original training data.

A.2 Original Predictors

Column name	Description	Value range
orderID	Order number	Natural number preceded by "a"
orderDate	Date of the order	Date in YYYY-MM-DD
articleID	Article number	Natural number preceded by "i"
colorCode	Color code for the article	Natural number
sizeCode	Size of the article	Natural number or string of letters
productGroup	Product group of the article	Natural number
quantity	quantity	Natural number including 0
price	Total value of the order position	Positive real number including 0
rrp	Recommended retail price	Positive real number including 0
voucherID	Voucher number	Natural number preceded by "v" and "0"
voucherAmount	Voucher value per order	Positive real number including 0
customerID	Customer number	Natural number preceded by "c"
deviceID	Device number	$\{1, 2, 3, 4, 5\}$
paymentMethod	Payment method	String of letters
returnQuantity	Quantity of returns	Natural number including 0

Table 1: Variable descriptions for original data. The response is returnQuantity.

A.3 Speculations on Translation of PaymentMethod

- advance payment = Vorauszahlung = VORAUS;
- credit card = Kreditkarte = KKE;
- cash on delivery = Nachnahme = NN;
- BillPay direct debit = BillPay Lastschrift = BPLS;
- BillPay invoice = BillPay Rechnungskauf = BPRG;
- BillPay PayLater = BPPL;
- invoice = Rechnungskauf = RG;
- ClickandBuy = CBA;

- some company's invoice = KGRG.

A.4 Exploratory Results

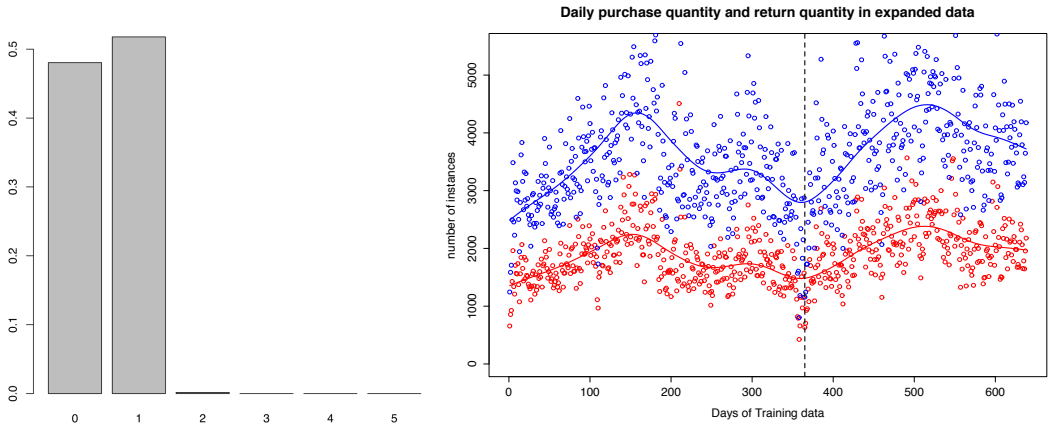

Figure 2: Left: Histogram of raw response in training data. Right: Scatter plot of purchase quantity (blue dots) and the quantity of returns (red dots), overlaid with smoothed mean purchase quantity (blue line) and mean quantity of returns (red line). The vertical dotted line indicates the end of year 2014. Seasonality effect can be observed.

orderID	articleID	voucherID	customerID
738698	3822	671	311369
Product group	Color code	Size code	
18	546	29	

BPLS	BPPL	BPRG	CBA	KGRG
11358	4248	1810036	179398	207
KKE	NN	PAYPALVC	RG	VORAUS
106748	17271	169009	13	26877

Table 2: Upper: Number of levels for raw categorical predictors. Lower: Frequency of payment method in training data.

A.5 Important Plots

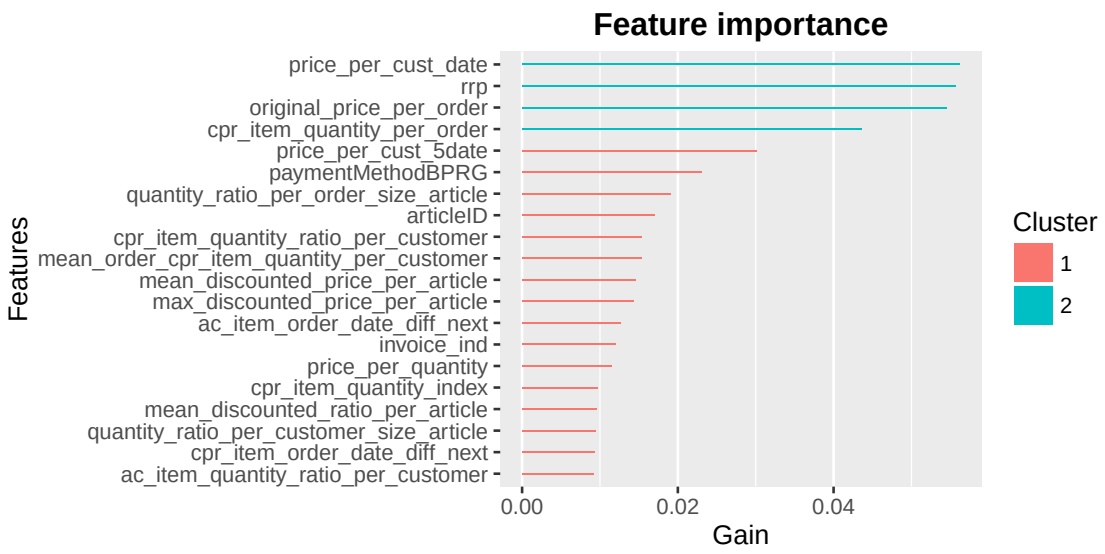

Figure 3: Importance plot of the top 20 features.

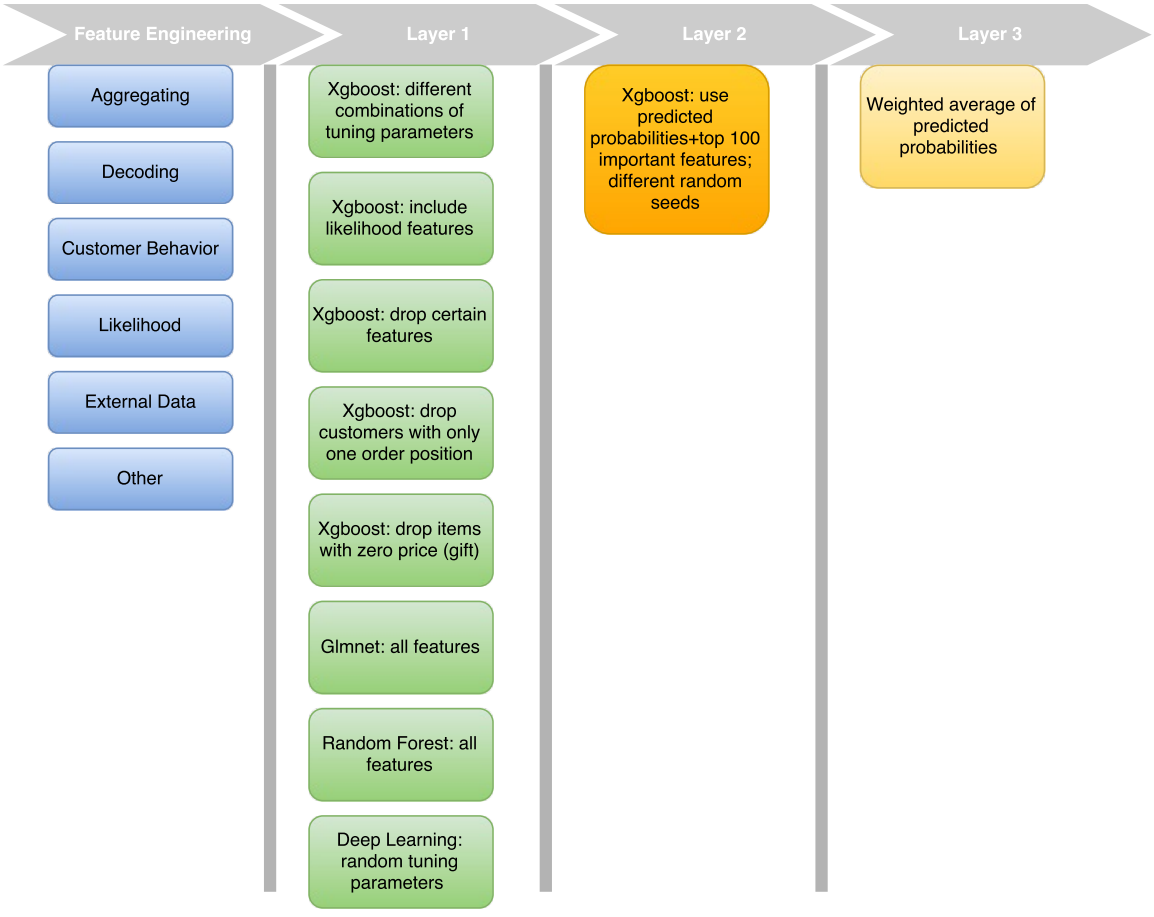

Figure 4: Flow chart of three-layer stacked generalization.