Using Decision Tree to predict repeat customers

Jia En Nicholette Li   Jing Rong Lim

Abstract

We focus on using feature engineering and decision trees to perform classification and feature selection on the data from *Kaggle*'s *Acquire Valued Shoppers Challenge*.

1. Introduction

Customer retention is important to many businesses as it is cheaper to build loyal relationships with a customer than to source for new customers[1]. A study by *Bain & Company* stated 25% to 95% increase in profits can be made just by increasing 5% of customer retention rates and a 30% rise in company value with an increase of 10% of customer retention[1]. From marketing to offering discounts to loyalty programs, companies have been continually innovating in order to increase customer retention, albeit at an initial cost to themselves. A good marketing strategy to look into would be product offers. Product offers aim to attract new and old customers alike with attractive product deals as an incentive to continue buying from them. However, this comes at the expense of businesses as these deals equates to lower revenue. Hence it is important that these costs translate to loyal customers that repeat product purchase from them within and outside of product offer periods.

Whether a customer decides to repeat a purchase is dependent on a myriad of factors. These can range from loyalty and trust to a particular company or brand, or maybe the product is a necessity, such as toothpaste. As Decision Tree algorithms inherently estimate the suitability of features during separation of classes and can handle both categorical and numerical features, we are interested in finding out the ranking of features in retaining customers with the use of product offers.

2. Related Work / Motivation

Our motivation to use Decision Tree algorithm to predict repeat customers comes from trying to find an algorithm that can handle both continuous and discrete variables as customer transactions have properties that are continuous (e.g. amount spent) and discrete (e.g. ID of company, category, brand). According to *A general purpose separability criterion* by Grabczewski and Duch, the construction of decision trees is a natural application of the "separability criterion", a criterion based on the idea that the best split is the one that splits the largest number of pairs from different classes[2]. As the cut-off point for continuous features can be a real number subset in a range of values for a feature, decision tree algorithms are suitable to handle it as they inherently try to find the best cut-off point during its splits. This applies to discrete features as well.

An added benefit of utilizing decision trees is that decision trees implicitly perform feature selection while performing classification. In *Feature Selection with Decision Tree Criterion*, Grabczewski and Jankowski states that decision tree algorithms "inherently estimate the suitability of features for separation of objects representing different classes"[3]. This makes it an interesting endeavor to find out what makes customers repeat a purchase and it can possibly give new insights to how customers tick.

A similar work to our project would be *Predicting Customer Shopping Lists from Point-of-Sale Purchase Data* by Cumby, Fano, Ghani and Krema[4]. Instead of predicting whether a customer would repeat a purchase with an offer, this research predicts what a customer would want to purchase from their past transactions using decision trees (specifically C4.5[5]), linear methods (such as Perceptron, Winnow and Naive Bayes) as well as hybrids of different algorithms. They accounted their research results by the accuracy, precision and coverage of how well the algorithms do in predicting a customer's potential shopping list, of which C4.5 have the highest precision (the number of true positive predictions)[4] at 42% and second highest accuracy (the total number of correct predictions over the total number of examples) at 73%. Promising results from the use of decision trees to predict customer's shopping list have led us to look into using it to predict whether a customer will repeat a purchase.

3. Methodology

3.1 Aim

With an input of number of items bought, total amount spent in 30 days/ 60 days/ 90 days/ 180 days/ overall transactions from a product company/ product category/ product brand, find out the usefulness of Decision Tree algorithm to predict repeat customers and determine the important features for predicting repeat customers.

3.2 Data

The dataset is acquired from the *Kaggle* competition, *Acquire Valued Shoppers Challenge* containing 1) customers' pre-offer transactions, 2) training history containing a product the customer bought and whether a repeat purchase was made, 3) testing history containing the predicted repeat success/ failure for a product and 4) a list of offers.

Data Type	Properties
Past Transactions	Customer ID, store, product department, product company, product category, product brand, date of purchase, product size, product size, product measure, purchase quantity, purchase amount
Training History	Customer ID, store, offer ID, geographical region, number of repeat trips, repeater, offer date
Testing History	Customer ID, store, offer ID, geographical region, number of repeat trips, repeater, offer date
Offers	Offer ID, offer category, offer quantity, offer company, offer value, offer brand

Table 1 showing the raw data acquired from the *Kaggle* competition: *Acquire Valued Shoppers Challenge*

The pre-offer transactions contains nearly 350 million rows of past customers transactions but we are using only ~15.4 million rows for this project.

3.3 Feature Engineering

Due to the nature of the data, such as one customer having multiple transactions from buying multiple products, it is necessary to merge the transactions of each customer into a row and engineer new features to make sense of the pre-offer transactions file. This is done through summing the quantities and amount spent for each product company, product category and product brand for all pre-offer transactions, for example:

- quantity_bought_from_Company1, ... (to company *l*)
- quantity_bought_from_Category1, ... (to category *m*)
- quantity_bought_from_Brand1, … (to brand *n*)

Upon considering the importance of the time proximity from the offer date, the amount spent for each company, category and brand within 30 days, 60 days, 90 days, 180 days before the offer date are also included as features. A feature to keep track of whether a customer has bought from a company, category, brand during the pre-offer transactions was also added for convenience.

The collated list of features engineered is as follows:

1. Total quantities bought from particular company/ category/ brand
2. Total amount spent on particular company/ category/ brand
3. Total amount spent on particular company/ category/ brand within 30 days before offer date
4. Total amount spent on particular company/ category/ brand within 60 days before offer date
5. Total amount spent on particular company/ category/ brand within 90 days before offer date
6. Total amount spent on particular company/ category/ brand within 180 days before offer date
7. Never bought from particular company/ category/ brand

3.4 Decision Tree Algorithm

Our model was implemented using Apache Spark's machine learning library (v1.5.1) that uses distributed CART (Classification And Regression Trees) to construct the tree based on numerical splitting criterion recursively applied to the training data. The Decision Tree Model implemented in *Apache Spark* is a greedy algorithm that performs a recursive binary partitioning of the feature space by selecting the best split from a set of possible splits to maximize the information gain at a tree node from the set:

$$\underset{s}{argmax}\ IG(D,s)$$

The above described algorithm is also better known as Hunt's algorithm, used in many existing and popular Decision Tree induction algorithms including ID3, C4.5, and CART. In this algorithm, a decision tree is grown recursively by partitioning the training dataset into successively purer subsets. Let D_t be the set of training records that are associated with node *t* and $y = \{y_1, y_2, \ldots, y_c\}$ be the class labels. We can then Hunt's algorithm as such[6]:

Step 1: If the records in D_t belong to the same class y_t, then *t* is a leaf node y_t.

Step 2: If records in D_t belong to more than one class, an **attribute test condition** is used to partition the records into smaller groups. For each outcome of the test condition, a child node will be created. The records in D_t are also distributed to the children according to these outcomes. This is then done recursively to each child node.

Node impurity and information gain

The node impurity is a measure of the homogeneity of the labels at the node. The current implementation of Apache Spark provides two impurity measures for classification - Gini impurity:

$$\sum_{i=1}^{C} f_i(1-f_i)$$

and entropy:

$$\sum_{i=1}^{C} -f_i log(f_i)$$

4. Results & Discussion

4.1 Classification

Building the Decision Tree Model

Legend:
A : company_108079383_spent90
B : company_1089520383_spentTotal
C : company_108079383_spent30
D : brand_5072_spent180
E : company_107127979_spent90
F : category_5824_neverBuy
G : company_105450050_neverBuy

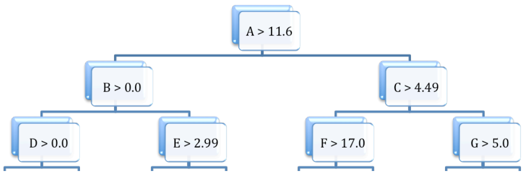
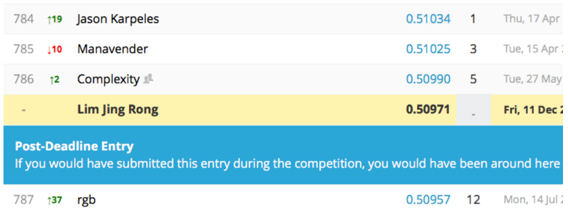


Figure 1 shows the illustration of the first 7 features/ depth 30 with 31131 nodes

The above Decision Tree Model was built using Entropy impurity measure, with a maximum depth of 30, which is the maximum depth the library allows. The decision to select the maximum depth possible was made after taking into careful consideration the amount of features the training dataset contains (close to 400), and it is thus unlikely for the Decision Tree Model to overfit the data. When training the model, a 70/30 split on the data was implemented to perform cross validation, which attained a test error of < 5%.

Submission to Kaggle

We ran the results from our decision trees against the *Kaggle* competition: *Acquire Valued Shoppers Challenge* to see where we stood amongst the submissions and we had achieved a 50.971% accuracy.



784	↑19	Jason Karpeles	0.51034	1	Thu, 17 Apr
785	↓10	Manavender	0.51025	3	Tue, 15 Apr
786	↑2	Complexity	0.50990	5	Tue, 27 May
-		**Lim Jing Rong**	**0.50971**	-	**Fri, 11 Dec**

Post-Deadline Entry
If you would have submitted this entry during the competition, you would have been around here

787	↑37	rgb	0.50957	12	Mon, 14 Jul

Figure 2 showing the possible ranking in Acquire Valued Shoppers Challenge.

Our prediction accuracy ranked us at 786, outperforming hundreds of other submissions. However, with an accuracy of 50.971%, we felt that our model could have performed better.

The top team of the submission achieved accuracy of 63.4%, which had outperform our model by a significant margin. Given this result, there's a possibility that our Decision Tree Model had either underfit or overfit our training dataset.

Using other Classification Algorithms
In an attempt to attain better results, our team tried tuning parameters and explored several other classification algorithms as well.

Algorithm & Parameters	Results
Decision trees : Max Depth 30	50.971
Gradient Boosted Trees with 50 iterations	50.175
Random forest : Max Depth 15 & 50 trees	50.145
Decision trees : Max Depth 10	50.169
Logistic Regression	50.168

Table 2 showing comparison of various Classification Algorithms ran on our engineered training data.

As shown in the above table, given that a Decision Tree of depth much shallower than our Model had perform worse than our current model, it is unlikely that our Decision Tree Model had overfitted the data. It is more likely that the model had underfitted the data due to insufficient depth of the tree.

While our Decision Tree Model performed the best out of the classification algorithms we tried, most of the classification models performed within 1% accuracy of our Decision Tree Model with a max depth of 30. As such, it is likely that the results are relatively algorithm agnostic, and the problem might be unrelated to the algorithm we picked.

Evaluation of our Classification Model
Our team have identified a few possible reasons on our results performing below expectations.

Firstly, we hypothesized of a possibility that our Decision Tree Model had underfitted and was unable to perform a good representation of the training data. The library we used, Apache Spark, allows for the maximum depth of the tree to only 30. Given that we have approximately 400 features, it is possible that the Decision Tree Model was incapable of fully capturing the features available to build the best possible model, thereby underfitting the dataset.

Approaching this from another perspective, there is also a high possibility that our engineered features are not representative of the customer's behavior. As shown in Table 2 above, implementing various different classification algorithms to our training data did not improve our accuracy when evaluated against Kaggle's test data. Therefore, we hypothesized that the engineered features could have been better improved to capture the essence of the massive training data we were presented with.

Lastly, there is also a possibility that we had lost valuable information during the data reduction process. The data reduction process involves filtering out transactions of products which did not appear in the offer data. Performing this filtering allowed for us to reduce the data set from 22GB to about 1GB, or from ~350M to ~15.5M rows of data. While this assumption is relatively safe and widely used online, our Training Model might have benefitted from these additional data to attain a more accurate result.

4.2 Feature Selection
Decision Trees implicitly applies feature selection whilst performing classification. When fitting a decision tree to training data, the top few nodes of which the tree is split are regarded as the important variables within the dataset and feature selection is thus completed automatically as the features are sorted top down by information gain. We used the heuristic, Separability of Split Value (SSV) criterion[2], for feature selection as one of the basic advantage of using SSV is that it can be applied to both continuous and discrete features, as well as compare the estimates of separability despite the substantial difference in the types of data.

The nodes found at the top of a Decision Tree are regarded as the most important variables in feature selection, and features that appeared infrequently are pruned, thus simplifying the Decision Tree Model. In this case, 112 out of 399 features which had not appeared at all, 58 features which have a frequency of less or equal to 10 occurrences, which are pruned from the Decision Tree and be ignored when performing classification.

Top 10 recurring features

Feature Label	**Count**
company_104127141_itemsTotal	12468
company_104127141_spentTotal	9504
company_106414464_neverBuy	3062
company_104127141_spent30	2904
company_1076211171_neverBuy	2788
company_104127141_neverBuy	2528
company_106414464_itemsTotal	2234
company_106414464_spentTotal	1862
company_107106878_spentTotal	1666
company_106414464_spent30	1598

Table 3 showing the top 10 recurring features in our Decision Tree Model.

We picked the top 10 recurring features of our Decision Tree Model and noticed a surprising trend that the first two features, company_104127141_itemsTotal and company_104127141_spentTotal had a huge difference from the 3rd feature. The third feature had relatively the same count as the 4th to 10th features. This is an interesting find as it means that the company with ID:104127141 had the most customers repeating their purchase from it. Even though we do not know what company it represents, we can hypothesize that it is a company that caters to the basic needs of the shopper.

Another insight from the top 10 recurring features list is that some companies are producing products that are unpopular such that not buying from the company is one of the main features in our Decision Tree Model.

5. Conclusion / Future work

Although our Decision Tree Model did not match up the top performers from Kaggle, the model still attained a test error of <5% upon cross validation and is therefore a relatively good model to predict customer repeats. As an additional benefit to using Decision Trees, it allowed us to perform implicit feature selection. Through the top 10 features, we discovered that a particular company has a greater impact in predicting customer repeats.

The project can be further improved by using more pre-offer transactions to provide a more holistic picture of how many items and how much a customer spent on a company, category or brand. This would likely increase our accuracy when tested on the test data. However, performing training and testing on such a large data set would take a much longer time and hence we were unable to do so for this project in a short time frame.

Another possible future work would be to train the data against Winnow as per the work done in *Predicting Customer Shopping Lists from Point-of-Sale Purchase Data*. In the work done by Cumby and his team, the Winnow algorithm had a 2% higher accuracy but 2% lower precision than their C4.5 decision trees. As this Kaggle competition is looking for accuracy in predicting whether a customer repeats a purchase, Winnow algorithm may perform better for this.

References

[1] Reichheld, F. (2001). Prescription for cutting costs. Bain & Company. Boston: Harvard Business School Publishing.
[2] K. Grabczewski and W. Duch. A general purpose separability criterion for classification systems. In Proceedings of the 4th Conference on Neural Networks and Their Applications, pages 203–208, Zakopane, Poland, June 1999.
[3] K. Grabczewski and Norbert Jankowski. Feature Selection with Decision Tree Criterion. Dept. of Comput. Methods, Nicolaus Copernicus Univ., Torun, Poland, Nov 2005
[4] Cumby C., Fano A., Ghani R., & Krema M. (2004). Predicting customer shopping from point-of-sale purchase data. KDD '04 Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, (pp. 402-409). New York.
[5] J.R. Quinlan. C4.5: Programs for Machine Learning. Morgan Kaufmann, 1992.
Classification: Basic Concepts. Decision Trees and Model Evaluation. In Introduction to Data Mining.
[6] Tan P. N., Steinbach M., Kumar V. Classification: Basic Concepts. Decision Trees and Model Evaluation. In Introduction to Data Mining.