Santander

Santander Product Recommendation

The Santander Data Group

Written by

Chenxi (Celia) Huang

William Raikes

Contents

I. Project Description

1. Project Name
Santander Product Recommendation

2. Our Goal

Based on customers' past behaviors and those of similar customers, to **predict which products** their existing customers will use **in the next month.**

For example, this is similar to what the data set looks like

	Features							Labels			
	fecha_dato	ncodpers	ind_empleado	pais_residenci	sexo	age	fecha_alta	ind_nuevo	ind_reca_fin_ult1	ind_tjcr_fin_ult1	ind_valo_fin_ult1
9704	1/28/2015	952138	N	ES	H	30	9/30/2011	0	0	0	0
120754	2/28/2015	952138	N	ES	H	30	9/30/2011	0	0	0	0
186516	3/28/2015	952138	N	ES	H	30	9/30/2011	0	0	0	0
252157	4/28/2015	952138	N	ES	H	30	9/30/2011	0	0	0	0
272539	5/28/2015	952138	N	ES	H	30	9/30/2011	0	0	0	0
338163	6/28/2015	952138	N	ES	H	30	9/30/2011	0	0	0	0
454973	7/28/2015	952138	N	ES	H	30	9/30/2011	0	0	0	0
494807	8/28/2015	952138	N	ES	H	30	9/30/2011	0	0	0	0
636849	9/28/2015	952138	N	ES	H	30	9/30/2011	0	0	0	0
674624	10/28/2015	952138	N	ES	H	30	9/30/2011	0	0	0	0
825030	11/28/2015	952138	N	ES	H	30	9/30/2011	0	0	0	0
863829	12/28/2015	952138	N	ES	H	30	9/30/2011	0	0	0	0
1030664	1/28/2016	952138	N	ES	H	30	9/30/2011	0	0	0	0
1127817	2/28/2016	952138	N	ES	H	30	9/30/2011	0	0	0	0
1191618	3/28/2016	952138	N	ES	H	31	9/30/2011	0	0	0	0
1293000	4/28/2016	952138	N	ES	H	31	9/30/2011	0	0	0	0
1345086	5/28/2016	952138	N	ES	H	31	9/30/2011	0	0	0	0

Note: The above is just an example.
The actual data sets are more complicated than this.

3. The Data

- number of observations = 13 million
- number of features = 24 per observation
- number of labels = 24 per observation

II. Data Processing and Cleaning

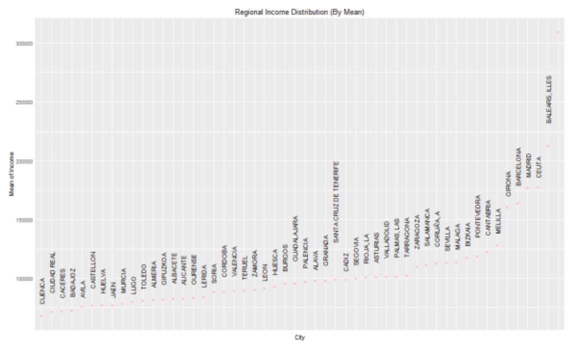
1. Problem: Super Large Data Set
Solution: sampling
Results: reduced number of observations from 13 million to 1 million

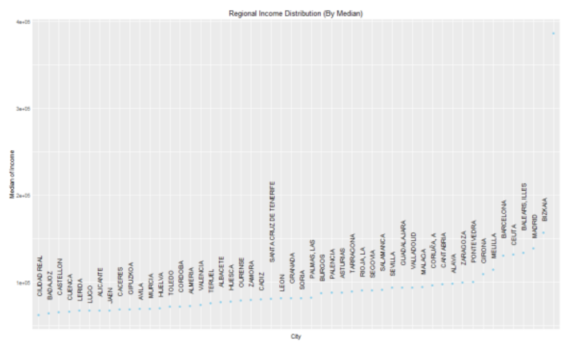
2. Problem: Empty Strings: “” (in categorical features)
Solution: treating it as one factor level
Results: assign “unknown” to empty strings

3. Problem: Missing Values (in continuous features)
Solution: Filling in values case by case
Results: filled in 11 columns, by observing data statistics and distributions

For example, for the feature “income”,

- We found using directly the total average not very satisfactory in terms of reflecting individual incomes;
- We divided data by “cities” and assign the median/mean income of that city to the missing income value in the corresponding data.





As indicated in the graphs above, whether by city median or city mean, the clusters are well separated. So it seems reasonable to use this approach.

**for our data, we chose city medians to fill in the missing values in “income.*

III. Feature Engineering and Selection

(I) Feature Engineering

1. Does time matter?

- Originally we thought about creating features based on the existing features.
- For instance, we created this “month” feature to reflect customers’ purchasing behaviors with regard to each month.
  Our reason behand this is that, during some particular periods of the year, e.g. Christmas, customers would be inclined to purchase more.
- However, the cross-validation error rate in the same model increased from 9% to 10% by doing so. Therefore, we abandoned this idea.

2. How to create features to reflect past behaviors?

- Thoughts: can current situations be used to predict future (martingale)?
- Solutions:
  Appending current month’s labels to original features.
  This way, we now have 24 more features for each data

For example, this is what the new data looks like:

New Features		New Labels
April’s Features	April’s Labels	May’s Labels

** We will discuss and evaluate the impacts of creating new features in this way in the model selection and evaluation part.*

(II) Feature Selection

- We only have 24 features to start with.
- Hard to do feature selection with multi-label classification problems.
- It is not easy to interpret just using usual ways, e.g. PCA, random forest and gbm to do feature selection here.

IV. Model Selection and Evaluations

Model Construction

The objective of our product recommendation algorithm is to compute a multi-label classification. After given input data, the algorithm should predict an outcome of multiple classifications, and discern which products an individual will have at any given time (in lieu of just predicting one product from many). This is a slightly more obscure approach, which is evident by the very few packages that use multi-label classification algorithms. Therefore, to work around this barrier, we chose to build an individual model for each product and use the 24 separate models as a single algorithm that produces a multi-labelled response.

A baseline model was constructed by finding how likely an individual would have a certain product. An average score was calculated based on how many months an individual had the specific product, and with a cutoff value of 0.5, we determined if the user was more likely to have the product than not in the near future. The more advanced methods employed were: rFerns, XGBoost, Random Forests, and SVM.

1. What is the problem?

Multi-label Classification, different from multi-class classification problems.

This means that normal algorithms like Multinomial, kNN or logistic regressions do not apply here.

2. Our Models (what we can do in R)

a. **Baseline (= clustering?)**
b. **rFerns in mlR**
c. **XGBoosting**
d. **Random Forrest**
e. **SVM**
f. **Not in R: ML-KNN, Neural Networks.**

3. Not in R:

ML-KNN, kernel methods for vector output, Neural Networks, which can be explored in the future.

IV. Model Selection (continued)

(A) The Baseline Model

1. Algorithms

(1) **Only Feature: customer ID**
(Ignoring all other features, e.g. cities, age, income.)

(2) Just consider the past purchasing behaviors of a certain customer A

(3) **Assign Predicated Value = Majority Label**

(4) Calculated means of each label column for customer A
If P(A purchases product 1) > 0.5, then predict yes (assign label value = 1)
If P(A purchases product 1) <= 0.5, then predict yes (assign label value = 0)

2. Cross Validation Error Rate = 0.87% (K=5)
= 0.88% (K=1)

3. Pros & Cons

Advantages	Disadvantages
no brainer. easy to understand and compute. Great performance	not using most features, in a sense similar to guessing

The baseline model performed surprisingly well, with a cross-validation error rate of 0.87%. Since this model was easy and efficient to compute, and had a high performance, it set a high bar to beat.

IV. Model Selection (continued)

(B) Multi-label Classification in R
{mlR} rFerns

1. Algorithms

- **{mlR} Machine Learning in R, came out in Oct 2016**
- **Usually: Problem transformation methods**
  to transform into binary/multiclass classification
- **Now we can: Algorithm adaptation method**
  to adapt multiclass algorithms so they can be applied directly

2. (Best) Cross Validation Error Rate = 24%

K=5	Regular Method	Regular Method + Added Features
Error Rate	24.00%	30.18%

3. Pros & Cons

Advantages	Disadvantages
predicting all labels at the same time.	demanding requirements on the format of the data, e.g. Labels = logicals
convenient & neat technique.	not very good results

The rFerns model had an error rate of 24% when using added features, and 30% without the added features. Even though this was the only method used for multi-label prediction, it had one of the worse predictive performances. It is probably due to its demanding requirements on the format of the data.

IV. Model Selection (continued)

(C) XG Boosting

1. Algorithms

Predicting labels column by column

2. (Best) Error Rate = 0.3%

K=5	Regular Method	Regular Method + Added Features
Error Rate	4.12%	0.29%

It is not hard to see that with the new features created, the performance was largely improved.

3. Pros & Cons

Advantages	Disadvantages
Great Performance	doesn't not take into account any of the prior months – strong assumption on the data!
Simply Implementation	

The XGBoost had the best cross-validated performance with an error rate of 0.29% with added features, and 4.12% without added features; this model also is efficient as processing time was very reasonable compared to the two remaining models.

IV. Model Selection (continued)

(D) Random forest

1. Algorithms
Predicting labels column by column

2. (Best) Error Rate = 3.81%

K=5	Regular Method	Regular Method + Added Features
Error Rate	4.09%	3.81%

As we can see from the table, with new features, there IS some improvement, which again is evidence that our conjecture is working.

3. Pros & Cons

Advantages	Disadvantages
enhance the strength of the model through averaging the results	Not very consistent outcomes among labels. 99.99% Vs 74.56%
Good results	Improvement not enhanced by a lot

IV. Model Selection (continued)

(E) SVM

1. Algorithms
Predicting labels column by column

2. Error Rate = 4.66%
Tuning parameter (radial kernel, default gamma, cost = 0.001)

3. Pros & Cons

Advantages	Disadvantages
Widely used model, stable	Hard to tune Takes way too long time
easy to interpret	Once can fit only one Column of labels
Good results	

** As it took way too long to run SVM with new features added, it was not pursued afterwards. But regarding the regular model, we imagine it would be pretty similar to random forest.*

IV. Conclusions

1. Feature Selection

- Appending labels as new features is working!
- Sometimes less is more (baseline's performance = 0.88% is pretty good)!

2. Model Selection

- Direct Multi-label Classification in {mlR} doesn't work too well here.
- XG boosting is the best

2. How to choose in the end?

- Baseline: stable
- New Features: strong assumptions. Works in the short-run.
- XG Boosting + New Features is our selected model

In conclusion:

The Feature Part:
From this analysis, it is apparent that the added features (i.e. the current month's product selection) provided more insight into what the individual would hold in the future. This is in addition to the fact that only one month of data was used to predict and test these models, as opposed to using all 17 months.

The Model Part:
The baseline model performed surprisingly well, with a cross-validation error rate of 0.87%. Since this model was easy and efficient to compute, and had a high performance, it set a high bar to beat. The rFerns model had an error rate of 24% when using added features, and 30% without the added features. Even though this was the only method used for multi-label prediction, it had one of the worse predictive performances. The XGBoost had the best cross-validated performance with an error rate of 0.29% with added features, and 4.12% without added features; this model also is efficient as processing time was very reasonable compared to the two remaining models. The SVM and Random Forest models performed similarly, in predictive performance and efficiency. Both models took an inordinate amount of time to build and validate, yet the error rates for both were 5% and 3.7%, respectively, without the added features. In fact, using the added features proved to be a time constraint and became infeasible to construct.

V. Future Considerations

1. How we calculated the cross-validation error rates?

The main objective of the project: predict additional products next month.
The error rate reflects which products will be owned, not which products were recently acquired

2. Feature Engineering

- It is hard to incorporate past behaviors into account
- Maybe we can append baseline labels as features because they are already the averaged results of past product purchasing records.

3. Combined Model?

Combining different models can lead to averaging the error rates. For example, we can calculate the majority votes for each label and assign it to the most likely scenarios.

4. R Shiny?

We can build Shiny App to reflect the predictions we made and present the results more vividly.

VI. References

- https://mlr-org.github.io/mlr-tutorial/release/html/multilabel/index.html#predict
- https://en.wikipedia.org/wiki/Multi-label_classification
- https://www.kaggle.com/c/santander-product-recommendation
- https://cran.r-project.org/web/packages/MLPUGS/vignettes/tutorial.html
- And many more!

Thank you for reading this report.

Celia (Chenxi) Huang
&
William Raikes