Santander Bank product recommendation

By Breaking Code
Ayush Sawlani (IMT2018014)
Ayush Mishra (IMT2018013)
Aarushi Goenka (IMT2018001)

***Abstract* - Trying to build a recommender system is helpful in ways more than one. As the overall motive of the Santander bank, or any other organisation is to increase its sales, understanding the customer and their buying habits can play a major role in boosting their sales, as well as better experience for customers. We tried to understand how different months and different seasonal trends and other various factors can make customers more likely or buy or drop any product. In the end we came up with a recommender system using traditional Machine Learning methods which can help Santander bank provide customers personalized product recommendations.**

***Index terms* - Feature Engineering, Logistic Regression, Random Forest Classifier, SGD Classifier, LGBoost, CatBoost, Hyperopt, Voting**

I. Introduction

Under their current system, a small number of Santander's customers receive many recommendations while many others rarely see any resulting in an uneven customer experience. In this competition, IIITB students were challenged to use Machine Learning to predict which products their existing customers will add or drop in the next month based on their past behavior and that of similar customers.

With a more effective recommendation system in place, businesses like Santander can better meet the individual needs of all customers and ensure their satisfaction no matter where they are in life.

II. Dataset

In this competition, we were provided with 1.5 years of customers' behavior data from Santander bank to predict what new products customers will purchase. The data starts at 2015-01-28 and ends at 2016-04-28. The data is basically a monthly record of every customer which is collected on 28th of each month.

The dataset had record of customers' basic information such as his/her age, monthly household salary, gender, residence, etc. as well as information specific to the bank such as customer type(active/inactive/primary etc.), customer seniority, etc. The train dataset also had the information on whether a particular customer possesses a particular bank product such as "credit card", "e-account", etc.

Given the dataset and the statement it was inferred that we had to build classifiers for all the products and use that to predict which products will be added or dropped in May of 2016.

III. Observations

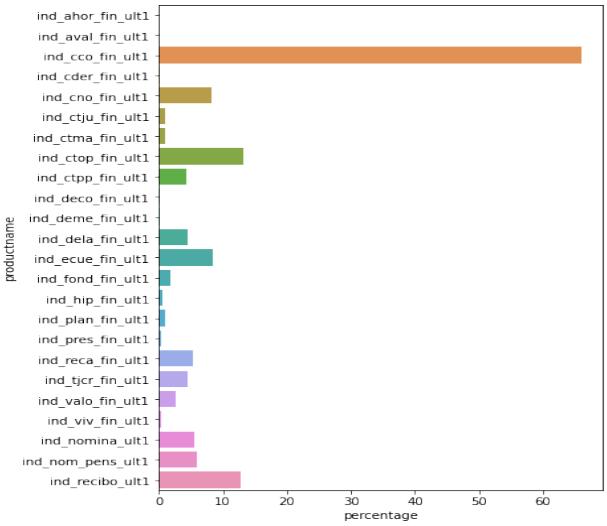
There were numerous insights that we got from the dataset and here are the main ones:

- The data was unclean on some grounds:[1]
  - After seeing the Null value count for many columns the number "27734" repeated a lot of times. After checking it was found that the dataset actually had 27734 NULL rows. They were dropped.
  - Columns such as conyouemp and ult_fec_cli_1t were dropped because they had too many null values.
  - The unique values of age and renta(gross household income) had ridiculous values(for example age had values like 164, 2, etc.) which were clearly outliers.
  - The column antiguedad(customer seniority) had garbage values at many places.

  Many more discrepancies were there in the dataset which needed to be handled.

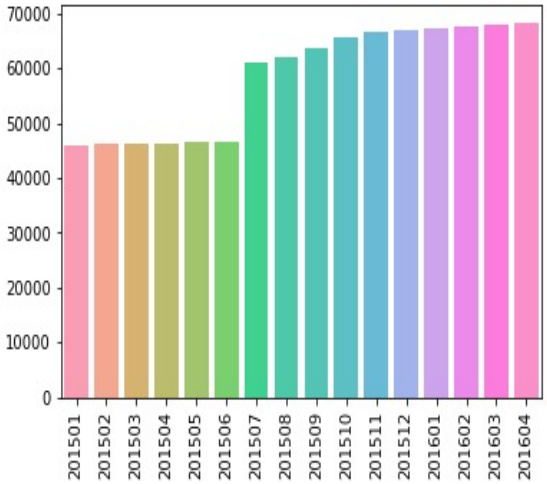
- It was observed that for only about 0.054 of the total customers, the products actually "toggled" in the month of April 2016. This gave us an idea of what kind of score we could expect and we could aim for.[2]

- Following is the product popularity graph

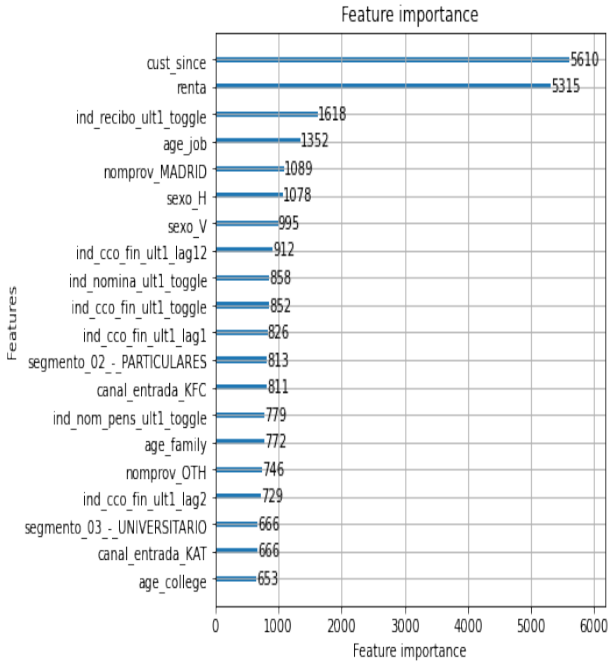


The above graph shows that the dataset is imbalanced for a lot of products, and hence may not be recommended by our classifier algorithm. We may be required to do sampling to handle this.

- We plotted the graph for the number of customers for each month and found that the number of customers remain almost constant except for a spike in July 2015 .



- Following is the feature importance graph of "recibo" product according to lightbgm classifier:[3]



This graph was more or less the same for all products, and it was inferred that customer seniority, household income, age of the customer and the previous product choices for that specific product(lags and toggles, will be explained later) always gave the best split.

IV. Pre-processing

A. Data cleaning and encoding

When we looked through the columns, we found that the dataset of this competition was mostly a clean dataset and would not require much data cleaning other than filling of null values and removing some outliers.

We filled the null values of renta (gross annual income) by the median of the province as we saw that the renta values varied for different provinces[1]. For, tiprel_1mes, we saw that all the customers whose values were null were new customers, that is, their date of joining the back was within the last 6 months, hence we filled it by 'A' which meant the customer was active. We filled all the other categorical columns by the mode.

We removed outliers in the data by removing the extreme 1% of the data for income, and removed values more than 95 and less than 12 for age.[1]

Since the column antiguedad had any garbage values, we replaced it with another column 'cust_since' which was the number of months since a customer was a member of the bank.

We applied normalisation on cust_since and age since we removed the outliers for them and they didn't follow normal distribution, whereas used standardisation for renta as it had outliers(even after we removed the extreme 1% of the data) and followed a normal distribution. For encoding, we used one hot encoding for all the categorical data. For canal_entrada, pais_residencia and nomprov, we only took the categories of top 10 categories and put the others in a category called "OTH" and put "UNK" for all null values. This helped us reduce the number of columns from more than a 100 to less than 25 after doing one hot encoding.

Features with large null values were dropped. Tipodom was also dropped because it only had 1 unique value. Province code had bijective mapping with province name hence the former was also dropped

B. Feature Engineering

For this competition, we realised adding new features were of utmost important. As this was a timeseries data, we found that most of the information was present in the lag feature of the products and of other non-product columns and hence we tried various combinations and features where we could extract new information related to the ownership of products in previous month.[4][5]

The major feature we added was the lag feature for the product columns. Adding just one lag feature, that is, the value of the product columns one month, increased the score of our most basic model from 0.02650 to 0.03546. After this, we added another lag column for the values of products 2 months prior, that is, lag2. We then played around with the number of lag features to take and of which months to take. We finally decided upon taking the values of lag1, lag2 and lag12, which are basically the ownership of products 1 month prior, 2 months prior and 12 months prior. We tried to add more lag

features but kept running out of memory on trying to add more than 3 lags. In the end, we decreased our timeline to only to April 2016 and took 12 lag features, that is, taking previous ownership data for all 12 months prior, therefore adding 12*24 new columns.

We then tried to observe the lag values for the non-product columns. We checked whether the values of some categorical columns like ind_actividad_cliente, segmento, province code, indrel_1mes, and indrell changed from what their values were 1 month ago, 6 month ago and 12 months ago. We found that there was a significant change in the values of ind_actividad_cliente and segmento between the current values and values 6 months ago. Hence, we decided to add lag6 values for these two columns.

Another important feature we added was the toggle feature. As the aim of the competition was to predict the products that a customer would add or drop, we decided to add a feature which would give a count of how many times a product was added or dropped in the last 6 months. We created 2 new columns for each product, 01_toggle and 10_toggle. 01_toggle gave the count of the time a customer added a product, that is, the product ownership goes from 0 (does not own) to 1 (owns), and 10_toggle gave the count when a customer dropped a product.

V. Model Selection

We experimented with a lot of models through this past 1-1.5 months, from linear models to tree based models , we tried all of it. Trying such a variety of models was a huge benefit for all of us because first of all it was a great learning experience and secondly it was a huge factor in us winning the competition. It played a very major role in the end when we did our Ensembling of different models which eventually led us to winning the competition.

Here are the models that we tried along our way in our project:

- Linear Regression[6] - In the early stages of our project we had figured out that a few of our product columns were linearly separable , so we started off with Logistic Regression as our first model,it was also the model we were all quite familiar with at that time.
  We did not perform any extensive hyperparameter tuning on Logistic Regression ,we just did whatever we could do manually through hit and trial and after a lot of feature engineering managed to get our best score of 0.04188 using Logistic Regression.

- SGD Classifier[7] - We used this quite late in our Project ,our main purpose to use this was so that we can use it in our Ensembling. We used the loss function in SGD which is basically Logistic Regression, we only used this because this was very fast compared to the Logistic Regression function.
  This was not much helpful.

- Random Forest Classifier[8] - This was one of the 3 tree models that we tried , even this we tried much later in our project and our sole purpose was to use it during Ensembling.We did not really tune our hyperparameters of RF Model much because we were short on time and we were only using it for Ensembling, we could only do as much it was possible to do manually.
  RF Classifier was giving quite good results even though not as good as LGBM ,but we decided to go ahead and submit a solution using only RF Classifier ,although we never submitted an ensemble solution using RF Classifier because we were getting better results using other different combination of models.

- Light Gradient Boosting Machine(LGBM)[9] - This is one of the models that we used in our project . Since we had 2 models running this was one of those 2 models that were used. LGBM was also the main model that we had boiled down to and were using it even before ensembling. We had also used Hyperopt for finding the suitable and best set of hyperparameters for LGBM and even though it did not really give us the best set of hyperparameters ,it gave us a clue that we were close to a good set of hyperparameters after which we just found out the hyperparameters manually.
  We submitted a lot of solutions using LGBM and the best solution using only LGBM without any ensembling was 0.04631.

- Cat Boost[10] - This was the final model that we used in our project. We had not used CatBoost before our Ensembling procedure ,so we never really had any exclusive submission using CatBoost but since we are running 2 models on our project for the purpose of Ensembling ,CatBoost is one of the models that is running. Even though we never really submitted any submission with only CatBoost we found out through local testing that this was performing better than LGBM and if we had used this instead of LBGM we would have gotten better scores.
  We never really did any hyperparameter tuning for it but we were able to find a reasonably good set of parameters using our experience from LGBM ,which performed quite well on our local testing. We submitted only Ensembling solutions using this and it performed really well in coalition with LGBM.

VI. HyperParameter Tuning

Hyperparameter tuning using searches such as RandomSearch[12] ,GridSearch[11] and Hyperopt[13] takes a huge amount of time and using it for every model that we used would take a huge amount of time especially considering we had 24 target columns.

So we performed hyperparameter search for only one model that was Light Gradient Boosting Machine (LGBM) and even on it we performed for only selected target columns which we

thought had a major impact on the final result.As mentioned above ,even though it did not give us the exact best set of hyperparameters for our model it gave us a clue to what our best set could be,after which we manually found out a pretty good set of hyperparameters using hit and trial methods.

For all the other models we found our best set of hyperparameters intuitively using hit and trial method and we were quite successful in finding a pretty good set of hyperparameters for almost all of them.

VII. Ensembling

Ensembling is the use of multiple predictive models together using techniques such as bagging and boosting which perform better than the standalone models.

We used the bagging technique for our Ensembling , we had weights associated with our models and our final result was the combination of both the models with their respective weightage. The fact that both of our models were run on slightly different feature engineered datasets gave us a really good result, which ultimately turned out to be our winning submission.

We locally tested our models to find out the perfect combination of weights for both of our models.

VIII. Post-processing

During one group discussion session, one of the teams(Survey Corps) presented with the idea that there were two products that were almost always brought together. Upon further analysis, we found that ind_nomina_ult1(Payroll) was never bought without ind_nom_pens_ult1(Pensions) if ind_nom_pens_ult1 was not owned in the previous month. This means that if both of them are not owned in a particular month, it is not possible for the probability of buying ind_nomina_ult1 to be greater than that of buying ind_nom_pens_ult1 as both would be bought together. Hence, we made the probabilities of nomina and nom_pens to be equal if the probability of nomina was more than nom_pens and both were not owned in the previous month. After this function, although a lot of values were affected, there was no increase in our score, hence we decided not to include it in our final submission.

IX. Training Details

A. Locat Training

In order to test our models locally, we split the data using test train split from sklearn[14], made predictions for the hold out set and summed the log loss[15] for 4 popular products: 'ind_recibo_ult1', 'ind_valo_fin_ult1', 'ind_tjcr_fin_ult1', 'ind_nomina_ult1'. As we tried to decrease the log loss for the hold out set, our score increased in the leaderboard. Therefore, it proved to be a good testing measure. Given the limited amount of submissions we had, this metric helped us a lot.

B. Timeline Selection

We tried various timelines for our training according to our intuition and then worked on selecting timeframes which gave the best results. First, we tried to capture seasonality by selecting data of the same month as our test data but of year 2015. Selecting our data from March 2015 - October 2015 gave us a good idea of seasonality. We tried to shorted our timeline to capture it even better, but found the score to decrease. We also selected April 2016 for our timeline to capture the recent trend and found this to be a good score as May 2016 (our test month) had trends similar to April 2016. We then went on to try various other months by observing their product popularity trends through the months, and observed that taking the months of April 2015, July 2015 and January-April 2016 gave us the best score. In the end however, one of our models in the ensemble had only data of april 2016 with 12 lag features.

X. Results

Below table depicts the score we got for different models:

SL No.	MODEL USED	BEST SCORE
1	Logistic Regression(LR)	0.04188
2	Random Forest Classifier	0.04574
3	Light Gradient Boosting Machine	0.04631
4	Ensemble (LGBM + CatBoost)	0.04659

XI. Conclusion

We would like to conclude that we were able to come up with an efficient model to predict what kind of products a customer would like to purchase in future and what kind of products he would like to drop.

Acknowledgment

We would like to thank our professors Professor GS Raghavan and Professor Neelam Sinha for imparting us the knowledge in the field of Machine Learning and giving us a great headstart in this world of Machine Learning.

Next we would like to thank the teams: survey corps, tamethatbench and modulo3 for having various fruitful discussions with us. We would also like to thank all the teams who participated in the contest. They gave us a tough fight which motivated us to try more.

Lastly we would also like to thank our amazing TA's who have been teaching us all the things that we have needed for the course and also for our project and the assignments and we would especially like to thank Tejas for his constant guidance and help in clearing all the doubts that we faced throughout this past 2-3 Months.

REFERENCES

[1] https://www.kaggle.com/apryor6/detailed-cleaning-visualization-python
[2] https://www.kaggle.com/sudalairajkumar/maximum-possible-score
[3] https://lightgbm.readthedocs.io/en/latest/pythonapi/lightgbm.plot_importance.html
[4] https://machinelearningmastery.com/basic-feature-engineering-time-series-data-python/
[5] https://ttvand.github.io/Second-place-in-the-Santander-product-Recommendation-Kaggle-competition/
[6] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
[7] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html
[8] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
[9] https://lightgbm.readthedocs.io/en/latest/
[10] https://catboost.ai/docs/concepts/about.html
[11] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
[12] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html
[13] http://hyperopt.github.io/hyperopt/
[14] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html
[15] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.log_loss.html