Santander Product Recommendation

Team Name: Survey Corps

Sai Rithwik M
IMT2018061
sai.rithwik@iiitb.org

Saksham Agarwal
IMT2018065
saksham.agarwal@iiitb.org

Sama Sai Karthik
IMT2018067
sai.karthik@iiitb.org

***Abstract*— Through this document we are trying to understand how with a given behavioural data of a customer over a period of time we could predict the top 5 changes in a customer's inventory of products.**

***Index Terms*— Bagging, Decision Tree, Exponential Smoothing, Holt-Winters, Label Encoder, Lags, LGBM Classifier, Logistic Regression, Min Max Normalisation, One Hot Encoder, One vs Rest Classifier, PCA, Stratified K Fold Validation, Target Encoder, T-SNE, Voting, XGB Classifier**

Problem Statement

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. You must predict which products their existing customers will add or drop in the next month based on their past behaviour 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.

Dataset

In this competition, we were provided with 1.5 years of customers behaviour data from Santander bank to predict what new products customers will purchase. The data starts at 2015-01-28 and has monthly records of products a customer has, such as *credit card*, *savings account*, etc. These products are the columns named: *ind_(xyz)_ult1*, which are the columns 25 to 48 in the training data. We had to predict the top 5 products that a customer will buy or drop in addition to what they already had at 2016-04-28. The training data had 12715856 rows, 951952 unique customers and 48 columns. Details regarding the data columns can be found in the link mentioned in references[1].

Introduction

Banking industry has grown up 5 fold in the last 10 years[2]. With such an unprecedented growth, the banks must be ready to cater to the needs of their customers by providing the right recommendations to them based on the behavioural patterns. Product recommendation engines perform a really good job in creating a sense of satisfaction among consumers during and even after their searching session. Be it an existing customer or a new customer, based on the behavioural patterns of a customer a product recommendation system accurately predicts the need of a customer. From a bank's, a perspective recommendation system also helps them identify which product to focus more on, for a particular customer, hence improving their efficiency.

Data science processes such as exploration, manual feature engineering, can become convoluted, and often call for empirical and domain knowledge. Most data analytics and product recommendation in retail banking revolve around the concept of behavioral similarity, for instance: studies and campaigns on customer retention, product recommendations. There has been a fairly good amount of research going on improving the product recommendation systems in banking sectors, and one of the most notable ones being devising systematic baselines for such models, which was extensively researched by Baldassini and Serrano[2]

The rest of the paper proceeds as follows **Sec. 1** describes about preprocessing on our dataset **Sec. 2** discusses about the inferences we drew from exploratory data analysis on our data **Sec. 3** speaks about why we chose a certain model **Sec. 4** covers on what all strategies we used on our model to improve the scores **Sec. 5** discusses about the challenges we faced while working on the project.

I. Preprocessing

One of the major advantages with this dataset is that it was moderately categorical. But due to the huge amount of data present it was really prone to have outliers and errors and handling all these was required.

Before we could go ahead with preprocessing, one of the most important things we did was to convert everything from Spanish to English so that we could understand how to proceed even better.

A. Imputation

The first observation we saw from our null value analysis was that columns like *pais_residencia*, *tiprel_1mes*, *ind_actividad_cliente* and all some more columns had the same null value count. This led to an intuition that maybe all of them had the null values at the same time and when we checked we found the same. But this being a time series data and the *ind_empleado* being an important factor they cannot be directly dropped, if such records were present in test. Since there were none such records in test we decided to drop these records.

The columns *conyuemp* and *ult_fec_cli_1t* had about 99% null data and hence we decided to drop these columns. Next we found out that our *cod_prov* and the *nomprov* represent the same thing with one being the label encoding of the other column, thus we dropped *nomprov*. Further, we went on to *segmento*. During our correlation analysis we found that this column was highly correlated with the age. Thus we filled the null values with

- age < 40 with *03 - UNIVERSITARIO*
- age < 50 with *02 - PARTICULARES*
- age > 50 with *01 - TOP*

Next, we looked at the records of the province codes that were not null and most of them had *pais_residencia* as Spain. So we decided to fill all the null categories that had province *ES* as 28 which was the mode and for the rest, we assigned a new province code '0'.

One more thing that we did to handle the null values in the *canal_entrada* was to find the *canal_entrada* of the customers who joined in the same year as the null value and filled it with the mode for that year. If there was no person joining in that year, we imputed directly with the mode of the entire dataframe. In column *sexo*, we took the modulus of 2 for the customer code and assigned with a H or V as the count for them was not very biased and we didn't get any trends from them. This was done so that a same customer is not assigned a different *sexo*. Finally, for the *renta* column, we decided to fill the null value with the median of the income of the non null values grouped on the basis of the *cod_prov*. While the above ones are for the behaviour of customers we also found out that there were a few columns in products which had null values. Since their count was very less, we had decided to fill it with a zero.

B. Outliers

Due to the huge size of this datframe it was bound to have many outliers. One of the major issues were the presence of strings in a numerical column, due to the presence of a Tab, during data entry. We easily solved it by converting the string to integers. There were also a few outliers like the presence of -999999 in *antiguedad* column. We replaced such outliers with 0.

C. Encoding

We had 2 different approaches for this. We shall explain both the methods we tried for encoding below.

The first and the more simple approach we tried was to label encode all the categorical columns. The second approach was to categorise the columns depending on the number of unique values and encode them appropriately.

In this method we had set up a condition that all the columns that had less than 4 unique values shall be encoded using One Hot Encoder, and the remaining using Target Encoding. One of the major disadvantages of Target Encoding is that it allows data leakage which is not well for validation[3]. So instead, we had to go for CatBoost Encoder which had the similar implementation but avoided data leakage[4]. On further research about an appropriate encoder for such dataset we had decided to go with Helmert Encoder[5], as we had observed a decent amount of ordered data in it, and decided to try it out too. But even after such precise encoding we had observed that our accuracies in the former method were better than the latter. Hence we decided to go with Label Encoding. One reason why we felt it didn't work out as planned was because we felt that the data was over encoded, and label encoding doesn't complicate it.

Subsequently we made a few more changes to our pipeline to improve the model. One of the most important additions has been scaling the entire encoded data using Min-Max Normalisation.

II. Exploratory Data Analysis

Before we could begin with encoding we surely wanted to see how the data was distributed and choose models accordingly. One of the most important thing to lookout was how correlated was behaviour of customers to the products they chose.

One relation that we saw was the relation between *cust_rel_type* and the *indrel_1mes*, where most of the customers seemed to be of type Primary and can be either Active (A) or Inactive (I) and if the customer type was an existing customer, they for sure are Inactive, thus we can see a correlation which can be used to fill up the null values for this.

cust_rel_type	customer_type	
A	1.0	367622
I	1.0	324555
P	3.0	32
I	2.0	1

Fig. 1. Analysis on Customer type

The account contracts for the customers have been there from 1995 and there seems to be an increase in the start date of account contracts in the later stages with a more seasonal peak (from July to December).

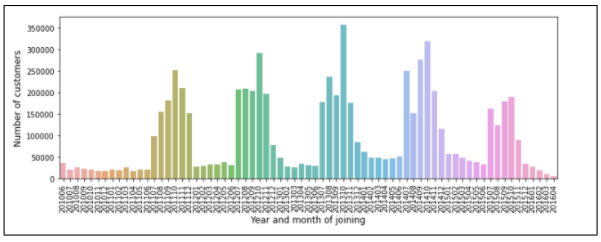

Fig. 2. Analysis on Existing Customers

The distribution was quite similar during the first half of the year but after the 6th month, there was a significant increase in the number of customers, which then ever slightly increased till the next year as well.

The renta column was heavily skewed and it was handled appropriately by taking it's logarithm.

We did some of our EDA to see how the products would look like when they were grouped according to a certain

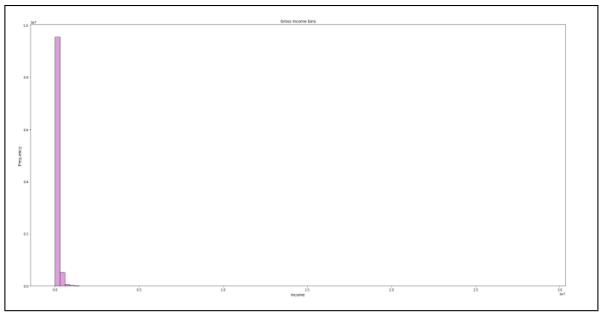

Fig. 3. renta before applying log

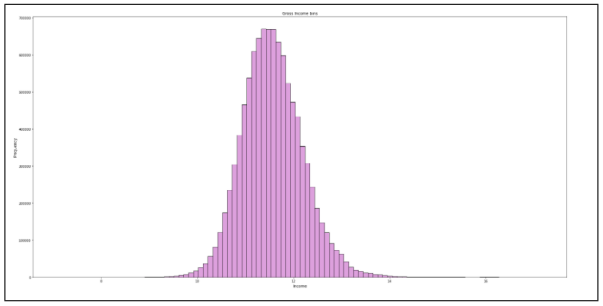

Fig. 4. renta after applying log

feature. Some of the most interesting features among these plots were ones with age. From here we can infer that most of our *ind_ctju_fin_ult1* products are found in the age group 0-20, then for *ind_hip_fin_ult1* maybe there is a condition where you can't get any before the age of 40. Also it seems that maybe, most of the *ind_ctju_fin_ult1* product holders turn to ind_cco_fin_ult1 as seen due to the boost in *ind_cco_fin_ult1* at the age 20. For remaining products there seems a general Gaussian trend for the products between the age of 20-50.

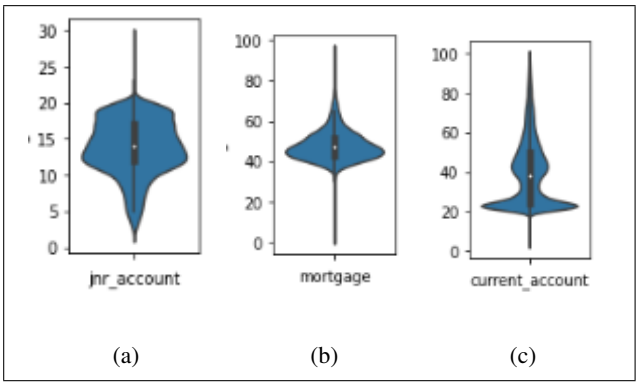

Fig. 5. Product Analysis

III. Model Selection

Initially we had identified that this is a Multi-Label classification problem where out of the 24 classes we have, we needed to segregate top-5 products based on how likely the product will toggle. This led us to choose classifier models like One-vs-Rest classifier which fits one classifier per class. While One-vs-Rest classifier classifier was working well, we were sceptical that it was overfitting, as we were not getting satisfactory accuracies during validation. So our natural choice was to choose a tree based model, specifically a boosting one. Out of the many classifiers available our natural choice was to go for XGBoost and LightGBM. XGBoost was our top preference because it had specific objective functions for Multi-Label Classification and evaluation of weak learner using MAP@5, which is really suitable for our case. Another reason why we chose a model which used decision trees was that, for a few products, when split based on certain attributes resulted in pure samples after splitting. For example:

- If *cod_prov* = 28.0 then *ind_aval_fin_ult1* was definitely 1
- If age < 20 most probably ind_ctju_fin_ult1

Hence, XGBoost was our go to algorithm. We had also checked out a few other models like AdaBoost etc. but one of the major disadvantages of using them are that they have too many hyperparameters to tune and since we have to predict for 24 products we didn't want to go the AdaBoost path. LightGBM gave us the speed by making all the predictions in under half-an-hour but it took a huge toll on our accuracy. Hence we wanted a good balance of accuracy and speed and XGBoost proved to be a good choice for us.

Once we had tried out few such models there was a slow growth in accuracy while we performed hyperparameter tuning. Hence a natural choice that struck to all the teammates was to go for voting. We had also considered stacking, but for stacking to work out, we needed diverse models which performed equally well. In our case, there was a considerable difference in the accuracies of diverse models, and hence we decided not to go with stacking ensembles. Finally, we had consolidated all our top performing models and voted on them to get the final score.

IV. Training Details

A. Lag

We present a perspective with respect to a product where we have dug up to see the differences. The product is *ind_cco_fin_ult1*. We observed a lot of volatility in this product thus making it a potential product to look out for final predictions.

We had analysed all the records of certain customers and we inferred that *ind_cco_fin_ult1* had lot of changes. We observed that although product name changed a lot, there wasn't a change in any other attribute, except the month of we are looking at. One such customer id is 536763. Now if we think about working of the boosting algorithms say, XGBoost, decision trees are base models used. If we expand on decision trees, we realise no matter how deep the tree grows it cannot distinguish label 0 and label 1 as none of the train attributes differ among these records.

Examining Fig 6, the probabilities that gradient boosting algorithm assigns in former case is around "0.5" signifying that algorithm isn't certain about the predictions. We observe this as the direct effect of the algorithm not having enough

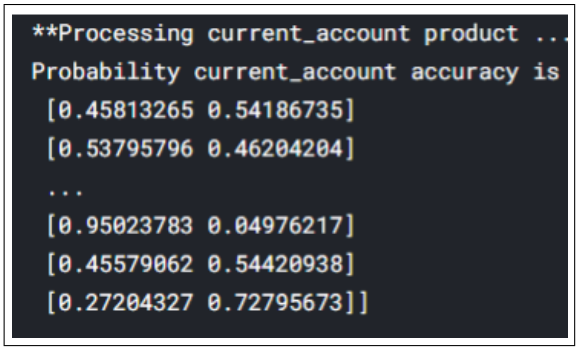

Fig. 6. Before Lag

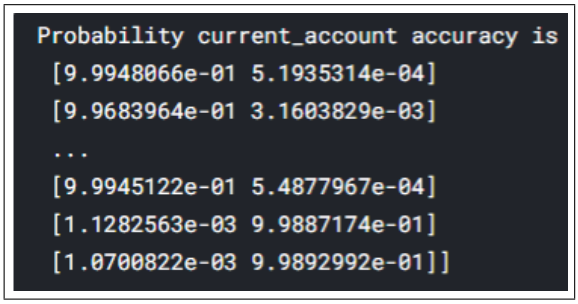

Fig. 7. After Lag

features to distinguish the labels. But when we send memory of how the product labels looked in previous months, which we term as *lags* there is a clear spike in how confident the gradient boosting algorithm becomes in predicting it's results. This is evident from Fig. 7.

The final result needs the most probable five products which are likely to toggle. Thus, we aren't interested in the predictions of the products directly but their probabilities predicted by algorithm which can be used to calculate how certain we are the product will toggle and rank products accordingly. This significance of probabilities is evident and only after adding lags we see that the certainty of the probabilities for model outputs has increased adding great value to the final result.

Till the end, we had tried to increase the number of lags without affecting the memory and the best point we found was between 4 to 6 lags, hence we had tried for 6 lags by the end of submissions.

We could have used a single model under a multi label setting. But we wanted to understand which features effected each of the products most. If we work with one model per one product, this could be done easily analysing each of the feature vector weight of the product wise model. These inferences could be used to build different feature lags for different models as we can't just pass lags of all attributes owing to limited RAM size with such huge data set.

B. Exponential Smoothing

Once we started observing that lags started to show improvement in accuracies, our next thought was to work with exponential smoothing. Exponential Smoothing is divided into 3 parts[6]:

- Checking the linear growth of the product over time
- Checking the trend of the product over time
- Checking the seasonality affects of a product over time

Initially, we made forecasts for products by linearly checking for growth, which used a weighted moving average with exponentially decreasing weights. Assuming the value for a product for a previously known time to be s_{t-1}, the formula to determine a prediction at time t is

$$s_t = \alpha x_t + (1-\alpha)s_{t-1}$$

Where α is the smoothing factor, which implies smaller α is, the more influence the previous observations have and the smoother the series is.

Additionally, using this with lags we had expected a boost in results as these kind of forecast methods usually work for prediction in a smaller scope of time. But on researching we had found a few issues, one of the most important being the above theory would not work properly in the presence of cyclic components and seasonality and methods like this are only accurate when a reasonable amount of continuity. As such, it's best suited for short-term forecasting as it assumes future patterns and trends will look like current patterns and trends. Due to this there wasn't much improvement in accuracy. While we tried to incorporate seasonality and trends the major disadvantage that we faced was time and shortage of memory. Even after several optimisations we could not implement additions to exponential smoothing model which fills in the issues like seasonality and trend and due to this major roadblock, we decided not to go ahead with exponential smoothing.

C. Memory Management and Optimisation

With huge train data set and a cap of 9 hours for a session in Kaggle, we had to optimize both memory and the way we processed records in our model. We tried using *pandas* library functions to the extent possible because predefined methods are generally faster than what we could write from scratch. The concepts of Joins, Merge, Apply, Grouping helped us tremendously in improving performance of our whole pipeline. For selecting the top5 products of each customer as per the probabilities obtained we used the *pd.apply()* method along axis=1 which brought down our processing time from 2.5 hours to few minutes, where initially we explicitly wrote *for loops* to process the top5 records of a row. The idea of *left join* helped us make a customized function where we could add what attributes and how many months lag we wanted to apply. We also got a chance to learn about gc(Garbage Collector) module in python to analyse the unused variable and freeing them. This allowed us to use lags of upto 6 months in our final notebooks.

V. Challenges & Scope

One of the main challenges we had faced during the entire contest was memory management. To tackle it we had used some naive methods, but there could have been

Model	Private Score	Public Score
Voting	0.04615	0.04652
XGBoost	0.04591	0.04628
LGBM	0.04471	0.04497
Catboost	0.04069	0.04111
One vs Rest Classifier	0.04068	0.04117

TABLE I
MODELS AND SCORES

more complex ways to optimise it even further. If we could have managed the memory properly, we could have tried out Exponential Smoothing to a complete extent, which we believe could have could have improved the accuracy of the model. Similarly, we could have tried out other pipelining methods like stacking which from our experience from assignment has shown a significant improvement in accuracy.

Overall we believe that this model with additional tweaks as the ones mentioned above can be used in banking institutions for better customer satisfaction.

CONCLUSION

This is a time series setup and for real time prediction of such data, we need to run the model time and again bringing in the notion of online learning. From our experimentation we have concluded that in order to predict the coming month's toggles accurately, we need to choose the data from the recent past. This fits very well with the notion of online learning as we can keep tuning the model parameters after having new data of current months. This was clearly evident as we saw a spike in accuracies of our model when we changed the sampling of train data from 2015-05-28(Data of the test month from previous year) to 2016-03-28(Data of recent past of test month). In the former case our score was 0.04117 and in the latter case it improved to 0.04628. We understood the importance of sampling because of this.

After 3 months of the competition, we had finally ended up at second position with a MAP@5 score of 0.04615 on private leaderboard.

VI. ACKNOWLEDGMENT

Firstly we would like to thank Prof. G Srinivasaraghavan and Prof. Neelam Sinha for teaching such a wonderful course. Without the concepts taught by them, we believe it would have been much harder to understand what we were doing. We would like to thank our Teaching Assistant Tejas Kotha for sparing time and helping us whenever we asked for. Without his valuable feedback, it would have been hard to implement the concepts. Special Thanks to Saiakash Konidena, Nikitha Adivi, members of Team Breaking Code and Team Modulo 3 with whom we had very fruitful discussions on how to approach this problem, and for sharing tips.

Overall this project has been a great learning experience for the entire team and we hope to implement the concepts we learnt here in further courses and competitions.

REFERENCES

[1] Dataset - Santander Product Recommendation, Kaggle
[2] Statistcs for international trade in Banking Services: Requirements, Availability and Prospects
[3] Daniele Micci-Barreca (July 2001) "A preprocessing scheme for high-cardinality categorical attributes in classification and prediction problems"
[4] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, Andrey Gulin (Jun 2017) "CatBoost: unbiased boosting with categorical features"
[5] Gregory Carey (2003). Coding Categorical Variables
[6] Eva Ostertagova, Oskar Ostertag (Dec. 2012) "Forecasting Using Simple Exponential Smoothing Method"