Santander Project
Team Name: Modulo3

Aditya Sheth
IMT2018003
CSE, IIIT Bangalore

Nipun Goel
, IMT2018052
CSE, IIIT Bangalore

Arpitha Malavalli
IMT2018504
CSE, IIIT Bangalore

***Abstract*—Customer behavior prediction identifies behaviors among groups of customers to predict how similar customers will behave under similar circumstances. This is beneficial as it increases customer satisfaction by giving highly-personalized predictions for every customer.**
Our problem statement required us to use historical data of the Santander Bank to predict which products/ accounts their customers are likely to start/drop at the beginning of a new month. This can be utilized by the bank to lend a hand to their customers through personalized product recommendations.
Our approach involved exploring various ways of correlating and grouping a large amount of time-series data available to come up with an accurate prediction model.

***Index Terms*—Feature Engineering,Logistic Regression, XGBoost, AdaBoost, Random Forest Classifier, LightGBM Classifier, Time-Series Analysis**

I. Introduction

It is always better to be proactive and not reactive. Enterprises have always tried to predict the future of their businesses. It allows them to make intelligent, forward-looking decisions through visualization and also better their user experience.
Enterprises today have access to an infinite amount of historical data, which they can use to make more sound, time-tested decisions.
Customer behavior models typically analyze years of customer data to predict what a group of customers will do at a certain point in time. If the model generated is right and most customers in the group will act as predicted by the model.
In our problem statement, the enterprise in question is a bank - Santander Bank. The objective of the model is to predict the top 5 accounts that are most likely to have some activity (activated/ deactivated), for every customer of the Bank.
This prediction must be made by

1) Identifying segments within customers based on their personal information like income, age, gender, etc
2) Tracking how customers move over time i.e the trends and behaviors within customers

Various methods of Machine Learning can be used to come up with the required customer behavior predictor. Some of them were explored by us, and have been explained in detail in the report. These predictions will ultimately help the bank to develop highly-personalized solutions for every customer.

II. Data Set

The dataset used in the project was made available under the IIITB ML Project: Santander Product Recommendation (InClass competition) on Kaggle. It has 1.5 years' worth of customer behavior data. Starting from 2015-01-28 to 2016-04-28, all the transactions made by customers are recorded by month.
Thus the data tracks the behavior of 951952 customers of the the Santander Bank over 16 months, resulting in a total of 610361088 data points. The training data also consists of 24 product columns of the form 'ind_(xyz)_ult1' which are the different variants of accounts that customers can have in the bank. Eg: Saving Account, Current Accounts, Payroll Account, Short-term deposits, Long-term deposits, Mortgage, Pensions, etc.
For a given customer_id ('ncodpers') for a given month, each product column is marked with 0/1 to signify if the account is closed/open respectively.
Using the other 23 attributes of a customer - like age, seniority, the region of residence, gross income- and their variation over the time-series data, we need to predict the top 5 products that a customer will buy in addition to what they already had on 2016-04-28.

III. Data Preprocessing

Preprocessing turns raw data into cleaner information that is more suitable to work with .

Following were the things we tried to clean our data:

1) *Dropping columns*
   Columns which didn't add any value to our dataset or were redundant were dropped. Such columns were "tipodom"(all value were 1's), "nomprov"(one to one mapping from feature "cod_prov") and "ult_fec_cli_1t"(majority values were NULL).

2) *Conyuemp*
   One of the other feature which had majority of the values as NULL was "conyuemp". This feature had Y/N type values with very few of them being Y. Initially we dropped it but then decided against it as we analysed that 'Y' in "conyuemp" adds special value

to a customer. So we simply filled NULL values with 'N' and kept the feature.

3) *Age*
"Age" had bimodal distribution and so outliers were filled with the mean of the nearest peak and other NULL values were filled with the mean of entire range. Following this we normalised this feature.

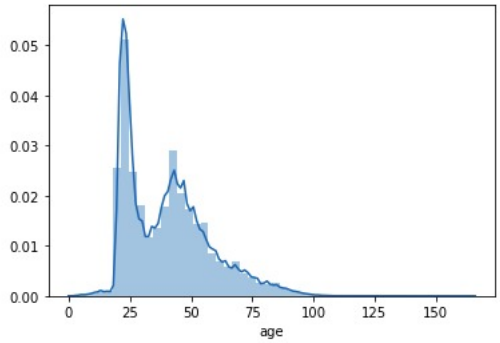
4) *Fecha_alta*
"Fecha_alta" is the date in 'YYYY-MM-DD' format on which the customer joined. Initially we extracted month part of this date and used as our feature in order to take account of seasonality in which the customer joined. Later we extracted year part of this date which improved the accuracy and so we decided to keep year as our feature.

5) *Renta*
"Renta" which is the gross income of the household seemed very important feature for our problem of predicting banking products. So to fill NULL values we grouped our data as per cod_prov and filled NULL values with the median as per cod_prov. After filling NULL values, the feature was normalised.

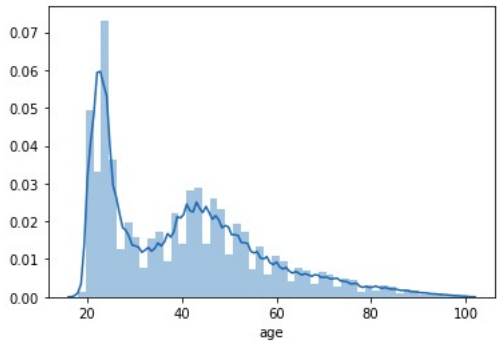
6) *Other features*
Rest of the features had very less number of NULL values and so we filled it with "UNKNOWN" and -1.0 for categorical and numerical features accordingly. For categorical feature "pais_residencia", we made a dictionary as per distance from the country of Santander bank which is 'ES'. Higher the distance of a country from Spain, higher the number assigned to it in label encoding. For all the other remaining categorical features, initially we tried one hot encoding. Then later we switched to label encoding which gave better result than one hot encoding.

IV. Data Visualisation

1) Age



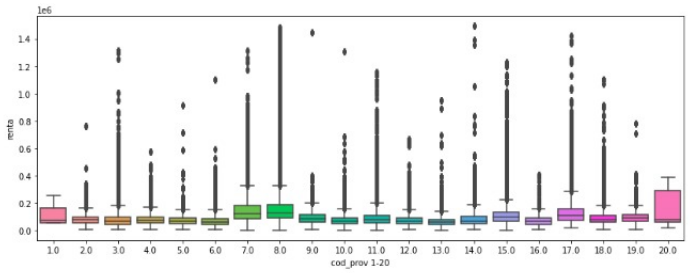
Bi-modal Distribution observed in age (with 2 peaks)

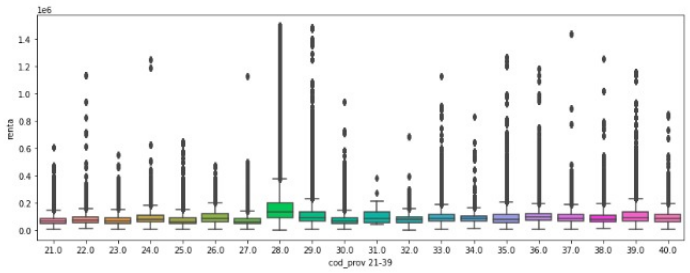


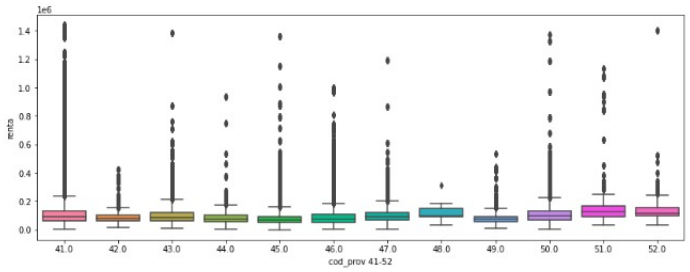
After the outliers are moved to the mean of the closest one.

2) Renta grouped by cod_prov

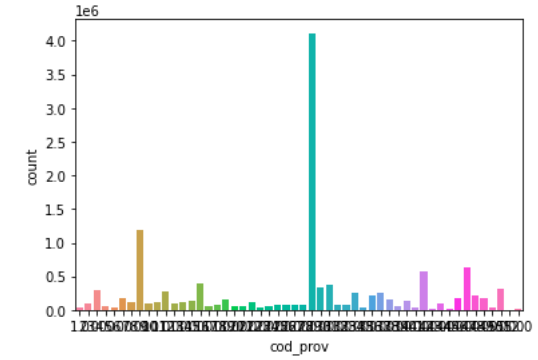
Since there is a different distribution of renta for every codprov, it is a good measure to group by and fill the null values in renta.

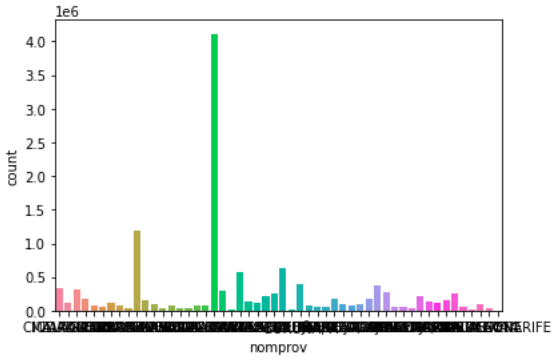




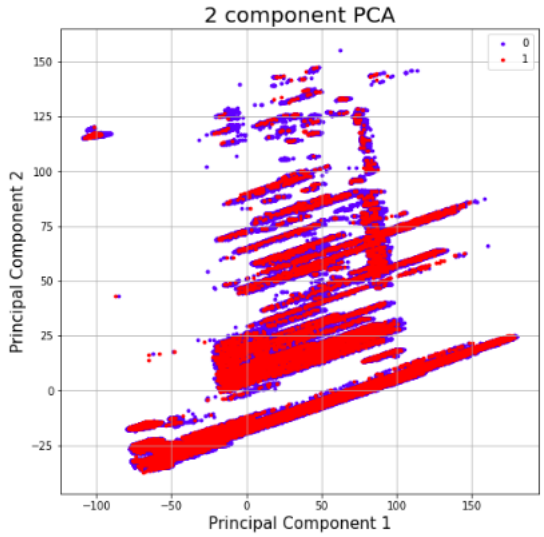


3) Cod_prov and nomprov
As can be seen from the below figures and also from the other numerical analysis done, it was interpreted that cod_prov and nomprov have one to one mapping. Thus it made sense to drop one of them. We dropped nomprov as it was categorical.





4) PCA analysis
From the PCA graph, we can see that the given features are not enough to bring a clear distribution between the two classes. Thus lag and toggle features were required to be added.



V. Feature Extraction

In time-series data like this, it is very important to extract-time based features from the raw data in order to make the training better.

We tried extracting some features which considers the time series aspect of our data and they are as follows:

1) We added lag features as it appeared to be of great value which was also evident from the visualisations as shown above. We added 3 lag features that is considered the product columns of the previous 3 months as features for training. This increased our score significantly.

2) Then we added one additional feature namely "product_name+sum" for each product that is in total 24 additional features. This feature contains the frequency of toggling of the product in consecutive months considering previous 8 months. So this feature gave information about how likely a particular customer is likely to toggle a particular product. Adding this feature increased our score slightly.

3) Then we tried to made the above feature more detailed by specifying the type of toggle that is 0to0, 0to1, 1to0 or 0to1. So instead of "product_name+sum", we added "product_namesum00", "product_namesum01", "product_namesum10" and "product_namesum11" as our features. These features in some sense define a customer with respect to a product taking into consideration the history of the customer with the product. Adding these features increased our score significantly.

4) In the end, we also added the product of $5th$ month of previous year(2015) as our feature in data in order to take account of seasonality. Adding this increased our score but by very low margin.

VI. Model Selection

The scoring method was based on Mean Average Precision @ 5 (MAP@5) metrics.
We tried several different models with basic hyperparameter tuning to get the following MAP@5 scores:

TABLE I
Method vs MAP@5 score

Model	MAP@5 score
Logistic Regression	0.02769
RandomForestClassifier	0.04320
AdaboostClassifier	0.04497
XGBClassifier	0.04562
CatBoostClassifier	0.04575
LightGBM Classifier	0.04597

It's clear from the above table that the Tree based learning

algorithms are performed much better compared to Regression based models(Logistic Regression) and Random Forest classifier.
We decided to move forward with LightGBM as it gave one of the best scores and was also the fastest to train among all Tree - based learning algorithms(Approximately 7 times faster than XGBoost).

VII. Results and Analysis

Here we describe the ideas we used to increase score.

1) First idea was to use basic data cleaning without any feature extraction for predicting the toggle probability of each customer(for every product).
   Then we sorted the probability of toggling in decreasing form and took the top 5 products that change for every customer.
   Score: 0.026
2) Next idea was to add previous month's product value for a customer as a lag feature for predicting next month's outcome.
   This idea gave a significant improvement on the previous idea and our score improved by around 34%.
   Score: 0.035
3) The previous idea was then followed by adding last 3 month's product values as lag features for predicting next month's outcome. Score: 0.039
4) Toggle feature addition over 4 months data.
   there are 4 subfeatures: toggle00, toggle01, toggle 10, toggle11.
   For example: toggle01 denotes that how many times a customer changed their product value from 0 to 1 in any consecutive months.
   Toggle values over last 7 months were added in this step.
   This gave a slight improvement of score.
   Score: 0.042
5) Toggle features addition over all the months.
   Score: 0.045

TABLE II
METHOD VS MAP@5 SCORE

Method	**MAP@5 score**
Basic Data Cleaning	0.026
1 month Lag added	0.035
3 month Lag added	0.039
Toggle features for past 7 months	0.042
Toggle features for all months added	0.045

Following the above written steps, we increased our score starting from 0.026 through 0.045 , hich is a significant improvement.

Note: All the given scores are based on training on LightGbm with the same hyper-parameter tuning.
We performed different variations of hyper-parameters on the last day to raise our score slightly from 0.045 to 0.046.

VIII. Issues Faced

1) In the initial few submissions, we were submitting only those products whose probability of toggling was greater than 0.5 . Thus in many cases, the number of products predicted was less than 5. We misunderstood the scoring metric MAP@5. We thought it is a penalty based metrics, where actually it is a reward based metrics. So it is always more optimal to predict exactly 5 products compared to predicting less than 5 products.
   Due to this silly error, our initial scores did not cross 0.018. After correcting this error, our score came out around 0.026.
2) While adding lag features, we didn't know of left-join technique using merge() function. We followed simple brute force technique in which the time taken to add lag features turned out to be around 2 hours. After changing brute force idea to using joins, the time reduced to couple of seconds. On a little bit of research, we found that join uses bit manipulation techniques along with parallel processing which reduces the runtime significantly.
3) RAM Limit: In our final model, we wanted to use all the month's data for adding toggle value but it was exceeding RAM limit as the total data was inflated during feature extraction process. To solve this issue, we performed output of the first 7 month data in one .csv file and another 7 month in a different .csv file. In the model.py file, we performed merge of the two different datasets and extracted the toggle values for all months combined.
   We believe that this issue would've been solved inside the notebook with careful optimizations of Memory usage. But we decided to take the other approach since now our notebook can start training the model without taking any time on cleaning and feature extraction part.

IX. Acknowledgement

The authors would like to thank professors for their continued guidance, support and encouragement. Further the authors would like to thank the teaching assistants for their valuable advice and encouragement.

Special Thanks to Tejas Bhaiya for constantly motivating and guiding us throughout the project.

Also we would like to mention teams Breaking Bad, TameThatBench, and Survey Corps. The members of these teams provided us with great help. Without their help we wouldn't know how to reduce the running time of feature addition from hours to seconds.
Also we would like to thank all the other teams in addition to above mentioned team, for providing great competition on the leaderboard. The healthy competition encouraged us to learn more about different models and techniques so as to get higher score.

X. References

1) “Scikit-learn documentation.” [Online].
   Available here
2) “Pandas documentation.” [Online].
   Available here
3) ”LightGBM Documentation.” [Online].
   Available here
4) ”XGBoost Documentation” [Online].
   Available here
5) ”Adaboost classifier” [Online].
   Available here
6) ”Notebook of SRK(When Less is More)” [Online].
   Available here
7) ”2nd place solution in discussion” [Online].
   Available here
8) ”3rd place solution in discussion” [Online].
   Available here
9) ”Detailed Cleaning/Visualisation notebook” [Online].
   Available here