SANTANDER PRODUCT RECOMMENDATION

INFO7390 – Advances in Data Sciences and Architecture
Professor: Nik Bear Brown

Tianyu Wang

001237975 | wang.tianyu1@husky.neu.edu

Abstract

Santander Bank is one of the country's top retail banks by deposits and a wholly owned subsidiary of one of the most respected banks in the world: Banco Santander. They focus on helping people make their banking hassle-free by providing simple ways to spend, save and manage money.

Under their current recommendation system, a small number of Santander's customers receive many recommendations while many others rarely see any resulting in an uneven customer experience. With a more effective recommendation system in place, Santander hopes to better meet the individual needs of all customers and ensure their satisfaction no matter where they are in life.

In this research project, I try to take an approximately 13.6 million row dataset containing information about customers of Santander Banks between January 2015 and May 2016 and to predict which products they would purchase in June 2016, which was withheld from us. The primary algorithm used is XGBoost which is based on GDBT (Gradient Boosting Decision Tree).

Keywords: Santander, recommendation system, XGBoost, GDBT

1. Overview

The new recommendation system will recommend the top 7 products among 24 products to each customer. The scoring was evaluated using mean average precision at 7 (MAP@7). The intuition behind this scoring metric is that it rewards solutions where the person actually added one of the items we recommended, and we get more points if the purchased item was earlier in the list of recommendations. We don't lose any points for recommending products to people that don't buy anything. Therefore, we should recommend exactly 7 products to each customer, and place the most likely ones earlier in the list. A list of recommended products can be obtained by sorting these from most-likely to least-likely.

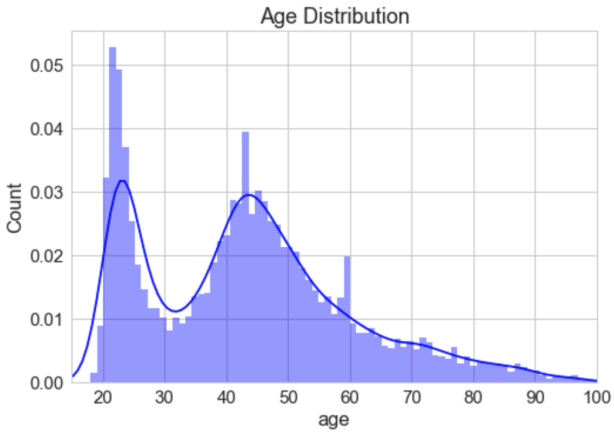
2. Data Source

I will use the dataset provided by Santander which contains 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 has monthly records of products a customer has, such as "credit card", "savings account", etc. I will predict what additional products a customer will get in the last month, 2016-06-28, in addition to what they already have at 2016-05-28.

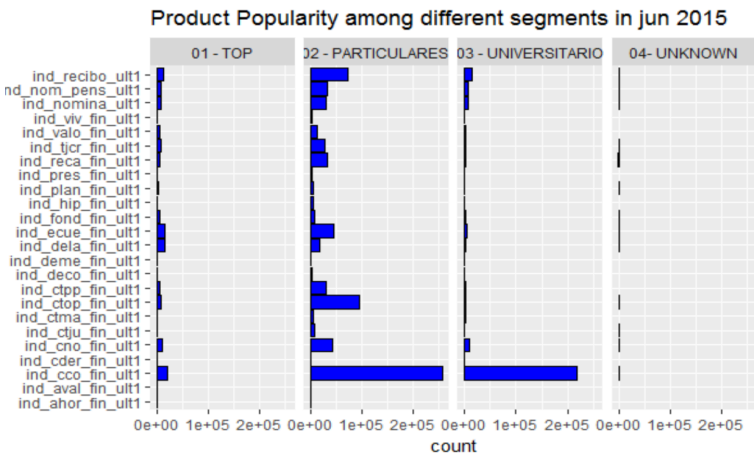
3. Exploratory Data Analysis

Exploratory data analysis is done on the dataset for data cleaning, adjusting some features, and visualize various important features.

In the age field, I find that in addition to NA, there are people with very small and very high ages. It's also interesting that the distribution is bimodal. There are a large number of university aged students, and then another peak around middle-age.



I also found that similar group of people tend to buy similar products under similar conditions. Analyzing the product popularity among different segments of people in June 2015 can be very useful for this problem. This feature is the base of the further prediction.



The ind_actividad_cliente field tells whether a customer is active or not. This field also contained NA values and those records were converted to inactive status. Count of inactive customers is almost constant throughout the year.

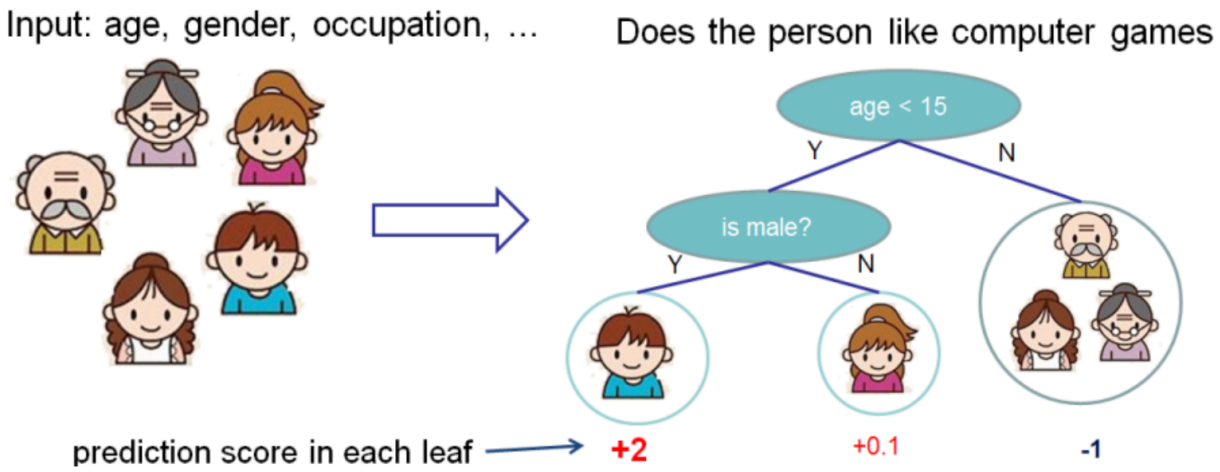
And the renta (Gross income of the household) is missing a lot of values. Rather than just filling them in with a median, it's better to fill the missing values by the median of renta for a particular province.

4. Solution Summary

The recommendation system in this research project is based on gradient boosted classification trees. There are a number of machine learning software packages that implement this concept, but the most popular by far in the data science community, and the one we used here, is XGBoost (eXtreme Gradient Boosting).

Gradient boosting is a machine learning strategy where you build many simple models. The simple model trees are very shallow, so we call them weak learners. This term means that they are quite inaccurate, but slightly better than random guessing. And the concept of boosting means that we take many weak learners and combine them into a stronger one. Using the mentioned scoring system instead of decisions also makes this process easy and straightforward to implement.

The following picture (from XGBoost Documentation) shows that how the scoring system works:



One another huge advantage of boosted tree over neutral network is that we actually see why and how the computer arrives to a decision. This is a remarkably simple method that leads to results of very respectable accuracy.

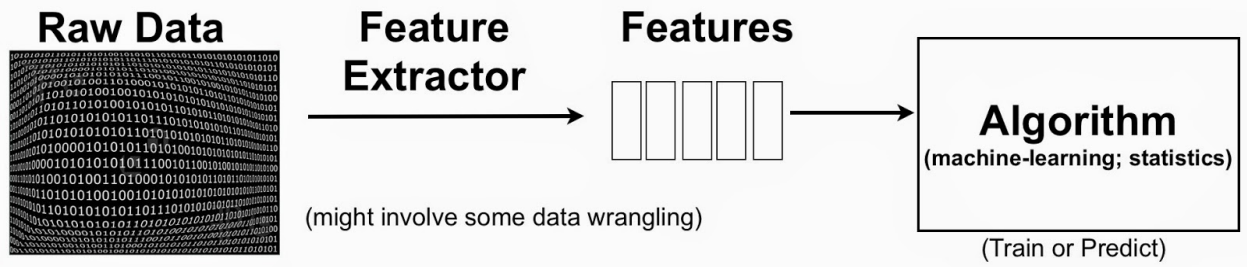
XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solve many data science problems in a fast and accurate way.

The core elements of my approach are the base models. These are all trained on a single month of data for all 24 products. To drop some columns with too low frequency is necessary. It is helpful for saving time and increasing efficiency. The base models are trained using all available historical information. This can only be achieved by calculating separate feature files for all months between February 2015 and May 2016.

Several feature preparation steps are required before the feature files can be generated. Restricting the base models to use only the top features for each lag-product pair speeds up the modeling and evaluation process. The ranked list of features is obtained by combining the feature gain ranks of the 5-fold cross validation on the base models trained using all features.

5. Feature Engineering

In real-world data analysis, it is possible that most of data is not in numeric type. So one of the important steps in using machine learning in practice is feature engineering: that is, taking whatever information you have about your problem and turning it into numbers that you can use to build your feature matrix.



In this project, I have defined three types of variables: numeric columns, categorical columns, target columns. The following codes show the detail

```
numericCols = ["age","antiguedad","renta"]

categoricalColumns = ["ind_empleado","pais_residencia","sexo","ind_nuevo","indrel",
                      "indrel_1mes","tiprel_1mes", "indresi", "indext", "canal_entrada",
                      "nomprov", "ind_actividad_cliente","segmento"]

targetsColumns = ["ind_ahor_fin_ult1", "ind_aval_fin_ult1","ind_cco_fin_ult1",
                  "ind_cder_fin_ult1", "ind_cno_fin_ult1", "ind_ctma_fin_ult1",
                  "ind_ctop_fin_ult1", "ind_ctpp_fin_ult1", "ind_deco_fin_ult1",
                  "ind_deme_fin_ult1", "ind_dela_fin_ult1", "ind_ecue_fin_ult1",
                  "ind_fond_fin_ult1", "ind_ctju_fin_ult1", "ind_plan_fin_ult1",
                  "ind_pres_fin_ult1", "ind_reca_fin_ult1", "ind_tjcr_fin_ult1",
                  "ind_valo_fin_ult1", "ind_viv_fin_ult1", "ind_nomina_ult1",
                  "ind_nom_pens_ult1","ind_recibo_ult1"]
```

Other features were added to incorporate dynamic information in the lag period of the 24 predictors. Many of these predictors are however static and added limited value to the overall performance. It would be great to study the impact of changing income on the product purchasing behavior but that was not possible given the static income values in the given data set.

6. Cold Walkthrough

Firstly, I cleaned the data including dropping outliers and filling null values. The following steps are based on the scrubbed dataset.

The following is the main script that does feature engineering and produces the data ready (mostly) to be fed into one of the models:

```
cols_to_use = ['pais_residencia', 'sexo', 'age', 'antiguedad', 'canal_entrada',
'cod_prov', 'renta', 'segmento']
trn = trn[cols_to_use]
tst = tst[cols_to_use]

# factorize = LabelEncode categorical features
categoricals = ['pais_residencia', 'sexo', 'canal_entrada', 'segmento']
for col in categoricals:
    temp, _ = pd.concat([trn[col], tst[col]], axis=0).factorize()
    trn[col] = temp[:trn.shape[0]]
    tst[col] = temp[trn.shape[0]:]
```

Then I did the cross validation:

```
# XGB Model Param
num_round = 500
early_stop = 10
xgb_params = {
    'booster': 'gbtree',

    # model complexity
    'max_depth': 2,  # higher, more complex

    # basic
    'nthread': 4,
    'num_class': 15,
    'objective': 'multi:softprob',
    'silent': 1,
    'eval_metric': 'mlogloss',
    'seed': 777,
}

trn_scores = []
vld_scores = []
best_iters = []
n_splits = 5
sss = StratifiedShuffleSplit(n_splits=n_splits, test_size=0.1, random_state=777)
for i, (t_ind, v_ind) in enumerate(sss.split(trn, y)):
    print('# Iter {} / {}'.format(i+1, n_splits))
    x_trn = np.asarray(trn)[t_ind]
    x_vld = np.asarray(trn)[v_ind]
    y_trn = np.asarray(y)[t_ind]
    y_vld = np.asarray(y)[v_ind]

    dtrn = xgb.DMatrix(x_trn, label=y_trn)
    dvld = xgb.DMatrix(x_vld, label=y_vld)
    watch_list = [(dtrn, 'train'), (dvld, 'eval')]
```

The following script is to refit and predict on the test data using methods in XGBoost:

```
dtrn = xgb.DMatrix(trn, label=y)
num_round = int(np.mean(best_iters) / 0.9)
bst = xgb.train(xgb_params, dtrn, num_round, verbose_eval=False)

dtst = xgb.DMatrix(tst)
preds = bst.predict(dtst)
preds = np.fliplr(np.argsort(preds, axis=1))
```

Finally, the script will submit a submission which is a csv file contains Santander product recommendations:

```
submit_cols = [target_cols[i] for i, col in enumerate(target_cols) if i in rem_targets]

final_preds = []
for pred in preds:
    top_products = []
    for i, product in enumerate(pred):
        top_products.append(submit_cols[product])
        if i == 6:
            break
    final_preds.append(' '.join(top_products))

t_index = pd.read_csv(TST, usecols=['ncodpers'])
test_id = t_index['ncodpers']
out_df = pd.DataFrame({'ncodpers': test_id, 'added_products': final_preds})
```

Every customer in the test dataset will get 7 product recommendations. The rank of 7 different products indicate the recommendation level. The result is like this:

	A	B
1	added_products	ncodpers
2	ind_recibo_ult1 ind_tjcr_fin_ult1 ind_nom_pens_ult1 ind_nomina_ult1 ind_reca_fin_ult1 ind_cco_fin_ult1 ind_fond_fin_ult1	15889
3	ind_recibo_ult1 ind_nomina_ult1 ind_nom_pens_ult1 ind_reca_fin_ult1 ind_cco_fin_ult1 ind_tjcr_fin_ult1 ind_cno_fin_ult1	1170544
4	ind_recibo_ult1 ind_nom_pens_ult1 ind_nomina_ult1 ind_cco_fin_ult1 ind_cno_fin_ult1 ind_tjcr_fin_ult1 ind_ecue_fin_ult1	1170545
5	ind_nom_pens_ult1 ind_nomina_ult1 ind_recibo_ult1 ind_cco_fin_ult1 ind_cno_fin_ult1 ind_tjcr_fin_ult1 ind_reca_fin_ult1	1170547
6	ind_nom_pens_ult1 ind_nomina_ult1 ind_recibo_ult1 ind_cco_fin_ult1 ind_cno_fin_ult1 ind_tjcr_fin_ult1 ind_ecue_fin_ult1	1170548
7	ind_recibo_ult1 ind_nom_pens_ult1 ind_nomina_ult1 ind_cco_fin_ult1 ind_cno_fin_ult1 ind_tjcr_fin_ult1 ind_ecue_fin_ult1	1170550
8	ind_recibo_ult1 ind_nom_pens_ult1 ind_nomina_ult1 ind_tjcr_fin_ult1 ind_reca_fin_ult1 ind_cco_fin_ult1 ind_cno_fin_ult1	1170552
9	ind_recibo_ult1 ind_nom_pens_ult1 ind_nomina_ult1 ind_cco_fin_ult1 ind_cno_fin_ult1 ind_tjcr_fin_ult1 ind_ecue_fin_ult1	1170553
10	ind_recibo_ult1 ind_nom_pens_ult1 ind_nomina_ult1 ind_cco_fin_ult1 ind_cno_fin_ult1 ind_tjcr_fin_ult1 ind_ecue_fin_ult1	1170555
11	ind_recibo_ult1 ind_nom_pens_ult1 ind_nomina_ult1 ind_cco_fin_ult1 ind_cno_fin_ult1 ind_tjcr_fin_ult1 ind_ecue_fin_ult1	1170557
12	ind_recibo_ult1 ind_nom_pens_ult1 ind_nomina_ult1 ind_cco_fin_ult1 ind_cno_fin_ult1 ind_tjcr_fin_ult1 ind_reca_fin_ult1	1170559
13	ind_recibo_ult1 ind_nom_pens_ult1 ind_nomina_ult1 ind_cco_fin_ult1 ind_cno_fin_ult1 ind_tjcr_fin_ult1 ind_ecue_fin_ult1	1170563
14	ind_nom_pens_ult1 ind_nomina_ult1 ind_recibo_ult1 ind_cco_fin_ult1 ind_cno_fin_ult1 ind_tjcr_fin_ult1 ind_ecue_fin_ult1	1170542
15	ind_recibo_ult1 ind_nom_pens_ult1 ind_nomina_ult1 ind_cco_fin_ult1 ind_cno_fin_ult1 ind_tjcr_fin_ult1 ind_ecue_fin_ult1	1170565
16	ind_recibo_ult1 ind_nom_pens_ult1 ind_nomina_ult1 ind_cco_fin_ult1 ind_cno_fin_ult1 ind_tjcr_fin_ult1 ind_ecue_fin_ult1	1170568
17	ind_nom_pens_ult1 ind_recibo_ult1 ind_nomina_ult1 ind_cco_fin_ult1 ind_cno_fin_ult1 ind_tjcr_fin_ult1 ind_dela_fin_ult1	1170570
18	ind_recibo_ult1 ind_nom_pens_ult1 ind_nomina_ult1 ind_cco_fin_ult1 ind_cno_fin_ult1 ind_tjcr_fin_ult1 ind_ecue_fin_ult1	1170576
19	ind_recibo_ult1 ind_nom_pens_ult1 ind_nomina_ult1 ind_tjcr_fin_ult1 ind_reca_fin_ult1 ind_cco_fin_ult1 ind_cno_fin_ult1	1170578
20	ind_recibo_ult1 ind_nom_pens_ult1 ind_nomina_ult1 ind_reca_fin_ult1 ind_tjcr_fin_ult1 ind_cco_fin_ult1 ind_cno_fin_ult1	1170579
21	ind_recibo_ult1 ind_nom_pens_ult1 ind_nomina_ult1 ind_cco_fin_ult1 ind_cno_fin_ult1 ind_tjcr_fin_ult1 ind_ecue_fin_ult1	1170581
22	ind_recibo_ult1 ind_nom_pens_ult1 ind_nomina_ult1 ind_cco_fin_ult1 ind_cno_fin_ult1 ind_tjcr_fin_ult1 ind_ecue_fin_ult1	1170583
23	ind_recibo_ult1 ind_nom_pens_ult1 ind_nomina_ult1 ind_cco_fin_ult1 ind_cno_fin_ult1 ind_tjcr_fin_ult1 ind_ecue_fin_ult1	1170585
24	ind_recibo_ult1 ind_nomina_ult1 ind_nom_pens_ult1 ind_reca_fin_ult1 ind_tjcr_fin_ult1 ind_cco_fin_ult1 ind_cno_fin_ult1	1170587
25	ind_recibo_ult1 ind_nom_pens_ult1 ind_nomina_ult1 ind_tjcr_fin_ult1 ind_cco_fin_ult1 ind_reca_fin_ult1 ind_cno_fin_ult1	1170588

Conclusion

This project about machine learning was a lot of fun, and also serves as an example of how different machine learning solutions can be from real-world implementations. Decision tree solution is really suitable for building a recommendation system. It can train this system based on all features, producing more and more accurate points of each branch. Based this iteration process, we can train our train dataset and give a prediction for target data.

Apart from this, we must consider the complexity of the model with the performance. A single XGBoost model trained on 2.29G data with predictor features was enough to get a nice recommendation precise. Conversely, if we set up several models and train all features, the produced result will be only slightly better. And it'll cost much more time to build and run. So that the solution should be decided with consideration of all aspects.

References:

[1] Benoit Descamps, Regression prediction intervals with XGBOOST, https://www.bigdatarepublic.nl/regression-prediction-intervals-with-xgboost/

[2] Burak Himmetoglu, Stacking Models for Improved Predictions, http://www.kdnuggets.com/2017/02/stacking-models-imropved-predictions.html

[3] Rory Mitchell, Gradient Boosting, Decision Trees and XGBoost with CUDA, https://news.developer.nvidia.com/gradient-boosting-decision-trees-and-xgboost-with-cuda/

[4] Github, initial pipeline script, https://github.com/kweonwooj/kaggle_santander_product_recommendation/blob/master/bare_minimum/code/main.py

[5] XGBoost Documentation, https://xgboost.readthedocs.io/en/latest/