Capstone Project: Retail Banking Recommender System Using Collaborative Filtering
Bolor Boldbaatar

Introduction

Although we see recommendation engine in every e-commerce website, it is not common for retail banking. Recommendation engine is similar to what industry leaders such as Amazon, Netflix and eBay are using to recommend next best offers based on purchase history and the purchase histories of similar customers.

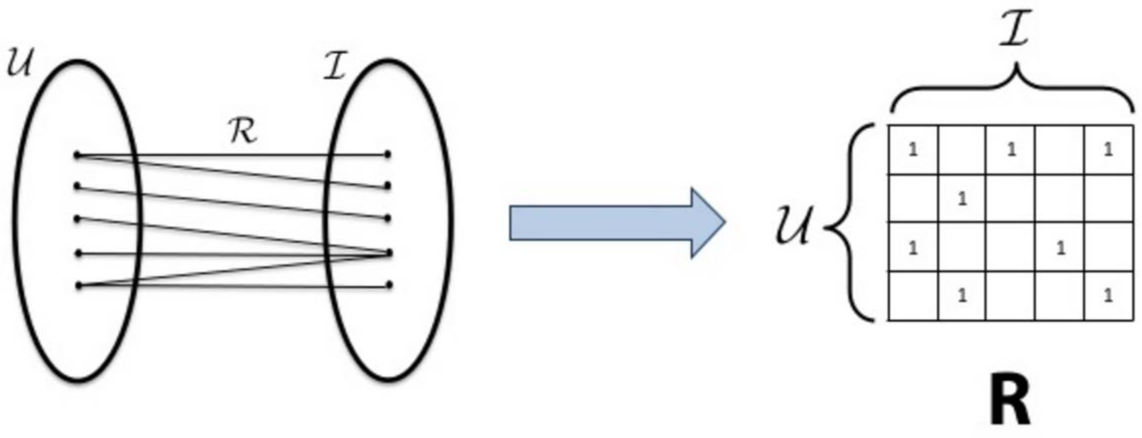
In the past, it was expensive to store data for all bank customers and customer interactions. And traditional product cross-sell and up-sell efforts are based on novelty items or popularity. Increasing use of alternative banking products and new generations of customers force retail banking industry to adapt to machine learning techniques to gain insights from historical customer-and-product interactions and recommend products tailored to customer needs (Sachdeva, 2016).

There are two main data-mining recommender system available today: Content-based and collaborative filtering. This project utilized the later recommender system due to the scalability of the problem that tries to solve. Collaborative filtering approach builds a model based on user past behavior and from similar decision made by other users. It acts as a relationship between users and products. Similarity of products is determined by the similarity of the ratings of those products by the users who have rated both products. The advantage of collaborative recommendation system is that it does not rely on machine based content and can be used for more complex data.

Usually user database consists of information on users’ behaviors, activities or preferences and collaborative filtering recommender system predicts what users will like based on their similarity to other users. The most recommendation systems assume that there are user ratings available in the dataset. In this case, a user first rate the item, or rank a collection of items. However, in real world scenario, that’s not always the case. In fact, most cases businesses face with binary, or positive-only data when they handle recommendation problems. For example, different items bought in Amazon, user likes in Facebook page, weblinks clicked on Google and etc. In such cases, user ratings do not consist of typical 1-5 Likert scale ratings, but with 0-1 data that represents user preferences, like observing the items that a user views, keeping a record of the items that a user purchases. This kind of positive-only data ratings recommender system is called a binary recommender system.

The below figure represents the how user and item ratings in binary recommender system (Figure 1).

Figure 1. Binary Recommendation System



Source: A tutorial at ECML PKDD (2015). Collaborative filtering with binary, positive-only data

Project client:

The client company, Santander bank, has roots from Spain and is a retail banking that is located mainly in European, North and South American countries. They created competition in Kaggle to solve their problem.

Project Problem Statement

- Currently, Santander bank system doesn't recognize the right products to their customers, so that some customers get too many offers while others don't have any.
- Their wish for the solution is to have a recommendation system to predict which products their existing customers will use in the next month based on their past behavior and that of similar customers.

Purpose of the Project

Therefore, the objective of this project is to find a recommendation system for predicting next bank products for each existing user. I will predict what additional products a customer will get in the last month, 2016-06-28.

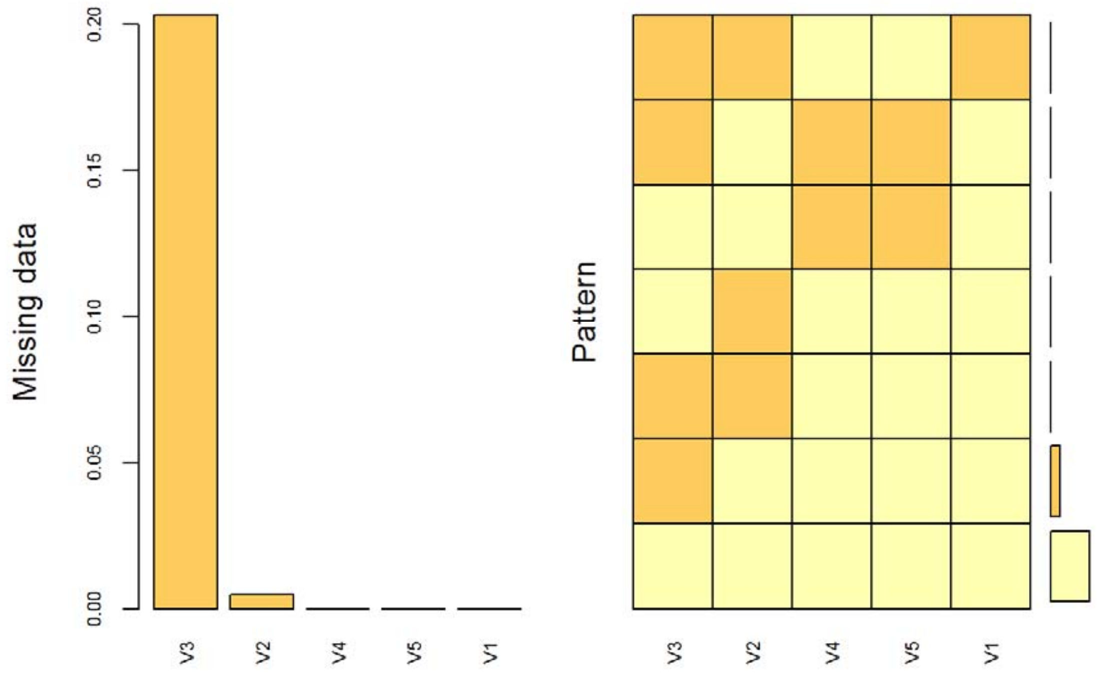
Data Exploration

Data includes 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., in addition to what they already have at 2016-05-28.

There are more than 13 million user data and 48 different variables (see Appendix 1 for name descriptions). Data was not only huge, but also all variables names were in Spanish. So the first thing I did was change the names to English interpretable names that I can understand. Next thing I did was try to understand the customer base. However, reading 2.2GB data on my laptop takes lots of memory, so many times I had to find different approach to explore the data than the usual way. I also have to admit I did very simple analysis than I initially planned to because I spent most of time to running the database that huge and waiting to see the results.

The initial transformation I did was simply delete the missing data that is not very useful for analysis. There were 27,734 users that do not have enough information across the dataset, and all is deleted. In the rest of data, I analyzed the missing data and found out that 80% of the data has no missing values. The most missing values was represented by customer income, which represents almost 20% of dataset (Figure 2). Then income variable was imputed by its mean value, so that I can do further analysis.

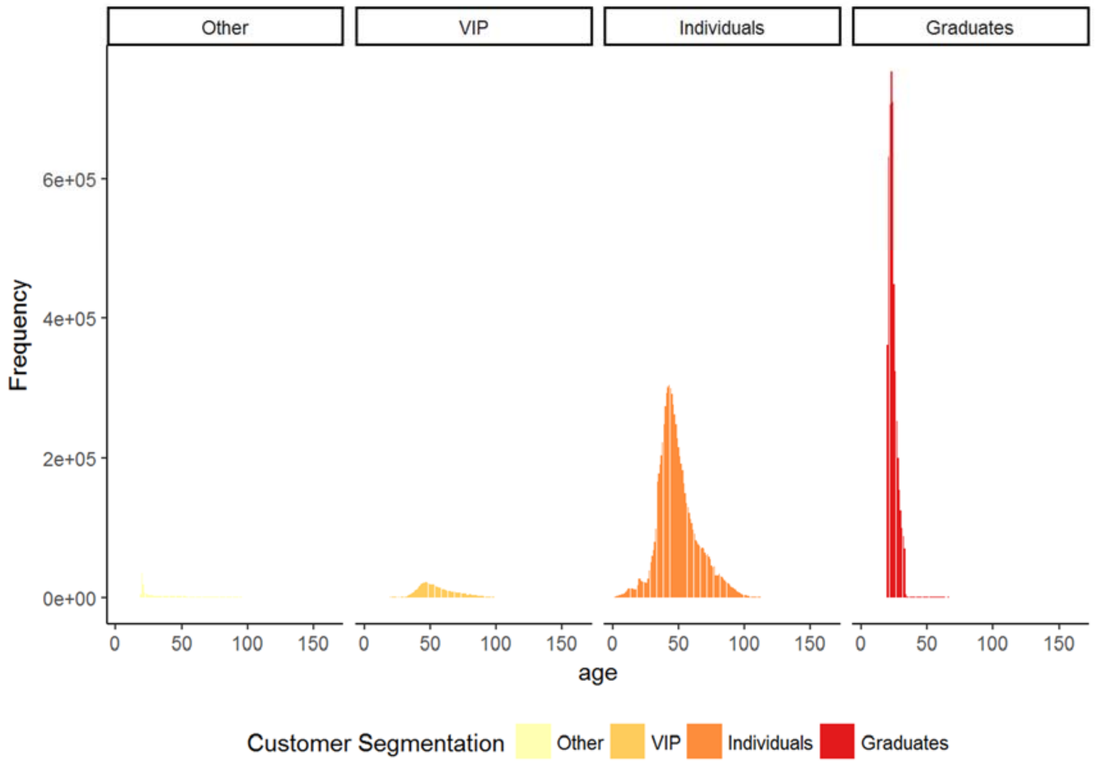
Figure 2. Missing data analysis



1. Exploring Customer Age

First thing, I did was to see the histogram of customer age and found out that there are two big spikes. So, I need to further analyze what's going on, I created another visualization that separated age histogram by customer segmentation, Graduates, Individuals, and VIP. There is very high spike on the "Graduates" segmentation that the majority are in their 20's (Figure 3).

Figure 3. Age Distribution for Each Customer Segmentation



The history of Santander bank also proves that since 2010, the bank has targeted university students and offer them current accounts and credit cards.

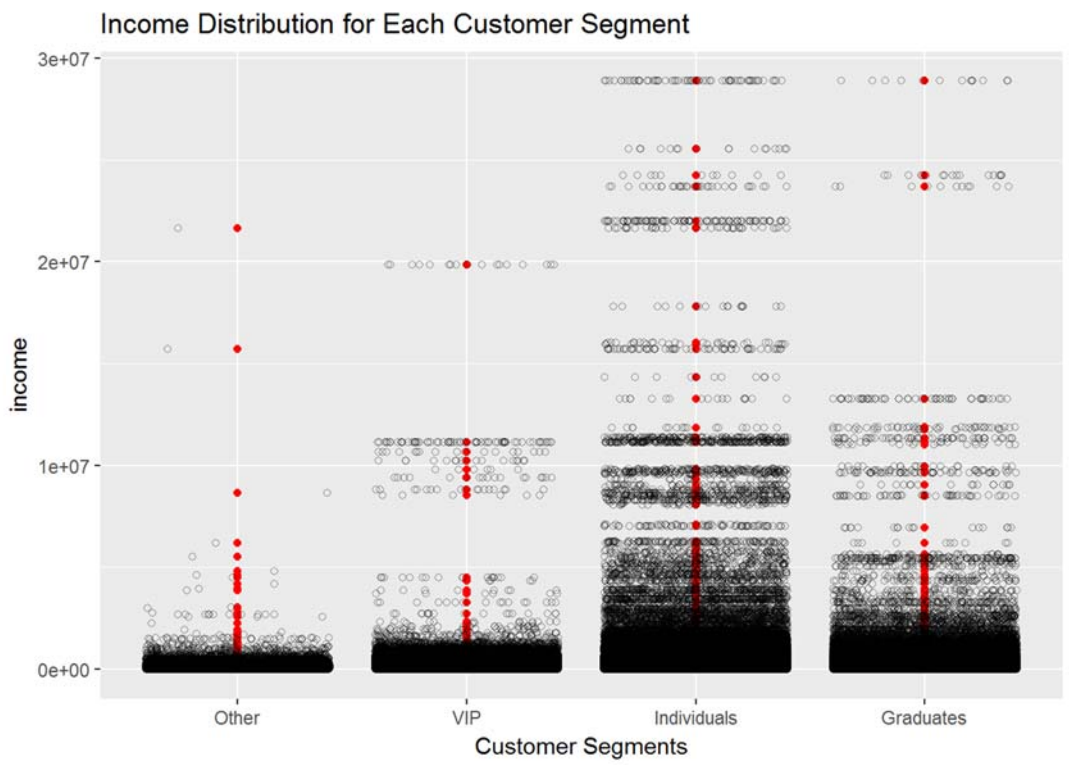
2. Exploring Customer Location

Next thing I explored was customer location. The majority of the customers are from Spain. In fact, 13,553,710 users or 99.31% of users are from Spain alone. So, their customers are not so much from other countries. This tells me that Santander customers are still very homogeneous to their origin. There is no need to explore the different countries customers this time.

3. Exploring Customer Income

Exploring customer income gave some interesting insights about their customer base. While looking at the income distribution among different customer segmentation, I discovered that VIP customers doesn't earn as much as Individuals and Graduates (Figure 4). Later, I also found out that their VIP customer base has been constant for over the years; while, Individuals and Graduates customer base have been grown. It is probably due to their product offers that are tailored to these customers.

Figure 4. Income Distribution for Each Customer Segment



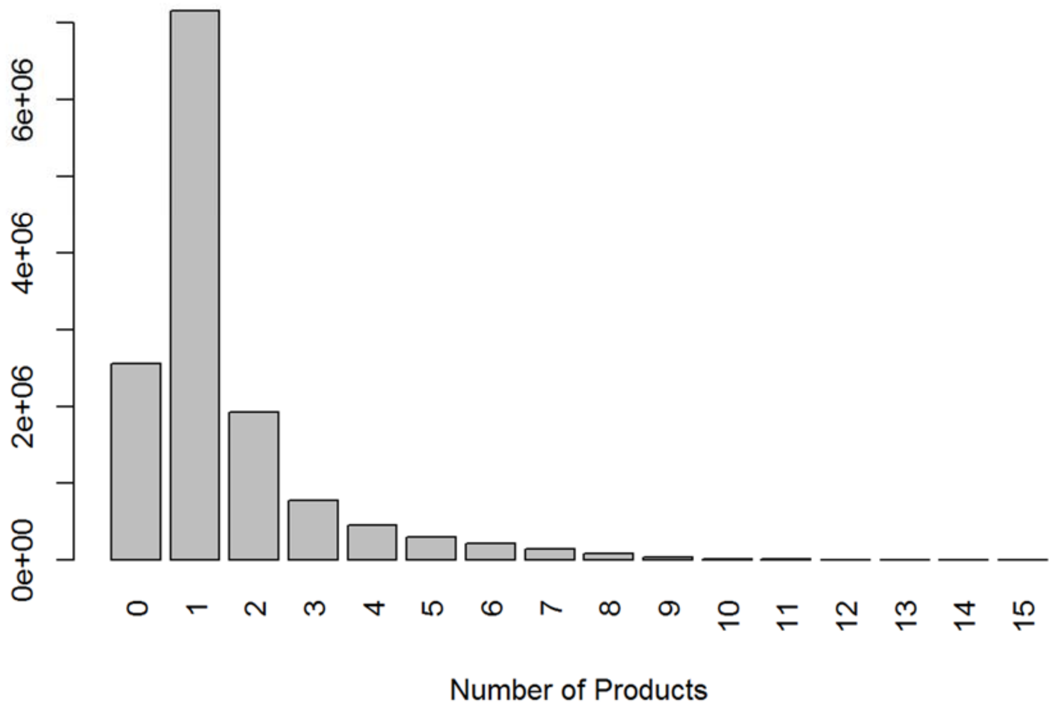
4. Exploring Bank Products

Even though I don't have detailed bank product information, exploring them gave me some idea of how products have been consumed. First of all, there is some confusion with the names. For example, both pension and payroll accounts have been entered twice in the database, but English descriptions are exactly same. Due to differentiate from one from other, I gave 2 different names. For pension product, I gave pension and pension_last. For payroll product, payroll and payroll_acc. I didn't further explore what these accounts represents due to time limitation.

Then I looked at how many total products each customer own and found that more than 2 million customers do not even have a single product. I don't know if it is family members of the users or due to promotional activity, but I deleted them all for the further analysis.

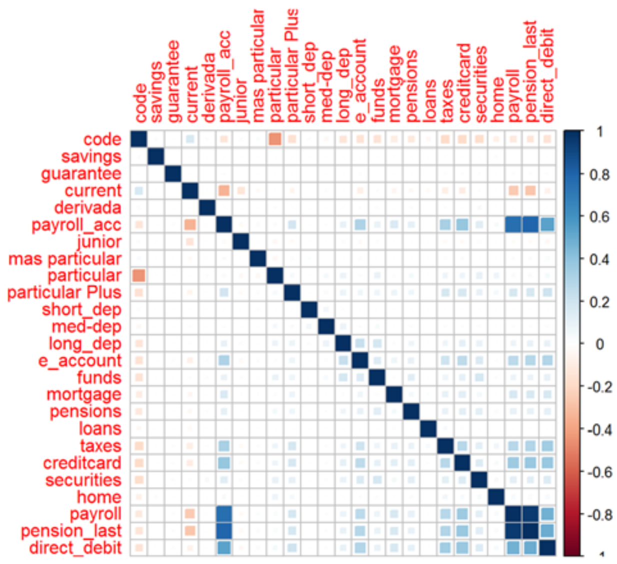
At most, users consume total of 15 products at the same time (Figure 5). The most popular product is current account and it is consumed by 8,938,150 users.

Figure 5. Total Number of Bank Products



Looking at the correlation of the products, payroll and pension_last products are the ones are highly correlated.

Figure 6. Correlation of Bank Products



Recommender System: Collaborative Filtering

I decided to use one of the popular methods of recommender system, Collaborative Filtering (CF), for my analysis to predict next recommendation of bank products. CF bases its predictions and recommendations on the ratings or behavior of other users in the system. This method assumes that other similar users' opinions can be selected and aggregated in such a way as to provide a reasonable prediction of the active user's preference. There are two ways you can use CF: user-user and item-item collaborative filtering.

User-user CF is first of the automated methods that has simple algorithmic approach. First, find other users whose past rating history or behavior is similar to the current user and use their ratings on other items to predict what current user will like. Even though this is a straightforward method, due to my large dataset, I cannot use user-user CF. First of all, it is more reasonable, if there are more items than users. In addition, as user base grows, it takes so much computational power than item-based approach. And if the user-base is more heterogeneous and many new customers, item-to-item method is the best to use. The best example is Amazon.com's recommender system.

Item-item CF is the method I used for my recommendation. This is most widely used collaborative filtering techniques today due to its scalability. Instead of using similarities between user ratings, item-based CF uses similarities between the rating patterns of items. If two items tend to have the same users like and dislike them, then they are similar and users are expected to have similar preferences for similar items.

For further analysis, I made additional transformations in the dataset. Here is the list of what I did:

1. I deleted all users who doesn't own any products
2. I only chose most active users
3. I subset only non-employee users
4. I subset users in the last month of the dataset.

As a result, my dataset has now only 300 thousand users compare to 13 million users in the beginning.

In addition, I chose only top 8 products for the analysis, which gave me much faster computational power to do CF analysis. My end result dataset looks as below (Figure 7).

Figure 7. Top 8 products

```
##     current        payroll_acc      particular       e_account
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.000
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.000
##  Median :1.0000   Median :0.0000   Median :0.0000   Median :0.000
##  Mean   :0.7418   Mean   :0.1825   Mean   :0.1711   Mean   :0.179
##  3rd Qu.:1.0000   3rd Qu.:0.0000   3rd Qu.:0.0000   3rd Qu.:0.000
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.000
##      taxes           payroll        pension_last     direct_debit
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000
##  Median :0.0000   Median :0.0000   Median :0.0000   Median :0.0000
##  Mean   :0.1073   Mean   :0.1223   Mean   :0.1338   Mean   :0.2857
##  3rd Qu.:0.0000   3rd Qu.:0.0000   3rd Qu.:0.0000   3rd Qu.:1.0000
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
```

Analysis1. Item-based collaborative filtering

After my data transformations, here is my steps to do item-based CF.

Algorithm:
1. Calculate similarity matrix between all items based on available users' ratings. For this could be used any preferred similarity measure. I used cosine function. The cosine similarity is a measure of similarity of two non-binary vector. The cosine similarity is defined by the following equation:

$$cos(A,B) = \frac{A \times B}{\|A\|\|B\|}$$

```
# Create a helper function to calculate the cosine between two vectors
  getCosine <- function(x,y)
  {
    this.cosine <- sum(x*y) / (sqrt(sum(x*x)) * sqrt(sum(y*y)))
    return(this.cosine)
  }

#
holder <- matrix(NA, nrow=ncol(bank.recom),ncol=ncol(bank.recom),dimnames=list(colnames(bank.recom),colnames(bank.recom)))

bank.recom.similarity <- as.data.frame(holder)


for(i in 1:ncol(bank.recom)) {
  for(j in 1:ncol(bank.recom)) {
    bank.recom.similarity[i,j]= getCosine(bank.recom[i],bank.recom[j])
  }
}

bank.recom.similarity <- as.data.frame(bank.recom.similarity)
head(bank.recom.similarity)
```

Final similarity matrix looks like this

```
##                current payroll_acc particular e_account     taxes
## current     1.00000000  0.04887032  0.3687681 0.2747616 0.1902221
## payroll_acc 0.04887032  1.00000000  0.1391024 0.3674849 0.3687115
## particular  0.36876805  0.13910241  1.0000000 0.1153420 0.1672927
## e_account   0.27476158  0.36748492  0.1153420 1.0000000 0.2688076
## taxes       0.19022212  0.36871153  0.1672927 0.2688076 1.0000000
## payroll     0.05957614  0.76568900  0.1245008 0.3153189 0.3153222
##                payroll pension_last direct_debit
## current     0.05957614   0.06127241    0.3545048
## payroll_acc 0.76568900   0.79755111    0.5742283
## particular  0.12450081   0.13023025    0.1897989
## e_account   0.31531889   0.33744902    0.3753149
## taxes       0.31532222   0.33478083    0.4053543
## payroll     1.00000000   0.95606375    0.5062049
```

2. For user:
2.1 Store only *n* closest items to each item;

# Get the top 2 neighbours for each,

# I only need to recommend 1 product for each user next month

```
bank.neighbours <- matrix(NA,
nrow=ncol(bank.recom.similarity),ncol=2,dimnames=list(colnames(bank.recom.similarity)))
```

2.2 Calculate predicted rating for each item based on available ratings of user *u* by weighting available ratings of users on similarities.

# Then I need to find the neighbours. This is another loop but runs much faster.

```
For (i in 1:ncol(bank.recom))
{
  bank.neighbours[i,] <-
(t(head(n=2,rownames(bank.recom.similarity[order(bank.recom.similarity[,i],decreasing=TRUE),
][i]))))
}
```

3.0 Results of recommendation

# First column is just identical of the product, so ignore it

```
head(bank.neighbours, n=8)
```

```
##             [,1]          [,2]
## current     "current"     "particular"
## payroll_acc "payroll_acc" "pension_last"
## particular  "particular"  "current"
## e_account   "e_account"   "direct_debit"
## taxes       "taxes"       "direct_debit"
## payroll     "payroll"     "pension_last"
## pension_last "pension_last" "payroll"
## direct_debit "direct_debit" "payroll_acc"
```

# Recommended products for each user

```
bank.final<-cbind(bank.matrix[,1],bank.neighbours[,2])
```

#Let's see recommended products for first 10 users

```
head(bank.final, n=10)
```

```
##      [,1]     [,2]
##  [1,] "657788" "particular"
##  [2,] "657795" "pension_last"
##  [3,] "657790" "current"
##  [4,] "657794" "direct_debit"
##  [5,] "657789" "direct_debit"
##  [6,] "657781" "pension_last"
##  [7,] "657779" "payroll"
##  [8,] "657770" "payroll_acc"
##  [9,] "657764" "particular"
## [10,] "657760" "pension_last"
```

Analysis2. User-based collaborative filtering

I also demonstrated user-based collaborative filtering based on first 5000 users.

```
bank.recomd<-bank.recomd[1:5000,]
```

I used *recommenderlab* package to create recommender model. To do this, I also need to transform my data to realRatingMatrix.

# First reshape the data to tidy data.

```
bank.recomd$user <- as.numeric(factor(bank.recomd$user))

bank.reshaped <- transform(melt(bank.recomd, id.var='user', na.rm=TRUE),
        variable=match(variable, names(bank.recomd)[-1]))
head(bank.reshaped)
```

```
##   user variable value
## 1 1973        1     1
## 2 1977        1     1
## 3 1975        1     0
## 4 1976        1     1
## 5 1974        1     1
## 6 1971        1     0
```

#Change dataset to SparseMatrix

```
sparse_ratings<-with(bank.reshaped, sparseMatrix(user, variable, x=value))
```

#recommenderlab works with realRatingMatrix object, which is created from sparse matrix

```
real_ratings <- new("realRatingMatrix", data = sparse_ratings)

head(as(real_ratings, "matrix"))
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
## [1,]    1    0    0    0    0    0    0    0
## [2,]    1    0    0    0    0    0    0    1
## [3,]    1    0    0    0    0    0    0    1
## [4,]    1    0    1    0    1    0    0    0
## [5,]    0    1    1    0    1    1    1    1
## [6,]    1    0    0    0    0    0    0    0
```

#Create a recommender model based on user ratings

```
bank.rec<-Recommender(real_ratings, method="UBCF", param=list(normalize = "center"))
getModel(bank.rec)
```

```
## $description
## [1] "UBCF-Real data: contains full or sample of data set"
##
## $data
## 5000 x 8 rating matrix of class 'realRatingMatrix' with 40000 ratings.
## Normalized using center on rows.
##
## $method
## [1] "cosine"
##
## $nn
## [1] 25
##
## $sample
## [1] FALSE
##
## $normalize
## [1] "center"
##
## $verbose
## [1] FALSE
```

# Split the data for train and test data

```
set.seed(1)
e <- evaluationScheme(real_ratings, method="split", train=0.8, given=3)
```

#Predicting recommendation

```
bank.pred<-predict(bank.rec, getData(e, "known"), type="ratings", n=1)
head(as(bank.pred, "matrix"), n=10)
```

```
##            [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
##  [1,]        NA        NA 0.2083333        NA 0.2083333 0.2083333
##  [2,]        NA 0.2083333 0.2083333        NA        NA 0.2083333
##  [3,] 0.2083333        NA        NA 0.2083333 0.2083333 0.2083333
##  [4,] 0.2083333        NA 0.2083333 0.2083333 0.2083333        NA
##  [5,] 0.0000000        NA        NA 0.0000000 0.0000000 0.0000000
##  [6,]        NA        NA 0.2083333        NA 0.2083333 0.2083333
##  [7,] 0.0000000 0.0000000        NA 0.0000000 0.0000000        NA
##  [8,]        NA 0.2083333 0.2083333        NA        NA 0.2083333
##  [9,] 0.0000000 0.0000000        NA 0.0000000 0.0000000        NA
## [10,]        NA        NA 0.2083333        NA 0.2083333 0.2083333
##            [,7]      [,8]
##  [1,] 0.2083333 0.2083333
##  [2,] 0.2083333 0.2083333
##  [3,]        NA 0.2083333
##  [4,]        NA 0.2083333
##  [5,] 0.0000000        NA
##  [6,] 0.2083333 0.2083333
##  [7,] 0.0000000        NA
##  [8,] 0.2083333 0.2083333
##  [9,] 0.0000000        NA
## [10,] 0.2083333 0.2083333
```

# Calculating RMSE

```
RMSE.ubcf <- calcPredictionAccuracy(bank.pred, getData(e, "unknown"))[1]
RMSE.ubcf
```

```
##    RMSE
## 0.45487
```

Conclusion

Overall, this project was overwhelming at first but was very intriguing and fulfilling. This was my first data mining project in large dataset of this size. Therefore, having results of recommended products at the end was exciting to conclude.

There are some limitations to this project. First of all, time to complete this project was only 2 weeks and one more week was granted to submit it. Second, my expertise of working in this level of large dataset was limited, so I faced lots of technical difficulties. The last, for demonstration purpose, I only created a model based on few number of users.

I know my conclusion wasn't complete. There should other steps to conclude my model and summarize by how accurate my model was and etc.

For item-based approach, I didn't use a model. I used memory-based algorithms, which simply memorizes all ratings and make recommendations based on relation between user-item and rest of the matrix. The end result simply print out the recommended products. This is only useful for offline calculation of recommender system.

However, if I need an online recommender system, I should create an accurate model. For this project, I only demonstrated user-based CF model based on only first 5000 ratings. Although my RMSE score was low, I don't have other models to compare with.

In addition, there are many ways to improve this model. For example, I can add user demographic dataset to the model to get more insights about users before I create a recommender model. Of course, there are also other modeling techniques other than collaborative filtering. To just name a few, Association rules, Cluster Analysis, Content-based filtering, and many more advanced techniques including hybrid approaches.

In conclusion, it is rewarding to learn about most popular recommender modeling method, a collaborative filtering, and find difference between user and item-based approaches. I hope this project is the only beginning of my data science career.

Reference

A tutorial at ECML PKDD (2015). Collaborative filtering with binary, positive-only data Retrieved from http://www.slideshare.net/koeverstrep/tutorial-bpocf

BigData Doc. (2014). Recommender System 101 – Step by step practical example in R Retrieved from http://bigdata-doctor.com/recommender-systems-101-practical-example-in-r/

Marafi, S. (2014). Collaborative filtering with R. Blog Post. Retrieved from http://www.salemmarafi.com/code/collaborative-filtering-r/

Sachdeva, K. (2016). A big data-enabled modern recommendation engine for retail banking. IBM Big Data & Analytics Hub. Retrieved from http://www.ibmbigdatahub.com/blog/big-data-enabled-modern-recommendation-engine-retail-banking

Appendix 1.

Column Name	Description
fecha_dato	The table is partitioned for this column
ncodpers	Customer code
ind_empleado	Employee index: A active, B ex employed, F filial, N not employee, P pasive
pais_residencia	Customer's Country residence
sexo	Customer's sex
age	Age
fecha_alta	The date in which the customer became as the first holder of a contract in the bank
ind_nuevo	New customer Index. 1 if the customer registered in the last 6 months.
antiguedad	Customer seniority (in months)
indrel	1 (First/Primary), 99 (Primary customer during the month but not at the end of the month)
ult_fec_cli_1t	Last date as primary customer (if he isn't at the end of the month)
indrel_1mes	Customer type at the beginning of the month ,1 (First/Primary customer), 2 (co-owner),P (Potential),3 (former primary), 4(former co-owner)
tiprel_1mes	Customer relation type at the beginning of the month, A (active), I (inactive), P (former customer),R (Potential)
indresi	Residence index (S (Yes) or N (No) if the residence country is the same than the bank country)
indext	Foreigner index (S (Yes) or N (No) if the customer's birth country is different than the bank country)
conyuemp	Spouse index. 1 if the customer is spouse of an employee
canal_entrada	channel used by the customer to join
indfall	Deceased index. N/S
tipodom	Addres type. 1, primary address
cod_prov	Province code (customer's address)
nomprov	Province name
ind_actividad_cliente	Activity index (1, active customer; 0, inactive customer)

Column Name	Description
renta	Gross income of the household
segmento	segmentation: 01 - VIP, 02 - Individuals 03 - college graduated
ind_ahor_fin_ult1	Saving Account
ind_aval_fin_ult1	Guarantees
ind_cco_fin_ult1	Current Accounts
ind_cder_fin_ult1	Derivada Account
ind_cno_fin_ult1	Payroll Account
ind_ctju_fin_ult1	Junior Account
ind_ctma_fin_ult1	Más particular Account
ind_ctop_fin_ult1	particular Account
ind_ctpp_fin_ult1	particular Plus Account
ind_deco_fin_ult1	Short-term deposits
ind_deme_fin_ult1	Medium-term deposits
ind_dela_fin_ult1	Long-term deposits
ind_ecue_fin_ult1	e-account
ind_fond_fin_ult1	Funds
ind_hip_fin_ult1	Mortgage
ind_plan_fin_ult1	Pensions
ind_pres_fin_ult1	Loans
ind_reca_fin_ult1	Taxes
ind_tjcr_fin_ult1	Credit Card
ind_valo_fin_ult1	Securities
ind_viv_fin_ult1	Home Account
ind_nomina_ult1	Payroll
ind_nom_pens_ult1	Pensions
ind_recibo_ult1	Direct Debit