Big Data Analytics Customer Profiling Project

Authors: Justin Tse, Cindy Xu

Github link: https://github.com/justinsynth/BigDataAnalyticsProjectGit

The Github does not contain the original data or the datasets that we modified.

Abstract

The goal for this project is to create a customer profile. We will be working with data provided by Santander bank. The data contains demographic and transactional behavior of customers over a 17-month period. Using this customer data, we want to create a model that can predict which products a customer will buy or drop in the next month. We also want to be able to measure each customer's level of satisfaction

Milestones

Our first milestone is to perform exploratory data analysis on the data and come up with a plan for the rest of the project.

Our second milestone is to create a model that will be able to predict which products a customer will buy or drop in the next month.

Our third milestone is to create a model that will be able to gauge customer satisfaction. To measure this, we will focus on customers that have become inactive over a ten-month period. We will try to predict which customers become inactive and determine indicators that suggest which customers will become inactive.

Related Work

Visa Rule Suggestion

The goal of the Visa Rule Suggestion Engine is to determine which transactions are legitimate and which transactions are fraudulent for merchants. Purchase habits can be easily identified for customers and fraud detection systems assist merchants whether transactions are legitimate or fraudulent by using different sorting algorithms. The method used is to obtain and segment transaction data, identify key indicators that correlate to the segmented data and suggest transaction rules to the user for the future. The general structure is to have a

The structure is first to have a rule suggestion engine and to create a profile for merchants. The engine first applies the suggested rule in passive mode to check if fraud probability is low. If it is then, add the suggested rule to the customer profile or to the suggested rules list; if it isn't, modify the rule and continue the above steps until the fraud is low. This work is similar because they are using past transactions to help train their model. One of our goals also is to find indicators that will suggest what a customer will do.

Using Data Mining to Build Customer Profiles Research Feature

This research feature by Gediminas Adomavicius and Alexander Tuzhilin details ways to build a customer understanding profile with data mining. The two main phases of the profile-building process are rule discovery and validation. They use data mining to detect rules and also to validate rules to build profiles. The model classifies the data into two basic types: factual—who the customer is— and transactional—what the customer does. It then generates association rules by using the Apriori Algorithm. When finding association rules for a particular customer, the model often generates large number of rules. Therefore, validating the discovered rules is an important requirement. Applying each validation operator results in the acceptance of some rules and the rejection of others. The model used is similar to a decision tree, this research is useful to us because it gives an idea of where we should start for our project.

Dataset

The dataset that we are using is provided by Santander bank. It contains demographic data on customers and it tracks which products each customer has. Each entry in the data set represents a customer in a single month, it contains the demographic data for that customer and what products they have in that month. There are 23 different products; these include items Savings Accounts, Credit Cards, and Loans. The dataset contains tracks customer information from January 2015 to June 2016. The dataset has roughly 13.5 million entries representing over 900,000 customers. It is important to note that each customer can have multiple entries in the dataset each representing a different month. In this data, customers can add products in each month, or drop products in each month.

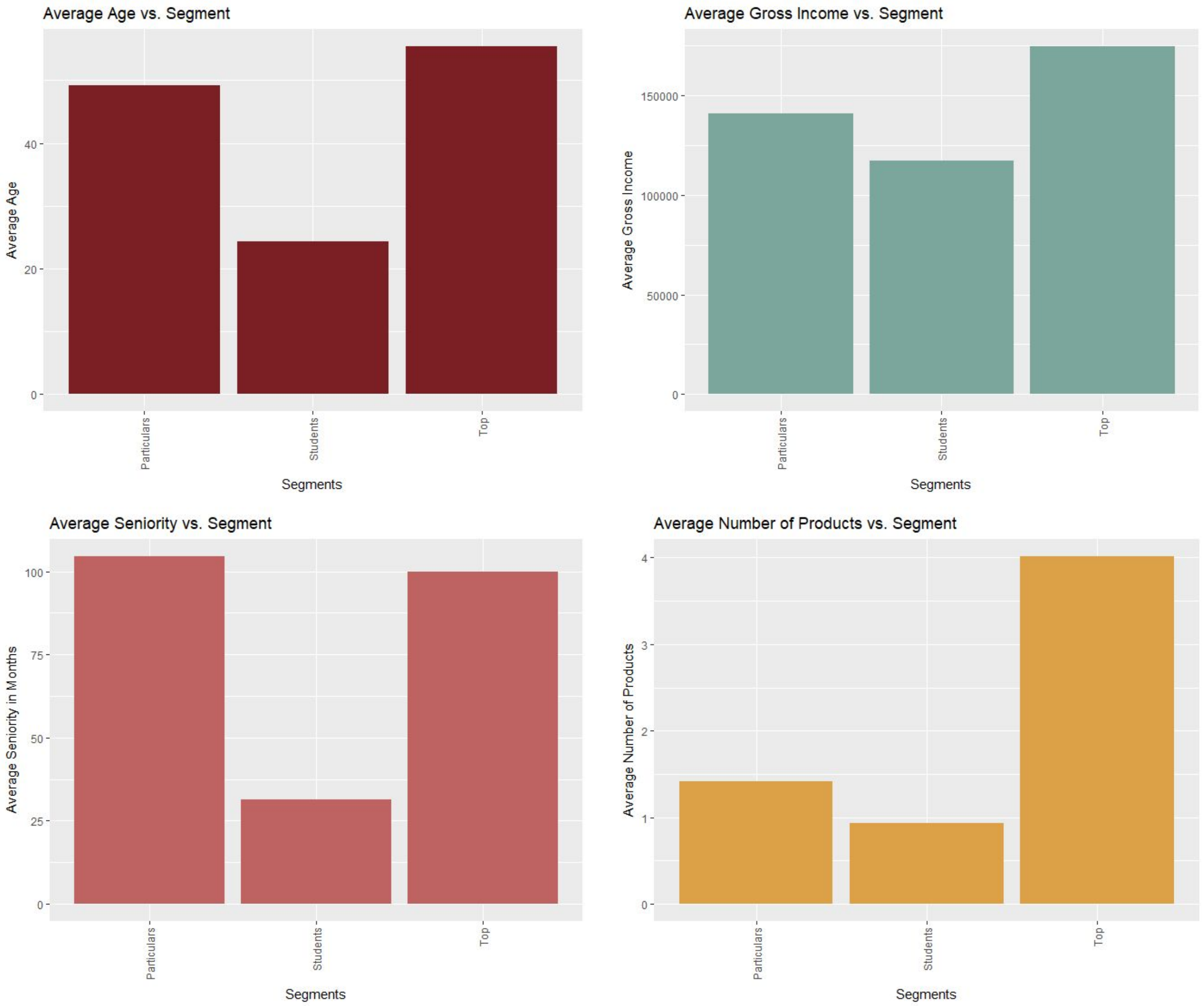


Figure 1: Graphs representing distributions of different features over segmentations of the

customer base. The segments are Particulars – normal customers. Students – university aged customers, Top – VIP customers.

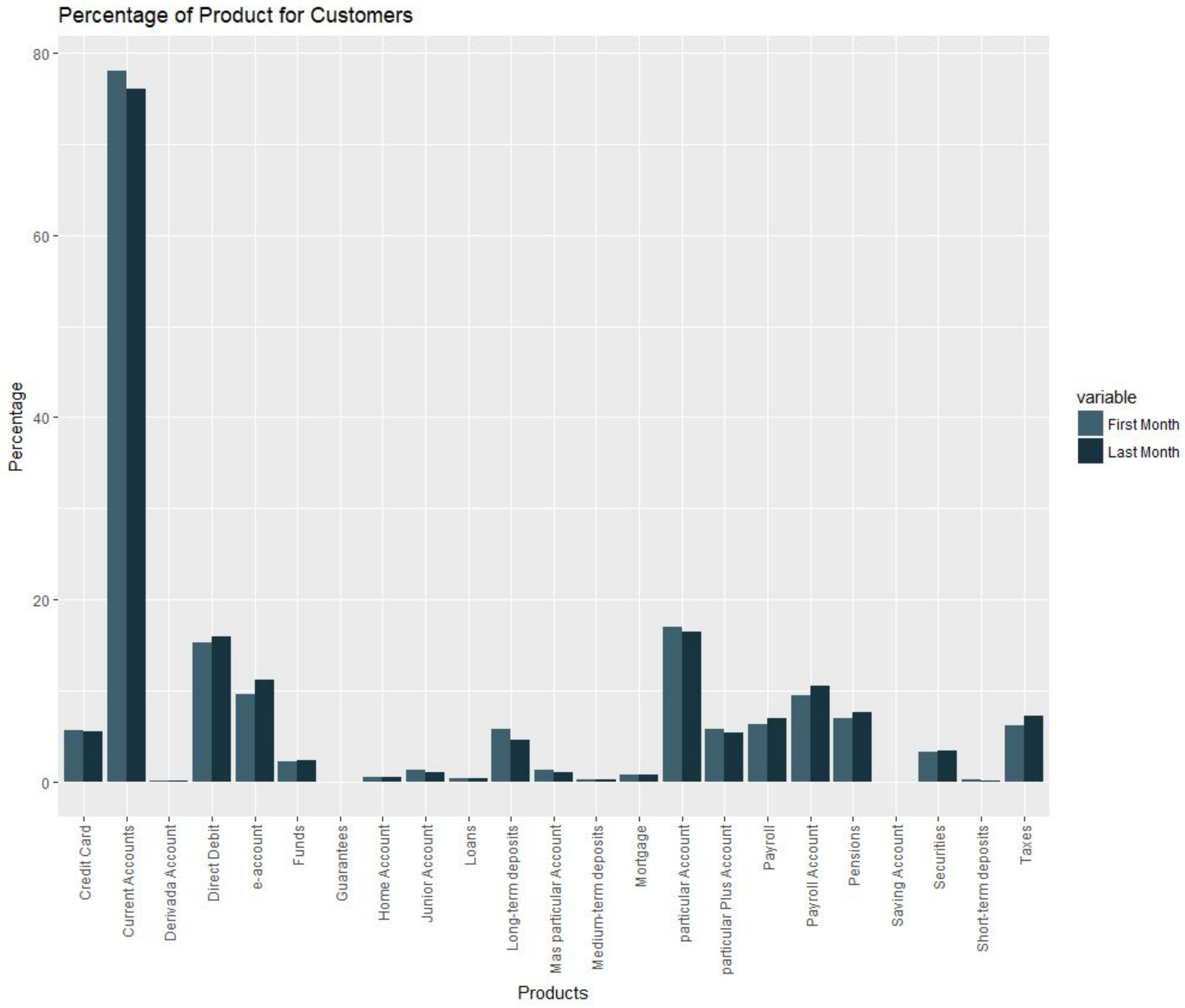


Figure 2: Graph representing how many customers have each product in in the first and last months.

There are very few missing values in our data, the only values that are missing are some entries for gross income and some entries for Payroll and Pensions. We set the missing values for gross income equal to the mean gross income and we fill the missing Payroll and Pension values with the value for the last known month. For our project, it is much more useful to have a unique entry for each customer. Since demographic data rarely changes, we modify the dataset so that each entry represents a unique customer and contains all of their past behavior. We also created a feature that represents how many products a customer has in the last known month. It is important to note that each customer can have a different amount of data. For example, there are customers that are added to the bank in the last 2 months of data collection, they will only have 2 months of data provided.

Our first task is to create a model that will predict which products a customer will buy or drop the next month. For our initial analysis, we decided to restrict the data that we used to lag2 and lag5 data. We start with these sets of data because if they give similar results, that suggests that it is not important to consider behavioral data past 5 months. If lag5 performed significantly better than lag 2, we would go back and consider other segments of the dataset.

For our second task (predicting customer satisfaction) we will want to use the past behavior to create many new features like number of products dropped in the last x months, or proportion of time inactive.

Milestone 2

For all of the following approaches, we first tested on a 10 percent sample of the data, and then trained on the full lag2 and lag5 data. We did this because some algorithms can take hours to train on the full data. Practicing on smaller datasets makes parameter tuning much easier. It is important to note that we retrained the algorithms for each individual product. This was necessary because of some of the optimizations that we used to improve our model's performance.

Logistic Regression

Our first approach is logistic regression. We choose this algorithm because it is faster than the other ones that we want to try and because it has a probabilistic interpretation. We felt that having a probability for each item would be very useful for the bank.

	Lag2 Data	Lag5 Data
Average Training Error	.01834	.00365
Average Test Error	.01944	.00360

In the table above, average error is the average error amongst all of the products that we are predicting. The lag2 data set has roughly the same performance as the lag5 dataset.

For our preliminary result the accuracy that we obtain is excellent, but it is misleading. It is high because we do not differentiate between the different cases for customer behavior. In the dataset, for each product, customers do not change their behavior roughly 99% of the time. This means if they have a product in one month they will typically have it in the next month and if they do not have a product in one month they will typically not have it in the next month. These cases are uninteresting to us. The bank is most interested in customers who are going to buy a product or customers who are going to drop a product.

	Lag2 Data
General Test Error	0.01943809

New Product Test Error	0.8502564
Dropped Product Test Error	0.5390374

New Product Test Error is the error among cases where customers buy a new product, Dropped Product Test Error is the error among cases where customers drop a current product. We will call these cases our special cases. In the special cases, logistic regression has abysmal performance.

To fix this issue, we used class rebalancing to get a better distribution of accuracies.

	Lag2 Data	Lag2 Data, Rebalanced
General Test Error	0.01943809	0.2616639
New Product Test Error	0.8502564	0.2208679
Dropped Product Test Error	0.5390374	0.1397276

After rebalancing we get much better results. For the above model, the we used a 1:1 ratio between special cases and general cases. When we do class rebalancing we train on less data but we still test on the same amount. It is also important to note that each product now has a different training population, items with more special cases will have more data to train on. So, the results above are a weighted average of accuracies. The products with more training data have their results weighted more heavily.

The class rebalancing that we used makes it so we cannot do multilabel classification since we use different data sets to train.

Random Forests

The next model that we test is random forests. The motivation behind choosing this algorithm is increased accuracy and effective feature suggestion. It was also suggested by Vishal who has experience with similar problems.

	Lag2, 35 trees, 10:1 ratio	Lag5, 50 trees 20:1 ratio	Lag2, 35 trees, 2x var/tree, 20:1 ratio
General Test Error	0.351458	0.2408037	.2201501
New Product Test Error	0.3041408	0.4981268	.4291711
Dropped Product Test Error	0.1895111	0.3582134	.2913976

Again, we found that lag2 data had roughly the same performance as lag5 data. It’s interesting to notice that in general, results with random forests are worse than our results with logistic regression. This could be due to the homogeneity of the raw behavior data from month to month.

Random forests is still useful because, it has an intuitive way to see variable importance importance.

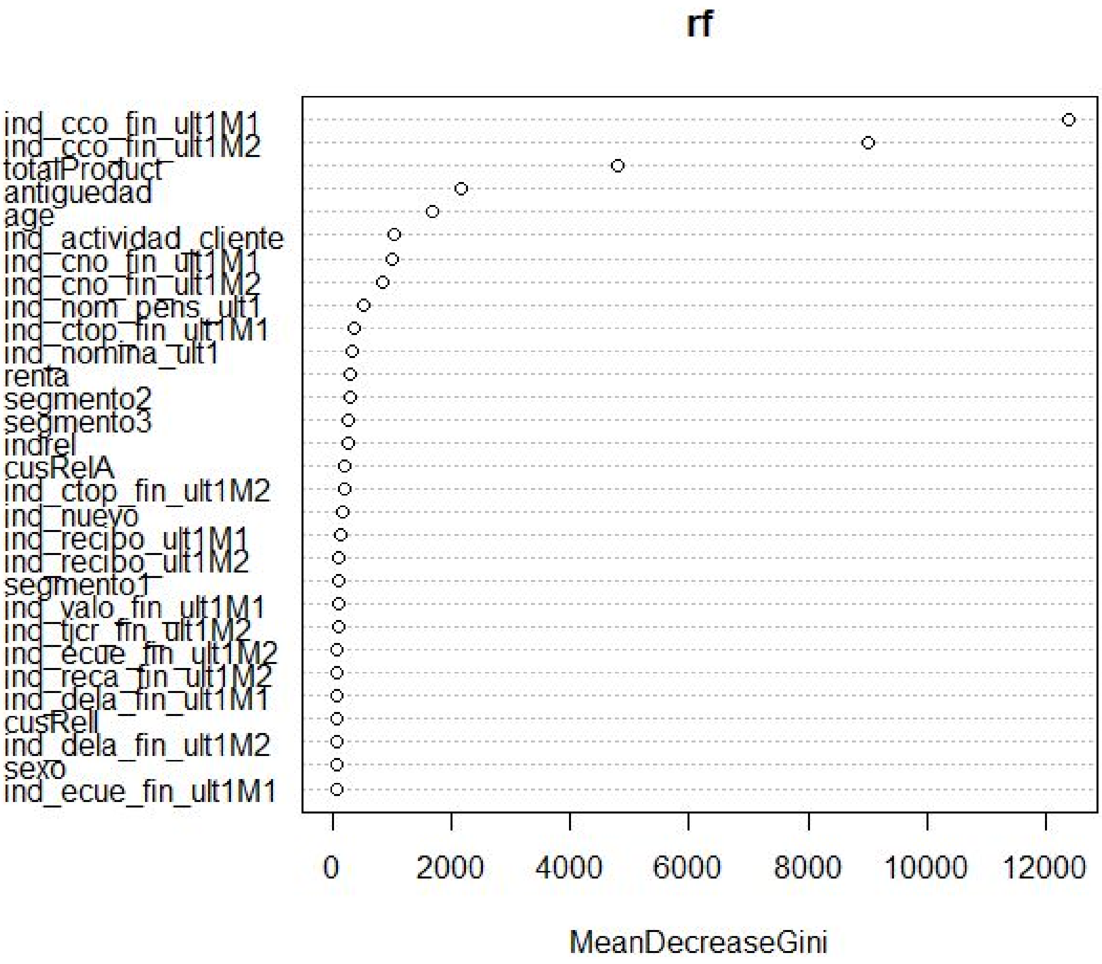


Figure 3: Variable importance plot for predicting product ind_cco_fin_ult from a random forests model with 35 trees, 10:1 rebalancing ratio trained over lag2 data.

In the dataset, variables beginning with ind are products. totalProduct is the total amount of products that a customer owns, antiguedad is the seniority of the customer, renta is gross income, and segmento2 and segment3 indicates the segment a customer is in.

We can see that past behavior is by far the most important indicator for future behavior. More specifically, past behavior for the same product is the most important, then past behavior for other products. Only some demographic information like seniority or gross income are important. The variable importance plot for other products and other random forests models were very similar to this one.

Gradient Boost

The last approach that we test for Milestone 2 is gradient tree boosting; we use the XGBoost package. The motivation behind choosing this algorithm is to get increased accuracy.

	Lag2, depth = 5, eta = .7, 2:1 ratio	Lag2, depth = 5, eta = .7, 10:1 ratio	Lag5, depth = .6, eta=.7, 10:1 ratio
General Test Error	.551679	.3745007	.4386165
New Product Test Error	.14655053	.3070853	.3051004
Dropped Product Test Error	.05731434	.1907504	.2030429

Like Random Forests, gradient boosting trees also does not perform as well as logistic regression. No matter the parameters or rebalancing ratios we used, we are unable to get better performance when considering all of the cases. Even if we only focused on the special cases, it is impossible to get better performance without completely sacrificing our performance in the general case.

Gradient boosting is still useful because we can extract variable importance data.

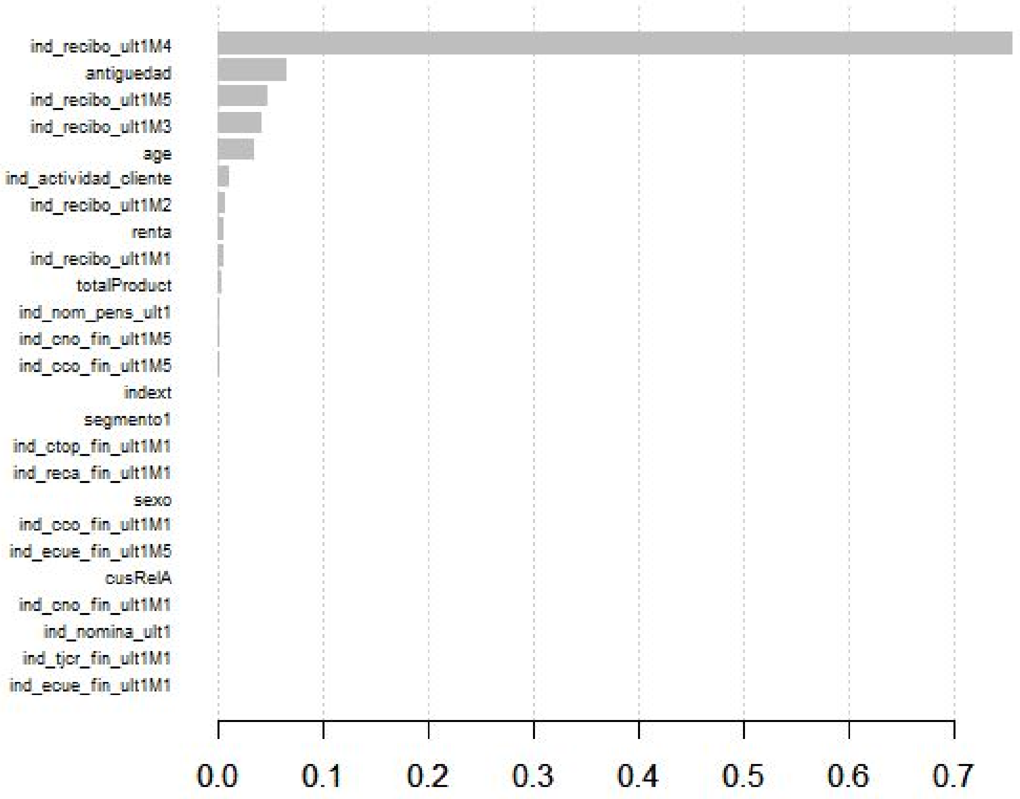


Figure 4: Variable importance plot for predicting ind_recibo_ult for an XGBoost model on lag5 data using a 10:1 rebalancing ratio. Parameters are maxdepth = 6, eta = .7, nrounds = 4, min_child_weight = 1.

In general, variable importance data from gradient boosting supports the results obtained from our experiments with random forests.

For random forests and gradient tree boosting, we did not have overfitting issues with any of the models, training error was generally very close to test error in each case. In XGBoost, parameter tuning is mainly used to reduce overfitting. However, for our problem training error was always worse than in linear regression. So, parameter tuning does not help much.

Milestone 3

Data Preparation and Engineering Features

For milestone 3, the dataset must be cleaned and prepared in a different way. We decided to focus on a 10 month period. Ideally, we would want a slightly longer period but we found that 10 months allowed us enough time to train our algorithms. We focused only on users that were active in the first month. There were 344,154 customers who were active in the first month and 17,652 became inactive by the end of the 10 month period.

One problem that we faced in the previous milestone was that there was a lot of repetition and homogeneity in the behavioral data. This could have been one of the problems that contributed to poor performance in our models, especially the performance of our random forests model. In order to fix this, we engineered new features that would summarize the raw behavioral data. Each feature represents a trait about a single customer over the 10 month period. The features are listed below.

TotalProductFirstMonth – Total number of products a customer has in the first month

TotalProductLastMonth – Total number of products a customer has in the last month

TotalProductDiff – TotalProductLastMonth – TotalProductFirstMonth

prodImp (Product Importance) –

$$\log \sum_{\text{products}} \mathbb{1}_{\text{customer owns product}} \cdot (\text{proportion customers who own product})^{-1}$$

We would have liked to also include total number of products added and total number of products dropped but it was too computationally expensive to look at the product changes between each month. Our final dataset is the demographic data, the engineered features and the raw behavioral data for months 5 and 10.

Logistic Regression

Our first approach for this milestone is logistic regression. We choose this algorithm because it is relatively fast and the probabilistic interpretation of the results can be useful to the bank. The result for the logistic regression on the dataset is below.

	Training Set	Testing Set
General Accuracy	.9613965	.9628069
Inactive Accuracy	.3410581	.3613041

Inactive Accuracy is the accuracy of predicting the customers who became inactive during the 10 month period. Inactive Accuracy is essentially the true negative rate of our classification model. Inactive Accuracy is very poor compared to General Accuracy because our test set is imbalanced. Like Milestone 2, to account for this we use class rebalancing.

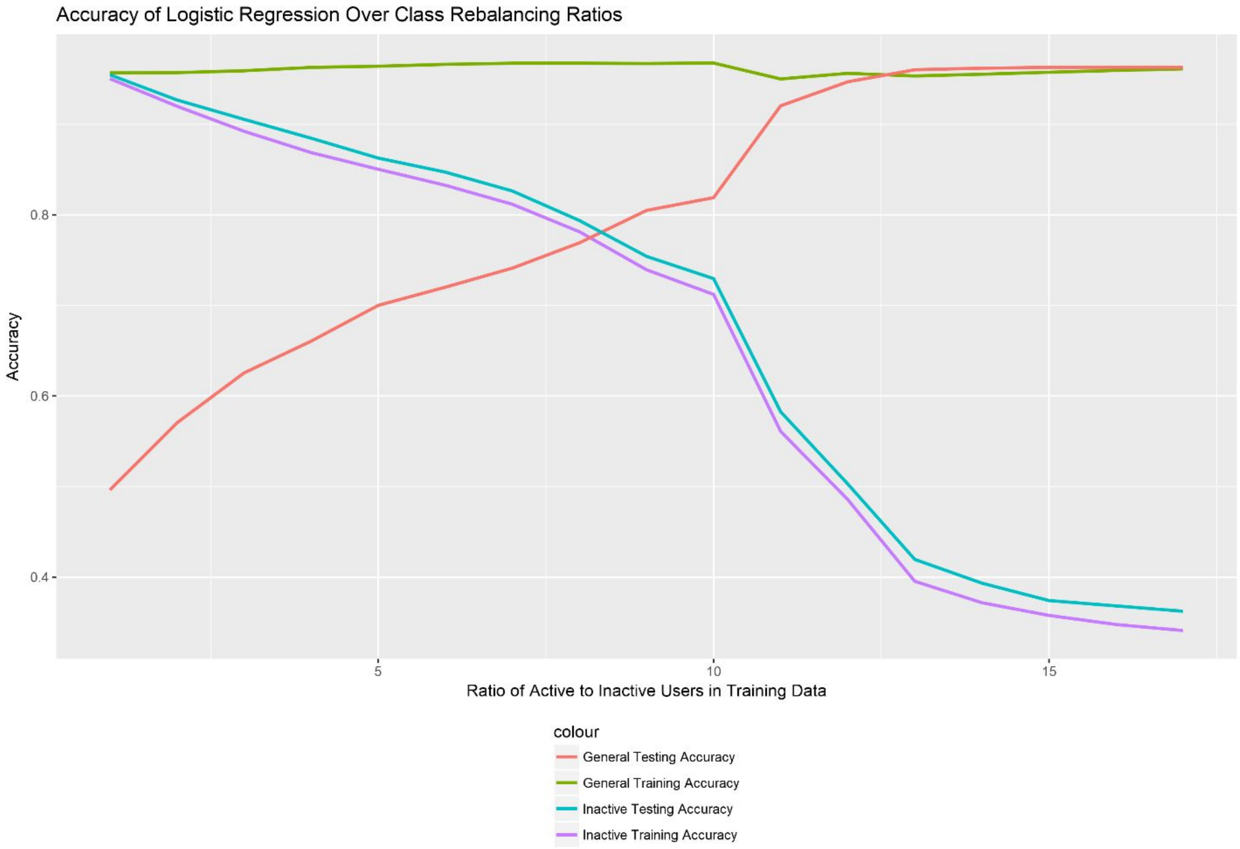


Figure 5: Accuracy of Logistic Regression over different class rebalancing ratios.

The graph above represents the results of logistic regression over several different class rebalancing ratios. The x axis represents the ratio of active cases to inactive cases. The raw data is below.

When the training data has a ratio of 8:1 of active customers to inactive customers, the general accuracy and inactive accuracy is roughly the same.

	Training Set	Testing Set
General Accuracy	.9673455	.7690542
Inactive Accuracy	.7813085	.7936932

The next aspect that we analyze is the effectiveness of the engineered features. To test this we trained the logistic regression model using the rebalanced training set with a 8:1 ratio of active customers to inactive customers on different subsets of features. Each of the subset of features used a different set of behavioral data. No Engineered Features only uses the raw behavioral data, No Month 5 Raw Data only uses half of the raw behavioral data, and No Raw Data only uses the engineered features.

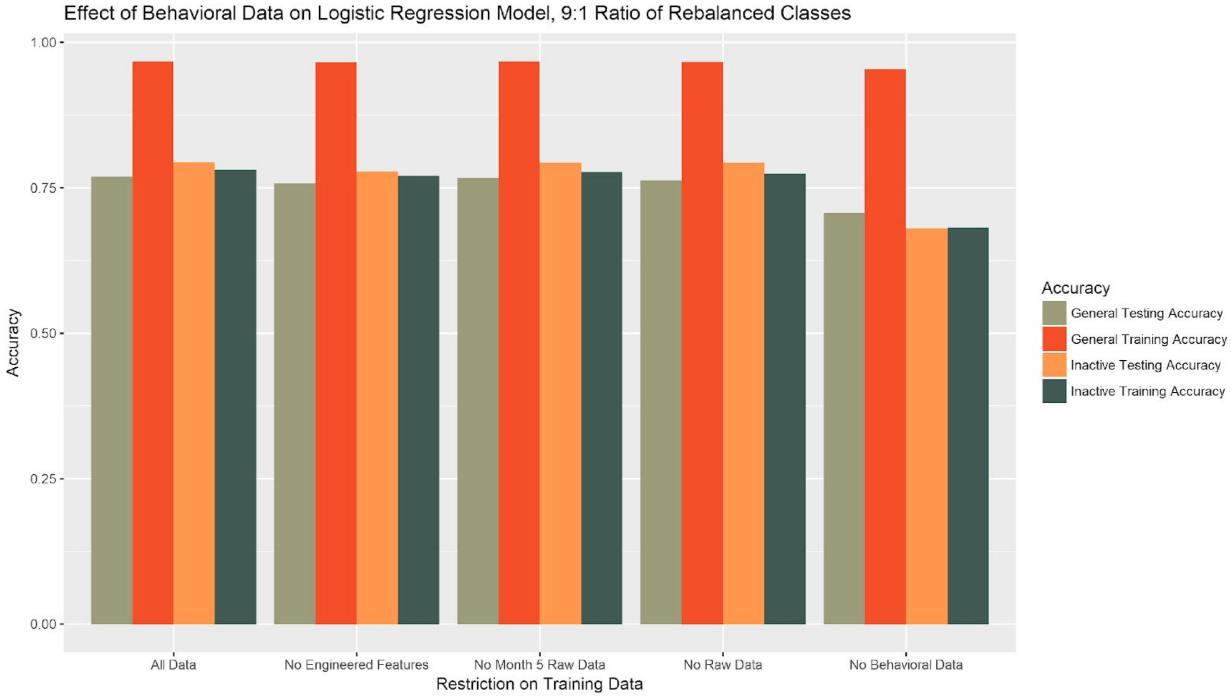


Figure 6: Accuracy of Logistic Regression over several different subsets of the training data. Data was trained on a 8:1 ratio of rebalanced classes.

All of the models with behavioral data gave very similar performance. This suggests that the engineered features efficiently summarize the raw behavioral data. Removing all of the behavioral data causes roughly 10% decrease in testing accuracies.

Random Forests

We first tried to find the best rebalancing ratio using the same techniques as above. We tried ratios from 1 to 10.

General Training Accuracy	Inactive Training Accuracy	General Testing Accuracy	Inactive Testing Accuracy	Rebalance Ratio
0.971881	0.973139	0.48168	0.958846	1
0.969725	0.949731	0.551504	0.93907	2
0.971574	0.925328	0.601772	0.916088	3
0.972577	0.899813	0.645649	0.897916	4
0.97348	0.885417	0.691733	0.87814	5
0.974477	0.866983	0.71989	0.864244	6
0.975056	0.850597	0.736278	0.849813	7
0.975584	0.832573	0.758158	0.828434	8
0.975591	0.799508	0.787273	0.799038	9
0.97632	0.790379	0.797152	0.790486	10

The best result we found was when we set the ratio of active cases to inactive cases to 9:1.

We then tuned the parameters with this training set, however the results stayed roughly the same. We tried configurations with up to 150 trees however the error seems to converge extremely quickly, usually within 5 trees. The following result is from a model with 50 trees with 9 variable per tree. Tuning the parameters gave a insignificant increase to testing accuracy at the cost of a significant drop off in training accuracy.

	Training Set	Testing Set
General Accuracy	.7844734	.7827691
Inactive Accuracy	.8131437	.8054516

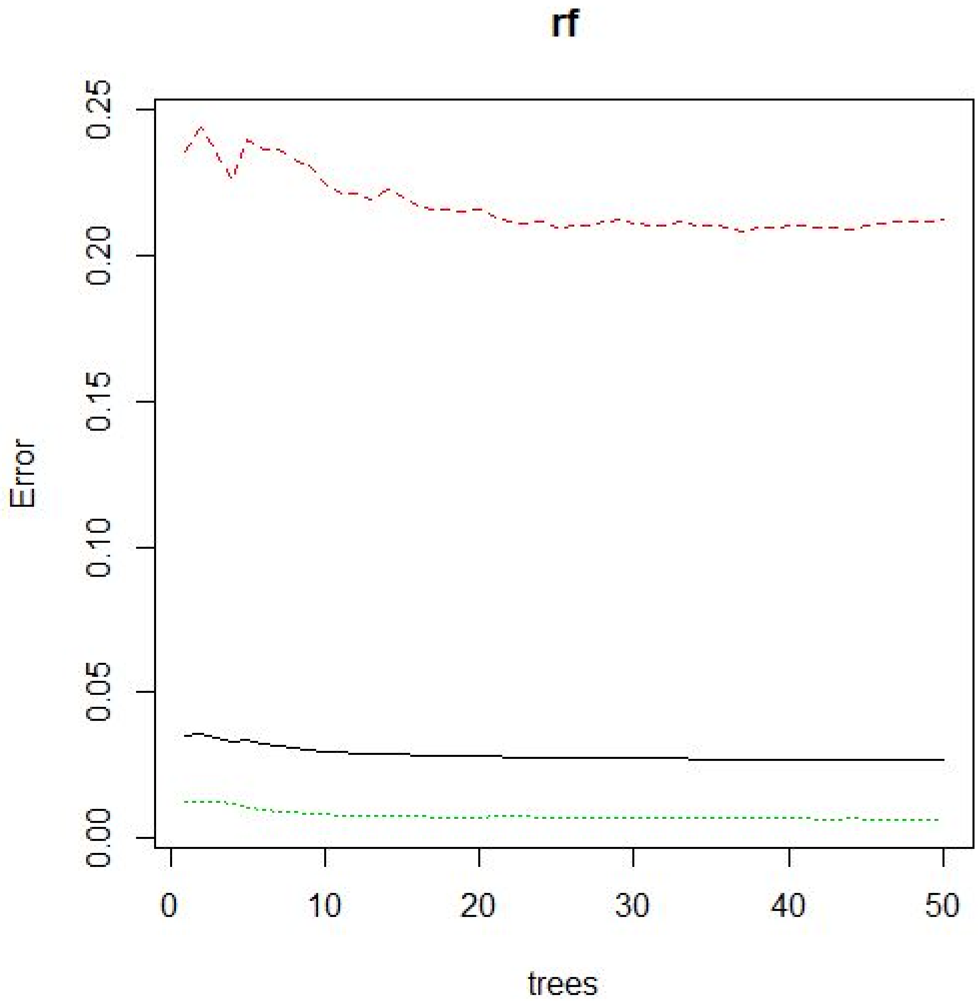
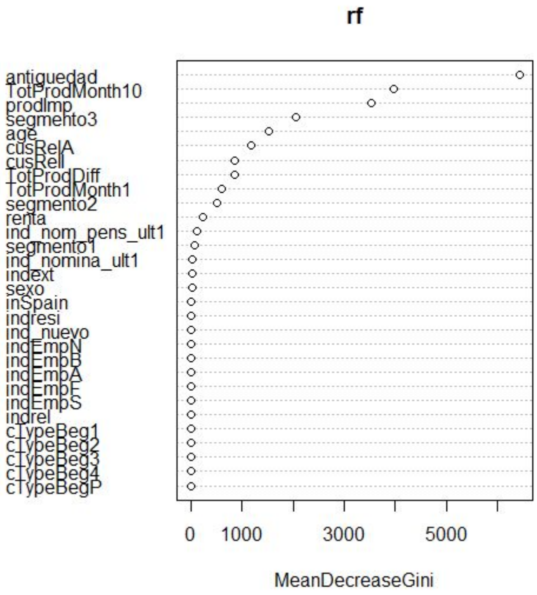


Figure 7: Evolution of error of random forest model as the number of trees grow. The green line represents error in the active case, the red line represents the error in the inactive case, the black line represents the general error rate.

It is useful to look at variable importance for the random forests models and assess whether the features that we engineered are useful.



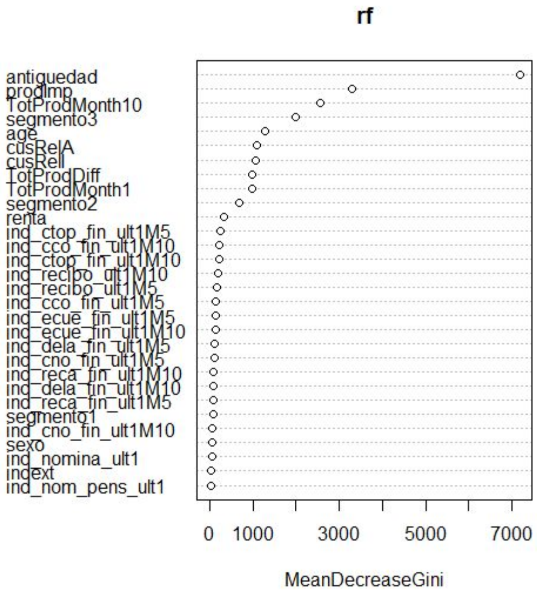


Figure 8: The graph on the left is the variable importance plot when there is no raw behavioral data. The graph on the right is the variable importance plot when all of the raw behavioral data is included.

These plots suggest that the features that we engineered are important. Unlike in the product prediction models, demographic data is vital to predicting which customers will become inactive. The most significant feature is antiguedad which is the seniority of the customer.

Neural Network

The next model that we try is neural networks. We did some research on why we want to use neural networks. With their remarkable ability to derive meaning from complicated or imprecise data, they can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques. A trained neural network can be thought of as an "expert" in the category of information it has been given to analyze.

The motivation behind choosing this algorithm is increased accuracy and effective feature suggestion. It does not impose any restrictions on the input variables (like how they should be distributed) and it can better model heteroskedasticity i.e. data with high volatility and non-constant variance, given its ability to learn hidden relationships in the data without imposing any fixed relationships in the data. This is something very useful in financial data where data volatility is very high. When we mention the effective feature suggestion, it’s because neural networks work well for non-linear data, it gives enough depth and it can extract different features to a new dimension space (like different combinations of features).

For milestone 3, we focus on customers’ satisfaction based on their activity. First, we directly use the training data set without rebalancing the data, it gives a really low training accuracy of around 48%. So, we rebalance the data with two labels at a 1:1 ratio.

The following is the result with rebalanced, unscaled data.

	Training Set	Testing Set
General Accuracy	.75	.74
Inactive Accuracy	.67	.66

It still gives a relatively low accuracy. So we then scale the data. We normalize the data by using sklearn. We do this because each feature of the financial data has different range. For example, the age of the customer has a much larger range than the feature representing if a customer is using a product. If we scale the data we should be able to achieve a better result.

	Training Set	Testing Set
General Accuracy	.8209562	.8057533
Inactive Accuracy	.8362008	.8400267

Compared with logistic regression, both the training and test general accuracy is low but inactive accuracy is pretty high. Compared with random forests, only the training general accuracy is low and the others are better.

Based on previous experience, neural networks should be able to achieve a pretty high accuracy, so we tried implementing a neural network in Keras to achieve a better result.

Here’s my architecture for neural network.

```
model.add(Dense(120, input_dim=77, activation='relu'))

model.add(Dense(120, activation='relu'))

model.add(Dense(1, activation='sigmoid'))
```

We use two fully connected layers with both 120 hidden units.

	Training Set	Testing Set
General Accuracy	.8409	.8290
Inactive Accuracy	.817591	.786745

There is little improvement.

Conclusion and Final Thoughts

For this project, the models that we built for both goals are quite similar. They are classification models that are trained on extremely imbalanced data. With class rebalancing accuracy between the different types of error can be manipulated but there is a limit to the accuracy that we can achieve. Even though the dataset is large, there is little data on the cases that we care about, especially for products that are rarer.

In general, it is easier to predict when a customer will drop a product than it is to predict when a customer will buy a new product.

When predicting whether or not a customer will become inactive, we were able to get consistent results. We were also able to create 4 new features that summarized the information contained in 230 features. This made it easier to train our models.

Contributions

Cindy:

Milestone 1: Research into building customer profiles

Milestone 2: Research into VISA Rule Suggestion Engine

Milestone 3: Neural Network Model

Justin:

Milestone 1: Exploratory data analysis on relevant features

Milestone 2: Data preparation, Data cleaning, Logistic Regression, Random Forests, Gradient Tree Boosting

Milestone 3: Logistic Regression, Random Forests

References

https://pdfs.semanticscholar.org/b298/e06b9ee4b3056c68a023035f228527a891a2.pdf

https://patents.google.com/patent/US20130054438

https://www.deeplearningtrack.com/single-post/2017/07/09/Introduction-to-NEURAL-NETWORKS-Advantages-and-Applications

https://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html