VU VRIJE UNIVERSITEIT AMSTERDAM

Faculty of Economics and Business Administration

Santander Customer Satisfaction

Derek van den Elsen

2580100

October 30, 2017

Research Paper Business Analytics

Dr. Mark Hoogendoorn (supervisor)

Contents

1 Introduction

Having adequate customer relations is paramount to success in any service industry. Identifying and analyzing your customer's contentment to improve customer retention can yield many benefits. The longer a client stays with an organisation, the more value he creates. There are higher costs attached to introducing and attracting new customers. The clients also have a better understanding of the organisation and can give positive word-of-mouth promotion. (Colgate et al., 1996) Datamining is essential in this process and this practice is widely applied across industries for instance FMCG retailers (Buckinx and van den Poel, 2005), telecommunications (Mozer et al., 2000) and banking (Clemes et al., 2010) and (Xie et al., 2009).

This paper focuses on Santander Bank, a large corporation focusing principally on the market in the northeast United States. Through means of a Kaggle competition (Santander, 2015), it is the objective to find an appropriate model to predict whether a client will be dissatisfied in the future based on certain characteristics. Having this model in place can ensure that Santander can take proactive steps to improve a customer's happiness before they would take their business elsewhere.

First the paper will discuss related work done on this, by now concluded, Kaggle case. Secondly it delves into the data we work with, analyzing groups of variables and individual features to give us insight in what is relevant. Thirdly several cleaning procedures that were employed to lead to better results are outlined. Fourthly we explain the performance measure of this competition and the three models: Logistic Regression, Random Forest and XGBoost that we utilize to tackle the problem. Lastly the tuning process and results are discussed. We reach an AUC score of 0.823152.

2 Related Work

This section looks into the work of various Kaggle competitors in different sections of the leaderboard. The private leaderboard score is mentioned first after which a small description details the work done. For reference the top position had a score of: 0.829072

0.828530: Silva et al. (2016) all seemingly independently do their own preprocessing, feature engineering and model selection and then combine all predictions of the models together in an ensemble using the R package optim. Preprocessing steps taken are for instance: replacing certain values by NA, dropping sparse, constant and duplicated features, normalization, log transforming features and one-hot encoding categorical features. Sophisticated feature engineering methods employed include t-Distributed Stochastic Neighbour Embedding, Principal Component Analysis and K-means. Models explored are: Follow the Proximally Regularized Leader, Regularized Greedy Forest, Adaboost, XGBoost, Neural Networks, Least Absolute Shrinkage and Selection Operator, Support Vector Machine, Random Forest and an Extra Trees Classifier.

0.826826: Yooyen and Ma (2016) are one of the few that find and handle some duplicated observations in the train set. Furthermore they do standard preprocessing steps like removing duplicated and constant features, normalizing, rescaling and handling missing values. They select features based on Pearson's Correlation Coefficient with the target and crossvalidation. Attempts at specifically handling the class imbalance and principal component analysis were fruitless. With success they added heuristic rules like `var15` < 23 means the target is 0. Their main models are based on decision trees.

0.8249: Wang (2016) applies hardly any preprocessing as he intends to use only decision trees that are relatively insensitive to this. He does extensively select features with the importance in the Gradient Boosting Classifier as criterion. After which he adds percentile change from feature to feature, and he applies selection again. Parameters are trained iteratively one at a time with a coarse to fine approach. Adaboost, Bagging Classifier, Extra Trees Classifier, Gradient Boosting, Random Forest and XGBoost are ensembled to lead to the final scores.

0.824332: Kumar (2015) tries linear regression with the top ten features, a support vector machine in combination with principal component analysis with limited success as a start. Neural Networks, tuned Random Forest and XGBoost yield him better results. XGBoost is seen as obvious best candidate and it is used with the top N features, which makes it apparent that anything beyond 5 features only adds very minor improvements to the crossvalidation score.

3 Data Exploration

The train set consists of 76020 observations and 370 features plus 1 binary target. The test set has a roughly equal amount of 75818 observations with the same features. There is a large imbalance with 96.04% being 0, meaning the customer was not dissatisfied and 3.96% being 1, signifying that the customer was dissatisfied. This is in line with the expectation of the customer satisfaction of a successful bank. There are no missing values in the train or test set, but some values might encode 'missing'. There are only numeric or possibly categorical features. No features seem to have substantial outliers.

The dataset is semi-anonymized, so it is unclear what a feature represents. The only clue we have is a header with a name for each feature that is clearly not randomly determined. For illustrative purposes, the first seven names are given in order from left to right: `ID`, `var3`, `var15`, `imp_ent_var16_ult1`, `imp_op_var39_comer_ult1`, `imp_op_var39_comer_ult3` and `imp_op_var40_comer_ult1`. Some of these words appear to be abbreviations for Spanish words like 'imp' for importe or amount. A non-comprehensive dictionary is discussed on the Kaggle forum (Andreu, 2015). On first glance, one can also infer that `imp_op_var39_comer_ult1` and `imp_op_var39_comer_ult3` are probably related and they are likely not related to `var3`. Looking at the distribution of the data can confirm these suspicions. We first distinguish some groups based on their name and broadly research each group in turn. Then we look into individual features that do not fit into a group in more depth. Aside from the clearly irrelevant ID variable, this comprises all the features. This is more practical than discussing all 370 features thoroughly.

3.1 Feature Groups

The sheer number of features in this dataset makes it hard to individually analyze and discuss each feature, so instead similar groups have been identified in the dataset, assisted by their corresponding names. In Table 1 it is visible on which substrings has been filtered. There is certain overlap between the groups as all 'Meses' features are also 'Num' features for example and all 'Delta' features are also either 'Imp' or 'Num' features.

Table 1: Variable Groupings

Substring	Example	Number (Raw)
'Num'	`num_var37_med_ult2`	87 (155)
'Ind'	`ind_var13_0`	46 (75)
'Saldo'	`saldo_medio_var5_hace3`	43 (71)
'Imp'	`imp_op_var39_comer_ult1`	21 (63)
'Delta'	`delta_imp_amort_var18_1y3`	3 (26)
'Meses'	`num_meses_var5_ult3`	8 (11)

The substring 'Num' likely stands for numeric variables. Typically, excluding the 'Delta' and 'Meses' subsets, these have values 0, 3, 6, 9 and further multiples of 3 as most common observations. These could indicate quarters for example. 0 and 3 tend to be most common and the distribution is usually unbalanced. The substring 'Ind' likely stands for indicator variables as all of them are 0 or 1. The distribution is usually unbalanced, but not consistently towards 1 or 0.

The substring 'Saldo' suggest the current actual amount on balance for certain financial products. A lot of these variables have an overwhelming amount of zero's, possibly being finished financial products or products that are not utilized in the first place. Other values are typically numeric and are of large scale like 6119500. Some values are also negative further providing evidence that this is a 'balance' type variable. Very similarly, 'Imp' can stand for Importe (Spanish for amount) and the distributions match this conjecture. Notable is that the scale tends to be smaller, so perhaps 'Saldo' is a sum of consecutive periods.

The substring 'Delta' signifies a difference of some kind, but the variable's distributions match ratio's, so it is possibly the ratio of an amount between a certain time period. Also here there is a massive imbalance towards the value 0. All 'Meses' variables are 'Num' variables, but are specifically taken as a subgroup, because they have a wildly different distribution. 'Meses' is Spanish for months and in the data these variables only take values 0, 1, 2 and 3. Some of the 'Meses' variables are even fairly balanced, which is fairly unique within this dataset.

3.2 Individual Features

These features all have a very short name and could be considered a group of their own. However upon closer inspection of at minimum two of them, it becomes apparent that they are clearly not related in the same manner as the previous groups of features. This individuality also means we need to consider each feature separately.

3.2.1 var3 (Nationality)

Table 2: var3

Maximum Value	238
Minimum Value	-999999
Unique Observation Count	208
Most common (count)	2 (74165)
2nd most common (count)	8 (138)
3rd most common (count)	-999999 (116)
4th most common (count)	9 (110)
5th most common (count)	3 (108)
Suspected Category	Categorical

`var3` is suspected to be nationality or country of residence. 208 Unique countries sounds like a plausible number that the bank can supply. 74165 observations are 2, which probably stand for the United States, the main market for this particular bank. A binary feature `var3_most_common` is made to put emphasis on this, which is 1 if `var3` is equal to 2 and 0 otherwise. -999999 likely encodes for missing values and another binary feature `var3_missing` accounts for this. After this feature is made the -999999 values are replaced by the most commonly occurring value 2.

3.2.2 var15 (Age)

Table 3: var15

Maximum Value	105
Minimum Value	5
Mean	33.212865
Unique Observation Count	100
Most common (count)	23 (20170)
2nd most common (count)	24 (6232)
3rd most common (count)	25 (4217)
4th most common (count)	26 (3270)
5th most common (count)	27 (2861)
Suspected Category	Numeric

`var15` is suspected to represents age as the minimum and maximum values are respectively 5 and 105, but the majority of the data is over 21. This data seems very biased to younger people and perhaps 23 is filled in if the age is unknown. For this reason we make another binary feature `var15_most_common`, which is 1 if `var15` is equal to 23 and 0 otherwise.

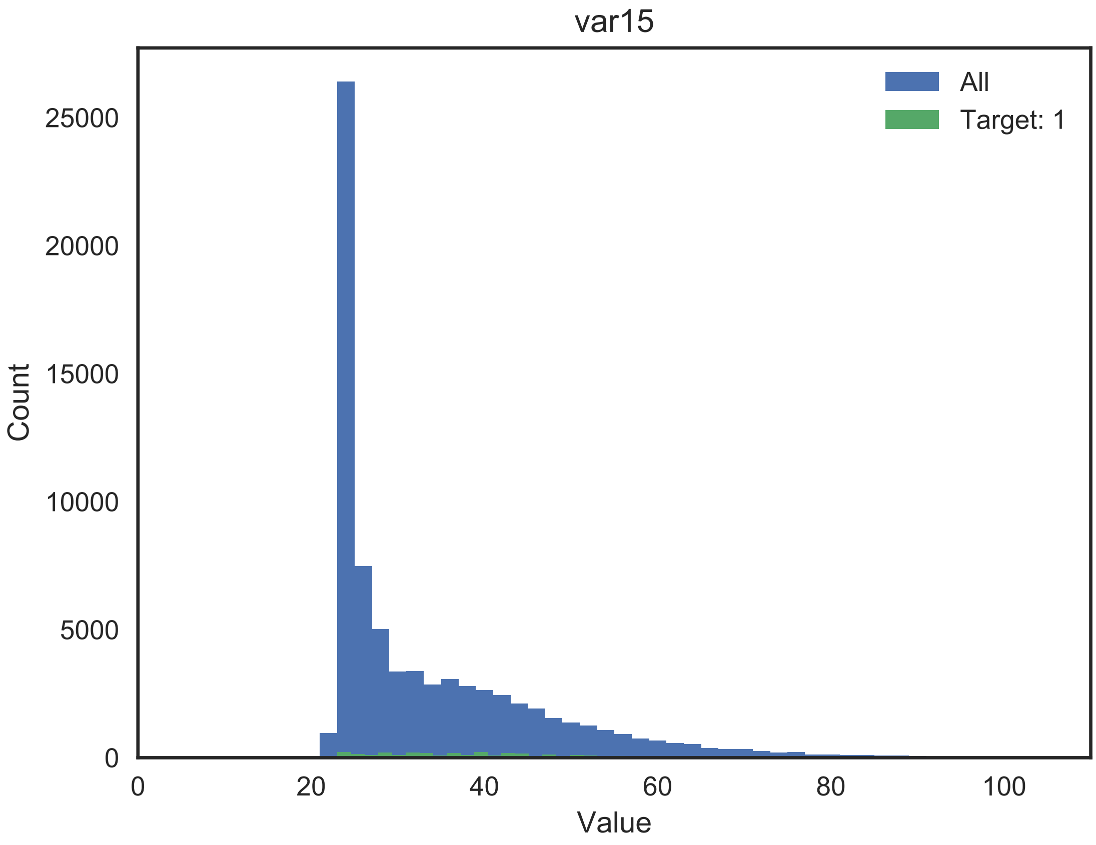


Figure 1: Histogram Age

3.2.3 var21

Table 4: var21

Maximum Value	30000
Minimum Value	0
Unique Observation Count	25
Most common (count)	0 (75152)
2nd most common (count)	900 (236)
3rd most common (count)	1800 (206)
4th most common (count)	4500 (96)
5th most common (count)	3000 (84)
Suspected Category	Numeric

Not all variables have an easy interpretation in this anonymized dataset and `var21` is a prime example. It is highly imbalanced and the non-zero values do not give off a likely meaning. Nevertheless it could still possibly be important, as it is clearly distinct from other variables.

3.2.4 var36

Table 5: var36

Unique Observation Count	5
Most common (count)	99 (30064)
2nd most common (count)	3 (22177)
3rd most common (count)	1 (14664)
4th most common (count)	2 (8704)
5th most common (count)	0 (411)
Suspected Category	Categorical

There is not much to be said about this particular variable, except that it is very likely categorical. One-hot encoding is applied such that it can be better understood by our classifiers. In the same vein, 99, which likely stands for missing values, is encoded properly.

3.2.5 var38 (Mortgage Value)

Table 6: var38

Maximum Value	22034740
Minimum Value	5163.75
Mean	117235
Unique Observation Count	57736
Most common (count)	117310.979016494 (14868)
2nd most common (count)	451931.22 (16)
3rd most common (count)	463625.16 (12)
4th most common (count)	288997.44 (11)
5th most common (count)	104563.80 (11)
Suspected Category	Numeric

This distribution ranges from high to low positive numbers with a very large number of the same value 117310. It is our conjecture that this represents the mortgage value of a customer or at least some kind of value indicating variable. If it is unknown for whatever reason the country average is instead filled in. For this reason we create a dummy variable `var38_most_common` that remembers this information. It is 1 if `var38` is 117310 and 0 otherwise. We visualize the distribution of the known values by making a histogram, excluding 117310 and cutting of the range at 350000, which excludes 1559 more observations, in Figure 2.

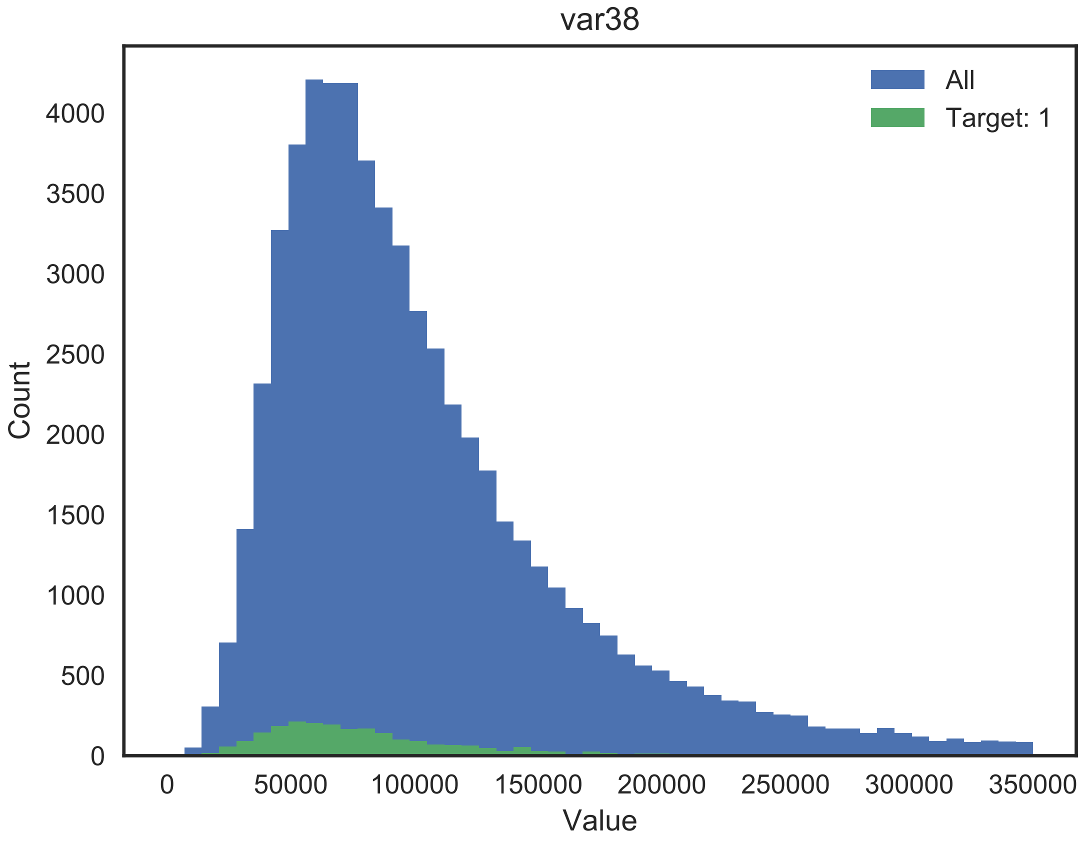


Figure 2: Histogram Mortgage

3.2.6 num_var4 (Number of Bank Products)

Table 7: num_var4

Maximum Value	7
Minimum Value	0
Unique Observation Count	8
Most common (count)	1 (38147)
2nd most common (count)	0 (19528)
3rd most common (count)	2 (12692)
4th most common (count)	3 (4377)
5th most common (count)	4 (1031)
Suspected Category	Numeric

According to dmi3kno (2015) this variable represents the number of bank products this client currently has with the bank. The distribution suggests that fewer people have multiple products with the bank and those tend to not be dissatisfied, giving this explanatory value. This is also intuitive, as clients investing multiple times probably have a good relationship with the bank and conversely the bank has more information to satisfy their client appropriately.

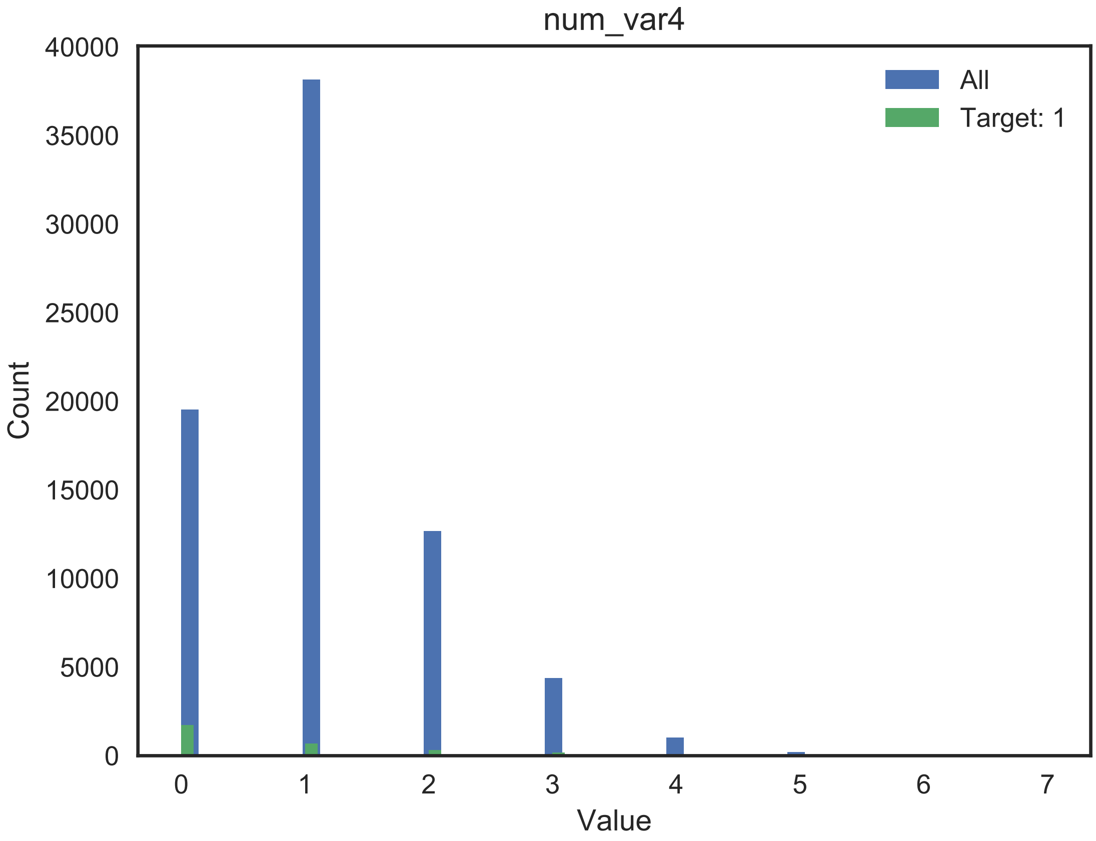


Figure 3: Histogram Bank Products

4 Preprocessing

4.1 Data Cleaning

On first glance the data is fairly clean, however there are definite problems for running certain features and observations through a machine learning algorithm. Figure 4 demonstrates the order that the cleaning steps were taken in. The left number represents the number of observations and the right the number of features.

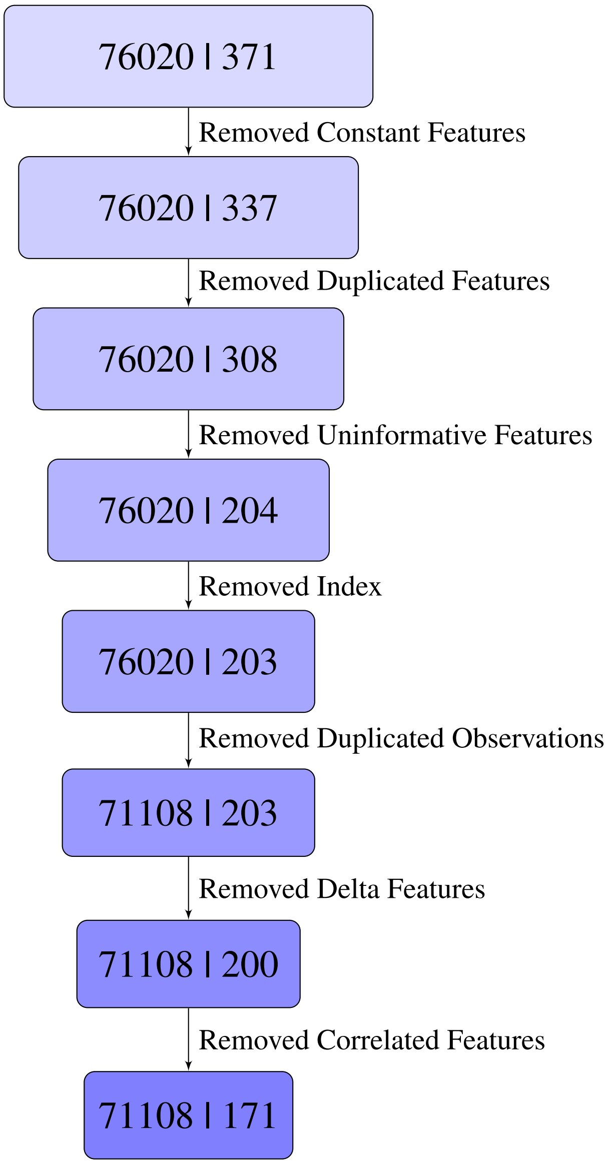

76020 | 371
Removed Constant Features
76020 | 337
Removed Duplicated Features
76020 | 308
Removed Uninformative Features
76020 | 204
Removed Index
76020 | 203
Removed Duplicated Observations
71108 | 203
Removed Delta Features
71108 | 200
Removed Correlated Features
71108 | 171

Figure 4: Cleaning Process

There are 34 features that are one constant value (zero for all checked cases). These features that do not vary at all cannot teach our classifier anything meaningful, so all features that have a standard deviation of zero are omitted. 29 features are exact duplicates of one another as well. We remove the redundancy by removing all but the first feature of a duplicate group. Note that the order of these cleaning procedures influences the number of removed features.

Some features have as little as 2 non-zero values. We categorize features that have very few non-zero values as uninformative for our classifier. We remove 104 features from the data by arbitrarily drawing a line at 100, effectively removing all features that have less than 100 observations that are not zero. Several of these were checked by eye and having a non-zero value was not very discriminatory regarding the Target value. For contextual purposes we show in Figure 5 what different choices would have meant for the dimension of the feature set. Finally we remove the index and in doing so assume the training data does appear randomly to us and not in a particular order.

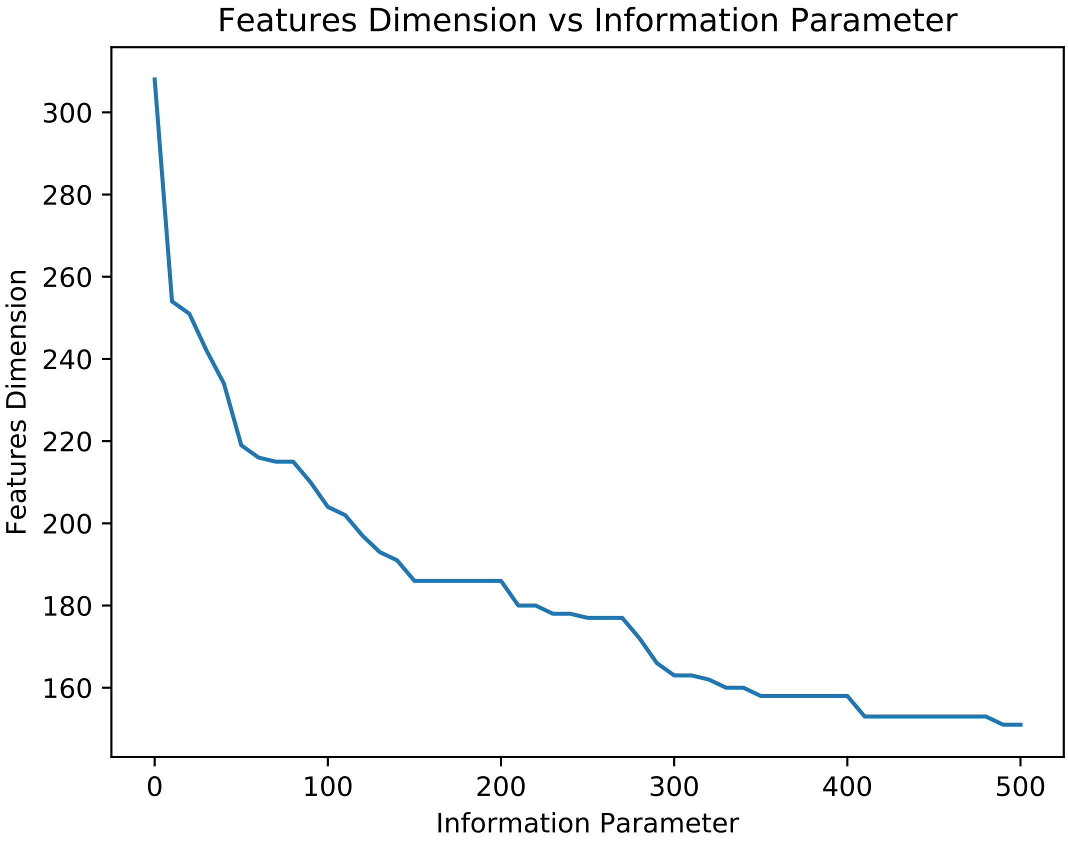


Figure 5: Information Parameter Range

There are also 4912 exact duplicates in features of the 76020 observations in the training data. Even worse, 109 of those duplicates have a differing target, which can only possibly 'confuse' a classifier. Although it can also be said that it will simply attach less confidence towards the importance of these observations, but to avoid the issue altogether these are all excluded from the dataset. The first of the duplicates that have the same target is kept around and 71108 observations remain after this procedure.

The three 'Delta' variables that are leftover are still relatively uninformative. They also do not appear as having great predictive power in any related literature studied. For these reasons they are also removed. They are the only variable group for which this is deemed appropriate.

4.2 Correlation

Correlation can be a powerful tool in machine learning. One of a pair of heavily correlated predictors can be removed without harming the predictive power of a model. Also very high absolute correlation with the target can indicate that this is an important variable. We apply the former and we look at the top features in the sense of being correlated with the target. We first apply normalization, where appropriate, to scale all variables between 0 and 1. We note that several of the features are hugely imbalanced and this type of scaling does not damage that. A feature X will become normalized feature Z with the following formula:

$$z_i = \frac{x_i - min(X)}{max(X) - min(X)} \quad \forall \quad i = 1, ..., length(X)$$

After applying this to `var15`, `var38`, 'Num's, 'Imp's and 'Saldo's, we plot the following Correlation matrix in Figure 6 and zoom in in Figure 7

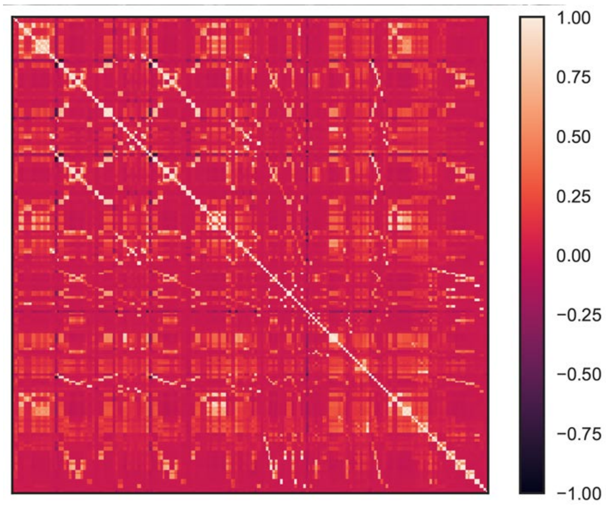


Figure 6: Correlation Matrix

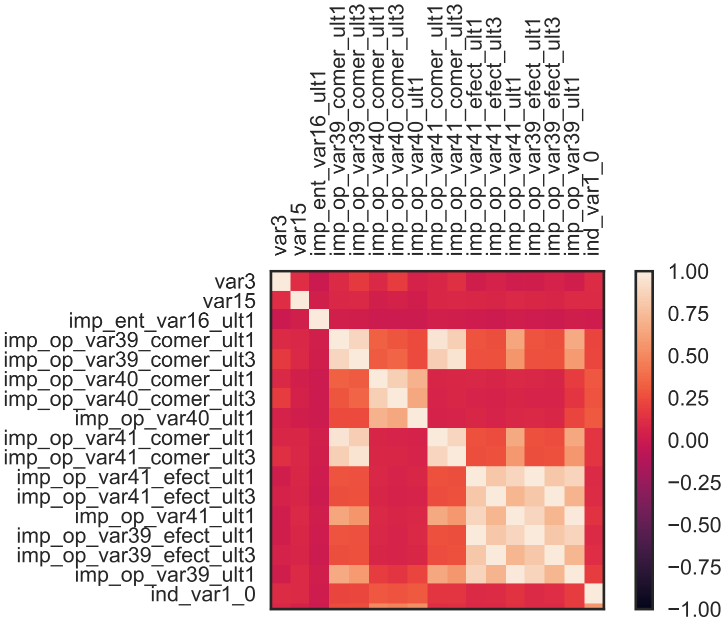


Figure 7: Correlation Matrix Zoom in

Here it is clearly visible that for instance imp_op_var39_comer_ult1 is heavily correlated with imp_op_var39_comer_ult3, further confirming that similar variable names are related, justifying the grouping of variables done earlier. Some features even have a correlation extremely close to 1 with each other like num_var1_0 and ind_var1_0 with 0.9988. All variable pairs that have higher absolute correlation than a conservative 0.99 are filtered out and the first of the pairs is deleted. An extension of this could delete one of the pair based on lower absolute correlation with the target. This removes 29 features and brings the cleaned dataset to 171 features total. It is visualized in Figure 8 what different correlation thresholds would mean for the dimensions of the feature set. We have been relatively conservative as the tree-based algorithms we use are adept at handling correlated features and the dataset is already comparatively small.

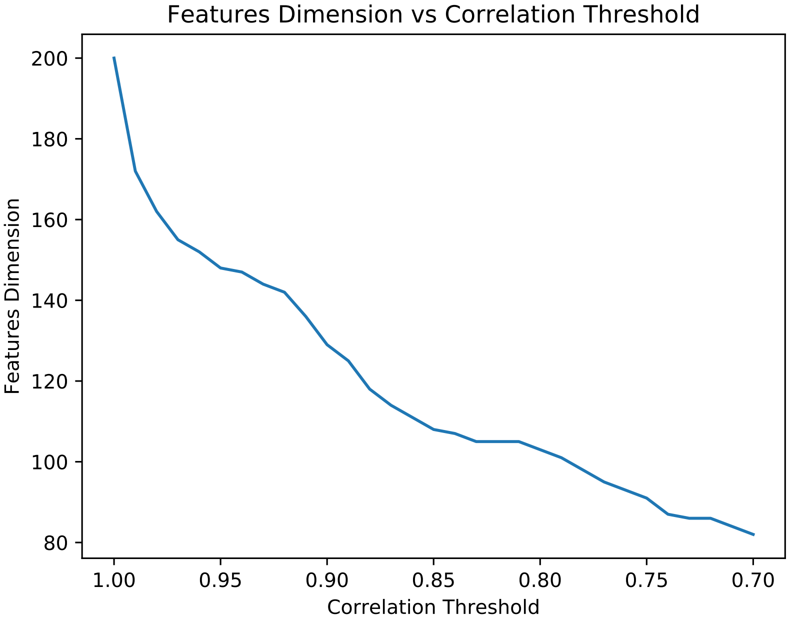


Figure 8: Correlation Thresholds

Furthermore some of the top correlations are showcased in Table 8. It is likely that these will be the important predictors later on.

Table 8: Top Correlations with Target

Negative	Positive
ind_var30 (-0.149756)	var36 (0.102401)
num_meses_var5_ult3 (-0.147362)	var15 (0.097341)
num_var30 (-0.137623)	ind_var8_0 (0.048493)
num_var42 (-0.134246)	imp_op_var41_efect_ult1 (0.030599)
ind_var5 (-0.133128)	ind_var8 (0.029038)

4.3 Feature Engineering

This dataset mostly lent itself for decoding what the existing data meant and making sure it is an appropriate format for machine learning algorithms. Nevertheless this led to the creation of some features, that were mentioned before, to accommodate expected oddities in the data. For instance the feature `var38_most_common` was created to ensure that the algorithm can recognize that this is likely a unique value outside of the numerical order that `var38` has. Outside of the ones mentioned before, the 'Meses' group was completely one-hot encoded, because they were small enough to warrant such an approach and seemed on first glance categorical.

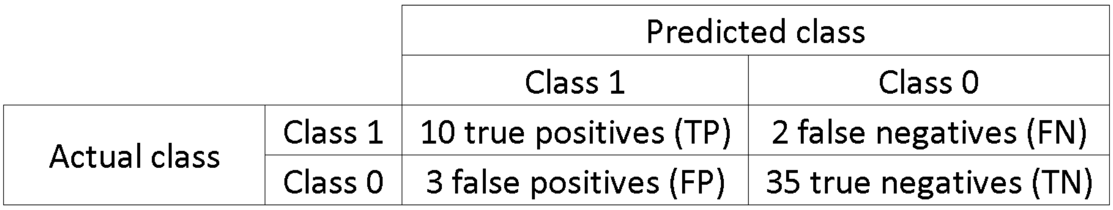
Furthermore it is noticeable that the data contains a lot of zero's. This usually stands for a lack of information or interaction and precisely that lack of knowing thy customer could hold predictive value for determining if the customer will possibly be dissatisfied in the future. For this reason `n0` was created, which simply sums the number of zero's that appear in the row. (dmi3kno, 2015) Special care is taken to not include the Target variable as this is a form of information leakage which disturbs the learning process of the models. Conversely a `n1` was created that should signify that the bank does know their customer and has a lot of interaction with them. This would be different, and not redundant in that way, than creating a variable that simply sums all spots in the row that are not zero. This brings the total dimensions of our training set to 71108 observations and 215 features to start the modeling process.

5 Modeling

5.1 Performance Measure

The performance measure of this Kaggle competition is the area under the receiver operating characteristic curve, AUROC or AUC for short. This metric deals well with the imbalance that is typical in churn prediction. (Burez and van den Poel, 2009) We build up to this concept by considering some simpler notions first. (Dernoncourt, 2015) If we consider a binary classifier we have four possible outcomes when we use it make a binary prediction and we call the collection of this a confusion matrix and an example is shown in Figure 9.

True negative: We predict 0 and the class is actually 0.
False negative: We predict 0 and the class is actually 1.
True positive: We predict 1 and the class is actually 1.
False positive: We predict 1 and the class is actually 1.



		Predicted class	
		Class 1	Class 0
Actual class	Class 1	10 true positives (TP)	2 false negatives (FN)
	Class 0	3 false positives (FP)	35 true negatives (TN)

Figure 9: Confusion Matrix Example

We define the following ratio's. The True Positive Rate corresponds to the proportion of positive data points that are correctly considered as positive, with respect to all positive data points. The higher the better, all else equal. The False Positive Rate corresponds to the proportion of negative data points that are incorrectly classified as positive, with respect to all negative data points. The lower the better all else equal.

$$\text{True Positive Rate (TPR): } \frac{\text{TP}}{\text{TP}+\text{FN}} \qquad \text{False Positive Rate (FPR): } \frac{\text{FP}}{\text{FP}+\text{TN}}$$

Binary classifiers usually predict with what probability they expect 1 to occur. The threshold where a high probability leading to predicting a 1 lies is an arbitrary decision. It is possible to consider every single possible threshold and plot the corresponding pair of TPR and FPR's of the resulting predictions in a ROC curve plot. An example is given in Figure 10. To obtain a single performance metric we can take the area under the ROC curve and effectively take into account both TPR and FPR. The baseline for this metric lies at 0.5, where we predict completely randomly. Anything performing worse can simply be inverted to do better.

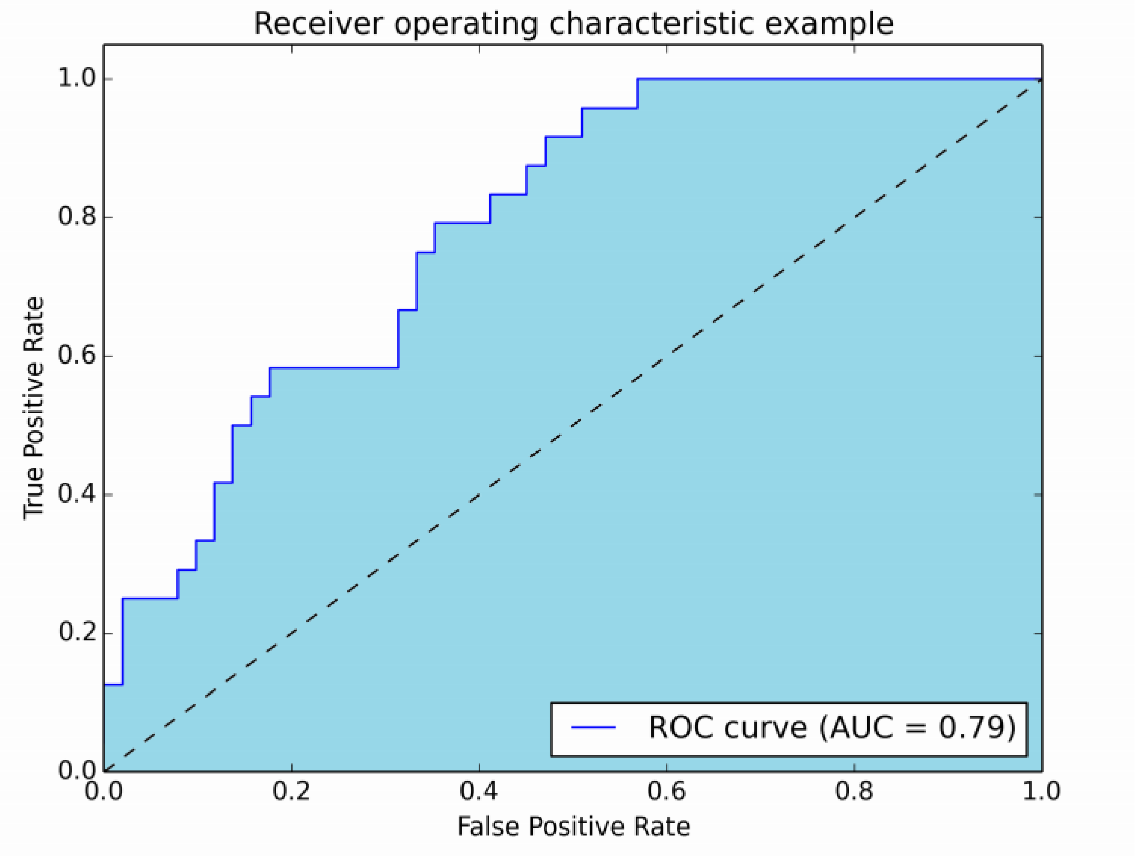


Figure 10: AUC Example

5.2 Solution Methods

We use 3 different methods to come to a solution: Logistic Regression, Random Forest from Scikit Learn (Pedregosa et al., 2011) and XGBoost (Chen and Guestrin, 2016). We apply 10-fold stratified crossvalidation on the train set to compare models internally, consequently fit the best model on the entirety of the train data and then predict labels for the test set and validate these on the Kaggle site for the final score. Stratification in this context refers to ensuring that each fold of the crossvalidation has a roughly equal distribution of classes as the original whole data set. Stratification tends to improve the accuracy of the crossvalidation as means of testing when dealing with imbalanced data. (Kohavi, 1995)

5.2.1 Logistic Regression

Logistic regression is a relatively 'simple' machine learning algorithm and we expect fast, but not great results. It tries to attach the best constant values to how features interact with the target, based on the train set, minimizing an error term. It then applies this same formula to the test data set. It is pursued here as a baseline in order to compare to more sophisticated models. For more details see (Bishop, 2006).

5.2.2 Decision Tree

The following two algorithms rely on multiple decision trees. This section roughly describes what a decision tree is. See (Breiman et al., 1984) for more details. To illustrate the concept of a decision tree we run a basic tree classifier implementation on our data and obtain Figure 11.

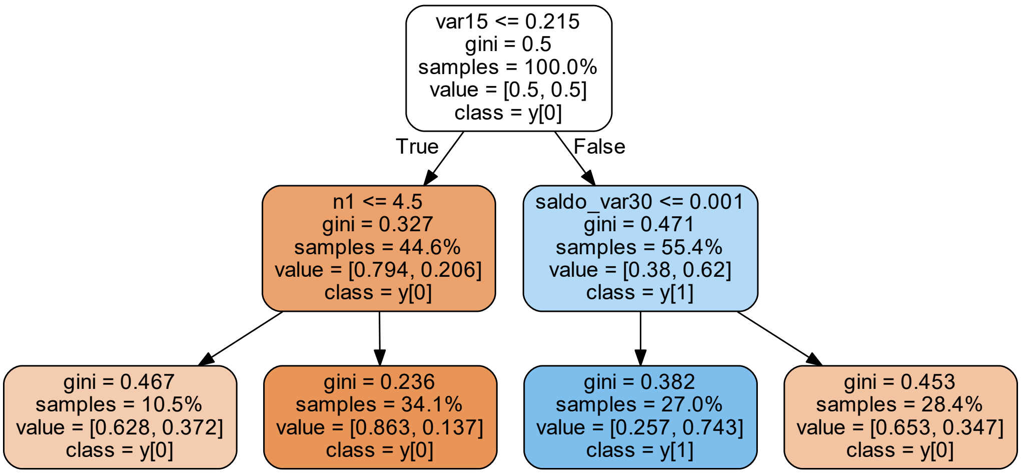


Figure 11: Decision Tree Example

The concept is fairly simple. If a prediction needs to be made, go from the top of the tree to the bottom. Go left at the top if `var15` is lower than 0.215, right otherwise. Repeat this process for all nodes until a leaf is reached and the prediction made corresponds to the class label of that leaf. This tree is constructed by at each depth level greedily finding the best feature to split on, maximizing information gain via the Gini impurity. Let p_0 be the proportion of observations that belong to class 0 and let p_1 be the proportion of observations that belong to class 1 out of all observations.

$$Gini = 1 - (p_0^2 + p_1^2)$$

This metric is an example of how to measure how pure a split is. It is more pure the lower it is with minimum 0 and maximum 0.5. A split is purer if in the branches the ratio of positive to negative examples is close to 0 or 1 or in other words the split is highly discriminatory. The maximum depth of this tree is set at 2 to keep it tractable and this effectively also keeps it from overfitting. A decision tree can otherwise perfectly match a training set, which does not generalize well.

5.2.3 Random Forest

The Random Forest model generates multiple decision trees. (Breiman, 2001) Only a subset of predictive features is now considered during each split that is randomly selected. The decision trees lead to one single prediction together by averaging all the predictions they give individually.

The parameters of a Random Forest model are really important and need to be tuned appropriately. N_ESTIMATORS determines the number of trees in the forest. MAX_FEATURES controls the maximum random amount of features to consider when determining a best split during the algorithm. MAX_DEPTH limits the depth of a tree in the random forest. MIN_SAMPLES_SPLIT determines the minimum amount of observations that need to be in a node for it to be considered for splits. MIN_SAMPLES_LEAF constrains that leaf nodes have at least this number of observations. N_JOBS controls the number of processors. Trees in a random forest can be made in parallel, so the more cores working, the less computation time needed. CLASS_WEIGHT can be set to 'balanced' to deal with imbalanced datasets.

5.2.4 XGBoost

XGBoost is a method where the outcome is formed by an combination of multiple trees. The trees are build iteratively. Every time a new tree is built it focuses on parts where the previous ones make mistakes, by assigning higher weights to these instances. This stands in contrast to Random Forest, where trees are made independently from each other.

The parameters are again of grave importance and there are even more. N_ESTIMATORS and MAX_DEPTH are the same as with the Random Forest model. LEARNING_RATE is a parameter that controls the speed and preciseness of the model. The lower it is, the more accurate your model becomes, but the more rounds it needs to converge. It represents a constant times how much the next tree built affects current predictions. SUBSAMPLE stands for the fraction of observations to be sampled randomly. COLSAMPLE_BYTREE denotes the fraction of features to take into consideration during the random sampling for each tree. MIN_CHILD_WEIGHT defines the minimum sum of weights of all observations required in a child. SCALE_POS_WEIGHT is a parameter to combat imbalancedness. XGBoost directly avoids overfitting by promoting simplicity of models in the objective via regularization, unlike Random Forest that only limits the way trees can grow by imposing restraints. (Chen, 2014)

$$\text{min } Obj(\Omega) = \text{Training Loss Function } + \text{Regularization}$$

$$\text{min } Obj(\Omega) = \sum_{i=1}^{n} l(y_i, \hat{y_i}) + \alpha \sum_{i=1}^{k} |w_i| + \lambda \sum_{i=1}^{k} w_i^2 + \gamma T$$

REG_ALPHA is a parameter for l1 regularization. LAMBDA is a parameter for l2 regularization. GAMMA is a regularization parameter that multiplies itself with the number of leaves. This regularization is in place to penalize the objective for building overtly complex trees to avoid overfitting.

6 Results

6.1 Tuning RF

It is difficult to determine what set of parameters is optimal for a specific problem. We utilize the GridSearchCV() (Buitinck et al., 2013) function with 5-fold stratified crossvalidation and scoring option area under the ROC curve to come to an answer. This basically boils down to trying all possible combinations of a set of predetermined parameter spaces. The parameter spaces and results of the grid search are shown in Table 9. These parameter spaces were seen as the optimal tradeoff between computation time and accurately tuning the model.

Table 9: GridSearchCV()

Parameter	Range	Best Value Combination
N_ESTIMATORS	[100, 500, 1000, 2000, 3000]	2000
MAX_FEATURES	['sqrt', 'log2']	'sqrt'
MAX_DEPTH	[None, 5, 20]	20
MIN_SAMPLES_LEAF	[10, 30, 50, 100]	50

With the aforementioned best parameter set we fit the model on the training data and make a plot of how relevant a specific feature is in Figure 12. These importances represent the total Gini impurity decrease weighted by the probability of reaching a node containing this feature, averaged over all trees. We can be pleased to see that several of the features we created like `var15_most_common` seem to be significant.

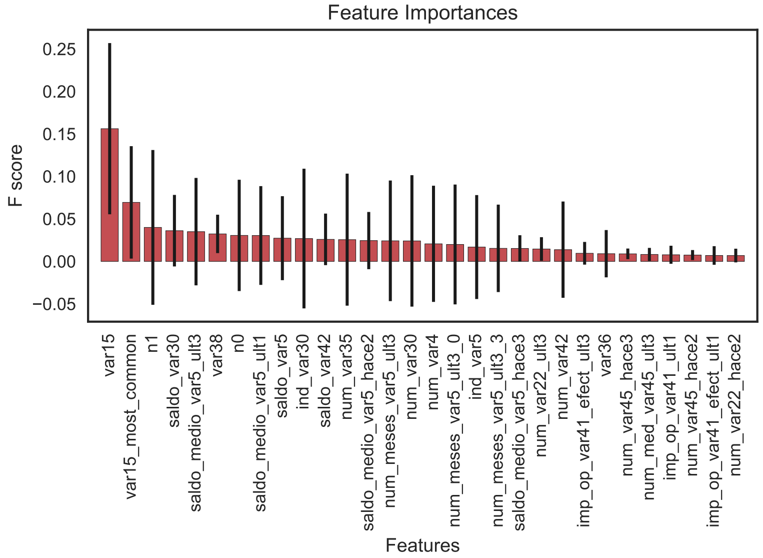


Figure 12: Feature Importance

We also plot one of the many trees the forest model creates in Figure 13, namely the very first one and we can see it is considerable in size.

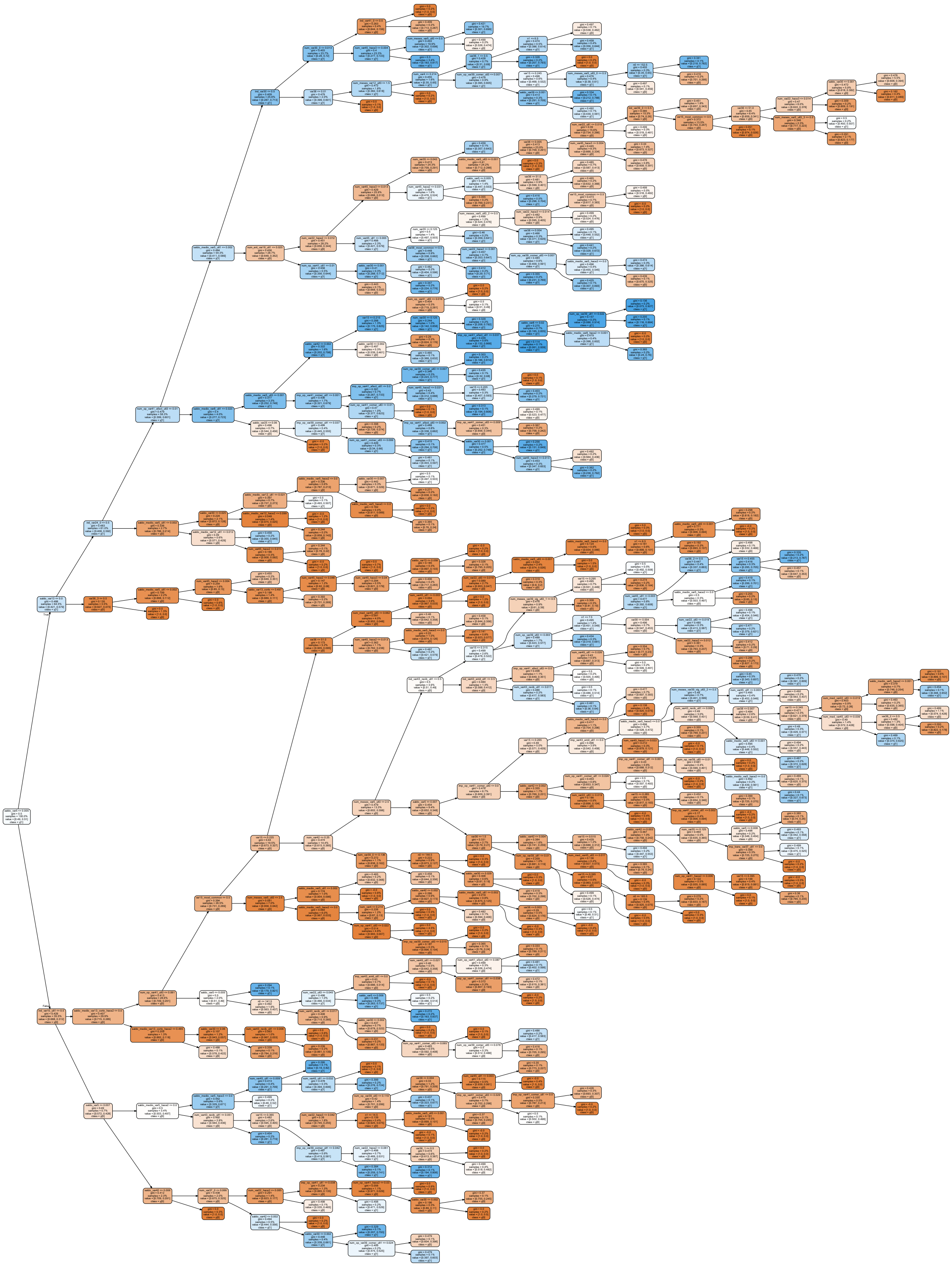


Figure 13: Tree 1

6.2 Tuning XGBoost

The numerous parameters in XGBoost make it intractable to simply apply a grid search and instead we utilize RandomizedSearchCV(). Instead of trying all possible combinations, this function samples a predetermined amount of sets of parameters in the parameter spaces specified. Instead of set values in the parameter space we typically have distributions. We run this once for 75 times and using the results of the first try, narrow the parameter spaces and run it again for 50 times. We first apply 5-fold stratified crossvalidation to compare and the second time we make it more precise by using 8-fold stratified crossvalidation. The results are shown in Table 10. We have chosen these starting ranges as broad ranges around recommended starting parameters in related work and here. (Jain, 2016)

Table 10: RandomizedSearchCV()

Parameter	Range 1	Best Value 1	Range 2	Best Value 2
N_ESTIMATORS	[100, 2000]	245	[100, 1000]	917
MAX_DEPTH	[3, 9]	6	[4, 6]	5
LEARNING_RATE	[0.01, 0.21]	0.04864	[0.001, 0.076]	0.01632
SUBSAMPLE	[0.6, 0.9]	0.85635	[0.75, 0.85]	0.75183
COLSAMPLE_BYTREE	[0.6, 0.9]	0.79057	[0.75, 0.85]	0.82017
MIN_CHILD_WEIGHT	[1,5]	1	1	1
REG_ALPHA	[0,0.1]	0.02276	[0,0.05]	0.04970
GAMMA	[0,0.2]	0.10552	[0,0.15]	0.038668

With the aforementioned best parameter set we fit the model on the training data and make a plot of how relevant a specific feature is in Figure 14. Like the Random Forest model we can see some made features are relevant. There is also a significant overlap between what is important, which is encouraging.

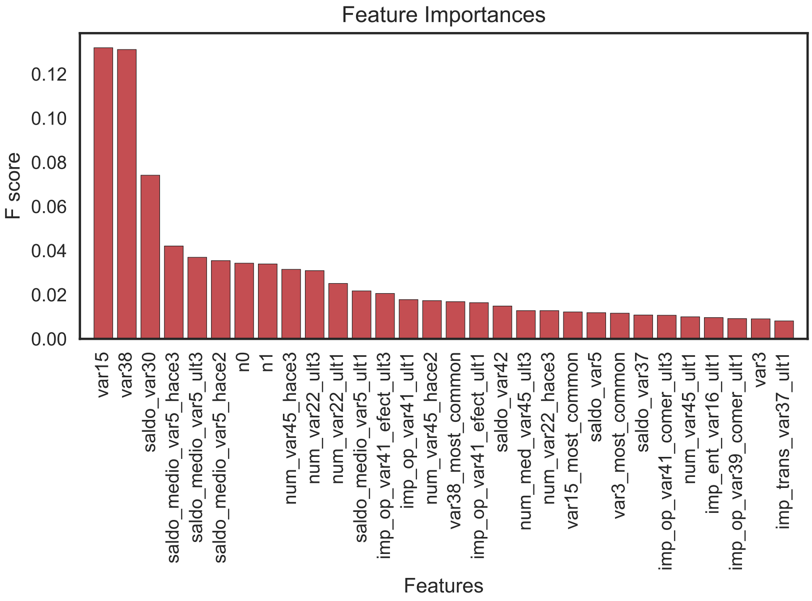


Figure 14: Feature Importance

We also show one of the many trees the XGB model creates in Figure 15, namely a randomly selected tree, which happens to be the 300th tree.

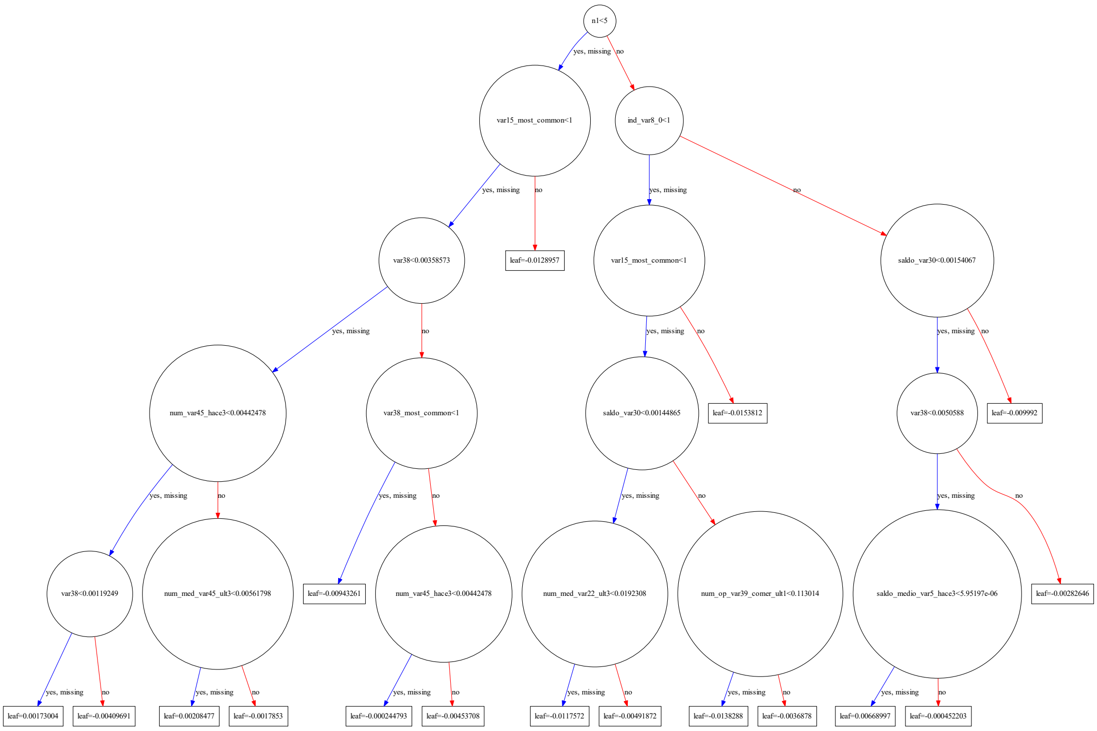


Figure 15: Tree 300

6.3 Results

There are a number of different performance indicators. Our local stratified 10-fold crossvalidation procedure was used to tune all parameters and select features. Note that a single fold takes up to 10 minutes at most and several operations can be ran in parallel. Kaggle also has a private and public leaderboard score. Approximately half of the submitted test set is used for the private leaderboard score and the other for the public leaderboard score. Since the competition had already completed the difference in these scores is superfluous for us. Nevertheless the complete results are showcased in Table 11.

Feature set 1 refers to only the top 30 features that have the highest importance scores according to the XGBoost model that uses all features. Feature set 2 refers to feature set 1 except some variables have been manually removed that seeemed obviously correlated like `'saldo_medio_var5_hace3'` and `'saldo_medio_var5_hace2'`. The one that was deemed most important by the model was kept. Feature set 3 refers to feature set 2 except adding some of the variables that were highly correlated with the target as seen in section 4.2. For a complete list of all involved features see Appendix A. Note that a single fold takes up to 10 minutes at most and several operations can be ran in parallel.

Table 11: Results

Model	Features	Local	Public	Private
Logistic Regression	All	0.80561 (+/- 0.02337)	0.804893	0.786949
Random Forest	All	0.82739 (+/- 0.01672)	0.822562	0.803286
XGBoost	All	0.84061 (+/- 0.01672)	0.837524	0.823152
XGBoost	Set 1	0.84035 (+/- 0.01763)	0.836194	0.822334
XGBoost	Set 2	0.83931 (+/- 0.01954)	0.836109	0.821644
XGBoost	Set 3	0.83898 (+/- 0.02624)	0.836397	0.821777

It is very peculiar that the private leaderboard score is consistently lower. This indicates a leaderboard shakeup, where the train set is not representative enough of the test set. For reference the top public leaderboard score is 0.845325 and the top private leaderboard score is 0.829072. Our scores are in far reach of that, however we did not endeavor to be at the top of the leaderboard and kept the test set like a secret until the end. All data exploration was done on solely the training data and normalization for instance was done using just the observations in the training data. We consider this more realistic as you can more truly validate your conclusions with completely new data, as opposed to the information spillover that happens if we had not done so. In a business sense a singular new client tends to appear, instead of an entire population that can for instance be appropriately normalized.

7 Conclusion

This paper researched how to preemptively understand if customers of Santander will be dissatisfied using Machine Learning. A semi-anonymized dataset, to protect the privacy of the customers, provided difficulties in asserting what could be relevant or not, especially in light of a huge feature set. However a thorough data analysis discerned the meaning and interpretation of several features. A Python implementation utilized the Logistic Regression, Random Forest and XGBoost algorithms, carefully tuned, in order to lead to predictions. Further research could for example employ different solution methods, apply more feature engineering or combine several models instead of trying singular models. More specifically they can increase the computation time that goes into tuning and for example make the correlation filtering dependent on the correlation with the target.

A Appendix Feature Sets

Set 1: var15, var38, saldo_var30, saldo_medio_var5_hace3, saldo_medio_var5_ult3, saldo_medio_var5_hace2, n0, n1, num_var45_hace3, num_var22_ult3, num_var22_ult1, saldo_medio_var5_ult1, imp_op_var41_efect_ult3, imp_op_var41_ult1, num_var45_hace2 var38_most_common, imp_op_var41_efect_ult1, saldo_var42, num_med_var45_ult3, num_var22_hace3, var15_most_common, saldo_var5, var3_most_common, saldo_var37, imp_op_var41_comer_ult3, num_var45_ult1, imp_ent_var16_ult1, imp_op_var39_comer_ul var3 and imp_trans_var37_ult1

Set 2: var15, var38, saldo_var30, saldo_medio_var5_hace3, n0, n1, num_var45_hace3, num_var22_ult3, imp_op_var41_efect_ult3, var38_most_common, saldo_var42, num_med_var45_ult3, var15_most_common, saldo_var5, var3_most_common, saldo_var37, imp_ent_var16_ult1, imp_op_var39_comer_ult1, var3, imp_trans_var37_ult ind_var8_0, num_meses_var5_ult3, num_meses_var39_vig_ult3_1, num_var4 and var36

Set 3: var15, var38, saldo_var30, saldo_medio_var5_hace3, n0, n1, num_var45_hace3, num_var22_ult3, imp_op_var41_efect_ult3, var38_most_common, saldo_var42, num_med_var45_ult3, var15_most_common, saldo_var5, var3_most_common, saldo_var37, imp_ent_var16_ult1, imp_op_var39_comer_ult1, var3, imp_trans_var37_ult ind_var8_0, num_meses_var5_ult3, num_meses_var39_vig_ult3_1, num_var4, var36, ind_var30, num_var42, and ind_var5

References

Andreu (2015). Predicting banking customer satisfaction. `https://www.kaggle.com/c/santander-customer-satisfaction/discussion/19291#110414`. Accessed: 10-15-2017.

Bishop, C. (2006). Logistic regression. In M. Jordan, J. Kleinberg, and B. Scholkopf (Eds.), *Pattern Recognition and Machine Learning*, pp. 205–207. Springer-Verlag New York.

Breiman, L. (2001). Random forests. *Machine Learning 45*(1), 5–32.

Breiman, L., J. Friedman, C. J. Stone, and R. Olshen (1984). *Classification and Regression Trees*. Wadsworth International Group.

Buckinx, W. and D. van den Poel (2005). Customer base analysis: partial defection of behaviourally loyal clients in a non-contractual fmcg retail setting. *European Journal of Operational Research 164*(1), 252–268.

Buitinck, L., G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux (2013). API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pp. 108–122.

Burez, J. and D. van den Poel (2009). Handling class imbalance in customer churn prediction. *Expert Systems with Applications 36*(3), 4626–4636.

Chen, T. (2014). Introduction to boosted trees. `https://homes.cs.washington.edu/~tqchen/pdf/BoostedTree.pdf`. Accessed: 10-25-2017.

Chen, T. and C. Guestrin (2016). Xgboost: A scalable tree boosting system. arxiv: Cs.lg 1603.02754v3.

Clemes, M. D., C. Gan, and D. Zhang (2010). Customer switching behaviour in the chinese retail banking industry. *International Journal of Bank Marketing 28*(7), 519–546.

Colgate, M., K. Stewart, and R. Kinsella (1996). Customer defection: a study of the student market in ireland. *International Journal of Bank Marketing 14*(3), 23–29.

Dernoncourt, F. (2015). What does auc stand for and what is it? `https://stats.stackexchange.com/questions/132777/what-does-auc-stand-for-and-what-is-it`. Accessed: 10-15-2017.

dmi3kno (2015). Exploring features. `https://www.kaggle.com/cast42/exploring-features`. Accessed: 10-15-2017.

Jain, A. (2016). Complete guide to parameter tuning in xgboost. `https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/`. Accessed: 10-25-2017.

Kohavi, R. (1995). A study of crossvalidation and bootstrap for accuracy estimation and model selection. In *Proceedings of IJCAI 1995*.

Kumar, A. (2015). Boosting customer satisfaction with gradient boosting. https://cseweb.ucsd.edu/classes/wi17/cse258-a/reports/a079.pdf. Accessed: 10-16-2017.

Mozer, M., R. Wolniewicz, D. Grimes, E. Johnson, and H. Kaushansky (2000). Predicting subscriber dissatisfaction and improving retention in the wireless telecommunications industry. *IEEE Transactions on Neural Networks 11*(3), 690 – 696.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research 12*, 2825–2830.

Santander (2015). Santander customer satisfaction. https://www.kaggle.com/c/santander-customer-satisfaction. Accessed: 10-15-2017.

Silva, L., G. Titericz, D. Efimov, I. Tanaka, D. Barusauskas, M. Michailidis, M. Muller, D. Polat, S. Semenov, and D. Altukhov (2016). Solution for santander customer satisfaction competition, 3rd place. https://github.com/diefimov/santander_2016/blob/master/README.pdf. Accessed: 10-16-2017.

Wang, S. (2016). Predicting banking customer satisfaction. https://shuaiw.github.io/assets/data-science-project-workflow/santander-customer-satisfaction.pdf. Accessed: 10-16-2017.

Xie, Y., X. Li, E. Ngai, and W. Ying (2009). Customer churn prediction using improved balanced random forests. *Expert Systems with Applications 36*(3), 5445–5449.

Yooyen, T. and K.-C. Ma (2016). Csci 567 spring 2016 mini-project. https://markcsie.github.io/documents/CSCI567Spring2016Project.pdf. Accessed: 10-16-2017.