Churn analysis

TABLE OF CONTENTS

1. Churn Prediction

In order to predict a binary variable, several methods can be applied. In the following section are presented the method which I used to predict if a customer is likely to churn. Then it will be illustrated the results obtained with these different classifiers and it will be picked the best model to put in production according with its time complexity and robustness of the estimates.

1.1 Methods

The models that has been utilized are :

1. Logistic Regression
2. Naive Bayes Classifier
3. Support Vector Machine
4. Decision Tree
5. Random Forest

In order to pick the best among those, I considered as measure of performance the accuracy of predictions, which is simply a measure that compare the results obtained by the model with the truth outcome of the dataset. As reminder, the baseline accuracy, i.e the ratio between churners and not churners, is 0.8305263. That means that our models should considerably have an higher accuracy, otherwise simply predicting that all customer will not churn will produce the same score, without any model trained.

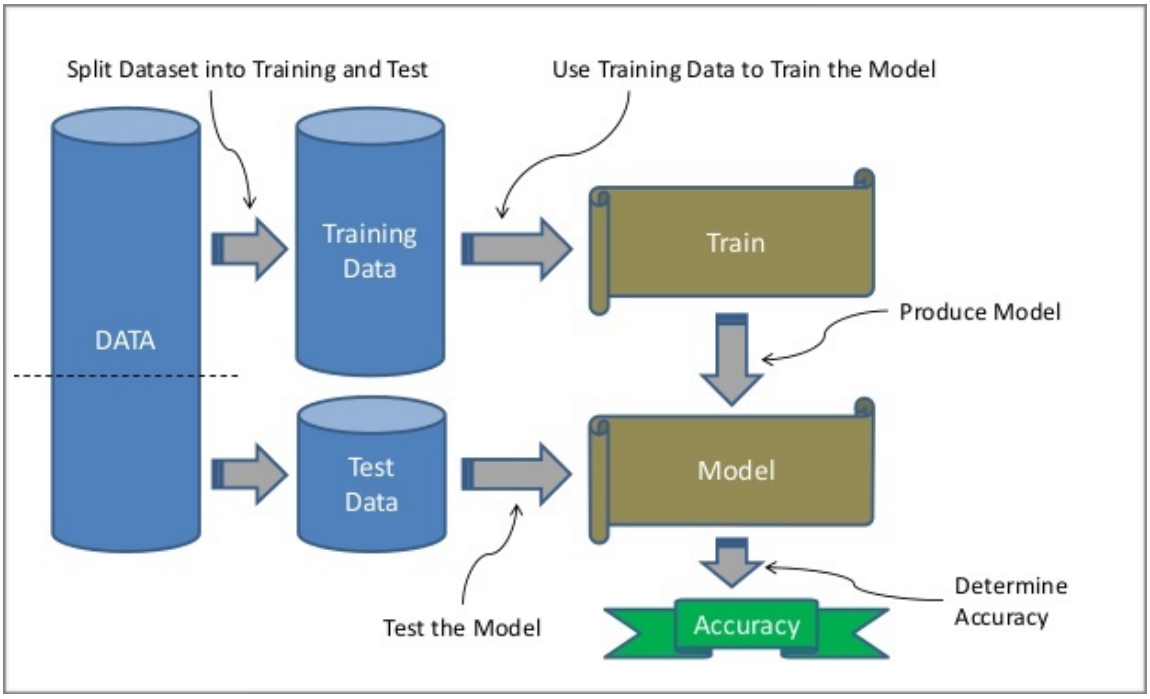
1.2 Approach

In order to evaluate the models, it is necessary to follow the procedure illustrated in the next image.

To get a first impression of how the models performs, I simply split our dataset in two parts, maintaining the proportion between the two classes:

- 80 % is used to train the models
- 20 % is used to test the models

Collecting the first results, has made it possibile to see how decision trees strongly outperform the other methods. From my personal knowledge and from that results, I proceeded the analysis optimizing this algorithm trough Random Forest, which use an ensemble of decision tree to predict the most common class.



1.3 Results

It's important that results are reproducible. This is made possible by setting the seed carefully. In that analysis I choose (54321) as random seed. This is used before each train / test splits of the dataset.

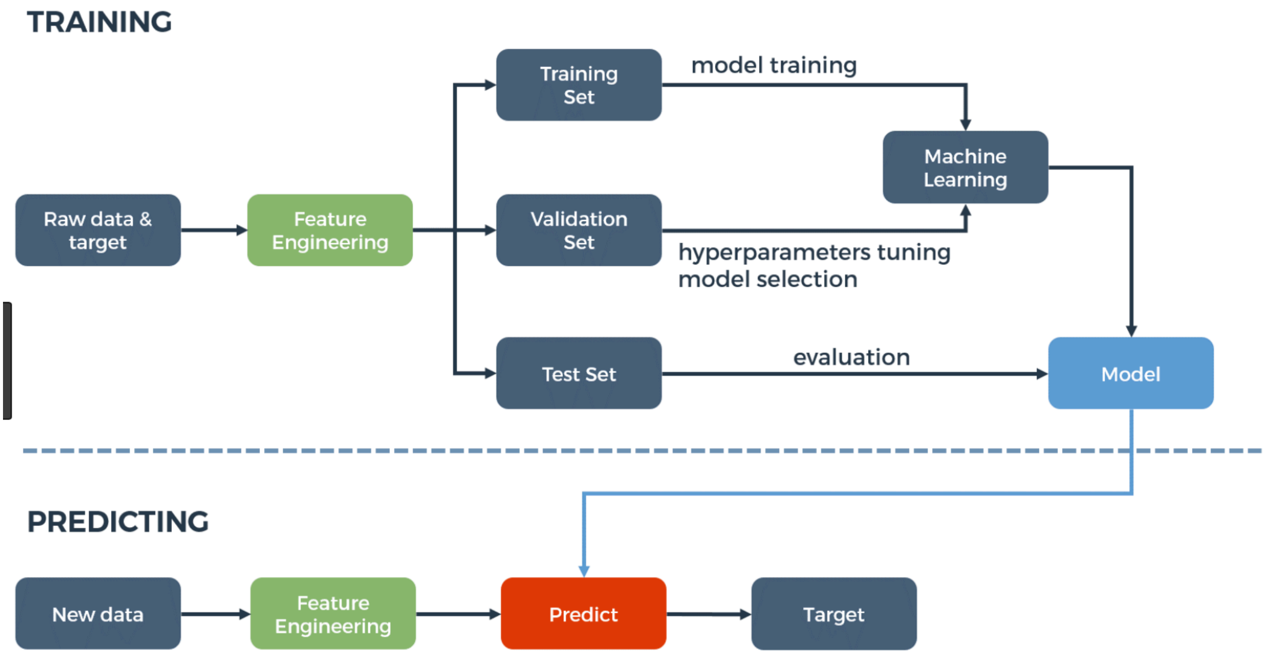
Here the results are calculating as follows:

1) set.seed(54321)
2) Select all variables except all Charge_variables (collinearity)
3) Split train 80 % - test 20%: Stratified random sampling for proportion of classes
4) Calculate the accuracy of each model
5) Pick the best best one
6) Tune the parameters
7) Calculate accuracy with Cross Validation techniques:
   - Bootstrap 100 times
   - 10 K-fold repeated 10 times

Regarding the Cross validation techniques, I picked this two because provide best results in term of bias and variance of the estimates. I tried also to random splitting train-test multiple times (100) and K-fold CV (K = 10) and the results are more unstable.

A logistic regression model is used as a benchmark to determined this results. For the other method I did not test the results according to Bootstrap resampling and K-fold CV because the difference with decision tree in prediction accuracy was to huge. It would be a waste of time. That manual approach is preferred to classical Machine Learning pipeline used in the package "caret" (classification and regression tree by Max Kuhn), that usually split the data into three different dataset :

(train - validation - test). This may be preferred in case of huge dataset, where the validation is used to tune hyper parameters and the best model is applied to a final unique test set. Considering the dataset provided (3333 customers) we may require as many data as possible for fitting the models and more importantly I wanted a results which is calculated multiple times to improve robustness. However with more data and more computer power (paraller processing) I would choose the caret pipeline to predict on a new test, because its much easier to train more advanced models. The "caret" procedure is illustrated in the following image.



Here are presented the results.

MODELS	*ACCURACY*	*BOOTSTRAP*	*K-FOLD (10*10)*
Baseline	**0.8305**		
LOGISTIC REGRESSION	-	**0.859190450** / sd: 0.006	**0.86081890** / sd: 0.012
SVM Linear	**0.855000**	-	-
SVM Poly	**0.86700**	-	-
SVM Radial	**0.85900**	-	-
NAIVE BAYES	**0.8695652**	-	-
NAIVE BAYES Laplace = 1/2	**0.8710645**	-	-
DECISION TREES	**0.9415292**	-	-
RANDOM FORESTS	**0.9565217**	-	-
RANDOM FORESTS Tuned	**0.964018**	**0.9501866 / sd: 0.006**	**0.9540343 / sd: 0.01059717**

The best model is depicted in green.

Tuning hyperparameter are :

- Num trees = 300
- M_try = 8
- Min_node_size = 2

With a Random Forest tuned in that way we achieve 95 % average accuracy. This is the best model to use it considering predictive power.

2. Clustering

2.1 Method

To perform clustering in practice, we try several different choices, and look for the one with the most useful or interpretable solution. With these methods, there is no single right answer any solution that exposes some interesting aspects of the data should be considered.

Regarding the algorithm I choose Hierarchical Clustering and K-Means. Giving as input all the variable that refer to “usage” the result was pretty bad in each method, using different configuration settings.

To manage this situation I created new features and I tried different combinations of them trying to obtained greatest results. However I did not find very interesting insights.

Between the two methods, Hierarchical Clustering gave slightly better results.

Variables used:

1) Total_Mins = Day_Mins + Eve_Mins + Night_Mins
2) Total_Calls = Day_Calls + Eve_Calls + Night_Calls
3) Intl_Mins

I discard from the analysis:

1) Account Length
2) Customer Service Calls
3) International Calls
4) VMail_Message

The reason of discarding that predictors is: this variables didn't improve the interpretability of the clustering procedure.

An in depth analysis should be carried to gather informations from this variables.

——- Linkage

According to results of *“agnes”* function in the library *“purr”* I should use Ward linkage. Moreover, If I see my customer segments as types and more or less spherical shapes with compaction(s) in the middle I'll choose Ward's linkage method , but never single linkage method.

——— Number of clusters

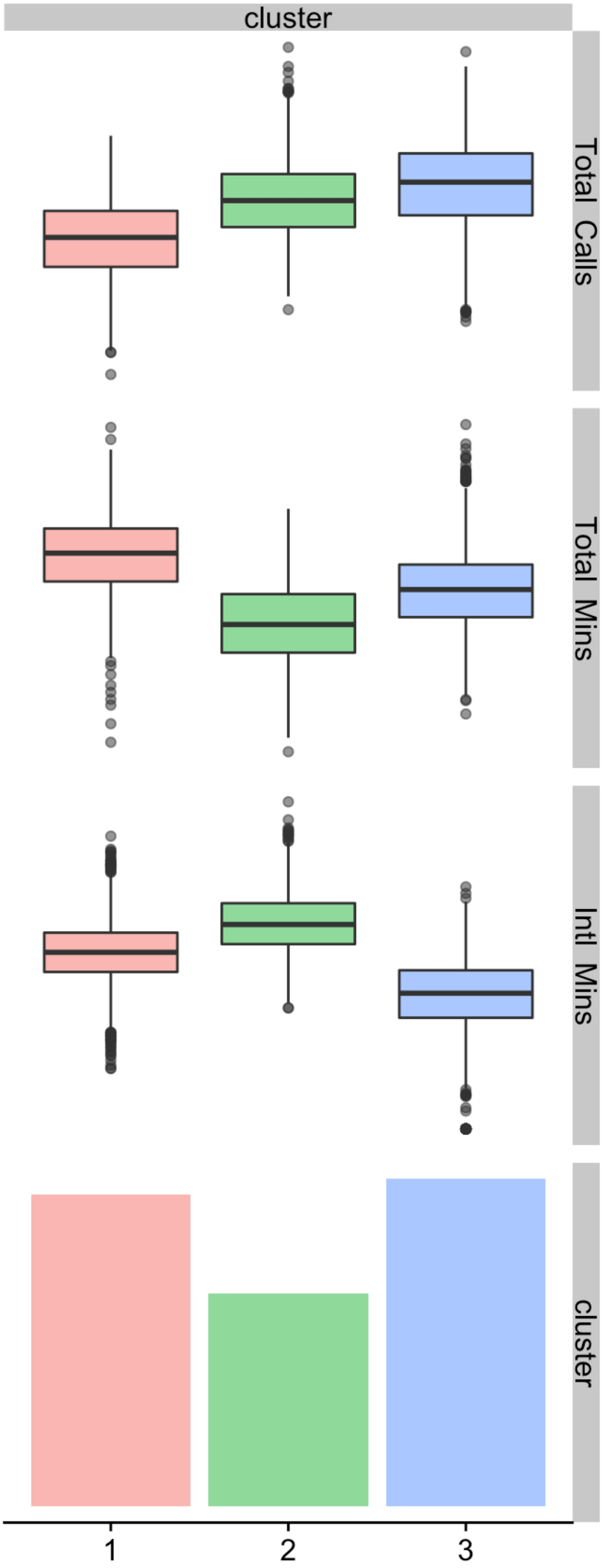
In order to choose the number of clusters both a silhouette analysis for K = 1…..10 and the Dunn index for the same number of clusters has been computed. Silhouette is a measure of connectedness, while the Dunn Index is the ratio of the smallest distance between observations not in the same cluster to the largest intra-cluster distance.

Using these variables the result are not very well defined, thus I tried with some visualization to find the correct number of clusters.
As a result I choose to clustering our customers in 4 different groups.

2.2 Results

Here are presented the result of the whole procedure.
As we can see we segment our customers into 3 different groups.

.



Group 1 (red):

[Long calls]

In average they usually spend more minutes at the phone even if their number of calls is the lowest.

Group 2 (green):

[Normal calls]

In these group there are the users that spends more time in international calls, but regarding the number of mins in calls on average they are the lower group.

Group 3(bue):

[Short calls]

Customers that have on average the highest number of calls, but they don’t use a lot international services, and comparing to group1 they spend less minutes in each call.