Final Project Economics For Data Science

Federico Ravenda[1]

[1]**Università degli Studi di Milano Bicocca, CdLM Clamses - STAT - matricola nr. 829449**

CONTENTS

1 **EDA** 1
2 **Data Entry Mistakes** 3
3 **Missing Values** 3
4 **Clustering** 3
5 **Customer Network** 6
6 **Matching** 7
7 **Churner Prediction** 9
7.1 Bonus Track - UnderSampling, OverSampling and SMOTE 10
8 **Profit Curve & Marketing Campaign** 10

INTRODUCTION

The prediction of new churners has become a critical issue for service providers around the world. To keep pace with the market, substantial investments were made to develop new anti-churn strategies, including machine learning models increasingly used in this field.
In this discussion we will use a pool of techniques seen during the Economics For Data Science class to analyze datasets which contain informations on museums memberships in different Italian municipalities and to predict new churners.
At the following link https://github.com/Fede-stack/E4DS can be found the github repository of the project with all the files useful for the analysis.

1 EDA

For this work, we had the availability of 3 different datasets containing information on museums, visitors and the memberships they have subscribed to. All three datasets share a common variable, the *unique identification code* of each customer, which will be used to merge the datasets (we decide to use a *inner join* to merge datasets so that there are only those customers who occur in all three datasets) The most important variables of the merged dataset are following summarised:

- **codcliente** *factor*. Id consumers
- **data_inizio** *Date*. Starting date of the card's subscription (validity 1 year)
- **importo** *numeric*. Price paid for the subscription.
- **totale_importo** *numeric*. sum of importo variable for each consumer

- **sconto** *factor*. Type of discount applied for each subscriptions
- **riduzione** *factor*. Type of price reduction for each subscriptions
- **agenzia_tipo** *factor*. Place where consumer bought the card
- **sesso** *factor, binary*. Sex of the consumer
- **comune** *factor*. City of the museum
- **ultimo_ingr.x** *Date*. Date of last visit
- **eta** *integer*. The year in which a consumer was born.
- **n_visite** *integer*. Number of visits
- **abb13** *Date*. Starting date of the card in 2013
- **abb14**. *Date*. Renewal date in 2014 (if renewed)
- **churn** *factor, binary*. Response variable. Will consumer churn?

The final dataset contains 69'574 observations. The classification task that we are about to carry out is strongly unbalanced towards the class of non-churners (Figure 1 shows the counting-differences between the two levels of response variable).

The dataset is made up of numerical, categorical and date variables. Variables in date format have been converted into numeric variables.
Variable *eta* was transformed in order to represent the age of each visitor. To do this, the year 2014 (the current one) was subtracted from the year of birth.
From the corrplot in Figure 2 (useful in order to highlight possible problems due to multicollinearity) we observe the correlation between quantitative variables, this shows a very high relationship between the variable *n_visite* and *total_import*.

From Figure 3 it seems that not all numerical variables are explanatory with respect to the target

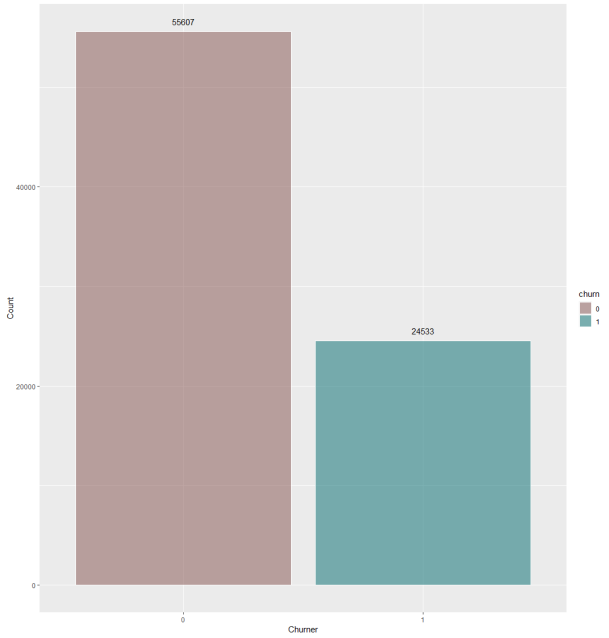

Figure 1. The barplot shows the number of churners and non-churners in the dataset

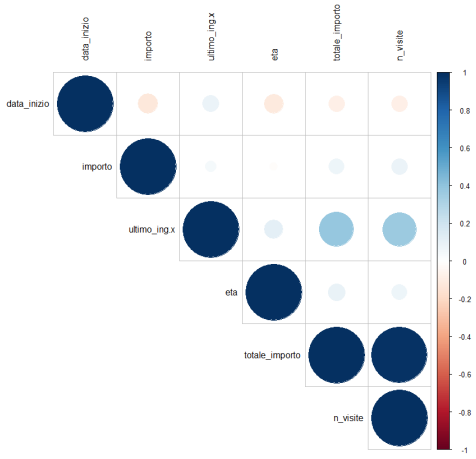

Figure 2. shows the correlations between numerical features

variable. The *ultimo_ingr.x* variable appears to be the variable where the difference between the two boxplots is most noticeable. Since *totale_importo* and *n_visite* variables are subject to strong dispersion we apply, as a *rule of thumb*, a logarithmic transformation to stabilize the strong variability and represent the boxplots in Figure 4.

The sex variable does not seem to be particularly discriminatory with respect to the target variable as shown in the Figure 5.

2 DATA ENTRY MISTAKES

During the EDA step we realized that the age variable had a number of particularly suspicious observations. In fact, the dataset contains 26 observations that have an age greater than 100 years and 4 observations that have a negative age. Since the number of anomalous observations is very low, considering the size of the dataset, it was decided to remove them. Looking at the variable **abb14**, it turns out, moreover, that there are 15 observations that are very distant from the remaining ones (strong outliers) who renewed their subscription at the end of November 2014, and therefore are to be considered customers for 2015.

3 MISSING VALUES

In Table 1 we can observe the relative and absolute frequency of missing values for each variable (only those containing missing values are reported).

1. *professione* contains only NAs values and it is therefore removed from the dataset.
2. *sesso* contains only 3% of the missing values. In this case we decide to remove the missing observations.
3. Missing values for *abb14* are churners or people who didn't subscribe neither in 2013 nor in 2014.
4. Missing values for *ultimo_ing* are people who never visited any museum, even if more than half of them bought a membership.

Features	pct_miss	n_miss
sesso	0.03	2410
professione	1.00	80140
ultimo_ing.x	0.10	8090
abb14	0.31	24548
eta	0.00	2

Table 1. shows, respectively, the relative and absolute frequencies of Missing Values for selected features

4 CLUSTERING

Before the clustering step we decide to remove variables:

- *abb13* because it is perfectly correlated with variable *data_inizio*, thus it contains only redundant informations.
- *abb14* because it takes values only for non-churners observations, while for churners it takes NA values. Thus, excluding missing data from this feature would mean that we are considering only non-churners.

To cluster data we use **SOM** algorithm (from `kohonen` R's package) mainly because, using this approach, clusters can be easily interpreted. **Self-Organising Maps (SOMs)** are an unsupervised data visualization technique that can be used to visualize high-dimensional data sets in lower (typically 2) dimensional representations.
SOM visualisation is made up of multiple "nodes". Each node has:

- A fixed position on the SOM grid
- A K-dimension weight vector (where K is the number of columns of the input data, each node on the grid will have values for these variables)
- Each sample in the input space is "mapped" to a node on the map grid. One node can represent several input samples.

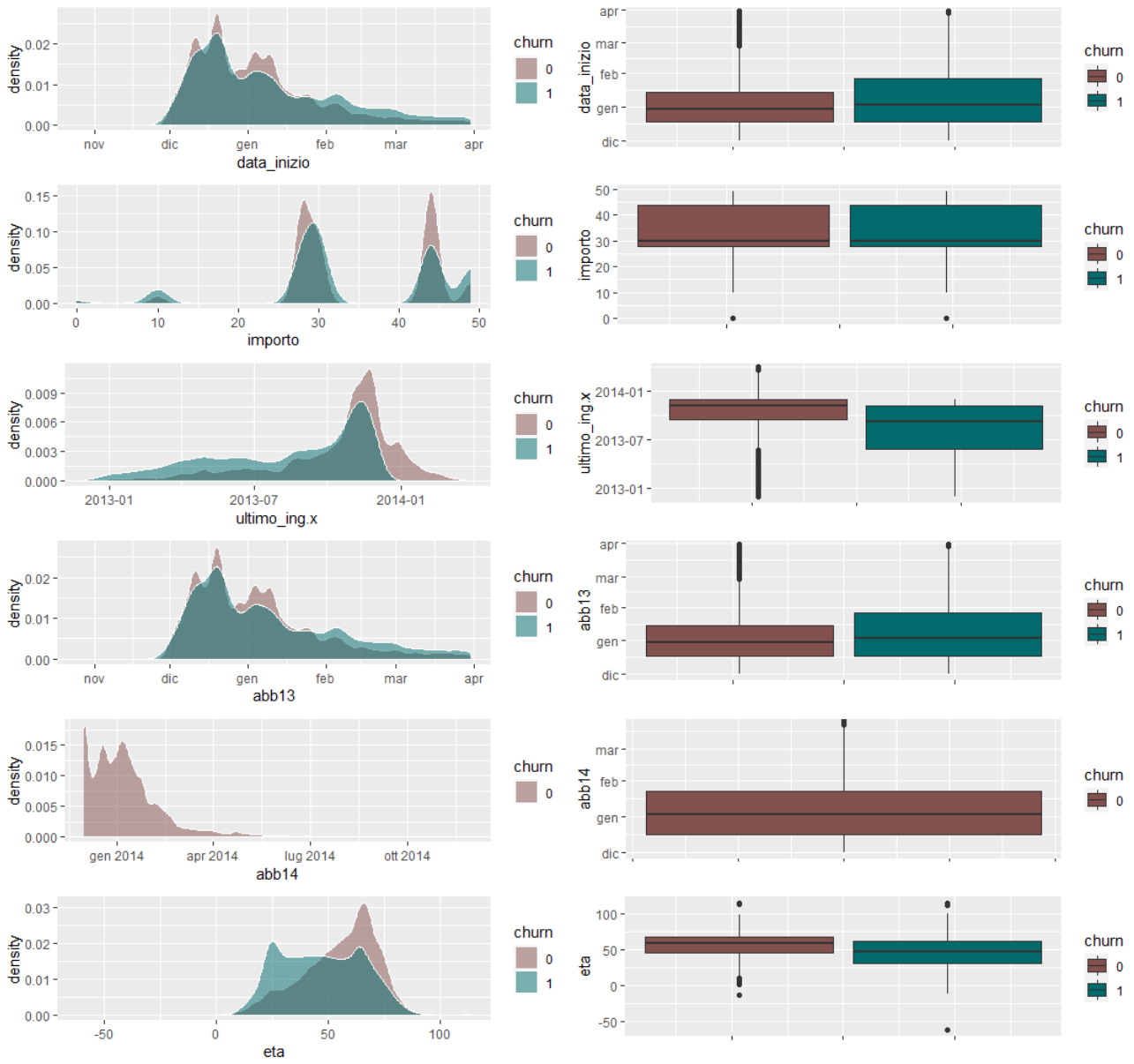

Figure 3. shows on the left side density distribution plots (conditional on the binary target variable) of a pool of selected variables, while on the right side conditional boxplots.

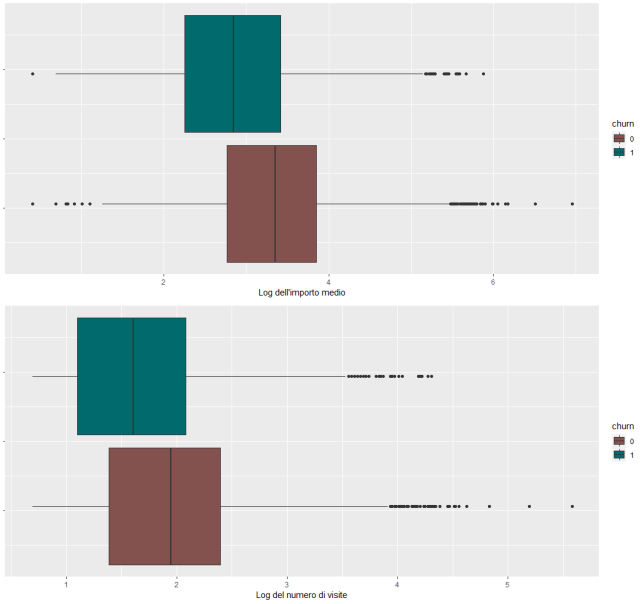

Figure 4. shows conditional boxplots of the 2 created variables totale_importo and n_visite

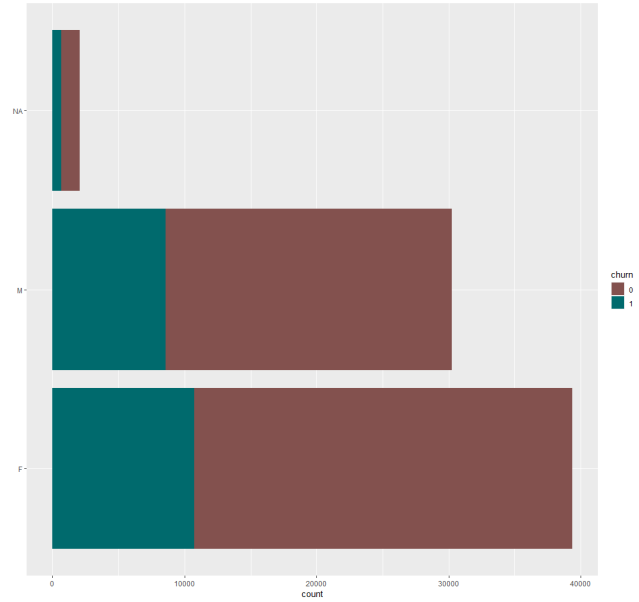

Figure 5. shows the number of observations related to sex conditioned on Churn

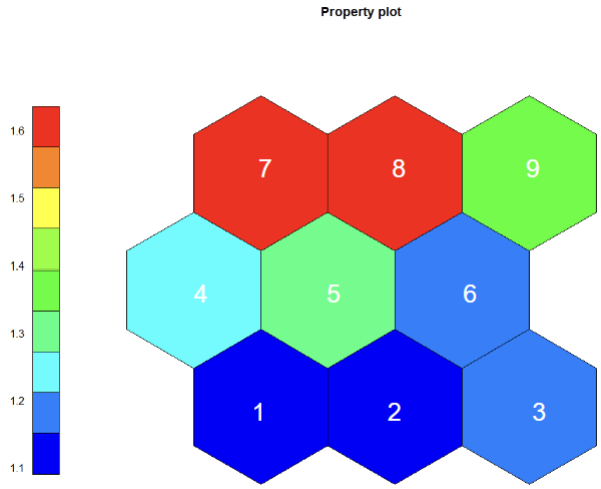

Figure 6. shows our SOM heatmap, number of clusters are reported in white

Typical SOM visualizations are "heatmaps". A heatmap shows the distribution of a variable across the SOM. To cluster data we used a **3x3** SOM with hexagonal topology. We then plotted the distribution of the variable *churn* on the 3x3 map to evaluate if there are clusters with more churners than the others.

Figure 6 shows the heatmap of our SOM and Table 2 shows the percentage of churners and non churners in each cluster.

This unsupervised method allows us to obtain good results, considering the peculiarity of the task (highly unbalanced dataset), creating clusters in which the percentage of churners is greater 50% (in clusters **7** it reaches 64% while in cluster **9** it reaches 62%). Clusters **1, 2, 4, 6** are mainly composed by non-churners.

Typically, a SOM investigative process involves the creation of multiple heatmaps, and then the comparison of these heatmaps to identify interesting areas on the map. In Figure 7 we can see the distributions of different variables across the map.

The `kohonen` package allow us also to visualize the quality of our generated SOM and to explore the relationships between variables in our

	Non-churners	Churners
1	0.91	0.09
2	0.91	0.09
3	0.82	0.18
4	0.74	0.26
5	0.67	0.33
6	0.84	0.16
7	0.36	0.64
8	0.38	0.62
9	0.64	0.36

Table 2. shows the percentage of churner and non-churner observations for each cluster indentified by the model

dataset.
In Figure 8 we can see how our SOM training progress over iterations. We can see that the distance from each node's weights to the samples represented by that node is reduced. Ideally, this distance should reach a minimum plateu (in this case this happen around iteration 2000). The representation in Figure 9 is called the **Neighbour Distance Matrix** or, more often, the **U-Matrix** which represents the distance between each node and its neighbours. In this case we can see that the blue clusters on the upper right side are very close.

5 CUSTOMER NETWORK

First, We consider the dataset in13 without *importo* variable. This will allow us to remove entrances from the same customer id in the same museum, day and time.
We then exclude people with less than 2 total entries from the dataset. This is because we consider as connected consumers with more than 2 entrances at the same time, i.e. if a person has never gone to a museum twice or more it's not interested for our analysis.
For the network's construction, an algorithm was created which calculates all the possible pairs of interactions between people at every day, hour and museum, in such a way as to form a matrix with two columns with the customer code of the

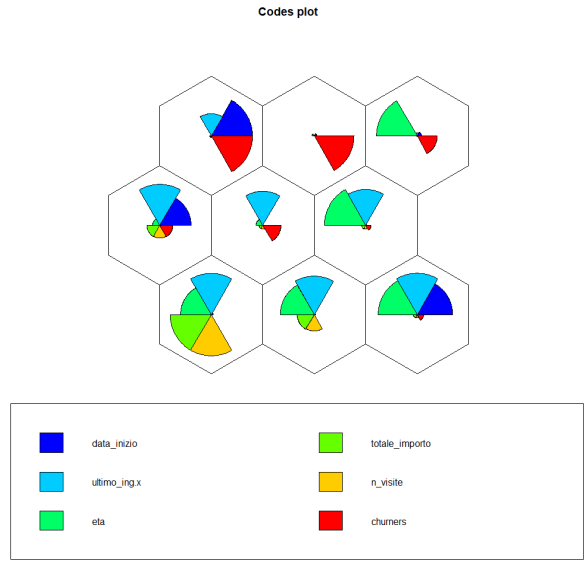

Figure 7. shows distributions of different features across the map

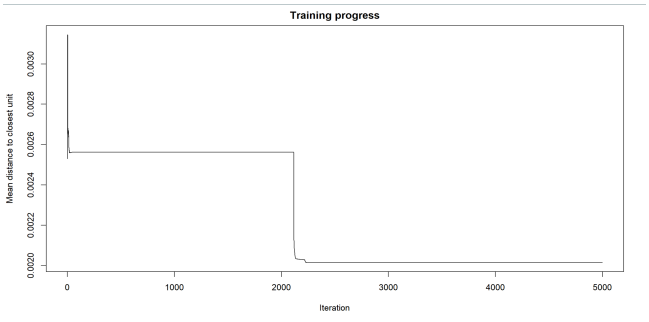

Figure 8. shows SOM's training process

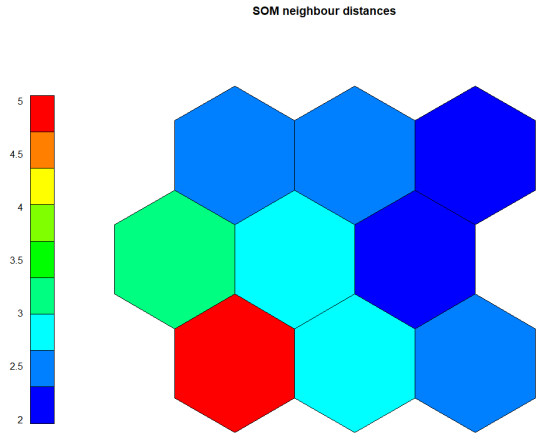

Figure 9. shows our SOM distance's map

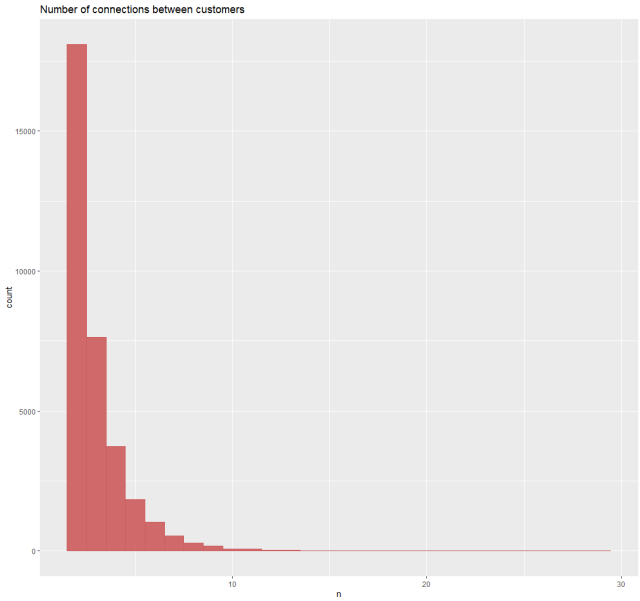

Figure 10. shows the distribution of the number of customer interactions

people concerned.
In Figure 10 we can see the number of customer interactions. We can notice how many are actually close to zero.
For visualization purpose, we decided to filter our network and to select those customers who appear at least 350 times in the connections. In Figure 11 we can see the created network composed of 1353 links and 74 nodes, the most connected ones.

6 MATCHING

First, the numerical variables were standardized, since many of them had very different orders of magnitude.
The matching was made by the variable "*sesso*", considering "*data_inizio*", "*sconto*", "*eta*", "*ultimo_ing.x*" and "*importo*".
After the matching we have 16'338 matched observations in Treated and Control and 3'022 unmatched Controls as we can see in Table 3. In Figure 12 we can observe the matching graphs: the distribution of propensity score between Matched Treated and Matched Control is very similar. We can also see that Unmatched Control units have a propensity score lower than 0,5.

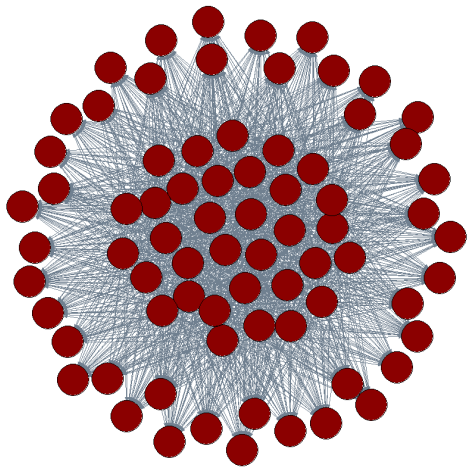

Figure 11. shows network with reduced nodes and edges

	Control	Treated
Matched	16338	16338
Unmatched	3022	0
Discarded	0	0

Table 3. shows a matrix of the sample sizes in the original (unmatched) and matched samples

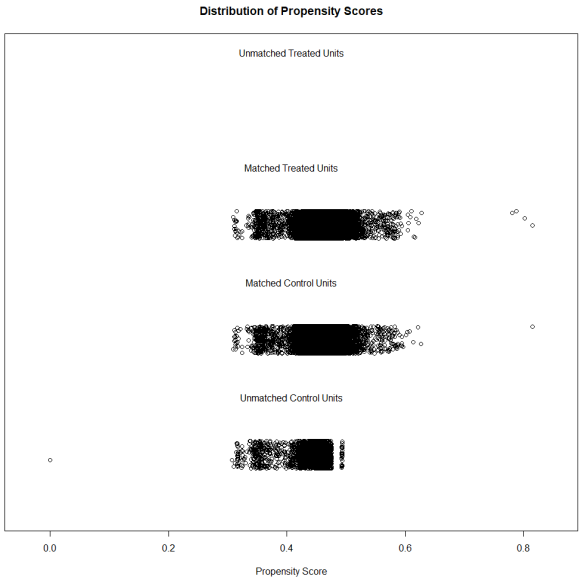

Figure 12. shows the propensity score distributions

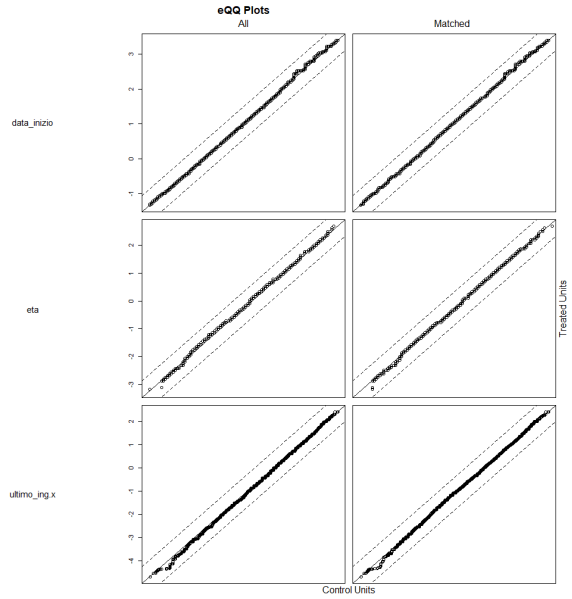

Figure 13. shows empirical quantile-quantile (eQQ) plots

In Figure 13 we can visualize the empirical quantile-quantile (eQQ) plots, created for covariates *eta*, *data_inizio*, *ultimo_ing.x* before and after matching. Points are approximately on the 45-degree line, which means that the distributions in the treatment and control groups are approximately equal.

Figure 14 shows the absolute standard mean difference between Treatment and Control before and after the matching. We can see that, even though most of the differences were already close to 0, after the matching we have a greater shrinkage towards 0. This plot is a simple way to display covariate balance before and after matching

From Figure 15 we can see that the propensity score in matched treated and control share the same support and in Figure 16 we can observe that the propensity score distribution is very similar

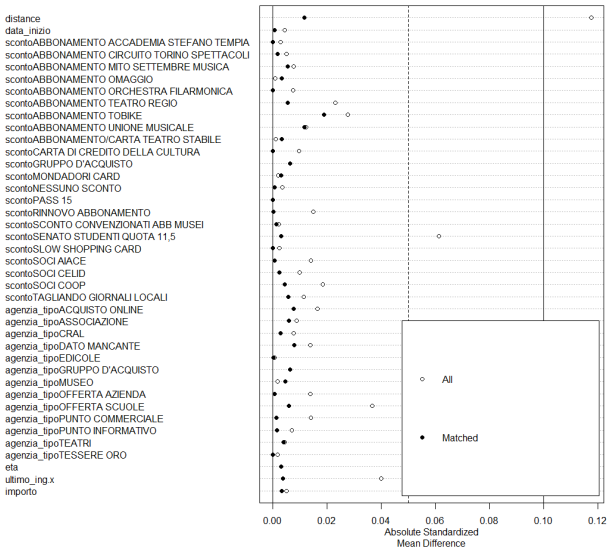

Figure 14. shows a dot plot with variable names on the y-axis and standardized mean differences on the x-axis.

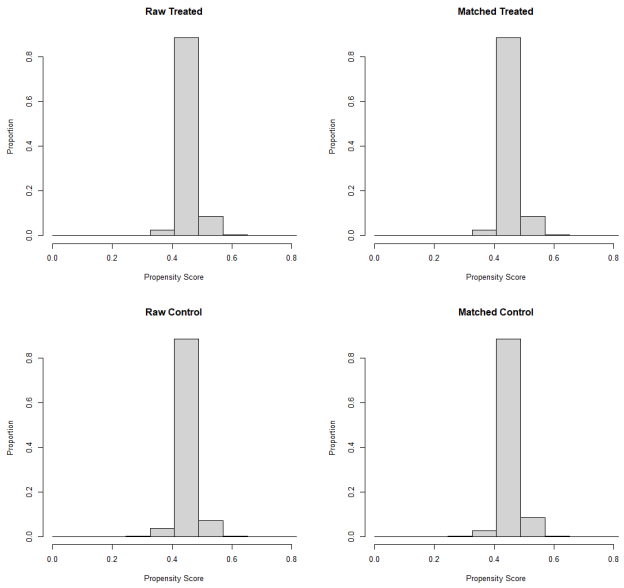

Figure 15. shows an histogram of distance values for the treatment and control groups before and after matching.

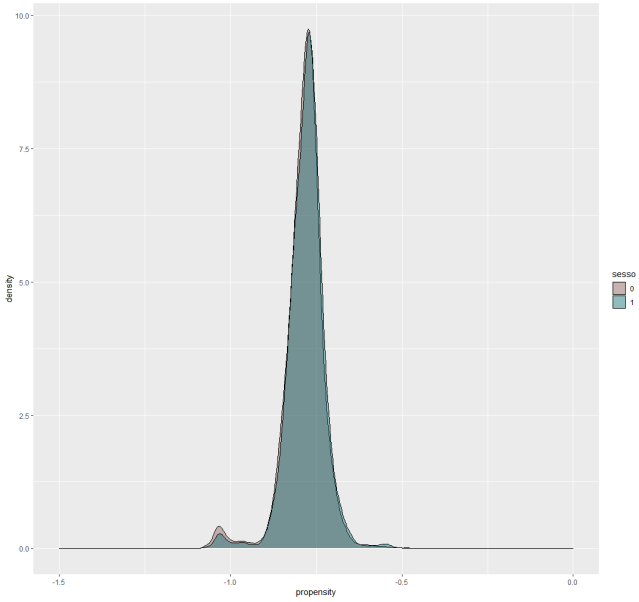

Figure 16. shows the propensity score distribution conditioned on sex

between Male (1) and Female (0).

We now fit a linear model for the target variable "*churn*" considering as features the variables "*sesso* " (with baseline Female), "*data_inizio*", "*età*" , "*ultimo_ingr.x*" and "*importo*". We decide to use a linear model for an easy interpretation of the coefficient "*sesso*". In order to see if gender has a causal effect on the probability of churning, we need to evaluate its linear coefficient estimate and pvalue. The coefficient is significative and equals **0,012**, so we can say that being a male has a slightly positive impact on the probability of churning.

7 CHURNER PREDICTION

Since the variable *comune* has an extremely high cardinality, we keep only those levels that are repeated more than 150 times, the others are classified as "*OTHERS*". We then exclude the variables *codcliente*, *cap* and *agenzia* from the analysis. We also used the network implemented in Section 5 to create the variable "grado", which indicates every node's degree.
The dataset is then divided into training and test sets by stratified holdout (80%-20%). We decided to stratify the sampling because, as mentioned in previous sections, we are facing a binary classification problem with unbalanced classes (Churners-Non Churners proportion is 78%-22%).

We decided to use the following algorithms for this task:

- `Catboost`: A boosting algorithm based on decision trees. It can handle very well multiple categorical variables
- `LightGBM`: a gradient boosting framework that makes use of tree based learning algorithms that is considered to be a very powerful algorithm when it comes to computation.
- `Random Forest`: An algorithm consisting of many decisions trees. It uses bagging and feature randomness when building each individual tree to try to create an uncorrelated forest of trees whose prediction by committee is more accurate than that of any individual tree.
- `Naive Bayes`: We select this algorithm for its simplicity

In Table 4 we observe different metrics associated with the selected models. Among the different performance measures, BAC (*Balanced Accuracy*) is preferred because it accounts for both the positive and negative outcome classes and doesn't mislead with imbalanced data, thus is a better metric to use with unbalanced classification tasks.
We can inferred that for almost all metrics, Random Forest is the winning model. In Figure 17 we observe the ROC curves of the different algorithms implemented. It is observed that the curve related to the Random Forest model is the one that lies above the others. This is also reflected in the value associated with its AUC (Area Under the Curve) which is the highest.

In Figure 18 we observe the density curves of the probabilities obtained by the 4 algorithms selected conditionally on the response variable. It is observed that the two curves overlap in all 4 cases, but in particular in the case of the Naive Bayes classifier, which, not surprisingly, is the model with the worst performances and the worst discriminatory power, as we can see from Table 4.

7.1 Bonus Track - UnderSampling, OverSampling and SMOTE

Since we are dealing with unbalanced classes, it was decided to use resampling methods to rebalance the observations belonging to the two classes of the target variable. A pipeline was created on `mlr3` in which the performances of the Random Forest algorithm were compared with the three different resampled datasets. Resampling Methods used are:

1. **Undersampling**: The majority class was decreased by 1/2.

2. **Oversampling**: The minority class has been resampled so that the observations of the minority class doubled.

3. **SMOTE**: From the minority class new artificial observations were created so that the number of observations of the minority class doubled.

Table 5 shows how the Random Forest trained on SMOTE has higher BAC value than the same model trained on the other resampled datasets and on the models trained on the non-resampled dataset, despite the lack of accuracy. Figure 19 shows the density curve of the probabilities returned by Random Forest trained on SMOTE's dataset conditioned on the response variable.

8 PROFIT CURVE & MARKETING CAMPAIGN

We can now generate the Profit Curve for each of the model considered trained on the non-resampled dataset. We calculate the gains and losses as:

- If a churner is contacted, the expected gain is 10% of "*importo*".
- If a non-churner is contacted, the profit of the campaign is 0 euro, because the person would have renewed the membership anyway.
- The cost to contact a churner is 2 euros.
- The cost to contact a churner is 1 euro.
- We considered the 20 cents per visit that the city of Turin pays to the card association as a profit. To include that earnings we multiplied 20 cents by the number of visits of each person, and we considered it as the actual earnings for each subscriber of a card in the city of Turin.

Figure 20 shows the *profit curves* of different algorithms.
The profit curve is maximized using Random Forest algorithm. Although the optimal number of instances is very similar between the three models, we can observe that Random Forest is the one that generates the highest profit.

If the card association planned to contact every single person who signed in 2013 there would

	Accuracy	Balanced Accuracy	Recall	Specificity	AUC	F1
FeatureLess	0.78	0.50	1.00	0.00	0.50	0.88
CatBoost	0.82	0.65	0.94	0.37	0.84	0.89
LightGBM	0.83	0.68	0.94	0.43	0.85	0.90
Random Forest	0.84	0.72	0.93	0.50	0.87	0.90
Naive Bayes	0.76	0.65	0.85	0.45	0.74	0.85

Table 4. shows different performance metrics of the selected ML algorithms

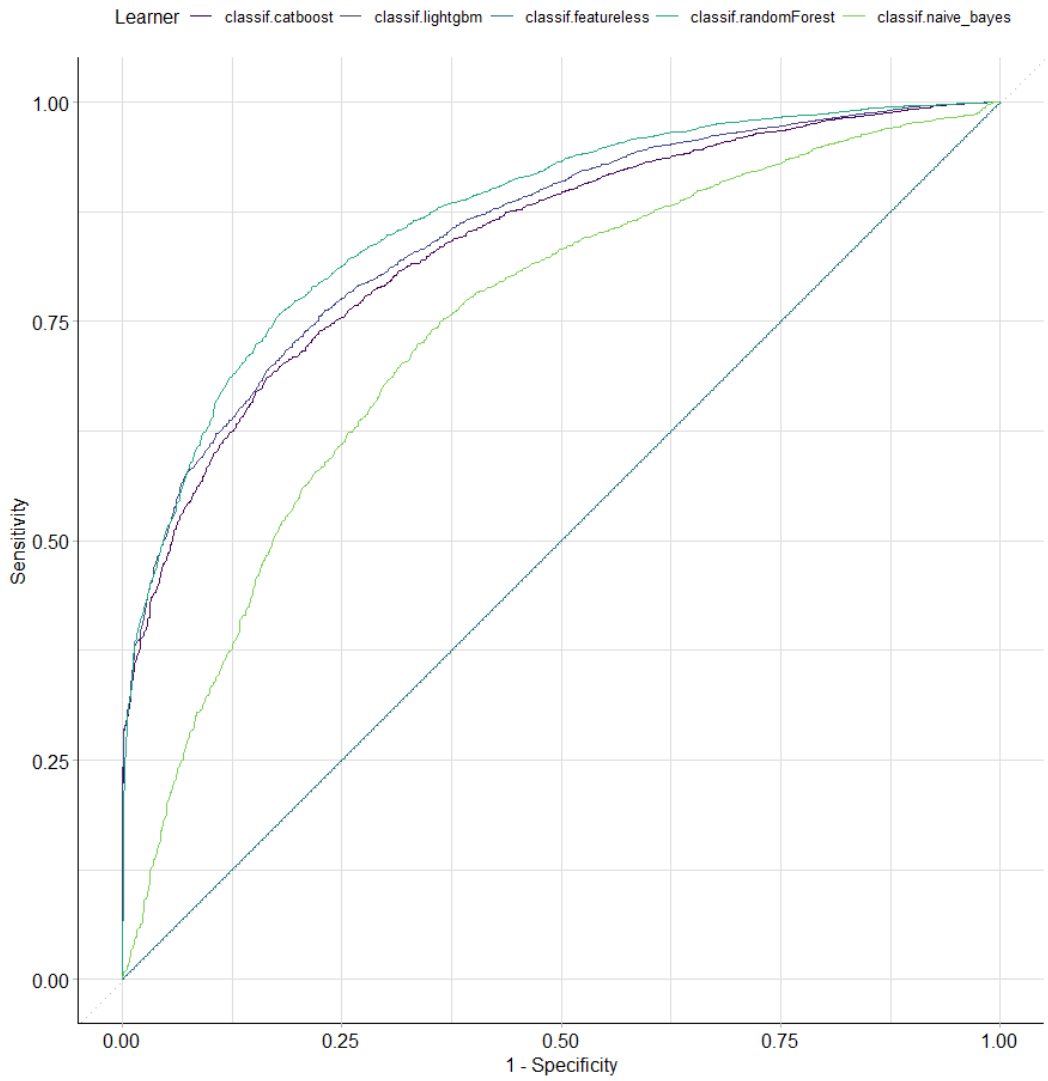

Figure 17. shows different ROC Curves, for every ML model

	Accuracy	Balanced Accuracy	Recall	Specificity	AUC	F1
Undersampling	0.80	0.71	0.87	0.56	0.83	0.87
Oversampling	0.81	0.71	0.88	0.54	0.83	0.88
SMOTE	0.77	0.74	0.79	0.68	0.83	0.84

Table 5. shows different performance metrics of Random Forest Model trained on Resampled Datasets

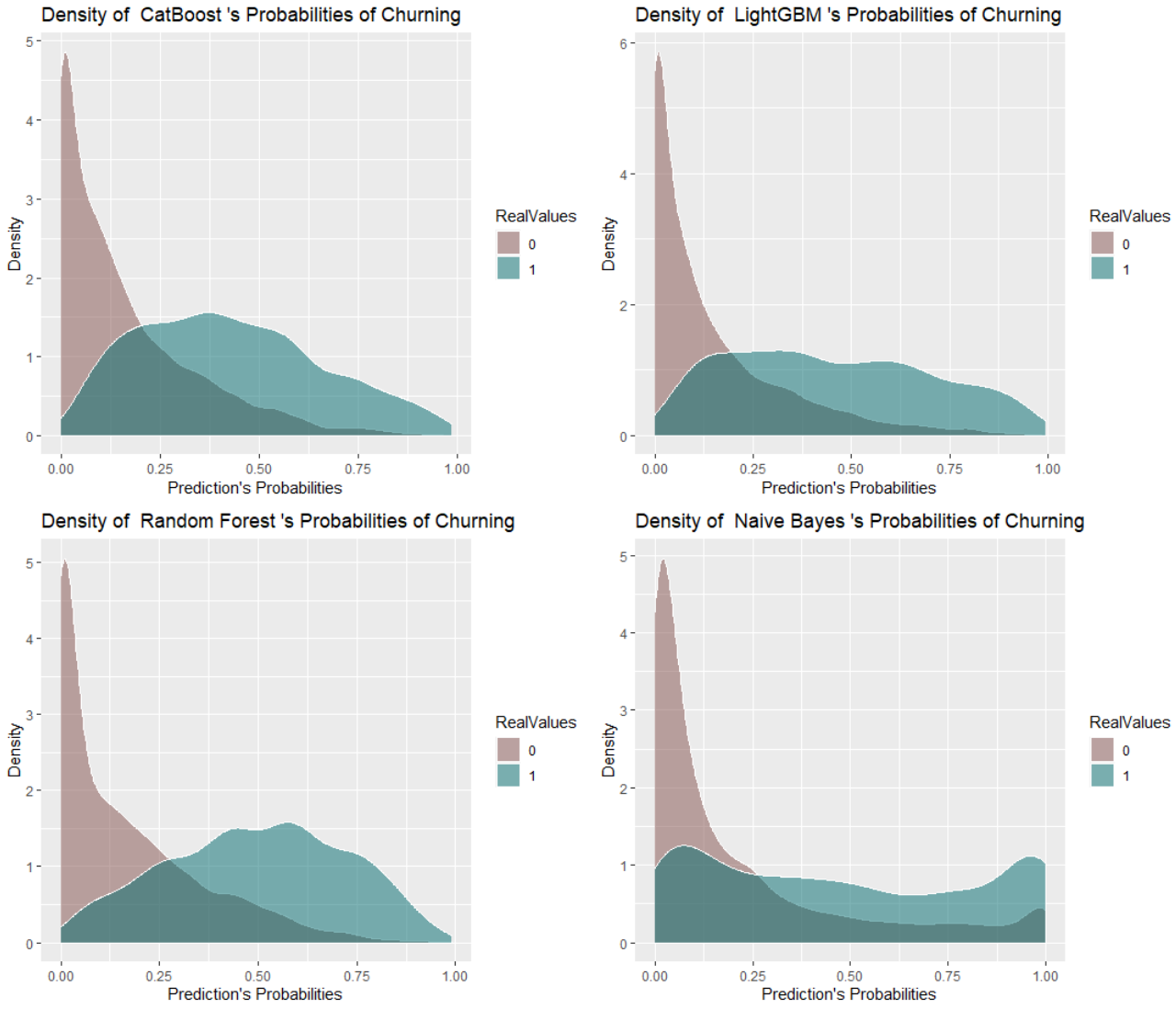

Figure 18. show density curves of the probabilities returned by the models

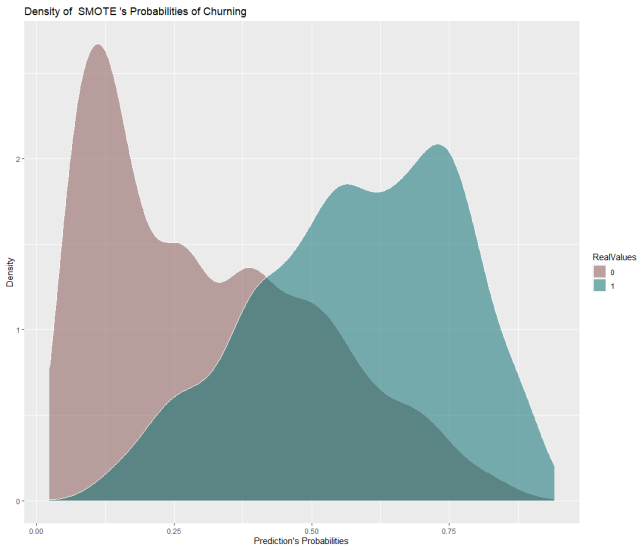

Figure 19. show density curves of the probabilities returned by Random Forest trained on SMOTE's dataset

be a loss. Using Random Forest to select the people who are more at risk, the campaign would be restricted to 1899 people, with a profit of 1'085,72€.

In Table 6 we summarize the values for each of the models trained with respect to the maximization of profit and the number of people to contact.

	Model	Maximum Profit	# of people
1	Random Forest	1085.72	1899
2	Naive Bayes	317.8	2139
3	CatBoost	842.72	1765
4	LightGBM	928.64	1722

Table 6. shows profit maximization of different algorithms

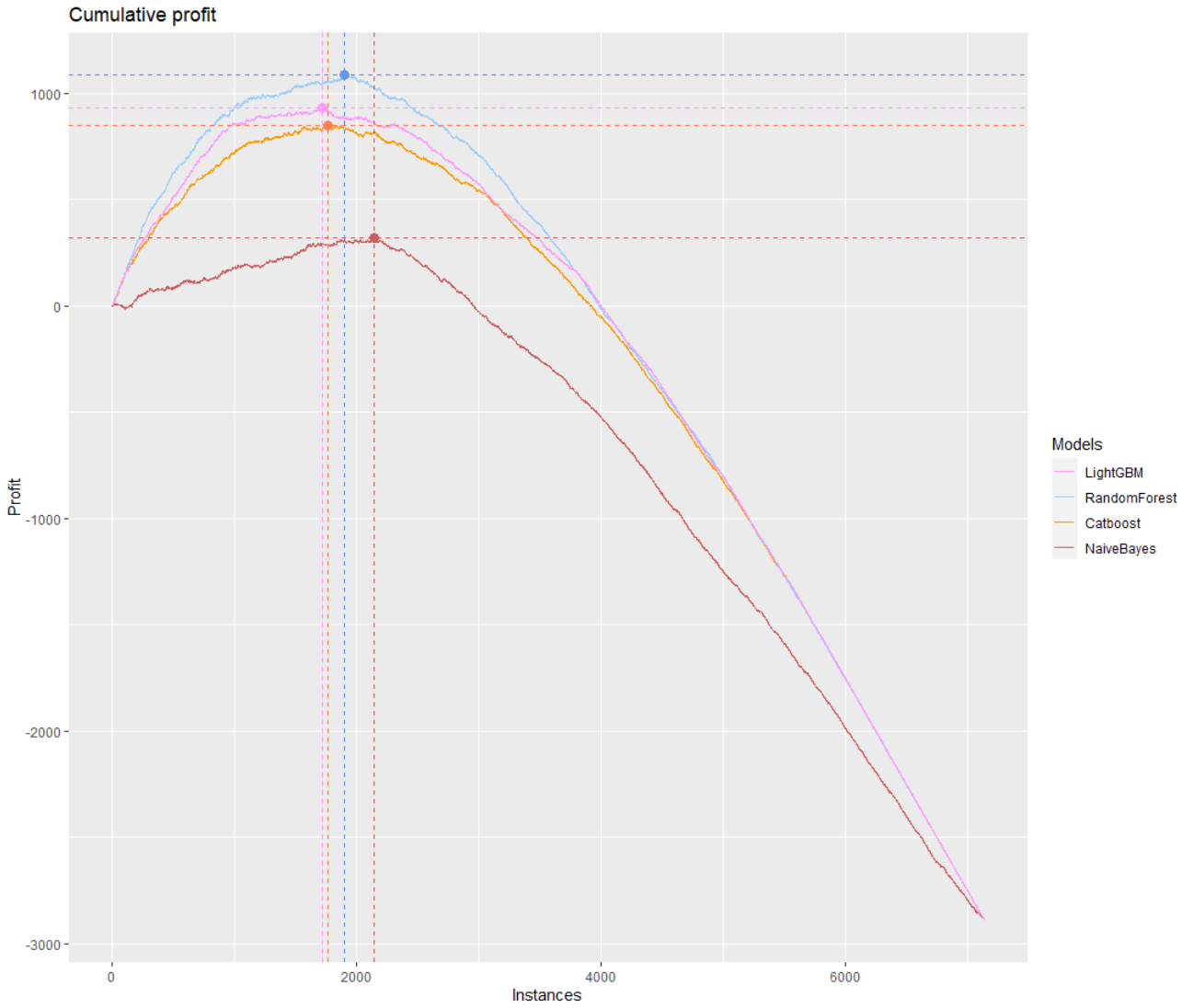

Figure 20. shows the profit curves of different algorithms