UNIVERSITÉ DE NEUCHÂTEL

Institut de statistique

Growing and Maintaining an Online Marketing Panel

Master Thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science in Statistics

by

Victor Steiner Ferreira

Thesis Director
Dr. Alina Matei
University of Neuchâtel,
Faculty of Science
Institute of Statistics

Neuchâtel, August 2018

List of Figures

List of Tables

Contents

Acknowledgments

To my wife Vivian who has given me all emotional and also daily support. To the canton of Zug and the organization Educationsuisse for the financial support. To the University of Neuchâtel for having accepted me and believed I could do it and Alina Matei who have accepted to orient this project but also was always supportive during many of the hard moments during this master. To all the people which have posted videos and all kinds of explanations about statistics on the Internet. To the friends I have done all along the time I have lived in Neuchâtel. To Netquest which gave me the possibility of using the company data for developing this thesis.

Abstract

This thesis approaches an online market panel form the point of view of panel demand and panel size control. The data was provided by Netquest company, Spain. In chapter one, a predictive model using the XGBoost algorithm is implemented in order to calculate the probabilities panelists will accept a Premium proposal in exchange for their Internet navigation data. The probabilities are then used in a real invitation process which is used to evaluate the model's performance in a real application. In chapter two, the life trajectories of the panelists invited to become Premium is studied using sequence analysis methods based on Optimal Matching methods for calculating dissimilarities sequences of statuses. Lastly, in chapter three, non-parametric survival analysis methods are used to study the retention of panelist who have accepted to share their Internet navigation data.

Keywords: XGBoost, Optimal Matching sequence analysis, Online Marketing Panels and Panel Demand, Retention analysis.

1 Introduction

This thesis is an applied statistics project. It addresses a specific business problem, the development of a new product. The objective is to have a deeper understanding on how this product has been evolving, how to control it and improve some of its processes.

The problematic to be addressed occurs in a company in the field of marketing research online panels and encompasses a complete business cycle within the Panel Demand area which is responsible for the panel size, growth and general use of the panel.

A project in applied business statistics must begin with a thorough understanding of the problem and the context from within it arises. The concepts involved are very specific and must be well detailed and characterized in order to be translated into a statistical framework. As discussed by Hahn and Doganaksoy (2012), many challenges are involved in such a process. Statistical skills are the essentials but interpersonal and communication skills are crucial to the success of the project.

A deep understanding of the concepts involved is essential when planning, extracting, cleaning and preparing the data sets to be used by the statistical analysis. In this project all data sets used had to be constructed from the aggregation of many other data sets in the company's databases using SQL (Structured Query Language). SQL is a query language used to query structured relational databases and based upon the relational algebra theory (Codd, 1970). The original databases were not built for statistical analysis purposes but to be used in the company's systems, therefore transformations were needed. Since the analysis involved historical data, many of the issues found when preparing and treating the data were related with past business decisions and learning about these decisions and actions took a lot of communication and research. Moreover, some of the concepts defining the variables used had also to be instantiated by means of new variables. Although the data preparation is not discussed in detail here, it was certainly the most time consuming task.

1.1 Online Panel Definition

An online panel, hereby called Access Panel, is a list of people who have accepted to be regularly invited to answer to online surveys in exchange to some kind of compensation. It is used as a sampling framework, although it is not the same thing as a sampling framework since it is not a complete list of the population posing many challenges when it comes to probabilistic sampling methods as discussed by Brick (2011).

The Access Panel has panelists in many countries, therefore cultural variability is an important factor to be taken into account. Panelists are invited to the panels through marketing campaigns. Therefore, the panels are not open to anyone who might want to be part of it and sizes can be kept under control. The philosophy behind it is the will of the company in developing a long term relationship with its panelists. Once a panelist is recruited, he or she get an online account where they can manage their participation in the panel and through which most of the communication between the company and panelists takes place. Panelists can also install the company's mobile application, which is known to help improving panelists' engagement in the panel.

To keep control of panel use, performance and size, there are many metrics available. Other are being developed now once online panels are a recent methodological tool when comparing with traditional surveys. A more thorough discussion on key metrics and online panels can be find in (Callegaro and DiSogra, 2008) where key concepts as the idea of an active panel, recruitment and response rates as well as profiling panelists are defined and discussed in depth.

In order to characterize the panelists in the Access Panel, all of them answer to a social demographic survey at the beginning of their lives as panelists. Additionally, panelists are also invited to answer profiling survey modules on a variety of of themes of interest.

All panelists tend to be regularly invited to surveys. The delay between these invitations has a target which is between 7 to 14 days but is subject to demand, seasonality and panel size variability. For each invitation leading to a participation in a survey, the panelist receives points that can be exchanged by products in the company's online marketplace.

According to each panelist history in the Access Panel, he or she is given a set of statuses describing his or her life trajectories in the Access Panel. Changes in status depend on rules as, for instance, a minimum participation rate, fraud detection, among other issues.

1.2 The New Product

In 2015, Netquest started developing a new product called Meter Panel or Behavioral Panel. The goal was to start collecting Internet navigation data from the panelists in order to develop online marketing and Internet behavioral data analysis products.

The decision was a reaction of the company to the continuous growth of online marketing and resulting new demands of traditional and new clients. Marketing research field is facing strong changes and traditional survey methods have limitations in a world of digital disruption as discussed in McQuivey (2013). The online manifestation of a person's life is almost as important as the real one in this context and the need to measure it is preeminent and the goal of this product.

The initial strategy adopted to start growing the Meter Panel was to invite a limited number of panelists who had a higher "engagement" in the Access Panel to become Premium panelists. The Premium proposal asks the panelist to install a tracker or plugin in one of his/her devices (laptop, smartphone or tablet) in order to share Internet navigation data. In exchange, the panelist receives an initial amount of points and also extra points for each survey they participate, depending on the number of devices the panelist has a tracker or plugin installed. For example, if a Non Premium panelist receives x points per participation, a Premium panelist would receive $x + y * z$ points, where y is the number of extra points and z the number of devices - up to three - with a tracker or plugin installed.

A crucial point in this process is that it has many steps controlled by different statuses assigned to the panelist as he/she progress on it. The sequence of possible statuses a panelist invited to become Premium may assume aims to:

- Control the communication flow between the company and the panelist during the process of conversion - the panelist receives an email when his or her status is changed;
- Give the panelist time to think and get exposed to the proposal its rules and advantages multiple times before deciding on it;
- Give the chance for the panelist to at any time refuse or to cancel the offer.

The subset of panelists in the Access Panel invited to become Premium panelists is defined as the Meter Panel and the panelists which actually started sharing Internet navigation data are called Meterized panelists.

1.3 Problem Definition

Building the Meter Panel brought many challenges and problems to the company which can be tackled using different statistical methods. These problems, from the perspective of Panel Demand, can be summarized into a three stages cycle which constitute the main chapters of this thesis.

1 **Invitation Process:** Who to invite in order to achieve the highest possible conversion to Meterized rate? In this chapter a Supervised Machine Learning Predictive Model is used to assign to each panelist in the Access Panel a probability of becoming a Meterized panelist;

2 **Life Trajectory or State Sequence Analysis:** In this chapter dissimilarity measures are used to cluster Meterized panelists' life trajectories into meaningful key

patterns. A regression tree model based on the dissimilarities and discrepancy analysis are applied to understand which of the available variables might lead to the ideal life trajectories;

3 **Retention Analysis:** For how long a panelist will keep sharing Internet navigation data? Survival analysis will be applied in order to determine the probabilities that a Meterized panelist will share Internet navigation data for specific periods, in other words retention of Meterized panelists. A Cox Proportional Hazard Model is used to understand the influence of specific variables in the survival curve.

These problems are all interrelated and define a holistic approach to the overall problem: how to grow and maintain the Meter Panel in terms of sizes seeking ideal meter life trajectories and long term retention.

2 Predictive model

2.1 Definition

At the beginning of the Meter Panel, invitations were planned based on a set of key concepts and recruitment marketing campaigns. More specifically, the seniority of a panelist meaning the time they had been in the Access Panel and the ratio of participation and invitations were taken into account as indicators of engagement.

As the panels started growing, the need for a better invitation procedure has aroused. The company has collected enough data which can now be used to develop a supervised learning predictive model in order to better target and calibrate the invitations to the Meter Panel.

A boosting ensemble algorithm called XGBoost (Extreme Gradient Boosting) (Chen and Guestrin, 2016) will be trained using R Core Team (2018) a language and environment for statistical computing. Two R libraries are used in the training implementation, caret (Kuhn et al., 2018) for controlling the implementation and XGBoost (Chen et al., 2018) for the algorithm itself. The model aims to calculate the conversion probabilities of a panelist in the Access Panel which is used then as a score to assess the potential each panelist has to become Meterized.

2.2 Theory

2.2.1 Learning Models

A supervised learning model is used to establish the link between the predictor variables vector $\mathbf{x_i} \in \mathbb{R}^m$ and the response or predicted variable $y_i \in \mathbb{R}$ where $i = (1, ..., n)$ forming the data set $\mathcal{D} = (\mathbf{x_i}, y_i)$ with dimensions $n * m$. This link is expressed through a mathematical relation between the predictors and the parameters $\omega_j \in \Omega$ to be learned, denoted by $y_i = f(\mathbf{x_i})$. The goal of statistical learning is to estimate $f(.)$ by $\hat{f}(.)$ which can be used then for predicting $\hat{y}_i$ for new data where y_i is unknown (James et al., 2013).

The expression supervised learning model has three terms that need to be contextualized. Supervised models are defined by the use of a response variable while unsupervised models study only the intrinsic relations of a data matrix. The term learning has its origins in the field of Machine Learning (Samuel, 1959) while the use of the term model indicates the strong relationship of this field with statistics. This is very well discussed in Breiman et al. (2001), where the author presents two approaches for modeling, one called "Data Modelling Culture" and other called "Algorithmic Modelling Culture". Both approaches seek to explain the relation between $\mathbf{x_i}$ and y_i and consider the relation $f(.)$ as a black box of a certain nature. In the first approach, the box has a stochastic model including the predictors $\mathbf{x_i}$, random noise ϵ and estimated parameters ω_j. In the second approach, the inside of the box is considered unknown and $f(\mathbf{x})$ is seen or modeled as a computational algorithm to predict the response variable. The function $f(.)$ is also called a learner in the machine learning jargon, since it is through it that the relation between predictors and response is learned.

The model used in this thesis application is an example of the historic development of the relationship between these two approaches. Therefore, in estimating $f(.)$ statistical models are used in the context of an algorithm implementation in a computer machine. As discussed by McCullagh (2002), a statistical model is a set of probability distributions on a sample space $\mathcal{S}$ and it can be parametric or non-parametric. A parametric statistical model has additionally a parameter space Θ and a function mapping the parameter space to the set of all possible probability distributions on S, expressed by $P : \Theta \mapsto \mathcal{P}(\mathcal{S})$. On the other hand, an algorithm can be seen as a procedure for computing a function (McQuivey, 2013), besides the most usual definition of the algorithm as a recipe or a set of rules precisely defining a sequence of operations (Stone, 1971).

Learning algorithms are "trained" using the observed data. The expression "training a model" is similar to the statistical idea of model fitting where the parameters are estimated and the fit is evaluated given an error function evaluating the differences between the observed data and the data generated by the model, assuming that the error follows a known probability distribution. Machine learning algorithms can be very powerful predicting the observations used in the training data, however the ultimate goal of training a predictive model is to predict new data that was not used for training the model. The risk here is that by learning all the details in the training data structure the model is too specialized and will not be able to generalize and predict well new data. This phenomenon is called over-fitting and is addressed in most machine learning algorithms by the implementation of tuning parameters also called hyper-parameters. Tuning parameters differ from the model parameters in the sense they are not estimated by a probabilistic model but by heuristic processes as for instance extensive hyper-parameters "grid-search" based on training/testing data split, cross-validation and bootstrap techniques (Friedman et al., 2001, p. 241).

The ultimate goal is to predict $\hat{y}_i$ which can have different interpretations depending on the task to be accomplished. In this thesis application, the task to be accomplished is a binary classification, therefore a logit transformation of the response variable will be used as follows:

$$\hat{y}_i = \log\left(\frac{\hat{p}}{(1-\hat{p})}\right) = \hat{f}(\mathbf{x_i}), \tag{1}$$

allowing the calculation of the probabilities of being in the positive class as:

$$\hat{p}(\mathbf{x_i}) = 1/(1+\exp(-\hat{f}(\mathbf{x}_i))), \tag{2}$$

which is easier to interpret and to use in practical applications.

2.2.2 Boosting Models and the XGBoost implementation

Learning algorithms are set as optimization problems using the objective function which is denoted by $\texttt{Obj}(\Theta) = L(\Theta) + \Omega(\Theta)$, where $L(\Theta)$ the loss function, is defined to measure how well the model fits the training data and $\Omega(\Theta)$, the regularization function, is defined to control the model complexity.

Optimizing the loss leads to better predictions by learning about the underlying distribution of the data while optimizing the regularization leads to less complex models which tend to have smaller variance and consequently more stable predictions. The objective function, through these two possibilities, incorporates what is known as the bias-variance tradeoff, trying to find the best balance between minimizing both, the variance and the bias of the model, at the same time.

For a binary logistic model, the most used loss function is defined as follows:

$$L(\Theta) = \sum_{i=1}^{n} L(y_i, \hat{y}_i) = y_i \log\left(1 + \mathrm{e}^{-\hat{y}_i}\right) + (1 - y_i)\log\left(1 + \mathrm{e}^{\hat{y}_i}\right). \tag{3}$$

The most common regularization functions used are L1 (Lasso Regression), $\lambda \sum_{j=1}^{m} |\beta_j|$, and L2 (Ridge Regression), $\lambda \sum_{j=1}^{m} \beta_j^2$, where β_i are the regression coefficients and λ is the penalty coefficient of the lagragian form of the regularization equations. Regularization terms are useful for avoiding over-fitting but if the regularization rate term λ is not well set it can lead to under-fitting. L1 punishes smaller coefficients and is better fit for feature selection while L2 punishes larger coefficients, better spreading error across the vector of model coefficients (Ng, 2004). The λ value will control how much the β_i coefficients will be affected. From a Bayesian perspective, one can interpret λ as the prior knowledge one has about the model's parameters, in other words, λ would represent the uncertainty of the model before being fit.

The main idea behind boosting algorithms is said to be born with the question "Can a set of weak learners create a single strong learner?" posed by Kearns (1988). The idea was further developed by Schapire (1990) and the boosting algorithms started to emerge and are still being improved until the present day.

Boosting algorithms are in general tree based supervised learning ensemble models. Ensemble because their predictions are built by averaging the individual predictions of many trees. They build upon Classification and Regression Trees (CART) theory, developed by Breiman et al. (1984), which are known for having high variance becoming very different given small changes in the training data. Techniques as Bagging (Breiman, 1996) and Random Forest (Breiman, 2001) were developed in order to bring randomness into the algorithms and decrease the variance of CART models. Boosting methods improve these algorithms even more by helping to reduce bias once at each new iteration a tree is built based on the error of the past iteration trying to improve the learners by focusing on areas where the model is not predicting well. Instead of using independent and fully grown trees as in the Random Forest algorithm, boosting trees are dependent, since each new tree focuses on improving the errors of preceding trees.

In simple terms, a boosting algorithm builds weak learners using them into an additive model resulting in a final strong learner. The model is weighted according to the accuracy of the weak learners and these weights are used to control the learning rate or shrinkage, reducing the influence of each individual weak learner (Friedman, 2000).

The first boosting algorithms were not adaptative, meaning they could not properly use the classification error in the weak learners to drive the construction of the next learner. The first adaptative boosting algorithm known as AdaBoost was developed by

Freund and Schapire (1997).

When using adaptative models, at the beginning of the algorithm all the observations in the data set are given equal weights, which are then updated at each new iteration. Wrongly classified observations receive bigger weights, hence, in the next iteration the new learner will focus on the wrongly classified observations.

A tree ensemble model with K trees can be denoted as follows [1],

$$\hat{y_i} = \sum_{k=1}^{K} f_k(\mathbf{x_i}), \quad f(x_i) \in \mathcal{F}, \tag{4}$$

where $\mathcal{F}$ is the functions space containing all possible regression trees such that,

$$\mathcal{F} = \{f(\mathbf{x}) = \mathbf{w_{q(x)}}\}(q : \mathbb{R}^m \mapsto T,\ \mathbf{w} \in \mathbb{R}^T), \tag{5}$$

where the tree structure is denoted by q which maps each case in $\mathbf{x_i} \in \mathcal{D}$ to one of the T leaves in the tree f_k. Note that T represents the total number of leaves for one tree. Thus, each tree f_k has a specific number T of leaves, also known as terminal nodes.

Trees are defined by a set of heuristics expressed in its hyper-parameters. These can be related with the definition of the objective function in multiple ways. For instance, splitting by information gain can be used as a loss functions, pruning procedures as maximum depth and leaf values smoothing using L2 can be implemented as regularization procedures.

The objective function in the case a XGBoost algorithm is defined as:

$$\texttt{Obj} = \sum_{i=1}^{n} L(y_i, \hat{y_i}) + \sum_{k=1}^{K} \Omega(f_k(\mathbf{x_i})), \tag{6}$$

where $\Omega(.)$ defines the complexity of the tree and the trees can be seen as the model parameters $\Theta \in \{f_1, f_2, ..., f_k\}$ we need to learn and $L(.)$ is the loss function. Learning a function in this context goes through the task of defining the objective function which will then map the tuning parameters defining a decision tree into the loss and regularization functions.

In our application the loss is the binary-logistic function. The standard regularization function $\Omega(.)$ for each tree f_k grown by the XGBoost algorithm is defined as follows:

$$\Omega(f_k) = \gamma T + \frac{1}{2}\lambda w^2, \tag{7}$$

[1]The mathematical development of the gradient boosting algorithm presented here is based on Chen and Guestrin (2016).

which is a simplified version of the regularization used in Regularized Greedy Forest by Johnson and Zhang (2014). The γ and the λ terms are lagrangian multipliers. The first penalizes the number of splits in each regression tree and the second penalizes the estimated regression coefficients. As discussed in the first paragraph of page 15, a lagrangian multiplier in a regularization function can be interpreted as the prior knowledge one has on the parameters before fitting the model. Therefore, there is no exact way of setting these values and, in general, techniques as cross-validation among others are used to approach the ideal region for both γ and λ in each specific case. In other words, the γ and λ values will always depend on the data being analyzed and the knowledge the research has about the specific problem at hand.

Once our parameters vector is not constituted by numbers but by functions defining trees, the problem of optimizing the objective function becomes very hard since simple gradient descent methods can not be applied. To solve this problem, boosting algorithms use an additive training strategy. A deep understanding of additive boosting models can be find in the paper (Friedman et al., 2000, p. 17), in Algorithm 3 he describes LogitBoost implementation for binary problems.

The prediction in round t is denoted by $\hat{y_i}^{(\mathrm{t})} = \hat{y_i}^{(\mathrm{t\ -\ 1})} * f_t(\mathbf{x_i})$. The goal becomes to find f_t in order to minimize the following objective function:

$$Obj^{(\mathrm{t})} = \sum_{i=1}^{n} L(y_i, \hat{y_i}^{(\mathrm{t\ -\ 1})} + f_t(\mathbf{x_i})) + \Omega(f_t) + c, \tag{8}$$

where c is a constant term.

Each new weak learner is greedily added taking into account the loss calculated in the past iterations. A greed algorithm is a heuristic solution for an optimization problem seeking to approach a global optimum by means of local optimization at each iteration of the algorithm. That is one of the main differences between boosting algorithms and random forests since in the second case trees are grown fully and are completely independent of each other (Breiman, 2001).

Despite the use of an additive model, Eq. (8) might still be very complicated to solve depending on the chosen loss function which is always a convex differentiable function. Therefore, to calculate the derivatives of the loss, an approximation approach based on Taylor series (Wikipedia, 2018) is the proposed solution to make calculations more feasible. Taylor series is a very powerful tool since it can be used to approximate non-polynomial functions using the derivative of simple polynomials. The basic idea of a Taylor series is to use higher order derivatives at a function point x to get information about the points around x and approximate them.

Based on a Taylor expansion of the objective function, Chen and Guestrin (2016) define $g_i = \partial_{\hat{y}(t-1)} L(y_i, \hat{y_i}^{(t-1)})$ and $h_i = \partial^2_{\hat{y}(t-1)} L(y_i, \hat{y_i}^{(t-1)})$ which are then used to rewrite the objective function as follows:

$$\begin{aligned} Obj^{(t)} &\simeq \sum_{i=1}^{n} \Big[L(y_i, \hat{y_i}^{t-1}) + g_i f_t(\mathbf{x_i}) + \frac{1}{2} h_i f_t^2(\mathbf{x_i}) \Big] + \Omega(f_t) + c \\ &= \sum_{i=1}^{n} \Big[g_i f_t(\mathbf{x_i}) + \frac{1}{2} h_i f_t^2(\mathbf{x_i}) \Big] + \Omega(f_t). \end{aligned} \tag{9}$$

Next step involves the definition of the instance set of leaf j denoted by $I_j = \{i | \mathbf{x_i} = j\}$ which is then used to rewrite Eq. (9) by expanding the regularization term as follows:

$$\begin{aligned} \tilde{Obj}^{(t)} &= \sum_{i=1}^{n} \Big[g_i f_t(\mathbf{x_i} w_i) \Big] + \gamma T + \frac{1}{2} \lambda \sum_{j=1}^{T} w_i^2 \\ &= \sum_{j=1}^{T} \Big[(\sum_{i \in I_j} g_i) w_i + \frac{1}{2} (\sum_{i \in I_j} h_i) w_i^2 \Big] + \gamma T. \end{aligned} \tag{10}$$

For each tree structure $q \in \mathcal{F}$, an optimal weight w_j* for each leaf j in the tree can be calculated as follows:

$$w_j* = -\frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda}. \tag{11}$$

Using Eq. (11) the optimal value of the objective function for the tree structure $q \in \mathcal{F}$ is calculated in (Eq. 12). This value has a similar interpretation to the one of the Gini impurity in CART (Breiman et al., 1984) defined as a measure of the probability of misclassification of a set of instances.

$$\tilde{Obj}^{(t)}(q) = -\frac{1}{2} \sum_{j=1}^{T} \frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda} + \gamma T. \tag{12}$$

In most practical situations, it is impossible to calculate the optimal objective function for all possible tree structures. At this point the greedy tree building process is used. Each tree is grown split by split. Each candidate split is evaluated by defining $I = I_R \cup I_L$ and the calculation of the loss reduction is as follows:

$$\tilde{Obj}_{(split)} = \frac{1}{2} \Big[\frac{(\sum_{i \in I_L} g_i)^2}{\sum_{i \in I_L} h_i + \lambda)} + \frac{(\sum_{i \in I_R} g_i)^2}{\sum_{i \in I_R} h_i + \lambda)} - \frac{(\sum_{i \in I} g_i)^2}{\sum_{i \in I} h_i + \lambda)} \Big] - \gamma. \tag{13}$$

The use of a regularized objective function can be further improved by controlling the shrinkage hyper-parameter η when tuning the algorithm. The shrinkage process works by multiplying each new weight in the additive model by the factor η reducing the influence of each single tree in the overall model and letting space for more weak learners to be trained. The smaller the value of η the smaller will be the learning rate of the algorithm preventing over-fitting as developed in Friedman (2000) and also Friedman (2002) where the author sustains that a $\eta <= 0.1$ would issue better generalization error. However, in practice a $\eta <= 0.1$ incurs the need of more iterations and consequently more training and data processing time, which might be complex if the algorithm can not be parallelized or if the computer used for training the model is not good enough.

In the article by Chen and Guestrin (2016), the authors discuss in more detail how the implementation of the XGBoost algorithm deals with split selection and system design. These features, specially the last one, make this implementation very fast and efficient, therefore, very suitable for business applications. For that reason it was chosen for the application to be described next.

The following algorithm from Kuhn and Johnson (2013) is a simple version of a gradient boosting algorithm for binary classification that can be adapted in many ways, but specially by defining the objective function. This algorithm has two basic tuning parameters, the tree depth and the number of iterations. Nevertheless, other shrinkage parameters may be used to control the learning rate. Since it is a model based on classification trees, other parameters for split criteria for defining the information gain measures - Gini Impurity or Entropy (Breiman et al., 1984) - are available, as for instance, minimum number of cases per split and per leaf node, minimum gain per split, number of features to use per split among many other options available.

```
1         Initialized all predictions to the sample log-odds: $f_i^{(0)} = \log \frac{\hat{p}}{1-\hat{p}}$.
2  for    *interation* $j = 1 \ldots M$ do
3    |    Compute the residual (i.e. gradient) $z_i = y_i - \hat{p}_i$
4    |    Randomly sample the training data
5    |    Train a tree model on the random subset using the residuals as the outcome
6    |    Compute the terminal node estimates of the Pearson residuals: $r_i = \frac{1/n \sum_{i=1}^{n}(y_i - \hat{p}_i)}{1/n \sum_{i=1}^{n}(\hat{p}_i)(1-\hat{p}_i)}$
7    |    Update the current model using $f_i = f_i + \lambda f^{(j)}$
8  end
```

Algorithm 1: Simple gradient boosting for classification (2-class).

Note that in step one, $\hat{p}$ is the sample proportion for one of the two classes and is the same for all observations in the data set. The authors do not make it clear how the Pearson residuals calculated in step 6 are used exactly. My interpretation is that, given this is an adaptative algorithm, observations for which the residuals are higher will be "tagged", in the sense that in the next iteration the tree to be grown will focus in correctly classifying observations for which the residuals were higher in the previous iteration.

A complete version of the XGBoost algorithm can be found in (Nielsen, 2016, p. 64) and is reproduced bellow in the Algorithm 2. Comparing both algorithms allows a deeper understand of the specifities of the XGBoost implementation among the more general boosting algorithm. Among the general Gradient Boosting algorithms XGBoost differs because it is an adaptative algorithm offering at each iteration the possibility of penalizing each individual tree. This is possible given specific regularization parameters only present in the XGBoost which help to control not only each individual tree (step 4 in Algorithm 2) but also the weights in each tree leaf (step 5 in the Algorithm 2), thus reducing the variance of each tree, which is one of the main problems of using individual trees models. This flexibility allows for characteristics as automatic feature selection and very important, it makes possible for the model to actively take the bias-variance tradeoff into consideration as discussed in (Nielsen, 2016, p. 91).

```
        Input: Data set $\mathcal{D}$.
        A loss function L.
        The number of iterations M.
        The learning rate $\eta$.
        The number of terminal nodes T.
1       Initialize $\hat{f}^{(0)}(x) = \hat{f}_0(x) = \hat{\theta}_0 = \arg\min_{\theta} \sum_{i=1}^{n} L(y_i, \theta)$;
2  for  m = 1,2,...,M do
3    |    $\hat{g}_m(x_i) = \left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i}\right]_{f(x)=\hat{f}^{(m-1)}(x)}$;
4    |    Determine the structure $\{\hat{R_{jm}}\}_{j=1}^{T}$ by selecting splits which maximize
     |    $Gain = \frac{1}{2}\left[\frac{G_L^2}{n_L} + \frac{G_R^2}{n_R} - \frac{G_j m^2}{n_j m}\right]$;
5    |    Determine the leaf weights $\left\{\hat{w}_j m\right\}_{j=1}^{T}$ for the learned structure by
     |    $\hat{w}_j m = \arg\min_{w_i} \sum_{i \in \hat{I}_j m} L(y_i, \hat{f}^{(m-1)}(x_i) + w_j)$;
6    |    $\hat{f}_m(x) = \eta \sum_{j=1}^{T} \hat{w}_{jm} I(x_i \in \hat{R}_{jm})$;
7    |    $\hat{f}^m = \hat{f}^{(m-1)}(x) + \hat{f}_m(X)$;
8  end
        Output: $\hat{f}_{(m)}(x) = \hat{f}^{(M)}(x) = \sum_{m=0}^{M} \hat{f}_m(x)$.
```

Algorithm 2: Gradient tree boosting.

2.3 Application

The choice of XGBoost in the company was largely driven by its success in many applications as, for instance, the many machine learning competitions it has won. A more thorough discussion about why this algorithm has been so successful can be find in (Nielsen, 2016).

The main goal of this model implementation is to assign to each panelist a probability of accepting the Premium proposal and start sharing their Internet navigation data in order to better target who will be invited.

This chapter section starts by describing the data used and feature engineering procedures. Subsequently, the model implementation is presented followed by its performance evaluation. Finally, the results of a real flagging procedure are presented and compared with expected results.

2.3.1 Data description and preparation

The data set built for this application has 421,166 observations. The predictors can be divided into five groups by their meaning in the context of the problem at hand. Each of these groups has an underlying hypothesis about the behaviour of the response variable and tries to capture it.

Response variable

- *get_to_active*: equals 1 if the panelist has shared Internet navigation data (get to active) and 0 otherwise (has never got to active):

get_to_active	Count (%)
0	305,364 (72.5%)
1	115,802 (27.5%)
Total	421,166 (100%)

Table 3: Response variable distribution.

The response variable distribution tends to be imbalanced. An imbalance variable in the field of Machine Learning might be defined by the difference of its distribution in relation to the uniform distribution. In other words, a balanced binary response variable would have 50% of the observation in each class. Most Machine Learning algorithms will have better predictive performance when predicting balanced response variables. It is hard to find academic literature on this theme, however, in practice it is very much discussed, for instance in Machine Learning blogs and threads on the Internet. Although not severe in the data set used here, class imbalance might cause problems and be misleading specially when using accuracy for evaluating the model performance. Several approaches exist for dealing with class imbalance (He and Ma, 2013), as for instance, over-sampling, under-sampling, penalized models and others. It will be seen later that the XGBoost algorithm implementation offers tuning parameters which are useful for controlling for imbalance in the response variable.

Predictors

1. Demographic variables: here it is known that panelists in certain countries are more open to share their data than others.

    - *country*: 13 countries - Spain (ES), Brazil (BR), Portugal (PT), Colombia (CO), Mexico (MX), Chile (CL), Argentine (AR), Peru (PE), Great Britain (GB), France (FR), Germany (DE), United States (US) and Italy (IT).
    - *gender*: Women = 0 and Men = 1.
    - *age*: in years from 16y to 98y with a mean age of 37y.
    - *ses_standard*[2]: standardized social class with the same five levels for each country, "unknown" = 0, "High" = 1, "MediumHigh" = 2, "MediumLow" = 3, "Low" = 4.

2. Panel dimension

    - *app*: equals 1 if a panelist has the company mobile application installed in one of his/her devices and 0 otherwise. This is an important variable since it is a strong indicator of engagement of the panelist with the company as well as an extra communication channel between both parts.
    - *seniority*: measures in days how old a panelist is in the Access Panel. The more senior a panelists is the higher the probability he or she will participate in surveys, thus, the higher the probability he or she will value the extra points offered by the Premium proposal.
    - *recruitment*: equals 1 if the panelist was invited through a flag process and 1 if invited by the Direct Meter process [3]. This is another important factor since Direct Meter invitations are targeting panelists who have no previous relationship with the company and have lower probability of understanding the real value of the extra points offered by the Premium proposal.

[2]For each country the official local definition of social class or information on income is measured on the panelists and later a standardization developed by the company is applied.

[3]Direct Meter: panelists are invited to the Access Panel and Meter Panel simultaneously. No conversion probability is estimated here in order better target who to invite or not.

3. Invitations & Participation dimension

    - ***invitations***: how many invitations to participate in surveys a panelist had from the first day in the Access Panel until the day of invitation.
    - ***participations***: how many invitations were converted into real participations. A panelist who has many participations is an engaged one and will be reminded of the extra points earned at each participation thus increasing the probability of retention of this panelist as a Meterized one.
    - ***invitations_30***: how many invitations to surveys in the 30 days before Premium proposal. This predictor intends to capture the momentus engagement of a panelist. For example, if a panelist who usually do many participations goes on vacations and stop answering to the survey invitations, it might be better not to offer the Premium proposal since there is a chance the panelist will miss it.
    - ***participations_30***: how many participations in the 30 days before Premium proposal. This plays a role altogether with the predictor `invitations_30`.

4. Points dimension

    - ***points***: how many points a panelist had when invited to become Premium.
    - ***exchanged_points***: how many points a panelist used to buy things before the Premium invitation. Here a panelist who spent many points buying things in the company online store will have a clear understanding of the real value of the extra points offered. Therefore, a higher probability of concluding the Premium proposal is an interesting offer.
    - ***n_exchanges***: how many times the panelists bought things in the company e-commerce. It has a similar interpretation as the previous predictor but focus on panelists which buy less expensive items but more frequently.

5. Internet Activities

    This set of predictors brings to the model the hypothesis that the more Internet activities a panelist does, the better are the chances he or she will not be resistant to share their Internet navigation data. Additionally, certain activities as Internet banking and buying online might be indicators of such behaviour.

    - A list of 15 Internet activities organized as yes or no questions ("flightpurchase", "onlinebanking", "shopping", "casualonlinegaming", "socialmedia", "gambling", "downloaduploadphotos", "downloaduploadaudio", "downloaduploadfilms", "downloaduploadvideos", "jobsearch", "informationsearch", "onlinecourses", "playonlinegames", "watchvideos"). This is the only set of variables which has missing values. No data imputation method was used and the missing values are handled by the XGBoost algorithm itself as discussed in the section 3.4 of Chen and Guestrin (2016) in the algorithm 3 representation.

The variables are all set to numeric type, even the categorical ones. This is a requirement for running the XGBoost algorithm in R. Categorical variables with more than one factor are dummified, meaning they are split into as many binary variables as factors in the variables.

```
library(caret)
# one_hot_encoding (dummify coding) factors with multiple categories
one_hot_encoding <- dummyVars(~ country + ses_standard,
                              data = training)
one_hot_encoding <- data.table(predict(one_hot_encoding, training))
# Adding dummy variables and deleting duplicated variables
training <- cbind(training[, -c(``country",␣``ses_standard")],
                  one_hot_encoding)
```

For splitting the data, the caret function createDataPartition(.) was used in order to assure the response variable distribution was the same in both data sets. The tradeoff here is that if you put too many cases into the training set you might get bad evaluation of the model's performance, however, too many cases in the testing data might lead to high variance in the training data. Common choices are 70/30 and 80/20 but it is not easy to find references on this topic. Sampling techniques as stratified sampling, similar to the approach used here, are used as discussed, among other options, in Reitermanov'a (2010). Since we have a data set with a huge number of cases either option should work well, therefore, a 70/30 split was adopted.

It is important to note that the split is done keeping the distribution of the response variable as specified in table 3. This will assure that both training and testing data sets are as close as possible to the original data set. A more complex approach could have been used in the case the distribution of the predictors were to be also kept in each data set. For instance, stratified sampling based on the predictors for which keeping the distribution is important would be an alternative.

```
# Splitting data into training and testing data sets.
set.seed(3185)
in_training <- createDataPartition(y = data$get_to_active,
                                   p = 0.7, list = FALSE)
training <- data[in_training, ]
```

```
testing <- data[-in_training, ]
```

2.3.2 Feature engineering

Feature engineering in Machine Learning is a key step for the success of the model. It is about how to model your predictors into new predictors and how to transform concepts and features of the problem being modeled into practical inputs for the model. There are not many academic sources on this issue that I could find but good practical references can be found in Brownlee (2014).

For the present model six variables[4] were created.

1. Ratio variables:

   - *inv_part_01*: the ratio between *invitations* and *participations*.
   - *inv_part_02*: the ratio between *invitations_30* and *participations_30*.

2. Logistic Regression [5] variables: These four variables intend to compress into one numerical predictor all the predictors in each of the first four predictive dimensions - Demographic, Panel, Invitations & Participation and Points, by calculating the probabilities of being code 1 in the response variable *get_to_active*.

   - *glm_demographics*: *get_to_active* ∼ *country* + *gender* + *age* + *ses_standard*
   - *glm_panel*: *get_to_active* ∼ *app* + *seniority* + *recruitment*
   - *glm_inv_part*: *get_to_active* ∼ *invitations* + *participations* + *invitations_30* + *participations_30*
   - *glm_points*: *get_to_active* ∼ *points* + *exchanged_points* + *n_exchanges*

For instance, for getting *glm_demographics*, a logistic model was built using *country*, *gender*, *age*, *ses_standard* as covariates for predicting *get_to_active*.

```
# Logistic model using demographic predictors
glm_demographics <-
  glm(
    get_to_active ~ country + gender + age + ses_standard,
    data = training,
    family = "binomial"
```

[4]Note that the social class variable was also created from the transformation and equalization of many other variables. However, this was a qualitative process more than mathematical, hence, it will not be considered as resulting from what we are calling here feature engineering.

[5]A complete presentation of the theory of logistic regression can be fin in Hosmer Jr et al. (2013).

```
  )

# Get the summary of the model
summary(glm_demographics)

Call:
glm(formula = get_to_active ~ country + gender + age + ses_standard,
    family = "binomial", data = training)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.4316  -0.8493  -0.6480   1.2548   2.7351

Coefficients:
                        Estimate Std. Error z value Pr(>|z|)
(Intercept)           -1.8197477  0.0557020 -32.669   <2e-16 ***
countryBR              0.2014928  0.0169451  11.891   <2e-16 ***
countryCL              0.8840749  0.0230966  38.277   <2e-16 ***
countryCO              0.3302521  0.0186921  17.668   <2e-16 ***
countryDE             -0.2457299  0.0225758 -10.885   <2e-16 ***
countryES              0.2642184  0.0181679  14.543   <2e-16 ***
countryFR             -0.7942222  0.0312506 -25.415   <2e-16 ***
countryGB             -0.4736912  0.0221738 -21.363   <2e-16 ***
countryIT             -0.6385111  0.0330118 -19.342   <2e-16 ***
countryMX              0.3575067  0.0187270  19.090   <2e-16 ***
countryPE             -0.3316342  0.0262979 -12.611   <2e-16 ***
countryPT              0.3775403  0.0377673   9.996   <2e-16 ***
countryUS             -0.3412111  0.0175168 -19.479   <2e-16 ***
genderMan              0.2049333  0.0085948  23.844   <2e-16 ***
age                   -0.0182443  0.0003247 -56.192   <2e-16 ***
ses_standardHigh       1.6392791  0.0539171  30.404   <2e-16 ***
ses_standardMediumHigh  1.5615552  0.0534697  29.204   <2e-16 ***
ses_standardMediumLow  1.3088772  0.0539137  24.277   <2e-16 ***
ses_standardLow        0.9792770  0.0546667  17.914   <2e-16 ***
---
Signif. codes:  0 (***) 0.001 (**) 0.01 (*) 0.05 (.) 0.1 () 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 346780  on 294816  degrees of freedom
Residual deviance: 329010  on 294798  degrees of freedom
```

```
AIC: 329048

Number of Fisher Scoring iterations: 4

# Adding the predicted probabilities to the training and testing data
training[,
    glm_demographics :=
    predict(glm_demographics, training, type = "response")]
testing[,
    glm_demographics :=
    predict(glm_demographics, testing, type = "response")]
```

This procedure works as a data reduction process but the idea is not to substitute predictors by the compressed version but, on the contrary, add new options increasing the split option of the model's trees. The use of these four predictors did not have a direct effect in the model accuracy, which is good since it would be possible that the noise coming from each model's error would have a negative effect. What was observed in fact is that these variables are among the first ten variables when measuring the implemented model feature importance. Meaning, in general, that these variables were used multiple times in the model's trees splits. This is actually a good example on how tree based models can deal with feature selection and collinearity issues internally. The most important variables end up being used in the first splits of the trees while less informative variables end up being selected less often.

It is important to note that the logistic models were all trained using only the training data set, therefore the testing data remained unseen. The probabilities were then added to the same training data set using the predict function in R. Next, we use predict function again to impute the predicted probabilities for the testing data set.

None of the numerical predictors were neither centered nor scaled since monotonic transformations of the data will not change how the trees are grown (Breiman et al., 1984, p. 29). This is an interesting feature since it makes interpretation of the predictors more clear and demands less work in the data cleaning and preparation step of the analysis.

2.3.3 Model hyper-parameters tuning and training

The model training has the following basic steps:

- Choice of the model evaluation metric to be used (ROC/AUC and Logarithmic Loss) to be maximized or minimized.
- An extensive tuning parameter process using grid-search and cross-validation is used

to find best regions for the values of the main parameters.

- A fine tuning procedure is run for setting the learning rate parameter η and regularization parameter γ (the last term in Eq. 13).

- Final model is trained using the R library XGBoost and the function xgb.train(.) with the final selection of tuning parameters.

Using the training data a grid with the combinations of tuning parameters to be tested is set. In this case 270 combinations of tuning parameters were tested. The number of combinations can grow really fast and it can be very time consuming to run all models, even more when using cross-validation. For that reason, the number of iterations was kept limited from 10 to 100 iterations and some parameters were kept constant as the learning parameter η and the regularization parameter γ also known as the minimum loss reduction parameter. A thorough description of the available tuning parameter for XGBoost can be found in XGBoost-Developers (2018) and also in Chen et al. (2018). The three parameters which vary the most in this grid search are: the maximum depth of the trees which can not be too big since the algorithm is based on weak learners; the proportion of predictors to take into consideration when evaluating each split; and the subsample parameter which limits the number of training instances to use when building each tree. These are all controls which focus on avoiding over-fitting.

The following code snippet sets the grid-search to be executed. A brief explanation of each parameter is also given beside the row in the code setting the vector for each specific parameter. Note that the parameters η and γ are kept constant and will be tuned later.

```
# Load the needed libraries
library(caret)
library(xgboost)
# Setting the grid and tuning parameter for training the model
xgb_train_grid <- expand.grid(
  eta = c(0.3), # Controls the learning rate
  max_depth = c(4,6,8), # The maximum depth of each tree
  nrounds = seq(10, 100, by = 10), # Number iterations
  gamma = c(0), # Minimum Loss Reduction
  colsample_bytree = c(0.4, 0.7, 1), # Subsample Ratio of predictors
  subsample = c(0.4, 0.7, 1) # Subsample Ratio at each iteration
)
```

Next the cross-validation which is a class of re-sampling methods for estimating the test error of a model (James et al., 2013, p. 29), is set. The training data is partitioned into k folds of equal size, the model is trained using $k-1$ folds and the testing error is estimated using the remaining fold. The process is repeated for all possible combinations of the k folds and the test error is then averaged as in Eq. (14).

$$CV_{(k)} = \frac{1}{k}\sum_{t=1}^{k} \epsilon_t. \tag{14}$$

A crucial point in setting the cross-validation grid search strategy is which error metric to use for comparing the models. For binary classification, the two most common metrics are the Receiver Operator Characteristic (ROC) curve (Fawcett, 2006) and the Logarithm Loss (Painsky and Wornell, 2018).

For defining the ROC metric we need to understand the idea of a confusion matrix for binary classification. A confusion matrix is a cross tabulation of the true values of the response variable against the predict ones. Many evaluation metrics for classification algorithms can be derived from this matrix in order to evaluate the quality of the predictions.

		True response		
		p	**n**	**Total**
Predicted response	**p′**	True Positive (TP)	False Negative (FP)	P′
	n′	False Positive (FN)	True Negative (TN)	N′
	Total	P	N	

Table 4: Confusion Matrix.

It is not the purpose to go through all of them here but only the two used for constructing the ROC curve which are the true positive rate $TPR = TP/P$ used on the y-axis and the false positive $FPR = FP/N$ rate on the x-axis. The ROC curve represents the relative tradeoff between benefits (TP) and costs (FP). In the context of our application, the tradeoff between the benefit of not inviting a panelist to become Premium and the cost of inviting a panelist who will not accept the invitation. In order to use the ROC curve to compare models, it must be expressed into a unique value. This is done by calculating the area under the ROC curve, denoted AUC. The common interpretation of the AUC is that it is the probability that the classifier will rank a randomly chosen positive instance higher than a randomly chosen negative instance (Fawcett, 2006, p. 868). The final goal is to find the tuning parameter combinations which will maximize the AUC calculated on the test instances of the cross-validation folds.

The logarithmic loss, on the contrary, needs to be minimized and basically works by penalizing wrong classifications. Mathematically, the logarithm loss for a binary classification problem is defined as:

$$logloss = -\frac{1}{n}\sum_{i=1}^{n}[y_i \log \hat{p}_i + (1-y_i)\log(1-\hat{p}_i)], \tag{15}$$

where n is the number of cases in the training data set, y_i are the true response variable and $\hat{p}_i$ is the probability predicted by the model. The intuition here is that if my actual response variable is $y_i = 1$, if my estimated probability is high, say $\hat{p}_i = 0.9$, the logarithmic loss will be small and vice-versa, penalizing more the wrong predictions which are very confident, as for instance, in this case, $\hat{p}_i = 0.1$ would be. This is an interesting metric in the context of our application since more than the classification the probabilities will be used to target who to invite to become a Premium panelist.

Both metrics were used for finding the ideal tuning parameters combination. However, no differences were found in the selected combinations, meaning the two metrics seem to agree in the context of the presented data set.

Using the caret function trainControl(.) we set a 5-fold Cross-Validation and the metrics to be used when evaluating the hyper-parameters combinations in the parameters grid. The argument “summaryFunction” is used to set the evaluation metric. When set equal to “twoClassSummary” the ROC curve and AUC will be used. This code is run twice, being the second run used to evlaute the grid search using the Logarithmic Loss by setting “summaryFunction” equal to “mnLogLoss”.

```r
# Setting training controls
xgb_train_control = trainControl(
  method = "cv",
  number = 5,
  summaryFunction = twoClassSummary, # mnLogLoss (used in the second run)
  classProbs = TRUE,
  verboseIter = TRUE,
  allowParallel = TRUE
)
```

Finally, the model is run for all combinations of tuning parameters in the grid keeping record of the cross-validation test error of each. Note that the following code is also run twice for considering both evaluation metrics.

```r
set.seed(3185)
```

```
start_time <- Sys.time() # keep control of running time
xgb_model_caret = train(
  x = data.matrix(training[, -5]),
  y = training[, get_to_active],
  trControl = xgb_train_control,
  tuneGrid = xgb_train_grid,
  metric = "ROC", # "logLoss" (also used)
  method = "xgbTree"
)
end_time <- Sys.time()
end_time - start_time # around around on hour to train
```

The best tuning parameters combination found using both metrics, ROC - with maximum value of 0.8949546 - and the Logarithmic Loss - with minimum value of 0.3465119 - has the following values for each one of the hyper-parameters:

nrounds	max_depth	η	γ	colsample_bytree	min_child_weight	subsample
60	6	0.3	0	0.7	1	1

Table 5: Best tuning parameters

The relation between the different combinations of tuning parameter and, for instance, the ROC metric can be observed in the figure 1. One can observe that a maximum depth of 8 will quickly lead to a bad performance as the number of iterations increase, getting worse as the subsample ration gets smaller. Also, it is interesting to observe that the smaller maximum depth improves slowly in the final iterations and it crosses with the maximum depth equal 6 and also tends to issue good results.

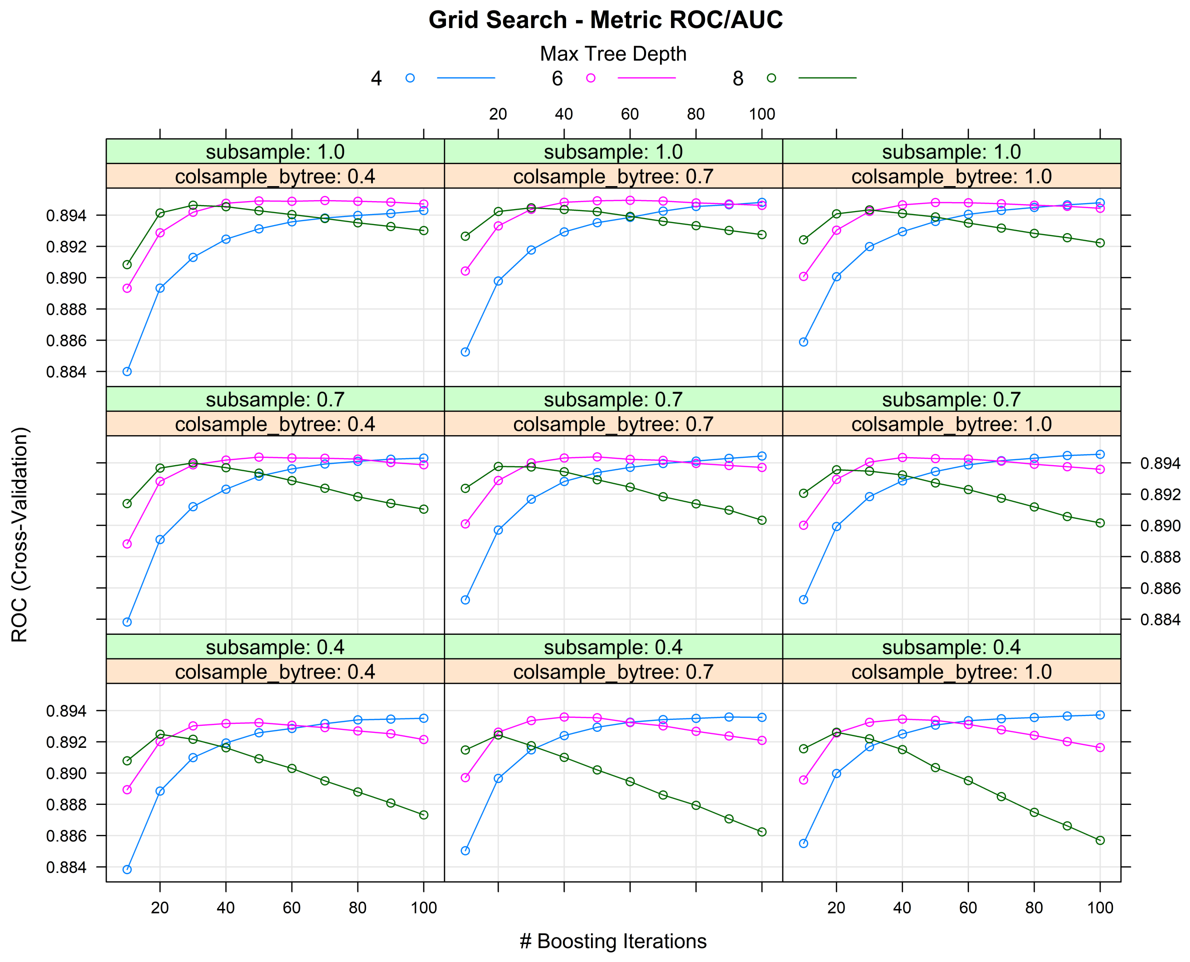


Figure 1: Grid search results using ROC metric.

A similar behaviour is observed when plotting the results based on the Logarithmic Loss metric as one can observe in figure 2. The only difference is that with the Logarithmic Loss the goal is to minimize the metric value.

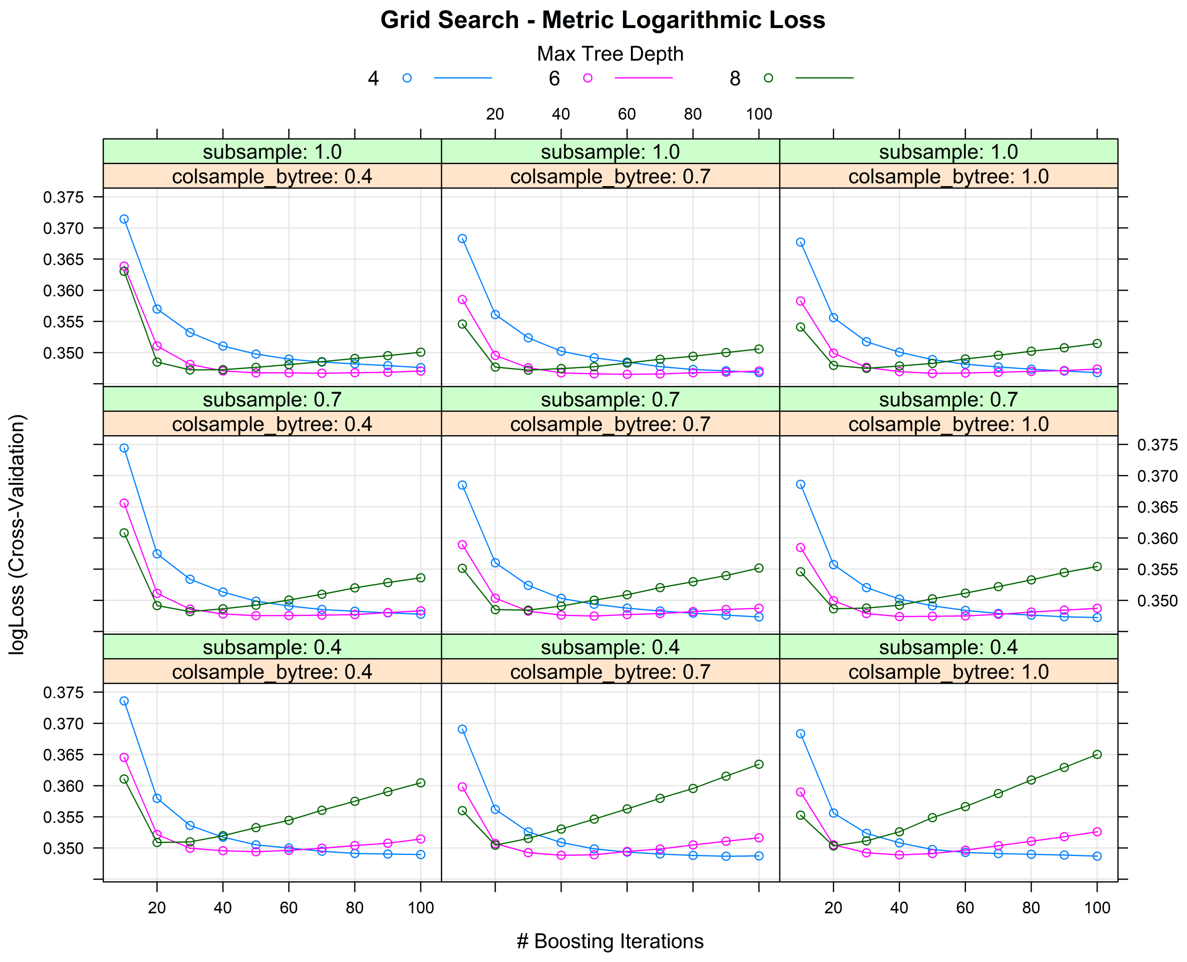


Figure 2: Grid search results using Logarithmic Loss metric.

Using the first tuning parameters combination, the next step is fine tuning important hyper-parameter as η and γ. With the function xgb.cv(.) from the XGBoost R library, the model will be trained using cross-validation and different values for these parameters. First, the learning parameter η is varied, the tried values were (0.5, 0.1, 0.2 and 0.3). The γ value was kept constant at 0.

An important point here is the use of the additional hyper-parameter "scale_pos_weight", used for controlling the imbalance in the response variable. This extra parameter uses a cost sensitive approach by changing the weights of the positive observations. The idea is to take care of the imbalance problem by increasing the loss if it fails to correctly predict the response. The proportion of negative class over positive class is used to set the value for this parameter. Using this parameter tends to have a negative impact in the overall accuracy of the model, however it tends also to improve the classification error for each specific class defined in terms of sensitivity (TP/P) and specificity (TN/N) based on the confusion matrix. The use of this parameter will establish the building of two different models, first a simple model which will not control for class imbalance and secondly a class imbalance model which will have such a control.

```
# Praparing data for XGBoost
matrix_training <- as.matrix(training[, -c("get_to_active")])
matrix_testing <- as.matrix(testing[, -c("get_to_active")])
response_training <- as.numeric(training[, get_to_active]) - 1
response_testing <- as.numeric(testing[, get_to_active]) - 1
# XGBoost works better if data is converted into the xgb.DMatrix format
data_training <-
    xgb.DMatrix(data = matrix_training, label = response_training)
data_testing <-
    xgb.DMatrix(data = matrix_testing, label = response_testing)
# Fine tuning eta - Setting parameters
params <- list(
  booster = "gbtree",
  objective = "binary:logistic",
  eta = 0.05, # values of eta tested (0.05, 0.1, 0.2 and 0.3)
  gamma = 0,
  max_depth = 6,
  subsample = 1,
  colsample_bytree = 0.7,
  # scale_pos_weight controls for the class imbalance problem
  scale_pos_weight = (length(response_training) -
                  sum(response_training))/sum(response_training))
# Run the model
xgbcv_eta <- xgb.cv(params = params
                       ,data = data_training
                       ,nrounds = 300
                       ,nfold = 5 # cross-validation set
                       ,showsd = T
                       ,stratified = T
                       ,print_every_n = 20
                       ,eval_metric = "auc") # AUC metric
```

The results are displayed in Table 6 which has the maximum ROC/AUC value for each η and the figure 3 which compares the training error and testing error across the iterations for each value of η.

Learning rate (η)	0.05	0.1	0.2	0.3
ROC/AUC max.	0.8953456	0.8957982	0.8952526	0.8947676
Number of iterations where (η) is max.	200	195	97	58

Table 6: Results fine tuning η.

The results are a clear example on how the learning rate parameters influences results, the higher the learning rate the smallest the number of iterations needed to reach the maximum value of ROC/AUC a model can get. Also, when observing the curves in figure 3, one observes that for $\eta = 0.05$ the ROC/AUC value grows much slower and seems to be still growing at the maximum iteration tested of 200. For that reason, η will be fixed at 0.05 thus increasing the number of iterations when fine tuning the hyper-parameter γ for regularization.

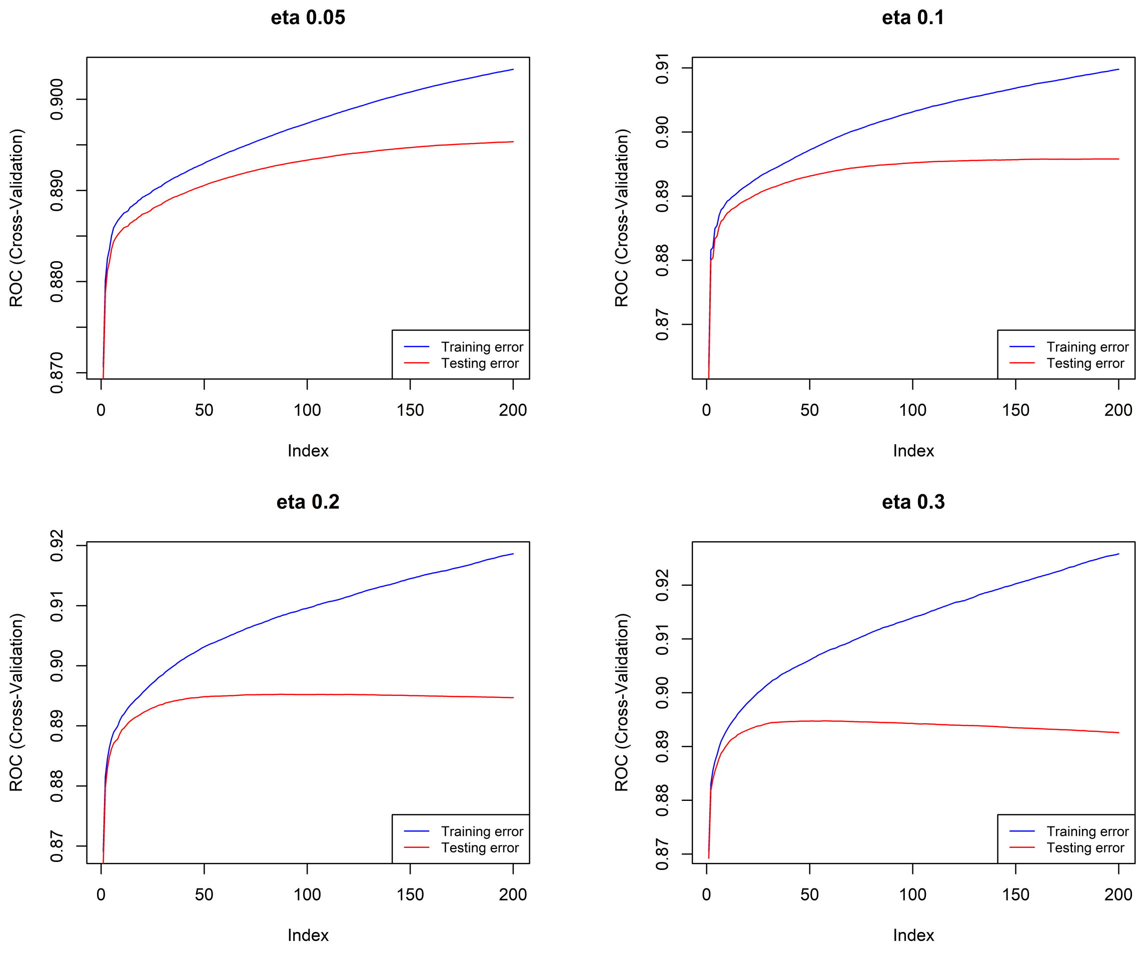


Figure 3: Test and training error for multiple values of the learning rate parameter η.

The last step before training the final model is to fine tune the regularization parameter γ, the last term in the Eq. 13. This parameter varies from 0 to infinite where 0 means no regularization. Mathematically, it is a Lagrangian Multiplier used for controlling the model's complexity. The higher the value of γ, the smaller the difference between test and training error tends to be. Fine tuning γ is in general a better option than using other parameters available as, for instance, "min_child_weight". The reason is that γ prune a shallow tree using the loss function instead of using the hessian weight (gradient derivative) as it is the case with "min_child_weight" (Laurae, 2018).

The results of fine tuning gamma for values 5, 10, 15 and 20, keeping $\eta = 0.05$ and number of iterations 300 a presented in the table 7 and figure 4.

Regularization parameter (γ)	5	10	15	20
ROC/AUC max,	0.89591	0.8959638	0.8958482	0.8954756
Number of iterations where (η) is max.	300	300	300	300

Table 7: Results fine tuning γ.

The results are very similar in terms of ROC/AUC metric. In all cases the maximum ROC/AUC is reached at iteration 300, indicating that we could try a higher number of iterations in the final training. The γ will be fixed at 10 since it led to the highest observed ROC/AUC.

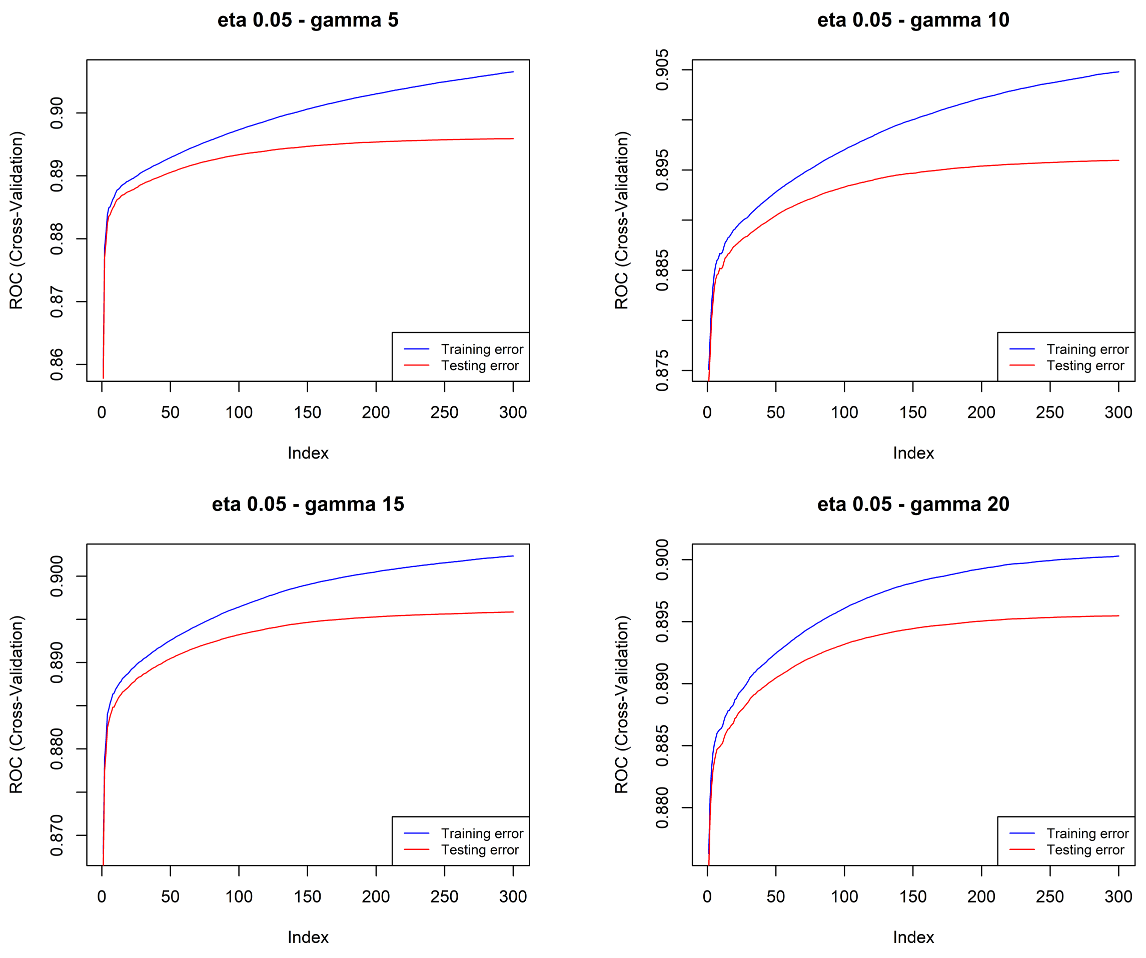


Figure 4: Test and training error for multiple values of the regularization parameter γ.

The final model was trained twice, first controlling for class imbalance using the parameter "scale_pos_weight" and next without it. The function used here is the xgb.train(.) from Chen et al. (2018).

```r
# Setting parameters for the imbalance final model
params <- list(
booster = "gbtree",
objective = "binary:logistic",
eta = 0.1,
gamma = 10,
max_depth = 6,
min_child_weight = 1,
subsample = 1,
colsample_bytree = 0.7,
# "scale_pos_weight" controls the balance of
# positive and negative class weights
scale_pos_weight = (length(response_training) -
sum(response_training))/sum(response_training))
# Training the final models (xgb_final_imbalance and xgb_final)
xgb_final_imbalance <- xgb.train(
params = params
,data = data_training
,nrounds = 350 # 50 extra iterations were allowed in the final model
,watchlist = list(val = data_testing, train = data_training)
,print_every_n = 20
,eval_metric = "auc"
)
```

The final model controlling for class imbalance has a validation maximum ROC/AUC value of 0.89669 at iteration 321 while the model not controlling for class imbalance had a ROC/AUC of 0.896501 at iteration 276. In the next section, we will evaluate the two models' performances using the testing data. Both models have the same maximum accuracy of 0.8479766, however, different cutoffs must be select to achieve this maximum accuracy in each model. The R library Sing et al. (2005) will be used for plotting the accuracy against the cutoff options, the ROC curves and calculate the area under the curve AUC.

2.3.4 Models Evaluation

We have two final models, the first one is "xgb_final", which has no class imbalance control and will be called simple model. The second, "xgb_final_imbalance" controls for class imbalance and will be called imbalance model. The goal of this section is to evaluate and compare the performance of both.

The first step will be to select a cutoff value for classifying cases based on the predicted probabilities. The selected cutoff is in general the one maximizing the model's accuracy. For both models the maximum accuracy is the same and has the value 0.8479766, meaning that this is the proportion of correctly classified observations. As one can observe in the figure 5 built using the R package ROCR Sing et al. (2005), for each model a different cutoff must be used in order to get the maximum accuracy from the classification. More specifically, for the simple model the cutoff should be at 0.498 and for the imbalance model it should be 0.721 if the goal is to have maximum accuracy.

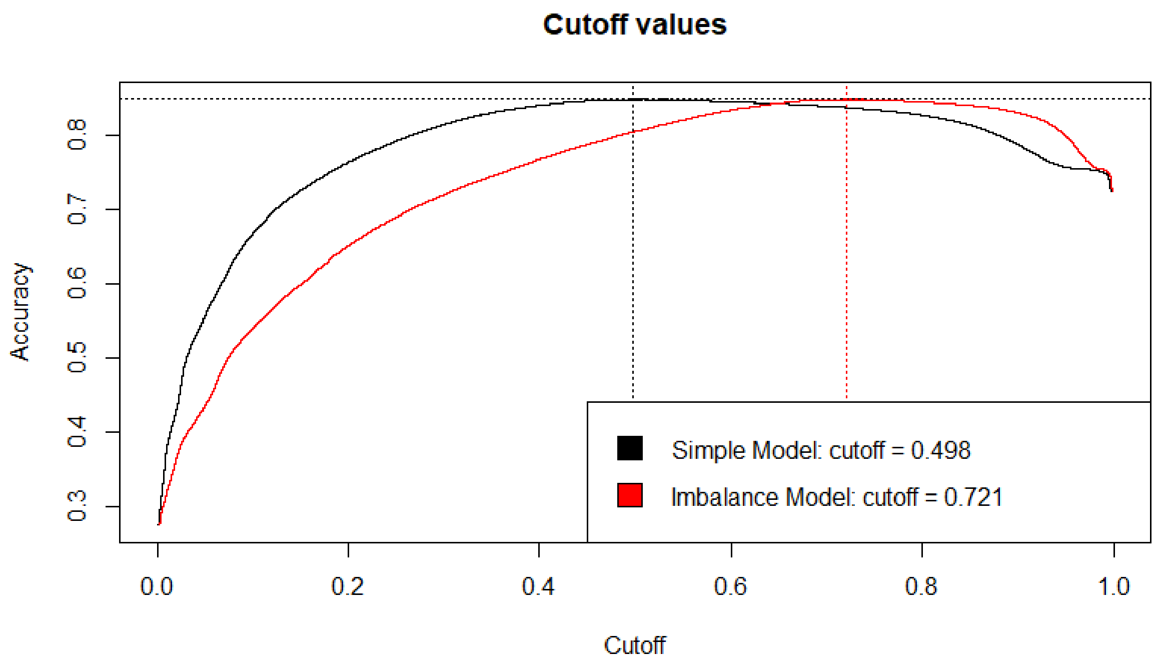


Figure 5: Accuracy against cutoff values for both models.

The next natural question would be how well do the predictions are able to separate the classes in each model. Again using the library ROCR a density plot of positive and negative classes against the cutoff can be drawn for assessing this information visually. Observing the figure 6 it is clear that the model controlling for the imbalance in the response variable achieves a much better separation of the classes. The active class seems to be bi-modal in both models. However, in the second the peaks are more accentuated. Despite the improvement in the ability in separating the classes, the area under the red curve in the imbalance model for cutoffs higher than 0.4 is bigger than in the simple model, hence the error in classifying the inactive class will also be higher.

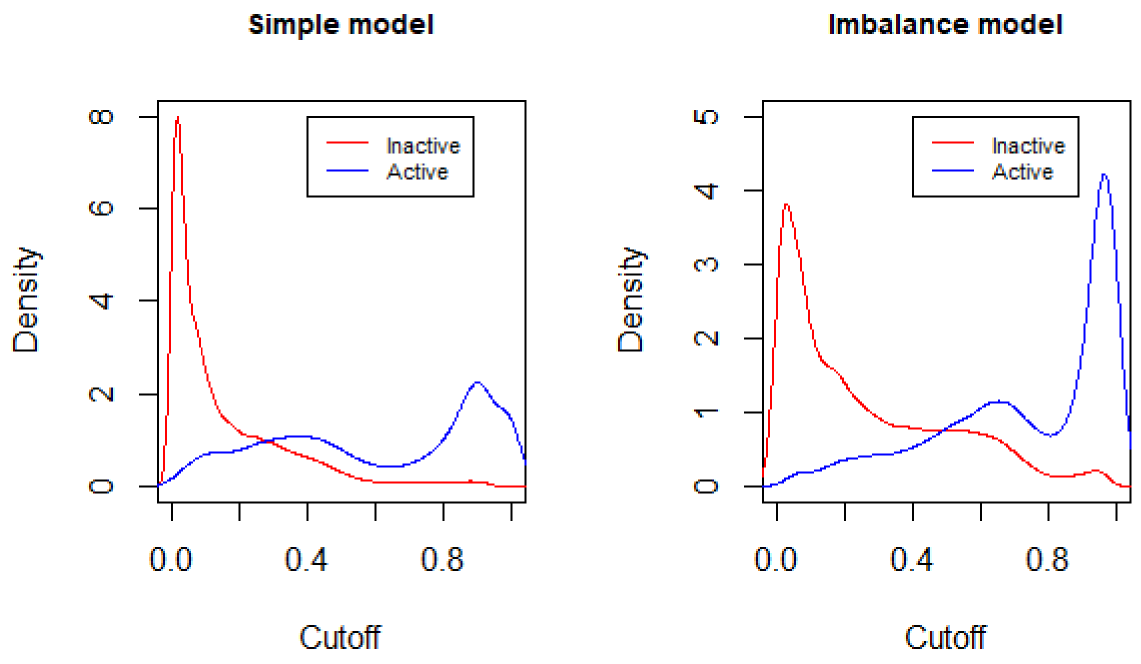


Figure 6: Class density and separation for both models.

Figure 7 displays the ROC (Receiver Operating Characteristic) curves (Fawcett, 2006) for both models. The curves are very similar and the models have almost the same AUC - 0.8965 for the simple model and 0.8967 for imbalance model. The area under the ROC curve will get smaller as the superposition of the density curves in figure 6 get bigger and vice-versa. ROC curves are commonly used to characterize the sensitivity/specificity tradeoff for binary classifiers by plotting the true positive rate - how often we correctly classify the positive class - against the false positive rate - how often the model wrongly classified the positive class. Each point along the curve corresponds to a possible classification cutoff and the colors of the curve help to make the link between the point in the curve and its corresponding cutoff value. As one can observe, the color distribution over each models' curve is not the same, meaning that for achieving the same true positive rate. In both models different cutoffs should be used. For maximizing the sensitivity in the simple model, a cutoff around 0.3 would be used while for maximizing it in the imbalance model, a cutoff around 0.5 would work better.

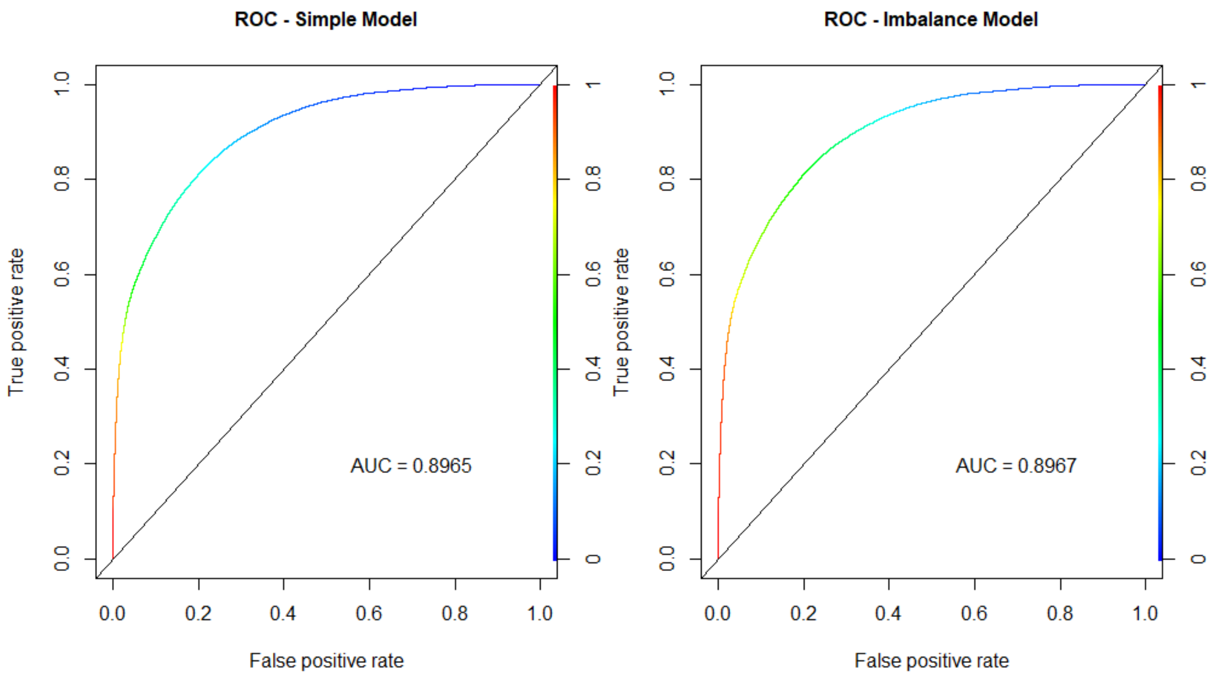


Figure 7: ROC curves for both models.

The problem here is that when using the cutoff for maximizing sensitivity one could get a worse accuracy or a worse specificity - the true negative rate. Selecting the best cutoff and balancing the sensitivity and specificity against accuracy depends partially on the application goals. In the present case, we want the best sensitivity we can get without deteriorating too much the model accuracy. In other words, converting invited panelists into active panelists is more important than wrongly inviting panelists which will not convert to active since the cost per invitation is almost null.

The point here is to quantify how much loss in accuracy is acceptable for which kind of improvement in sensitivity. The two models will be compared in such a manner that for the simple model the cutoff maximizing accuracy (0.49) will be used while for the imbalance model the cutoff maximizing sensitivity (0.50) will be applied. For comparing the models the confusion matrices in Tables 8 and 9 and the derived metrics in Table 10 will be used.

	Reference	
Predicted	Inactive	Active
Inactive'	87035	14635
Active'	4574	20105

Table 8: Confusion Matrix: Simple Model

	Reference	
Predicted	Inactive	Active
Inactive'	73720	6726
Active'	17889	28014

Table 9: Confusion Matrix: Imbalance Model

Confusion Matrix Metrics

Metrics	**Simple Model**	**Imbalance Model**
Accuracy	0.848	0.8052
No Information Rate	0.725	0.725
Kappa	0.581	0.5557
Sensitivity	0.5787	0.8064
Specificity	0.9501	0.8047
Pos Pred Value	0.8147	0.6103
Neg Pred Value	0.8561	0.9164
Prevalence	0.2750	0.2750
Detection Rate	0.1591	0.2217
Detection Prevalence	0.1953	0.3633
Balanced Accuracy	0.7644	0.8056
Positive Class	Active	Active

Table 10: Comparing confusion matrix metrics for both models.

For evaluating accuracy, the ideal starting point is to compare it with the no information rate which indicates how often the model would be wrong if it always predicted the majority class. The no information error rate is the error rate that would be observed if the predictors and the response variable were independent. In practice the no information is calculated as the largest class percentage in the data.

The difference between accuracy and the no information rate can be assessed by measures of agreement, as for instance, the Cohen's kappa (κ) (Cohen, 1960) coefficient of agreement which, despite of all the criticism as presented in Pontius Jr and Millones (2011), is still being used in practice. This metric is defined as follows:

$$\kappa \equiv \frac{p_0 - p_e}{1 - p_e} = 1 - \frac{1 - p_0}{1 - p_e}, \tag{16}$$

where p_0 is the model's accuracy and p_e is probability of random agreement defined as:

$$p_e = \frac{1}{N^2} \sum_k n_{k1} n_{k2}, \tag{17}$$

where k is the number of categories, in our case 2, N is the total number of cases in the data set, n_{k1} is the number of cases the model predicted as being in the negative class and n_{k2} is the number of cases the model predicted as being in the positive class. The value p_o can be interpreted as proportion of units which agree and p_e as the expected proportion of agreement as discussed in Rosenfield and Fitzpatrick-Lins (1986). The closer κ is to 1 the better the models' performance in comparison with the null information rate model. The κ is part of a family of of bi-variate agreement coefficients (Krippendorff, 1970) that can be easily interpreted as:

$$Agreement = 1 - \frac{observed \quad disagreement}{expected \quad disagreement}. \tag{18}$$

Observing the results in table 10, the simple model has a better accuracy however, the difference in the κ metric is very small. When comparing the sensitivity - the true positive rate or the proportion of active panelists correctly classified by the model - of both models, the difference is impressive, sensitivity goes from 0.58 in the simple model up to 0.81 in the imbalance model. The improvement in sensitivity however, comes with a small decrease in the value for specificity. This results were already reflected in the density-cutoff plot 6 where the imbalance model appeared with a much more clear separation of the classes.

Besides the small difference in the κ values, a second approach to evaluate the size in the accuracy difference between the models is to look at what is called the balanced accuracy. As discussed by Brodersen et al. (2010), accuracy by itself has two important problems. First, since it is estimated by averaging the accuracy of each of the cross validation folds, thus presenting limitations for calculating confidence intervals. Secondly, it leads to an optimistic estimate when a biased classifier is tested on an imbalanced data set, what might be the case here. The authors approach to overcome the problems is to substitute the non-parametric approach of the simple accuracy calculating the posterior distribution of the balanced accuracy. In table 10 the balanced accuracy for the simple model is smaller than the balanced accuracy in the imbalance model. This reflects the better proportionality between sensitivity and specificity observed in the imbalance model. As mentioned in (Brodersen et al., 2010, p 2), if a learner performs equally well predicting both classes the balanced accuracy reduces to the simple accuracy, exactly as observed in the imbalance model results.

Prevalence is the proportion of positive class in the data set, which is the same for both models. However, the detection prevalence which is the proportion of cases the model predicts as positive over all cases is bigger in the imbalance model as well as the detection rate which measures the proportion of cases correctly classified as positive among all cases in the data set.

The last two metrics are the positive predicted value or the proportion of true positive among all values predicted as positive and the proportion of true negatives among all cases predicted as negative. From this point of view the simple model is more balanced. The imbalance model predicts many more cases as positive in order to increase sensitivity causing a decrease in the positive predicted value.

Conclusion is that both models are good enough for being used. Using the simple model would assure the best accuracy and would better predict panelists who would not accept the Premium proposal while the imbalance model would better predict panelist who would accept the Premium proposal, being a more balanced model in terms of sensitivity and specificity despite the smaller value for accuracy.

2.4 Model interpretation - Variable Importance

The last step is to evaluate which are the variables that most contribute for the predictions. This might be helpful when explaining the model for a business audience, since it is easy to interpret. But it has also some drawbacks, since different measures of variable importance might have very different results.

Variable importance can be calculated using the function xgb.importance(.) of the library XGBoost. This functions offers three different methods for calculating variable importance:

- Gain: based on the total reduction of loss or impurity contributed by all splits for a given feature (Breiman et al., 1984).
- Cover: metric of the number of observation related to this feature.
- Weight: percentage representing the relative number of times a feature has been taken into trees.

These measures are criticized because they might lead to inconsistent results, meaning a model can change such that it will rely more in a certain feature but the feature importance might decrease instead of increase. This problem is discussed in Lundberg et al. (2018), where the authors discuss the limitations of the three most common measures of variable importance and develop a new approach called SHAP (SHapley Additive exPlanation), which offers a solution for the inconsistency problem of the mentioned methods. However, the implementation in R of the SHAP approach is fairly new and still unstable. Therefore, the models will be evaluated using the most common approach based on the gain.

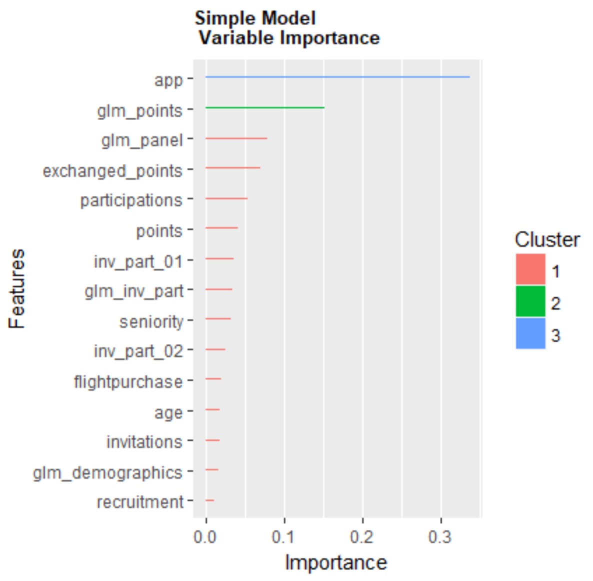
A comparison of both models variable importance based on gain for the 15 most important variables can be observed in the figure 8. The first point to observe is that the first three variables are exactly the same although "glm_points" have a higher importance in the imbalance model. When comparing the first 14 variables one can see that they are the same for both models but appear in different orders. Only the last variable is different for each model.

By far, the most important variable is “app”, which stands for panelists who have the company mobile application installed or not. This is a very interesting information indicating that the company should give more incentive for panelists installing the app before sending them the Premium proposal.

An interesting point of the plot is the fact that all the variables created using feature engineering are among the most important variables, which is an example of the power such techniques have. The majority of variables are, nevertheless, indicators of the panelists' relationship and status in the Access Panel. Demographic variables, however, are in smaller numbers among the most important ones. Only the “glm_demographics” and “age” appear in the list. This is an important finding once demographic variables are used to balance the panel sizes given the population distributions in each country. If predicting conversion to active was strongly dependent on the demographic variables, balancing the panel sizes would be more complicated, since the invitations are based on the estimated probabilities.

It is also clear that the model relies heavily on the invitations and participation dynamic.

So it is important that the company is able to keep a constant flow of invitations so panelists can engage more, increasing participation and accumulation of points, thus having bigger chances of buying in the company online store. This makes sense, since for seeing the value of the Premium proposal a panelist must have an understanding of what the proposal means in terms of real value. In other words, what they would be able to buy if they accept the proposal. Also, since the proposal rewards more panelists who participate, the proposal will not be appealing neither for panelists who do not participate regularly nor for panelists who do not receive enough invitations to participate and would be willing to do so. Controlling this dynamic brings many challenges to the company, since the number of invitations depends on the demand of clients for projects, which are seasonal. That means, for instance, that invitations in periods of the year with less projects should be avoided. Also if there is a limited number of projects, invitations should be focused into those panelists which are known to be more participative, so their engagement is kept strong.



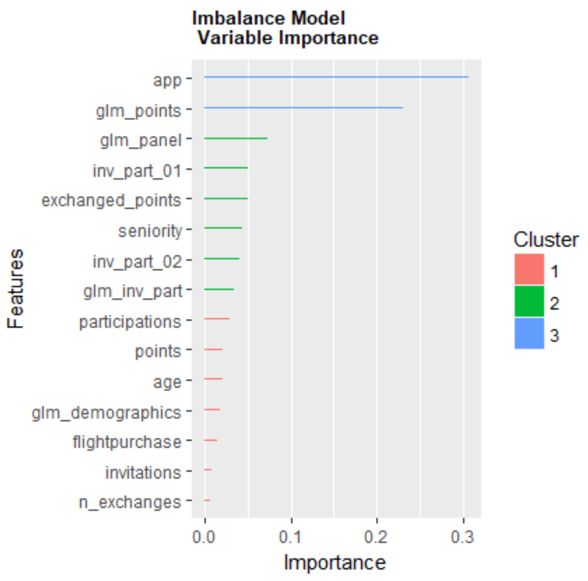


Figure 8: Variable importance for both models.

Finally, the last comment refers to the variables of Internet activity. Only one of those figures - flightpurchase - is among the most important variables, meaning these group of variables might be excluded from the model. It is also possible that the fact that these variables have many missing values has harmed the potential they might have. Therefore, a second alternative could be incentivate more panelists to answer such questions and experiment in training future versions of this model.

2.5 Results and Discussion

In this section the results of a real invitation or flagging procedure will be presented and the performance of the model in a real application will be assessed.

The company needed to increase the number of Premium panelists in a specific group of countries - Argentina, Brazil, Chile, Colombia and Mexico. These panelists also needed to have specific demographic characteristics regarding gender, age, social class and geographic position. For each of these characteristics, a non-crossed quota was established based on the needs of balancing the Meter Panel with respect to the population distribution.

Using the XGBoost simple model, the conversion probabilities were calculated and assigned to each panelist pertaining to the targeted groups. These probabilities were ordered from the highest to lowest inside each group and a cumulative sum was used to determine how many panelists should be invited in order to reach the targeted quotas.

Initially, a cutoff of 0.5 was established. However, for many groups it was too high and panelists with smaller probabilities ended up being invited in an attempt to reach the targeted quotas. For some groups, the overall estimated probabilities were smaller than for others, meaning that a higher number of panelists would have to be invited in the first ones in order to achieve similar results as in the second ones. Although the invitation of panelists with probabilities below the cutoff was not recommended, the pressure for achieving the wanted quotas were stronger and panelists with probabilities under 0.5 were also invited.

Negotiating strategies based on statistical results is not always easy in a business context, specially when these results involve chance. An interesting discussion about chance and how to talk about it to non-statisticians audience can be found in (Blastland and Dilnot, 2008, Chapter 5) specially focusing on how non-intuitive chance related concepts can be. Complex models as XGBoost are not easy to explain for an audience of non-statisticians and ideally the results must be somehow converted to a financial framework since business decisions are largely driven by financial results.

In the specific case of our application, the argument used for trying to demote the managers of inviting people with probabilities smaller than 0.5 was that a low probability of conversion might indicate a smaller chance of retaining the panelist sharing data for long periods. For instance, a panelist which has 0.2 chance of converting to active might share data for only 1 or 2 days what would not be enough for covering the cost of 100 points paid to the panelist independently of the the amount of time this panelist will keep sharing Internet navigation data. However, the argument was not strong enough and the invitations were made.

The table 11 summarizes the results of the flagging procedure after 96 days from the invitation. Overall, 72% of the expected conversion was achieved. Argentina showed the best results while Brazil was close to only 50%.

	Argentine	Brazil	Chile	Colombia	Mexico	Total
Achieved Conversion	832 (90%)	527 (54%)	860 (74%)	611 (80%)	805 (68%)	3635 (72%)
Expected Conversion	927	979	1.170	765	1.182	5.023
Total Invited	1.700	1.723	1.918	2.472	2.228	10.041

Table 11: Results of an invitation procedure carried out in 02-05-2018.

Why the achieved conversion was not the expected?

The first reason is the limitation inherent to our model. Accuracy was around 84% and the simple model which was used here had a better specificity of 0.95 and a sensitivity of only 0.58, so it performed better predicting panelists who would not accept the Premium proposal than the ones who would have accepted it.

Secondly, other sources of variability played a role here. The invitation is designed as a process which has many steps associated with different statuses assigned to each panelists.

The transition between one status and another depends on many circumstantial rules. For instance, a panelist must participate in a survey in the Access Panel in order to see the invitation. This means the process is dependent on the demand for projects requiring a specific profile of panelists at the same time they are invited.

In the next chapter, the entire process from the moment a panelist get invited will be explained in detail. Each panelist invited to become Premium has a life trajectory inside the Meter Panel, which is described by the statuses sequences and the associated rules and actions. Such life trajectories will be described and studied in depth. But it is important to understand that at each step expressed by a status there is a risk associated of losing the panelist which is reflected in the fact the real flagging did not achieve its full expected conversion.

3 Status Sequence Analysis

3.1 Definition

The Meter Panel consists of all panelists in the Access Panelists which have been invited to become a Premium/Meterized panelist and start sharing their Internet navigation data in exchange for extra points.

Incentives for accepting the Premium Proposal are:

- 100 points when the first device is installed;
- 5 extra points per active device at the end of each survey while the user is still a Premium participant and the device is active. With a maximum of 15 points.

There are two main types of invitation:

- **Flagging:** a Predictive Model calculates the probability of a panelists accepting the Premium proposal and panelists are then invited based on their assigned probabilities and the Meter Quotas control.
- **Direct Meter:** panelists are invited to the Access Panel and Meter Panel simultaneously. No conversion probability is estimated here.

The life trajectory of a panelist in the Meter Panel is controlled/described by a series of statuses and associated transition rules.

Transition rules are of two types:

- number of participations in surveys;
- number of days.

Figure 9 has a diagram describing how the different statuses are associated, by which transition rules they are linked and which communication action is taken when transitioning. The arrows without a transition rule specified depend on actions taken by the panelist or by a Netquest agent so they are reactive links and not rule based.

There are **three classes of statuses** which have different colors associated with - green, blue and red:

1. Green statuses in the diagram 9 encompasses panelists which can potentially accept the proposal and become active:
   - **PRE_PROPOSAL_START:** have been invited but have not seen the proposal yet;
   - **PROPOSAL:** have seen the Premium proposal at least once;
   - **REMINDER:** have done 5 participations after seeing the proposal but has neither accepted it nor refused it;
   - **ACCEPTED:** have accepted the proposal and received instructions on how to install the meter application
   - **NOT_INIT:** have not installed the meter application 5 days after accepting the proposal.
2. Blue statuses in the diagram 9 encompass panelists which have installed the tracker and are sharing data (ACTIVE) or have shared data once before (INACTIVE):
   - **ACTIVE:** panelist has installed the meter application and is sharing data;
   - **INACTIVE** panelist has stopped sharing data 7 or more days ago.
3. Red statuses in the diagram 9 encompass panelists which are not part of the Meter Panel anymore:
   - **REFUSED:** refusal at the moment of the invitation;
   - **OUT:** panelist has not accepted the proposal or has not installed the application after having accepted the proposal;
   - **CANCELLED:** a panelist or the company itself can cancel a Premium account at any time;
   - **LOST:** panelist has been inactive for a very long time.

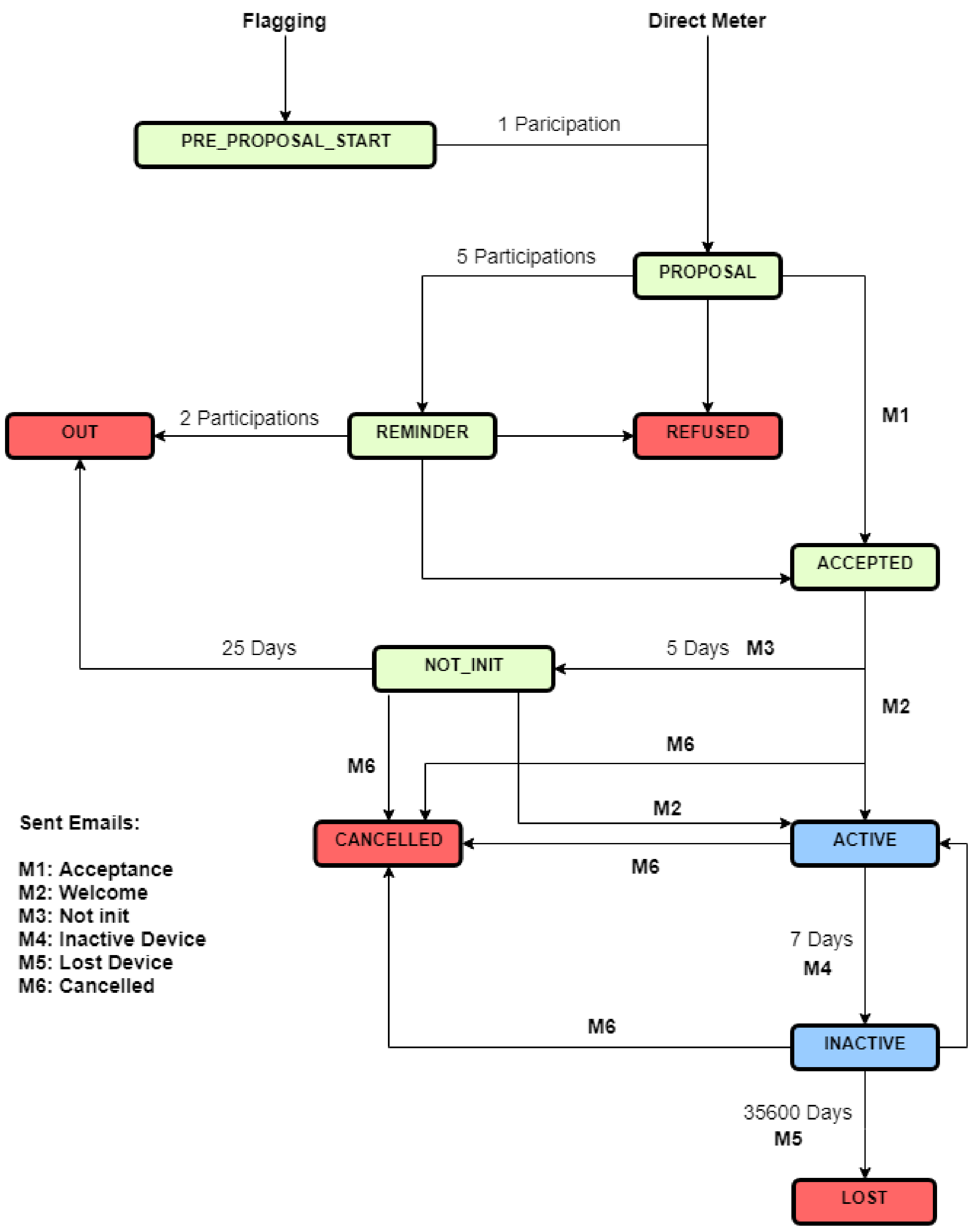


Figure 9: Meter Status Diagram.

This is a dynamic and complex process and it is important to have summary pictures to describe it and try to make it more intuitive.

Continuing the evaluation of the results of the flag presented in table 11, the following chart describes the curves of the statuses along the first 96 days after the invitation.

The horizontal axis represents the days from the moment each panelist got assigned the status PRE_PROPOSAL_START and the vertical axis counts how many panelists were at each status in each day. Therefore, for each day the total number of panelists is always the same. In the title it is specified “Only state = 70”, meaning only active panelists in the Access Panel were considered. It is important to note that the Meter Panel is a subset of the Access Panel, therefore panelists which are excluded from Access Panel will be set as CANCELLED in the Meter Panel.

Note that the time base measure is in days, however, a panelist might change of status multiple times in the same day. In this chart we will always consider the last status of the day. That is why in day 1 we have panelists with status ACTIVE although the first status they had in the same day was PRE_PROPOSAL_START.

Lines are grouped by colors spectrum. Green lines for the initial statuses, grey/black for intermediate, blue for ACTIVE and INACTIVE and red for panelists out of the Meter Panel.

One can observe how fast the conversion to ACTIVE is and its relation with panelists becoming INACTIVE. Also, we see how consistent the results are since after 96 days the number of active panelists is stable and not dropping. Since the number of INACTIVE panelists is growing, it means there is a transference mechanism going on, new ACTIVE panelists take the place of the ones going to INACTIVE, at least for the observed period.

The number of panelists with status OUT starts to grow a bit faster from the 30th day, which is consistent with the transition rules since a panelist in status NOt_INIT will stay a max of 25 days like that before going to status OUT. Other similar statuses as CANCELLED and REFUSED are residual.

Even after 96 days, many panelists still in status PRE_PROPOSAL_START and PROPOSAL. The first ones have never seen the proposal and the second ones have seen it at least once but are ignoring it. These two statuses work as filters, in general panelists which do not participate end up getting stuck on these steps.

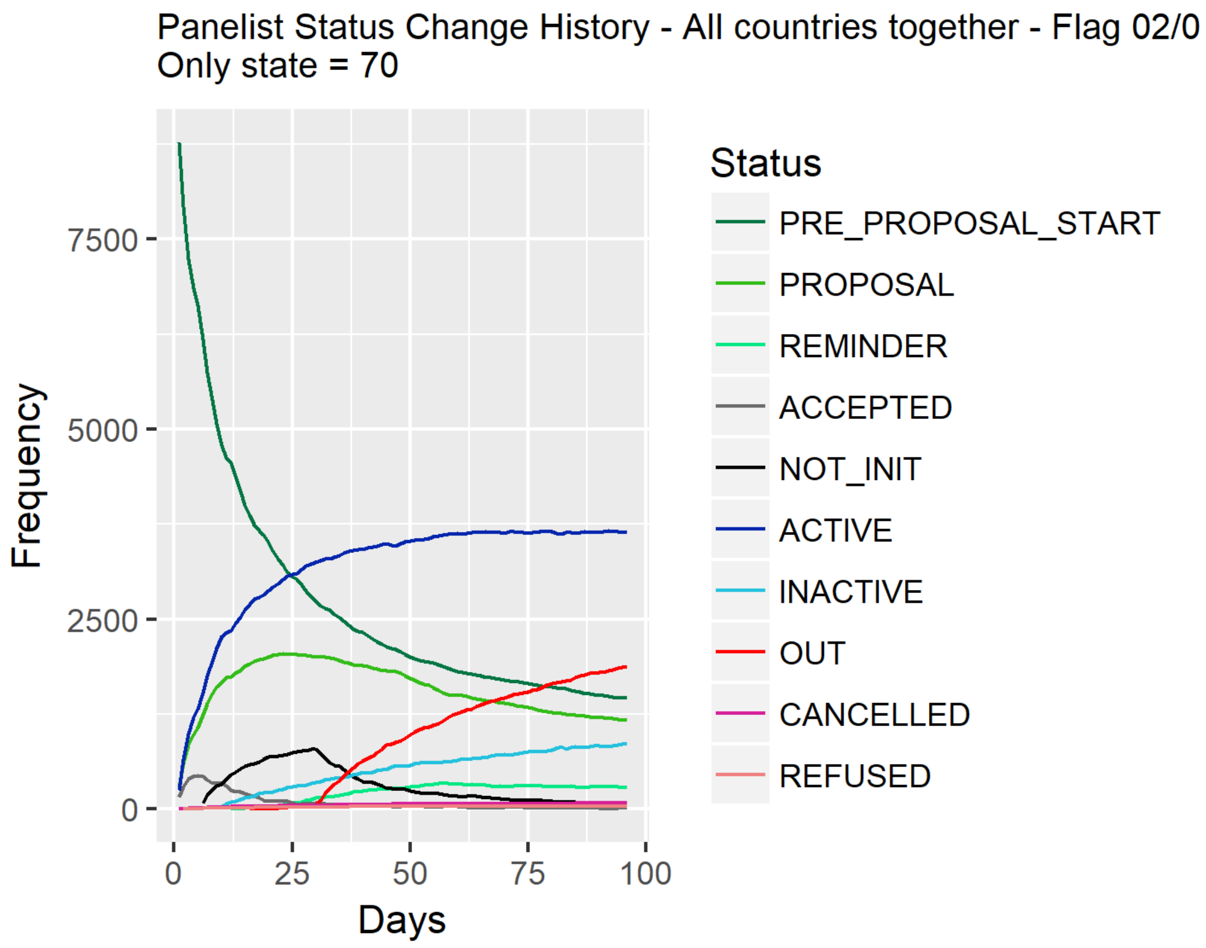


Figure 10: Flag Results - Status change along the first 96 days after the invitation.

As mentioned before, there are two forms of inviting panelists to become Premium. First, there is the flag using the conversion probabilities from the XGBoost and second, there is the Direct Meter. It is also very interesting to compare the statuses patterns change across time for the flag invitation against the direct meter invitation patterns.

Figure 11 has the status trajectories 96 days after invitation for a group of panelists invited through Direct Meter in US and GB. Many differences are observable, for instance, the majority of panelists do not leave the first status - PROPOSAL - in the first two months after being invited. The curve for ACTIVE is very different than the one observed before. Most of the panelists becoming active do so in the first days and then drop fast after the seventh day. A considerable number of panelists which have ACCEPTED the proposal go to NOT_INIT and 25 days later to OUT.

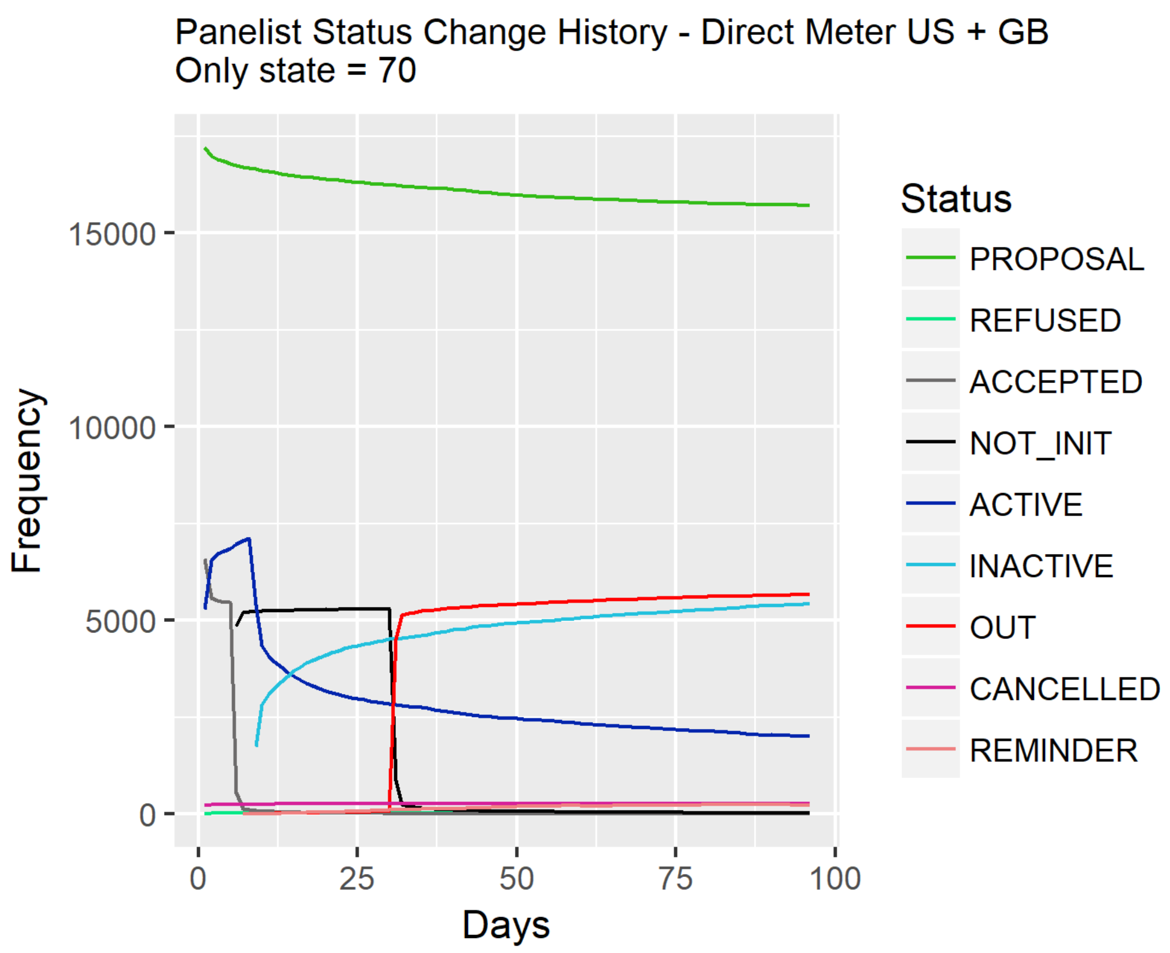


Figure 11: Direct Meter Results - Example for the first 96 days after invitation.

The comparison indicates that inviting panelists based on the probabilities of conversion calculated by the XGBoost seems to be a more successful strategy.

The goals of this chapter are:

- summarize the most common sequences;
- use a dissimilarity measure to calculate the distances between different sequences;
- use the calculated dissimilarities to cluster sequences into similar groups;
- search for the characteristics of the panelists which are different from group to group in order to try to understand if there are variables that might help to predict specific sequence patterns (regression trees will be used where the response variable is the calculated dissimilarity).

3.2 Theory

The life trajectory of a panelist in the Meter Panel will analyzed using the R library TraMineR (Trajectory Miner in R) (Gabadinho et al., 2011) for rendering and analyzing categorical state sequence data.

First, a categorical sequence is defined as an ordered list of statuses constituting an alphabet. This list has often a chronological order forming a complex object which has many important dimension to be taken into consideration.

- **experienced states:** the statuses constituting the alphabet;
- **distribution:** the analysis of the total time spent in each status of the alphabet;
- **timing:** the point in time along the observed time range in which a panelist is given a certain status;
- **spell duration or spacing:** the consecutive time spent in the same status;
- **sequencing:** the order in which the statuses present in a sequence appear.

The theoretical framework for this chapter is strongly based on the articles published by the authors of the library. The first step of the analysis will be to calculate the dissimilarities among the status sequences describing the panelists' life trajectories in the Meter Panel. A in depth discussion about the dissimilarity measures available in the TraMineR package can by find in the article Studer and Ritschard (2016).

Three general approaches for constructing a dissimilarity measure exist, one based on the measurement of the distances between distributions, a second one based on the count of common attributes of the sequences being compared and the last one which is based on editing the sequences measuring the costs of the operations that are necessary to transform one sequence into the another.

Table 1 in the article by Studer and Ritschard (2016) has a complete summary of the most used dissimilarity measures. The simple Optimal Matching (OM) method (Abbott and Forrest, 1986) will be used to determine the dissimilarities among the the status sequences based on the definition of a substitution matrix and indel value costs.

The simple idea behind the OM method is that for measuring the dissimilarity between two sequences one has to evaluate how complex it would be to transform one sequence into another given a set of possible operations which are: (1) substituting a status in a time point by a different one, without changing the length of the sequence; (2) including or deleting points, what would change the length of the sequences. The number of operations and the costs attributed to each one of them is used then to define the value of the dissimilarity.

The costs are specied by a single matrix including the substitution costs and also the indel costs which can be specied as a substitution cost for an additional ‘null’ or ‘empty’ status.

The are three basic ways of defining the substitution costs:

- **Theory-based costs:** here a priori knowledge of the researcher on the specific theme analyzed is used. In the present application one could define a higher cost for going from status ACTIVE to INACTIVE then for going from ACCEPTED to ACTIVE. This is the most arbitrary approach, specially at the moment of defining the values of the costs;
- **Costs based on state attributes:** in this approach, for each status a list of attributes is defined and different values are given. Next the vector of attribute values for each status can be used for calculating the distances. This is a less arbitrary approach and has some mathematical advantages since, for instance, it satisfies the triangle inequality;
- **Data-driven costs:** here solutions based on the information from the data are proposed as for instance considering the statuses transition rates, assigning higher costs for less common transitions. Other possibilities exist and are presented by Studer and Ritschard (2016).

For indel costs less attention is given in general. In the context of the present application, since all sequences have the same length the influence of this type of transformation when using OM methods is not very relevant.

The next step in the analysis will be to use the calculated matrix of dissimilarities to build clusters of panelists which have similar life trajectories in the Meter Panel. The cluster will be calculated using the R library cluster by Maechler et al. (2018). The build the cluster the Ward’s method will be used. A thourough discussion on the method can be found in Murtagh and Legendre (2011) and practical examples in R can be found in Kassambara (2017).

The clusters of panelists with similar life trajectories many possible practical applications for the company, among which defining if the groups have different profiles given key indicator variables as well as defining clear targets for different marketing campaigns and variable retribution systems when doing a feedback invitation to panelists not converting to ACTIVE. It can also be used in feature engineering procedures, for instance, a predictive model can be built for predicting to which cluster a panelist who was not invited to the Meter Panel would pertain. The new variable could than be added to the model developed in the second chapter.

Lastly, TraMineR will be used for study the relationship between status sequences and covariates. In the article by Studer et al. (2011), the authors describe how ANOVA inspired methods can be used to understand how existing covariates can explain the discrepancy between the life trajectories. A pseudo R^2 can be calculated for measuring the strength of sequence–covariate associations. The article also discuss how to build a regression tree for uncover the discriminant structure of covariates. An alternatively to the pseudo R^2 is the F value which compares the explained discrepancy to the residual discrepancy. The discrepancies are calculated using the calculated dissimilarities and the pseudo R^2 and the F value definitions can be find in (Studer et al., 2011, p 5-6).

Multi-factor discrepancy analysis and tree-structured analysis of sequences are presented in the two last sections of the above mentioned article. The regression trees results are very simple and easy to visualize given a comprehensible view of the effect of covariates on the sequences discrepancies.

3.3 Application

Using TraMineR starts creating a sequence object as follows:

```r
# We create the alphabet
meter.alph <- seqstatl(flag[, 7:102])
# the labels
meter.lab <-
  c(
    "ACCEPTED", "ACTIVE",
    "CANCELLED", "INACTIVE",
    "NOT_INIT", "OUT",
    "PRE_PROPOSAL_START", "PROPOSAL",
    "REFUSED", "REMINDER"
  )
# the short labels.
meter.shortlab <-
  c("ACC", "ACT",
    "CAN", "INA",
    "NOT", "OUT",
    "PRE", "PRO",
    "REF", "REM")
# Create a state sequence object
meter.seq <- seqdef(
  flag[, 7:102],
  alphabet = meter.alph,
  states = meter.shortlab,
  labels = meter.lab,
  xtstep = 6,
  cpal = cbPalette
)
```

```
 [>] state coding:
       [alphabet]          [label]  [long label]
     1  ACCEPTED            ACC      ACCEPTED
     2  ACTIVE              ACT      ACTIVE
     3  CANCELLED           CAN      CANCELLED
     4  INACTIVE            INA      INACTIVE
     5  NOT_INIT            NOT      NOT_INIT
     6  OUT                 OUT      OUT
     7  PRE_PROPOSAL_START  PRE      PRE_PROPOSAL_START
     8  PROPOSAL            PRO      PROPOSAL
     9  REFUSED             REF      REFUSED
    10  REMINDER            REM      REMINDER
 [>] 9488 sequences in the data set
 [>] min/max sequence length: 96/96
```

All the plots in the analysis will share the same colors system, therefore the legend for the statuses and respective associated colors is indicated in figure 12:

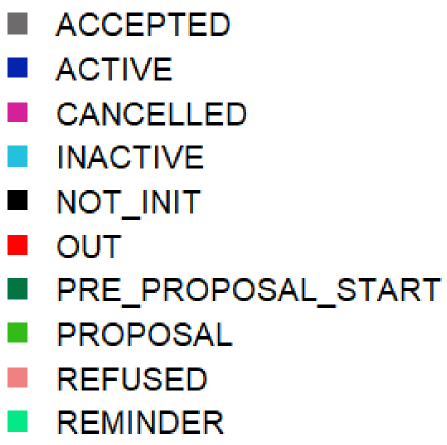


Figure 12: Legend for status color identification used across all charts in this section.

The first visualization in figure 13 is a cross sectional plot which has the proportion of panelists in each one of the statuses for each of the 96 days observed. The plot is a instantaneous photo of the resulting of the flagging during the period. For instance, the dark blue area indicates how the number of ACTIVE panelists grow on time while the light blue displays the emergency of the INACTIVE cases. The red area shows in which moment panelists start getting status OUT. An important point here is that during the first three months, once the maximum number of ACTIVE panelists was achieved it has been since sustained. However, since the number of INACTIVE is also growing, that means there is a cyclical movement in which the conversion to ACTIVE still happening along the time replacing the panelists which are becoming INACTIVE.

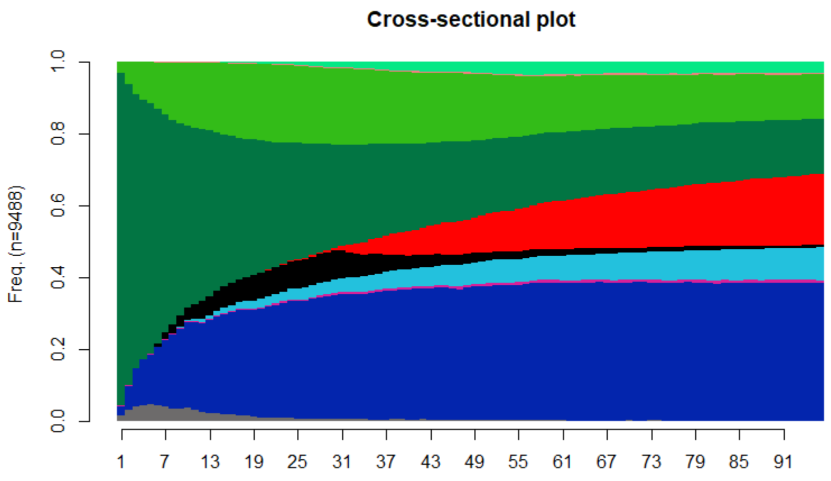


Figure 13: Daily proportion of panelists by status.

The full potential of this sort of chart can be observed when comparing the cross sections for categorical predictors. Since the variable "panelist_app_installed" was the most important in the predictive model it is the best choice for exemplify this point.

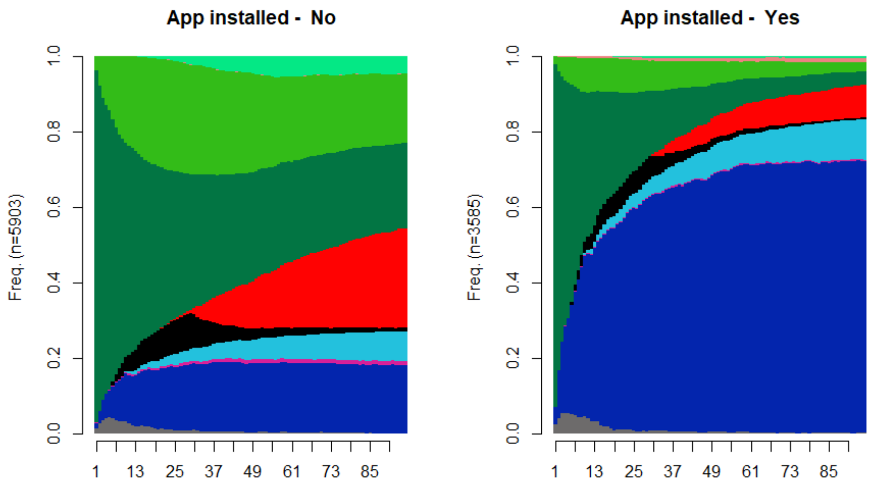


Figure 14: Daily proportion of panelists by status and "panelist_app_installed".

The difference is impressive and once again highlights the importance of the variable “panelist_app_installed” as a predictor of ACTIVE panelists.

An interesting interpretation of this difference can be made based on the calculation of the cross sectional entropy for each of the two categories in “panelist_app_installed”. Entropy here is a measure of variability or uncertainty regarding sequence predictability.

In figure 15, one can observe that among the panelists who do not have the company application installed the entropy reaches much higher levels while for the ones who does have the app the entropy reaches its maximum around the 13th day and as the proportion of ACTIVE panelists become dominant the entropy starts to drop.

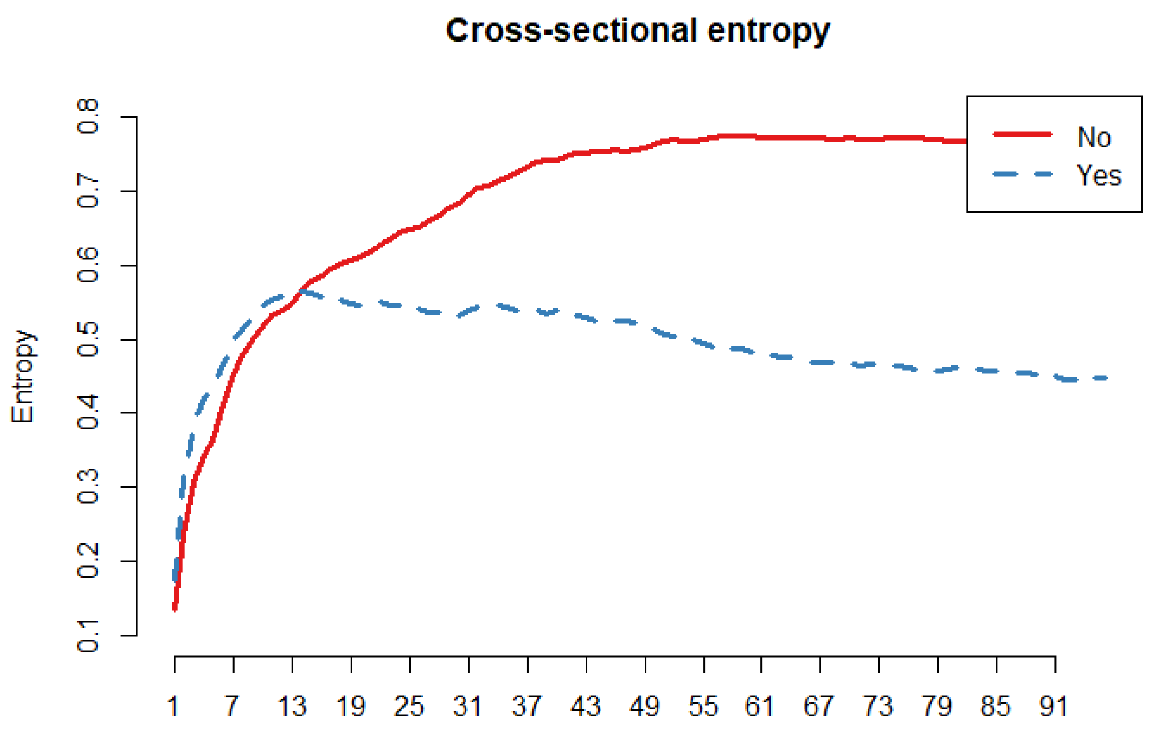


Figure 15: Cross sectional entropy for the variable “panelist_app_installed”.

Entropy values can be used as the dependent variable for training linear models based on the predictors used in chapter 2. This model can be then used in the feature engineering process for creating an additional predictor (Gabadinho et al., 2011, p 8). Other measures of sequence complexity exist besides entropy and are discussed in the mentioned article.

Next step is to build the clusters of sequences. The following peace of code does the work.

```
# Creates the substitution cost matrix based on
# "TRATE" (derived from the observed transition rates)
submat   <- seqsubm(meter.seq, method = "TRATE")
# Calculating the dissimlarity matrix using OM method
# Indel cost is set to 1
dist.om <- seqdist(meter.seq,
                   method = "OM",
                   indel = 1,
                   sm = submat)
# Load cluster library
library(cluster)
# Creates the cluster using Ward method
# and the dissimilarities matrix.
clusterward <- agnes(dist.om, diss = TRUE, method = "ward")
# Build four clusters
cl1.4 <- cutree(clusterward, k = 4)
cl1.4fac <-
  factor(cl1.4,
         labels = c(
           "To␣ACTIVE",
           "PROPOSAL␣-␣OUT",
           "ACCEPTED␣-␣OUT",
           "PRE_PROPOSAL_STAR"
         ))
# Plot the cross-sections for each cluster
seqdplot(
  meter.seq,
  group = cl1.4fac,
  with.legend = FALSE,
  border = NA,
  cex.main = 0.9
)
```

The Ward based clusters calculated based on the OM dissimilarities can be visualized in the figure 16. The results are very simple since they can be easily interpreted. The cluster in the top left has almost all cases converting into ACTIVE. Cluster in the top right has cases the have never ACCEPTED the Premium proposal but since these are panelists doing more participations in surveys in the Access Panel they have moved forward into further statuses other then PRE_PROPOSAL_START. Bottom right cluster contains the cases of panelists who have not even seen the proposal since they still all as PRE_PROPOSAL_START. These are panelists who have probably a very low level of engagement and probably had also low conversion probabilities resulting from the XGBoost model. This might be an example to be used when trying to convince managers to not push the numbers beyond the limits indicated be the XGBoost model in future invitations. Finally, cluster in the bottom left holds the cases of panelists who have ACCEPTED the Premium proposal but have never installed the meter application. These are potential candidates for a feedback survey which could be used to understand the reasons for not installing it once some interest in doing so was signalized.

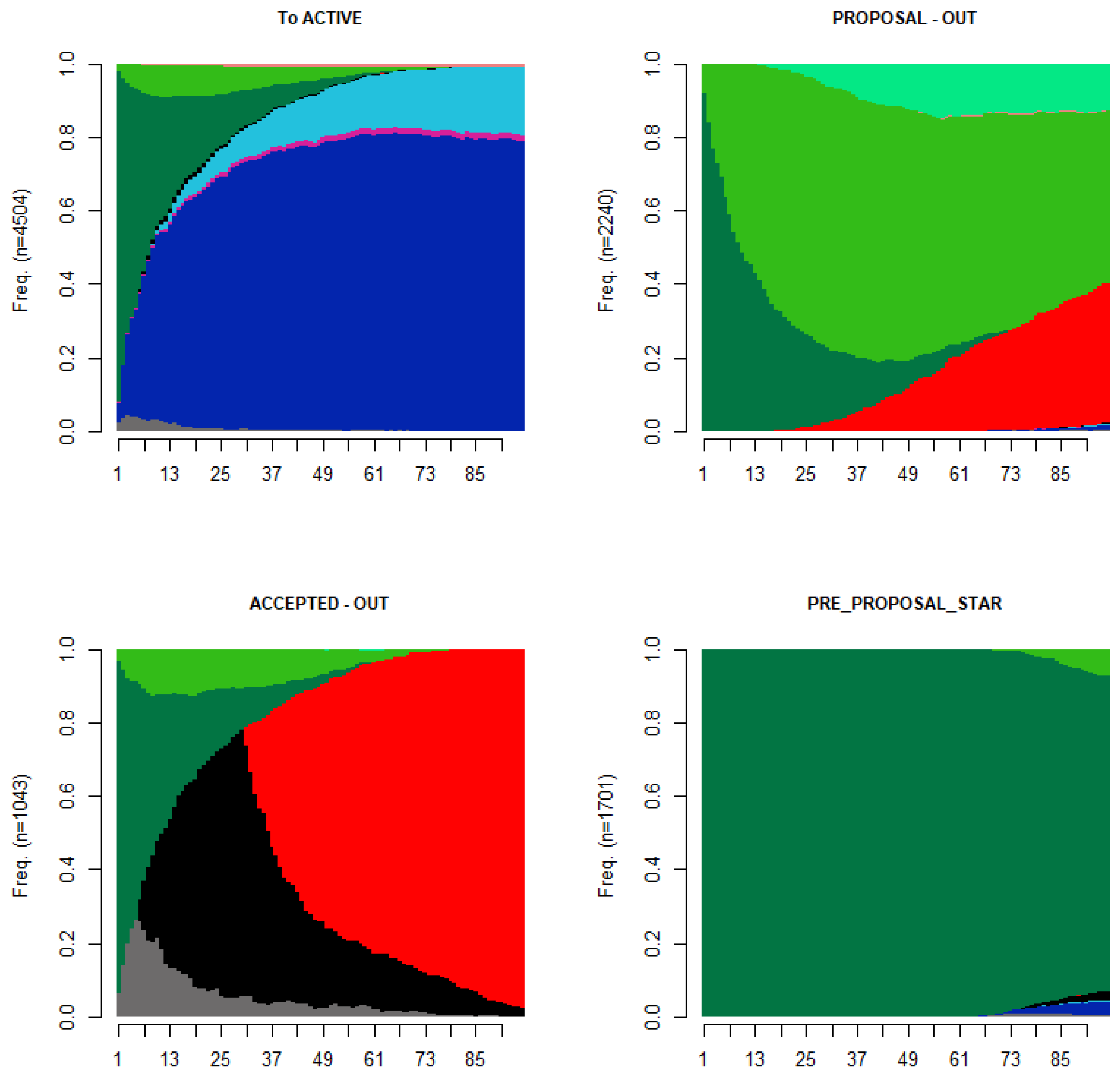


Figure 16: Cross sectional plots for the clusters built using the OM dissimilarities.

The last step in this chapter application is the construction of the regression tree model based on the dissimilarities and discrepancy analysis. The following peace of code fits first an ANOVA like univariate discrepance analysis using the variable “panelist_app_installed”. Next it fits and plots a regression tree model.

```
# Running an ANOVA-like analysis for panelist_app_installed
da <-
  dissassoc(meter_dist.om,
            group = panelist_app_installed,
            R = 1000)
print(da)

Pseudo ANOVA table:
             SS   df          MSE
Exp    65487.09    1 65487.08668
Res   575164.53 9486    60.63299
Total 640651.62 9487    67.52942

Test values  (p-values based on 1000 permutation):
                      t0 p.value
Pseudo F   1080.0570464   0.001
Pseudo Fbf 1175.1438189   0.001
Pseudo R2     0.1022195   0.001
Bartlett    130.4955225   0.001
Levene     1307.0042346   0.001

Inconclusive intervals:
0.00383  <  0.01  <  0.0162
0.03649  <  0.05  <  0.0635

Discrepancy per level:
         n discrepancy
No    5903    68.18820
Yes   3585    48.15887
Total 9488    67.52230

# Growing a sequence regression tree
dt <-
  seqtree(
    meter.seq ~ country +
    gender + age + social_class +
    panelist_app_installed,
    weighted = FALSE,
    data = flag,
    diss = meter_dist.om,
    R = 1000
  )
# Plotting the tree model
seqtreedisplay(
  dt,
  type = "d",
  border = NA,
  with.legend = F,
  cex.quality = 3.5,
  cex.main = 2,
  filename = "dt_meterseqtree.png"
)
```

The results for for the univariate discrepancy analysis show that the pseudo R^2 and pseudo F, although significative at a 5% level, are not very high, indicating that the strength of the association between the predictors and the sequences is not very strong. The values for discrepancy are higher for the sequences of panelists who do not have the company's app installed.

Finally, figure 17 has the results for the regression tree. The global pseudo R^2 is 0.142 and it is significative at a 5% level. The first split, as expected, is based on the variable "panelist_app_installed". In the side where "panelist_app_installed" equals "No", the remaining splits are based on country and social class. In the side where "panelist_app_installed" equal "Yes", the remaining splits are mainly based in the variables social class and age.

3.4 Results and Discussion

This chapter was useful to understand better the life trajectories of panelists in the Meter Panel given the available statuses and associated rules. Different statistical tools were used and insights on how to identify and approach different groups of panelist with different life trajectories were discussed. Alternative feature engineering variable construction based on complexity measures of sequences as entropy and cluster analysis were also discussed.

The regression tree model was useful to indicate that although the variable "panelist_app_installed" is very powerful explaining the sequences discrepancy, demographic predictors as country, age and social class are not. This conclusion will be further explored in the next chapter on survival/retention analysis.

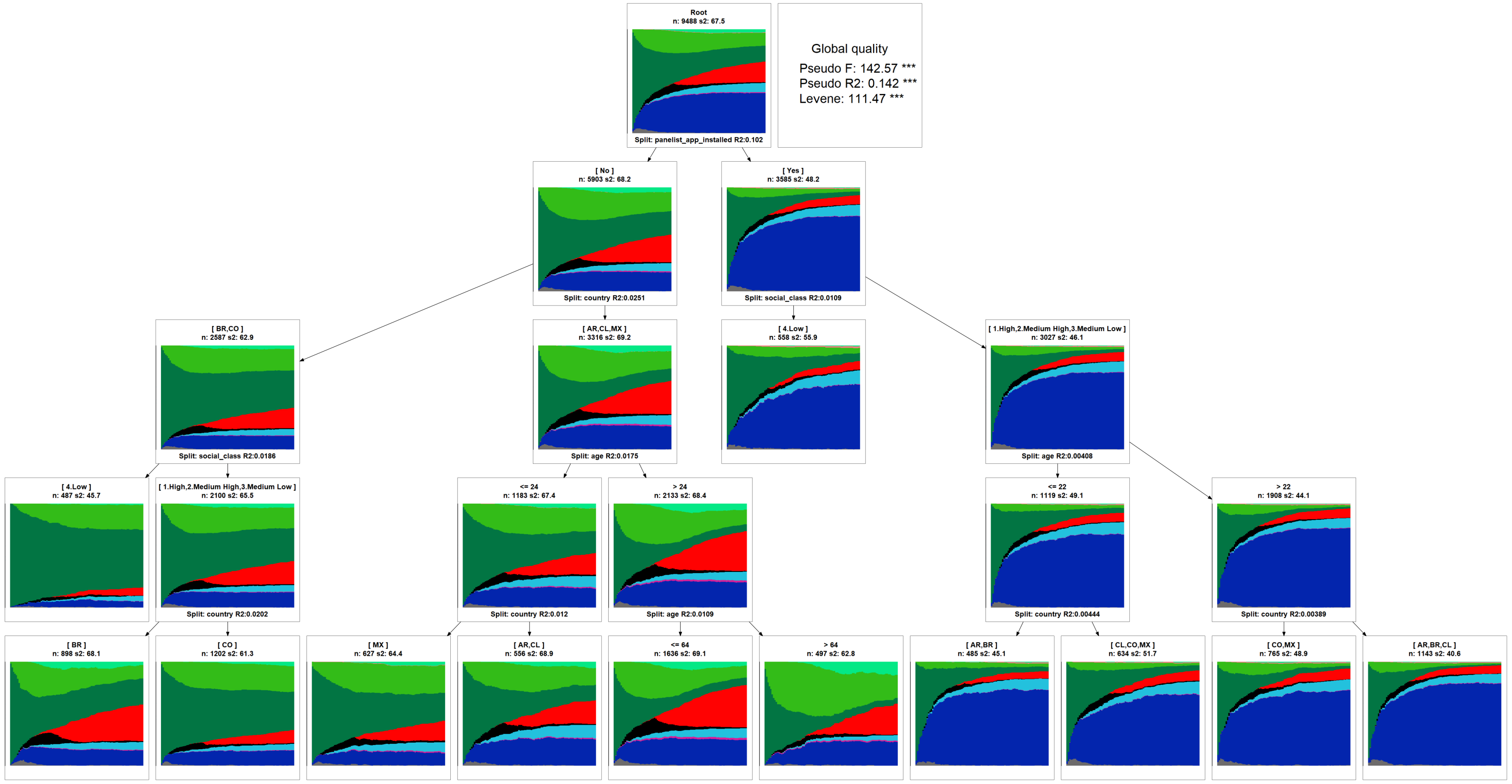


Figure 17: Regression tree based on the calculated dissimilarity matrix.

In figure 17 one can observe that once sequences are split by the variable panelist_app_installed, the subsequent splits produce very similar cross-sectional plots, meaning the variables using for splitting the sequences are not as good separating dissimilar sequences. That is already a pointing for the conclusion that the analysis need more variables - which are unknown at this point - capable of explaining more thoroughly the dissimilarities in the life trajectories of Meterized panelists.

4 Retention Analysis

4.1 Definition

In this last chapter, the goal is to apply survival analysis to understand the retention of Meterized panelists.

Retention is defined here as the time a Mterized panelist stills ACTIVE up to the moment he or she gets to the status INACTIVE for the first time. It is the dependent variable of the survival model and is denoted by the continuous random variable T with support $[0, \infty)$. Panelists who have never get to ACTIVE are not considered and were filtered out. Only the first 60 days from the the moment a panelist gets to ACTIVE are considered, therefore, panelists which converted to ACTIVE but had not at least 60 days as ACTIVE were also excluded. Finally, panelists who did not become INACTIVE during the period of the experiment were right censored.

In order to measure the survival time of an ACTIVE panelist the survival function can be used to estimate the probability that a Meterized panelist will be active after a specified time t. This is a very useful information since it gives an idea of how fast the size of the Meter Panel is decreasing and also the mean number of days till a panelist becomes INACTIVE. Such information can be used for determining the periodicity with which new invitations should be made in order to keep the size of the panel stable and to increase or decrease it depending on variability of the demand for projects. Finally, it can also be used to plan a the communication strategy during the days for which the risk of becoming INACTIVE is higher.

The hazard function estimated the risk of a Meterized panelist to become inactive at a specified time t, or the instantaneous rate of occurrence of the event. This information is also useful since specific communication and marketing actions can be taken slightly before the moments which have the higher risk.

The chapter starts using a simple non-parametric statistic, the Kaplan–Meier estimator (Kaplan and Meier, 1958) for the survival function. Next, the Cox Proportional Hazard model (Cox, 1992) will be used to include the predictors in the analysis and evaluate the impact each of them have in the survival curve.

4.2 Theory

4.2.1 Kaplan-Meier survival curves

The basic building blocks in survival analysis are the survival function and the hazard function.

Define the cumulative distribution function on the support of T as

$$F(t) = P(T \leq t) \qquad t > 0. \tag{19}$$

The survival function on the support of T is defined as

$$S(t) = P(T \geq t) = 1 - F(t) \qquad t > 0. \tag{20}$$

The hazard function on the support of T is defined as

$$h(t) = \frac{f(t)}{S(t)} \qquad t > 0. \tag{21}$$

The cumulative hazard function on the support of T is defined as

$$H(t) = -\log(S(t)) \qquad t > 0. \tag{22}$$

The Kaplan–Meier estimator will be used to estimate the survival function. Since it is a non-parametric estimator no underlying distribution needs to be assumed. It is defined as:

$$\hat{S}(t) = \prod_{i:t_i \leq t} \left(1 - \frac{d_i}{n_i}\right), \tag{23}$$

where t_i is a time point where at least one panelist became INACTIVE, d_i is the number of panelists which became INACTIVE at t_i and n_i is the number of panelists which were still ACTIVE at t_i but were also at risk of becoming INACTIVE. It is a non-increasing step function. As described in (Diez, 2013, p. 27), the variance for the Kaplan-Meier estimator can be calculated using the delta method (Elandt-Johnson and Johnson, 1980, p. 69-72) for approximating the variance of a continuous transformation $g(.)$ of the random variable T. The method chooses the log function as $g(.)$ and defines and uses the transformation to approximate the variance of $\log(\hat{S}(t))$ as follows:

$$var\Big(\log \hat{S}(t)\Big) = \sum_{i:t_i \le t} var\left(\log\left(1-\frac{d_i}{n_i}\right)\right) \approx \sum_{i:t_i \le t} var\left(\log\left(\frac{d_i}{n_i(n_i-d_i)}\right)\right). \tag{24}$$

Next, the author uses again the delta method to get the estimated variance for $\hat{S}(t)$ itself as defined in Eq. 25.

$$v\hat{a}r\Big(\hat{S}(t)\Big) \approx \Big[\hat{S}(t)\Big]^2 \sum_{i:t_i \le t} var\left(\frac{d_i}{n_i(n_i-d_i)}\right). \tag{25}$$

Nevertheless, this last definition is problematic because confidence intervals computed based on it may extend below 0 or above 1. The alternative presented is a variance calculated based on the log-log transformation of $\hat{S}(t)$ as follows:

$$v\hat{a}r\Big(\log\Big[-\log \hat{S}(t)\Big]\Big) \approx \frac{1}{[\log \hat{S}(t)]^2} \sum_{i:t_i \le t} var\left(\frac{d_i}{n_i(n_i-d_i)}\right). \tag{26}$$

The median for the retention time is defined as the smallest t so the survival function is less than or equal to 0.5. The calculation of the $1-\alpha$ confidence interval is based in the following inequality:

$$-z_{\alpha/2} \le \frac{g\Big(\hat{S(t)} - g(0.5)\Big)}{\sqrt{var\Big[g\Big(\hat{S}(t)\Big)\Big]}} \le z_{\alpha/2}, \tag{27}$$

where $g(.) = \log[-log(.)]$ and $var\Big[g\Big(\hat{S}(t)\Big)\Big]$ is given by Eq. 26, and $z_{\alpha/2}$ is the $\alpha/2$ quantile of the N(0,1) distribution.

Survival curves can be built taking into consideration different categorical predictors. For instance, one can check the survival curve for demographic variables as country, gender, age grouped and social class. In the context of this application type of comparisons may bring important insights. Knowing a certain subset of Meterized panelists has a steeper survival curve in the first active days allows the developing of different communication or rewarding approaches for specific groups and might improve the retention of these panelists.

A log-rank test (Bland and Altman, 2004) will be used to test if the difference between curves for different predictors is significative or not. The test is used to test the null hypothesis that there is no difference between the populations given the survival probabilities. The log-rank test has the same assumptions as the Kaplan Meier survival curve, that censoring is random, the survival probabilities are the same independently of how long it took to a panelist become ACTIVE.

4.2.2 Cox Proportional Hazard Model

The Cox Model allows going from the univariate analysis seen with the Kaplan-Meier curves to a multivariate analysis, taking into consideration the influence of the different predictors in the estimated survival curve.

The Cox model is expressed by the hazard function defined in Eq. (21) and it can be interpreted as the risk of a panelist becoming INACTIVE at a specified time t. The model can be expressed as follows:

$$h(t) = h_0(t) * \exp(\beta_1 * x_1 + \beta_2 * x_2 + \ldots + \beta_p * x_p), \tag{28}$$

where the p x_i variables are the predictors to be considered, the β_i are the model's coefficients measuring the effect size of the predictors in the hazard function $h(t)$ dependent on the time t, measured in days in this application. The term $h_0(t)$ is similar to the intercept in a logistic regression, it is the baseline hazard function for when all the predictors take the value 0.

Similar to a logistic regression, Cox models a ratio based and $\exp(\beta_i)$ is defined as the hazard ratio (HR). The interpretation is simple, if the HR is greater than one (same as β_i being bigger than 0), it means the predictor x_i is positively associated the survival probability, therefore the survival time will decrease as x_i increases. The opposite relation will be true when HR is smaller than 1. An HR equal to one indicates the absence of effect of the predictor on the hazard function.

For using the model appropriately its assumptions must be verified. The following three will be tested:

- Schoenfeld residuals to check the proportional hazards assumption. The idea here is that the hazerd curves must be proportional and should not cross each other. For instance, for two panelists, k and j, the respective hazards are: $h_k(t) = h_0(t) * \exp(\sum_{i=1}^{n} \beta * x_i)$ and $h_j(t) = h_0(t) * \exp(\sum_{i=1}^{n} \beta * x_i)$, the hazard ratio $h_k(t)/h_j(t)$ should be independent from t. That is why the model is called Cox Proportional Hazard Model.

- Martingale residual to assess nonlinearity;

- Deviance residual which is a symmetric transformation of the Martinguale residuals, to examine influential observations.

In the next section of the chapter, the described methods will be applied to the ACTIVE panelists survival time of the flagged panelists results presented in table 11, given the restrictions described at the beginning of the present chapter.

4.3 Application

This application relies basically on two R libraries. First, Therneau (2015) for estimating the models and second, the library Kassambara and Kosinski (2018) for drawing the plots. The Surv(.) function in the survival library is used to create the survival object by specifying a vector of survival time and the censored cases. This object will be the input for the function survfit(.) which can be used to estimate the Kaplan-Meier curve among other alternatives.

Figure 18 has the Kaplan-Meier estimated survival curve for 1,572 panelists. One will quickly note that during the first 7 days none of the panelists went INACTIVE. This is a consequence of the status sequence transition rules described in the diagram 10. An important rule is that a panelist will be considered inactive only after 7 days without sharing data with the company. Thus, the drop seen in the 8th day reflects the cases of panelist who have become ACTIVE and shared data for only one day. These are the worst possible cases since the panelist will receive 100 points for becoming ACTIVE independently of for how long they will keep sharing Internet navigation data.

Note that after the 60th we have the censored cases for which the time to INACTIVE is known although before the time range observed.

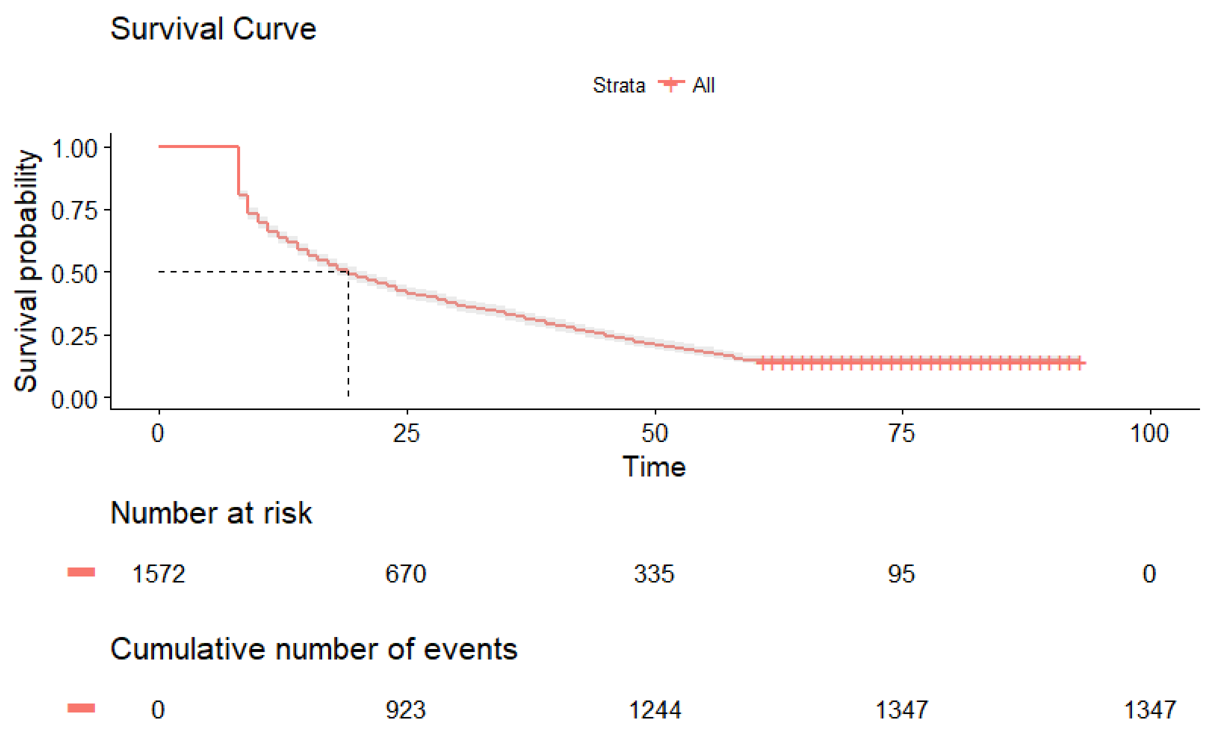


Figure 18: Survival curve for the first 60 days of a panelist in the Meter Panel after becoming ACTIVE.

The estimated median survival time was 19 days, therefore one can conclude that half of the panelists converted to ACTIVE stopped sharing data in the first 12 days of activity. However, the curve is steeper between days 7 to 12. That indicates half of the points paid for these panelists are at risk of not being paid back since more days of activity are needed in general for using the data in real projects.

This conclusion indicate a flaw in the predictive model developed in chapter two, in the sense that, although it is reasonably good predicting conversion to ACTIVE, it is not taking into consideration the retention dimension. Therefore, the need for predicting retention and finding predictors which might enhance this dimensionality will be crucial for the company.

The 95% confidence interval in the results bellow is calculated by the function using the log-log method. Starting from the following trasformation of the estimated survival function, $\hat{L}(t) = \log(-\log(\hat{S}(t)))$, the interval can be built as presented by Anderson et al. (1982).

```
2267 observations deleted due to missingness
   n  events  median 0.95LCL 0.95UCL
1572    1347      19      17      21
```

The risk over time ca be assessed by the cumulative hazard curve, defined by Eq.

(22) and plotted in the figure 19. The cumulative hazard is integrating the instantaneous hazard rate over time. In other words it measures the total amount of risk that has been accumulated up to a specified time t (Cleves et al., 2008, p 8-15).

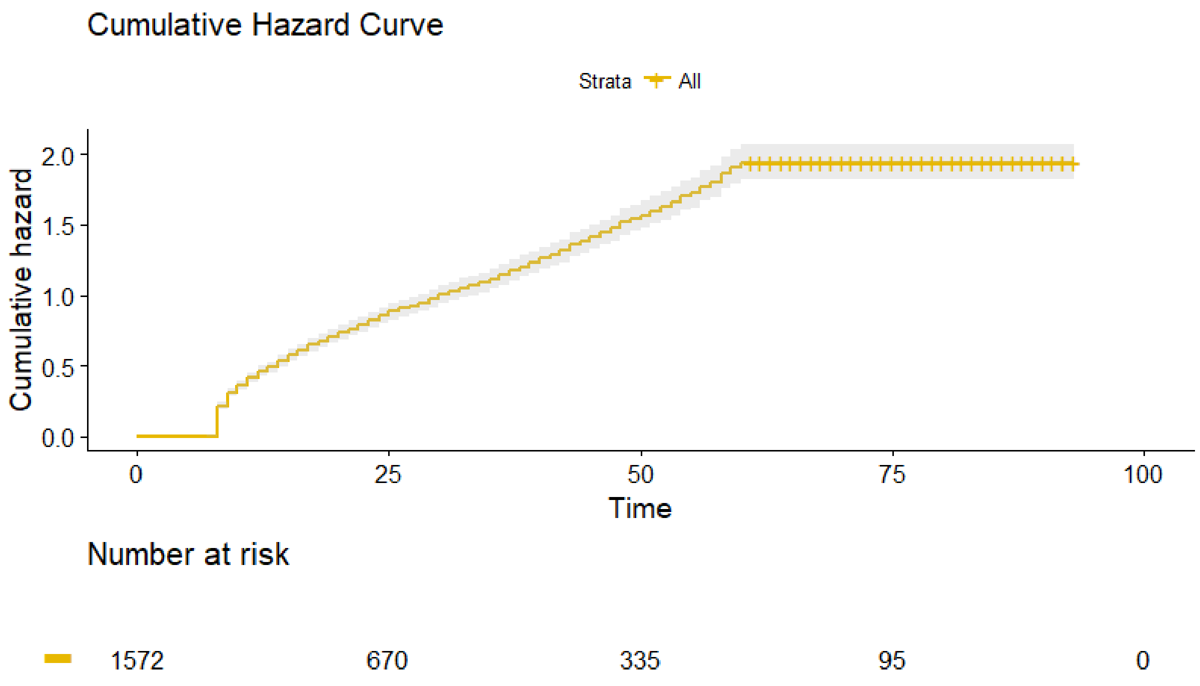


Figure 19: Cumulative Hazard curve for the first 60 days of a panelist in the Meter Panel after becoming ACTIVE.

The next step in the analysis was to estimate the survival curves by the main categorical predictors. Demographic predictors as country, gender, age and social class were tested as well as the variable indicating if the panelist had or not the company application installed [6]. None of the variables showed significative differences in their estimated survival curves based on the log-rank test ($p < 0.05$). In the figure 20 one can observe the estimated survival curves for each of the countries.

[6]Note that here the referred application is not the meter one, used for measuring Internet navigation, but a general use application that panelist use for answering to surveys and managing their panelist account.

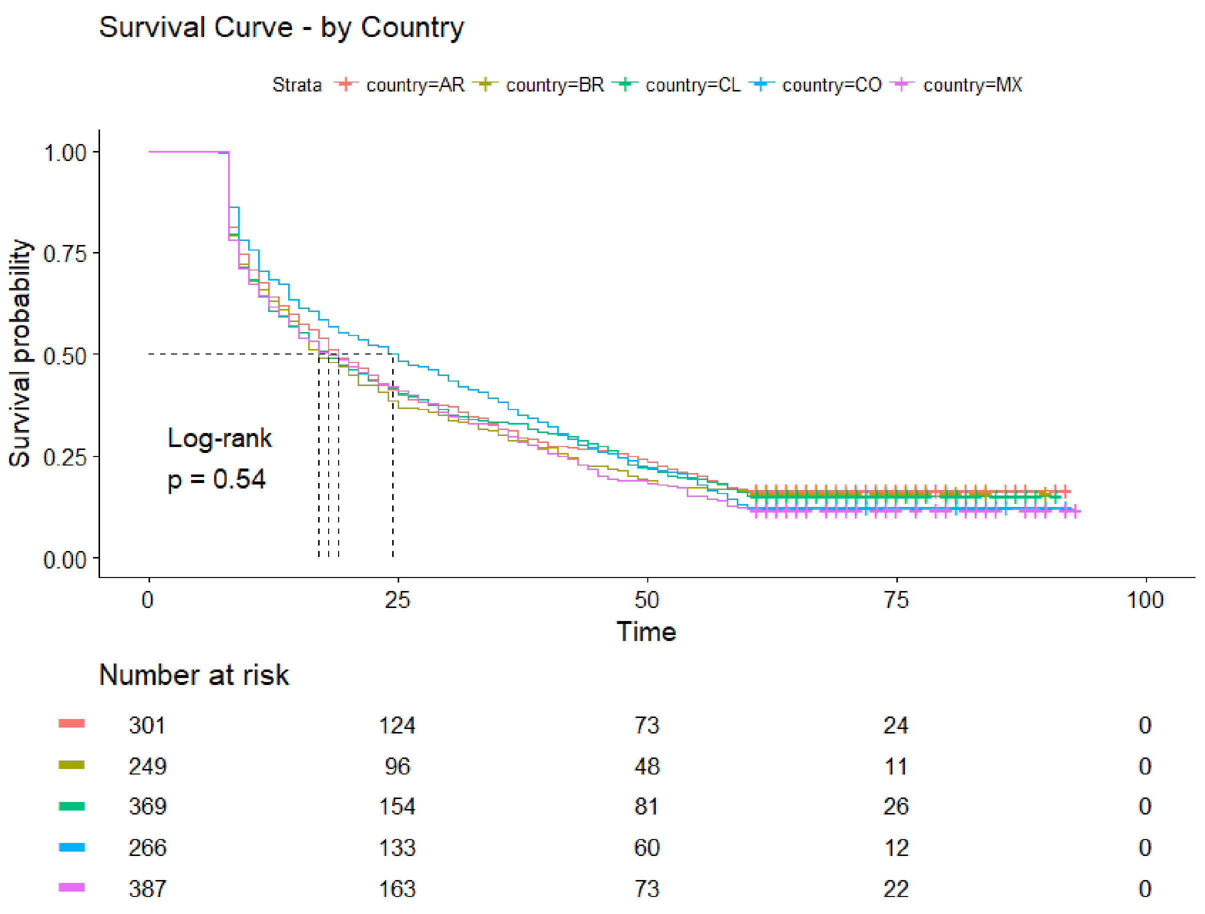


Figure 20: Survival curve per country and log-rank test p-value.

Despite the high p-value for the log-rank test indicating there is no difference among the curves, Colombia (CO) seems to be different. The country has a median survival time of 24 days, bigger than all other countries. To go further, a pairwise log-rank test was used, so the differences were estimated curve by curve using the function pairwise_survdiff(.) from the survival package in R. Based on the test results bellow we can not conlude that the curves are different.

```
        Pairwise comparisons using Log-Rank test

data:  survival_data and country

   AR   BR   CL   CO
BR 0.84 -    -    -
CL 0.84 0.88 -    -
CO 0.90 0.84 0.84 -
MX 0.84 0.84 0.84 0.84

P value adjustment method: BH
```

The fact that the estimated survival curves for all the categorical predictors showed no significative differences indicates already that hour predictors are not strong enough for predicting retention.

Despite of this fact, a Cox model (Cox, 2018) was used as an alternative to confirm this point and also include other predictors other than the categorical ones in the model. The results for this model are presented in the figure 24 and the results are pretty bad.

The figure 24 displays the hazard ratios for all predictors with their confidence intervals and significance levels. Most of the rations are very close to one indicating small effect size on the survival time. The confidence intervals are big and all of them cross the base line with the value 1, meaning one can be very unsure if a predictor have a positive or negative effect. Finally, apart from the variable "time_to_active" which measures how many days a panelist has taken till become ACTIVE, all other variable are not significative at a 95% confidence level.

The assumptions of the model were tested in order to try to understand the possible reasons for such a poor performance (cox, 2018). For testing the proportional hazard assumption the function cox.zph(.) in the library survival was used. The function correlates the set of scaled Schoenfeld residuals with time for each predictor, to test for independence between residuals and time. Additionally, it performs a global test for the model as a whole. Results are presented bellow:

```
                                rho   chisq p-value
countryBR                  -0.01067  0.1554 0.69345
countryCL                  -0.00648  0.0561 0.81277
countryCO                   0.05929  4.8371 0.02785
countryMX                   0.01862  0.4833 0.48694
genderMale                  0.00742  0.0757 0.78325
age18-24                   -0.02777  1.0425 0.30724
age25-35                   -0.02886  1.1291 0.28798
age36-45                   -0.03404  1.6011 0.20575
age46-55                    0.00960  0.1273 0.72120
age56-65                   -0.01826  0.4511 0.50181
age65+                      0.03491  1.6689 0.19640
social_class2.Medium High   0.06383  5.5988 0.01797
social_class3.Medium Low    0.06343  5.6742 0.01722
social_class4.Low           0.06513  5.9167 0.01500
panelist_app_installedYes   0.03867  2.0777 0.14946
total_points                0.03793  1.9375 0.16394
participations              0.00781  0.0858 0.76959
time_to_active              0.06424  5.7434 0.01655
seniority                   0.00680  0.0612 0.80463
```

```
GLOBAL                                          NA 41.5096 0.00206
```

A non-significative p-value supports the absence of relationship between residuals and time. For most of the predictor the p-value is not significative, however, the global test was significative.

In figure 21 we plot the Schoenfeld residuals for two countries Chile and Colombia. These countries had very different p-values for the individual Schoenfeld tests, therefore, a comparison of the residuals for both will be useful. In evaluaing the chart one must look for systematic departures from a horizontal line whch are indicative of non-proportional hazards. Despite the differences in the tests' p-values it is not easy to visually identify the differences in the residuals for the two predictors.

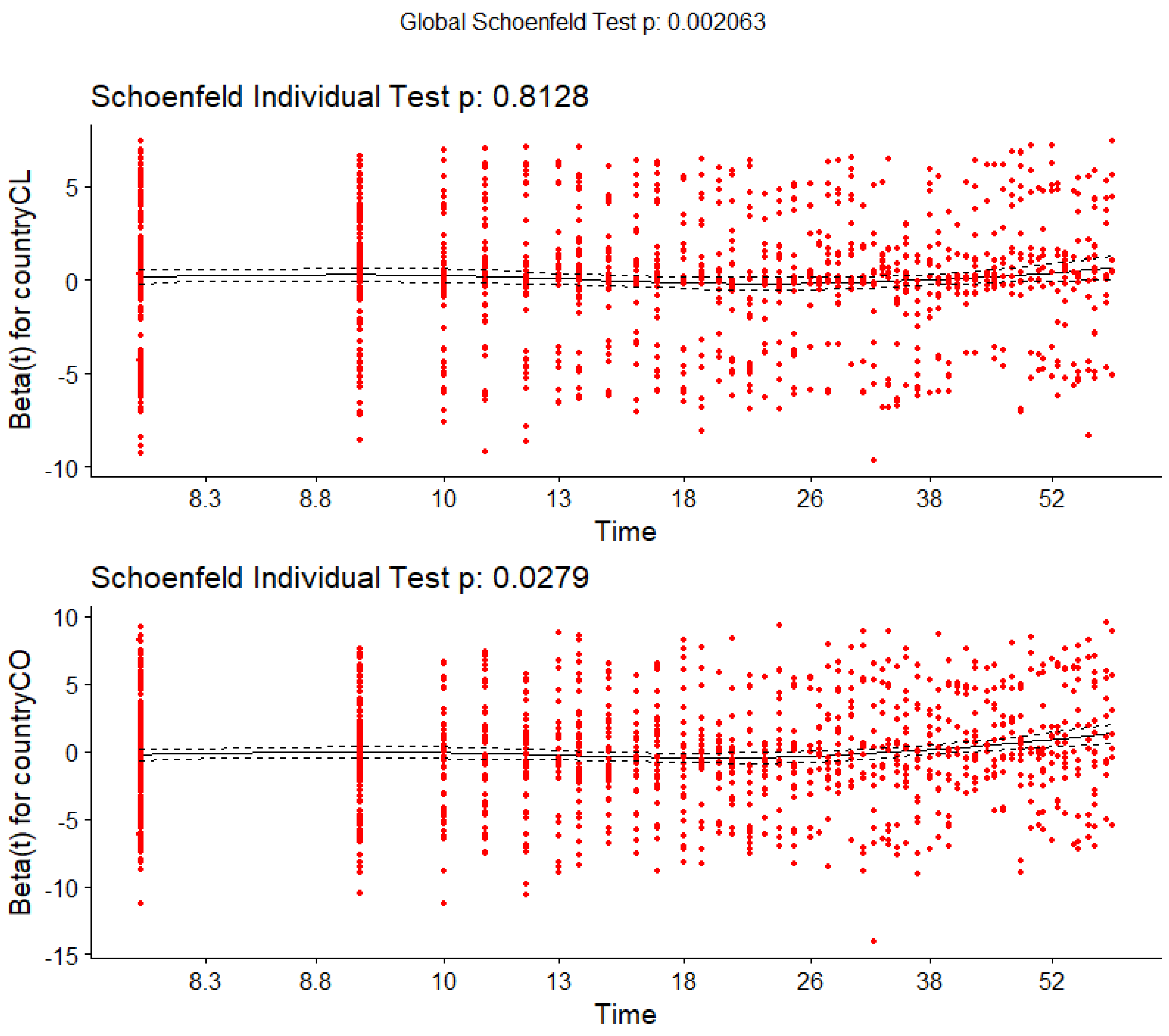


Figure 21: Schoenfeld residuals and smoothed spiline test plot.

Next step is testing for influentional observations, what can be done using the function ggcoxdiagnostics(cox_model, type = "deviance") to visualize the deviance residuals which is a is a normalized transform of the martingale residual. In this diagnostic, positive residual values indicate shorter survival times and negative longer survival times.

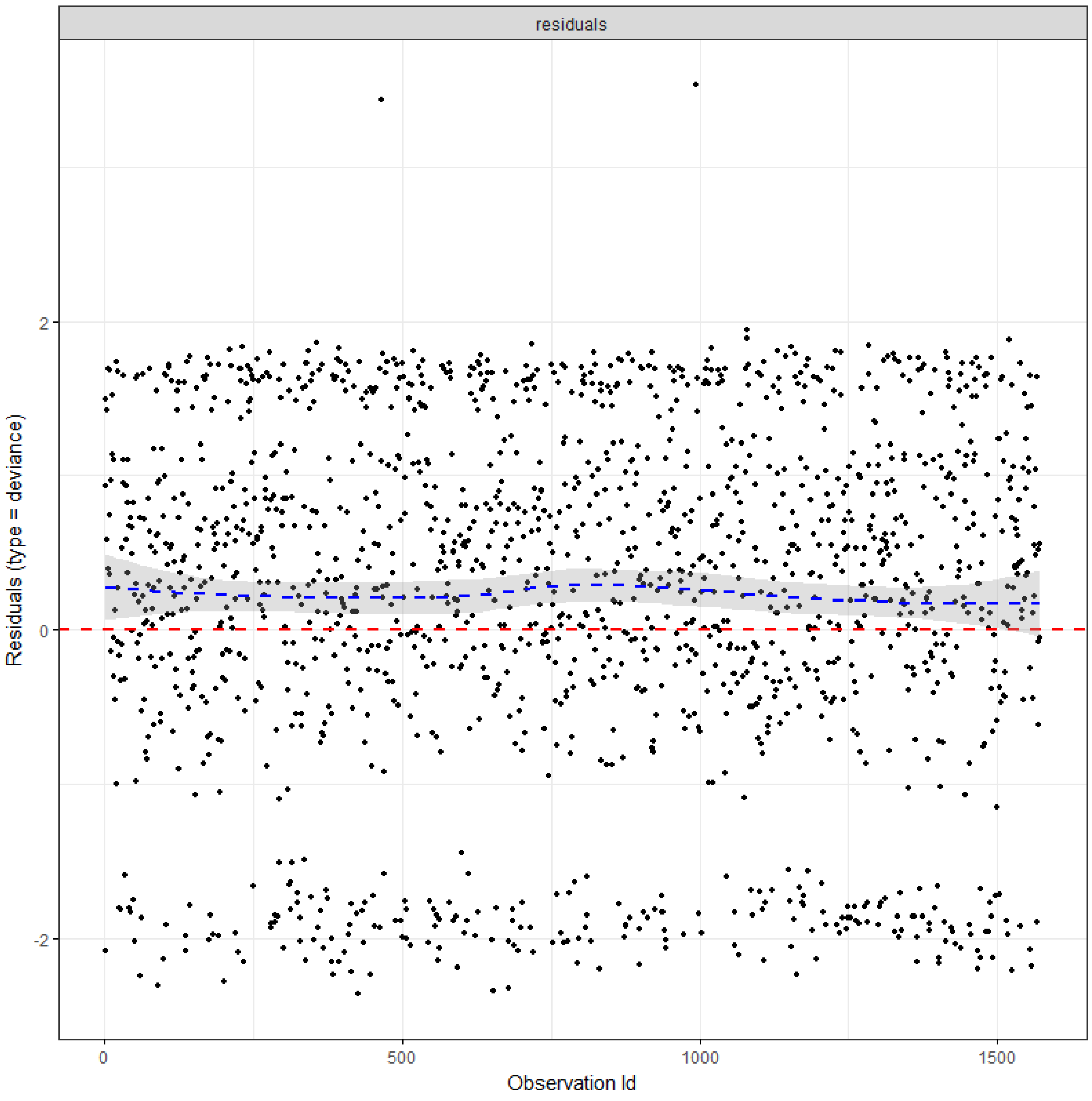


Figure 22: Deviance residuals plot for the Cox proportional hazard model.

Figure 22 has peculiar results. First, two outliers are found which must have shorter time than all the other observations. Looking at the data, there are two cases which have survival time of 7 days followed by 302 with a survival time of 8 days. Overall, the residual are not symetrically distributed around 0. The blue line in the chart indicates this block displacement. This might be a related with the 7 days to INACTIVE rule resulting almost no case of INACTIVE panelists in the first 7 days. For treating that, ideally, the model should be built using an alternative definition of retention, allowing to measure inactivations taking place during the first 7 days.

Besides that, one can observe a group of isolated observations with longer survival times. It is not clear if the chart is taking into consideration the censored cases with survival times bigger than 60 days.

The last assumption to be tested is the one about the non linearity of continuous predictors. Here, the Martingale residuals of a null model can be plotted against these predictors in order to assess their form. The funciton ggcoxfunctional(.) in the survminer R package was used to draw the plots in figure 23.

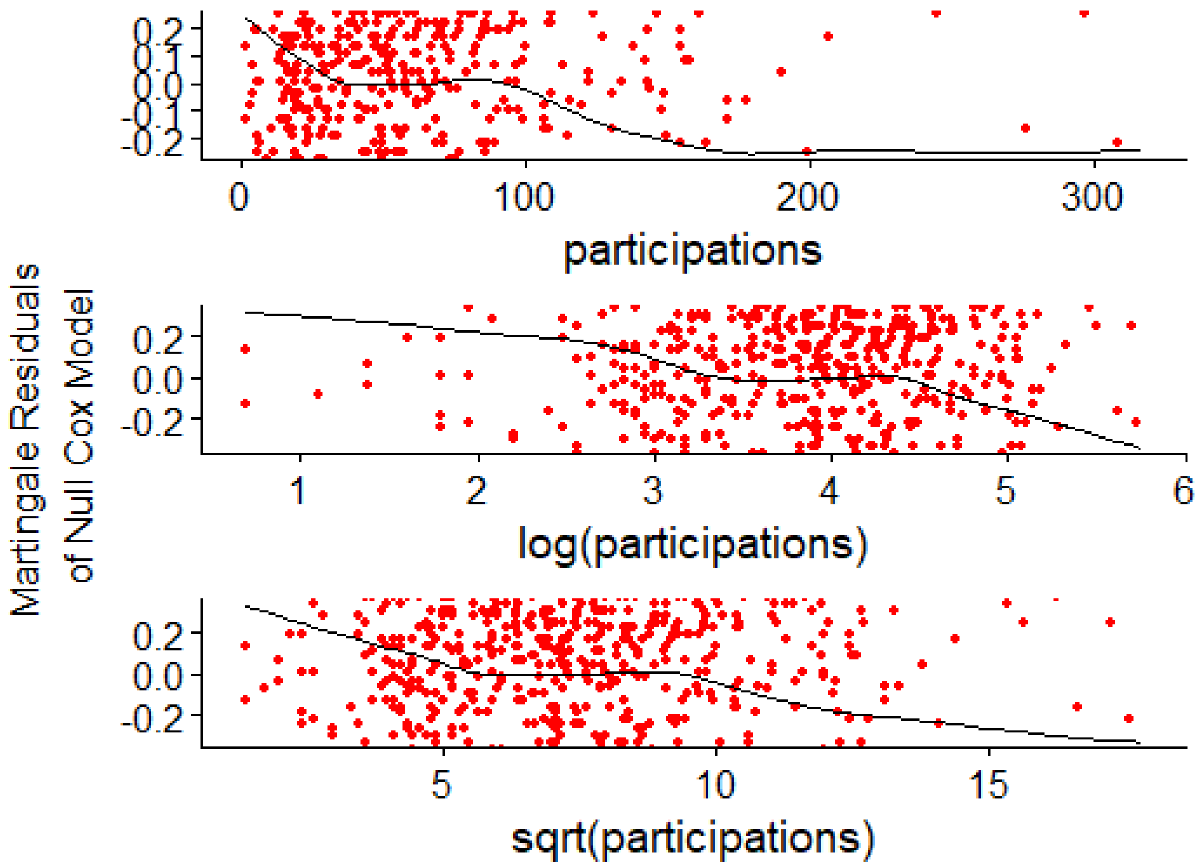


Figure 23: Testing non-linearity of predictor "participations" used in the Cox model.

As observed in figure 23, for the specific case of the variable "participations", all the continuous variables used in the model had a slightly non linear functional shape. This is possibly one of the factors contributing for the poor performance of the model.

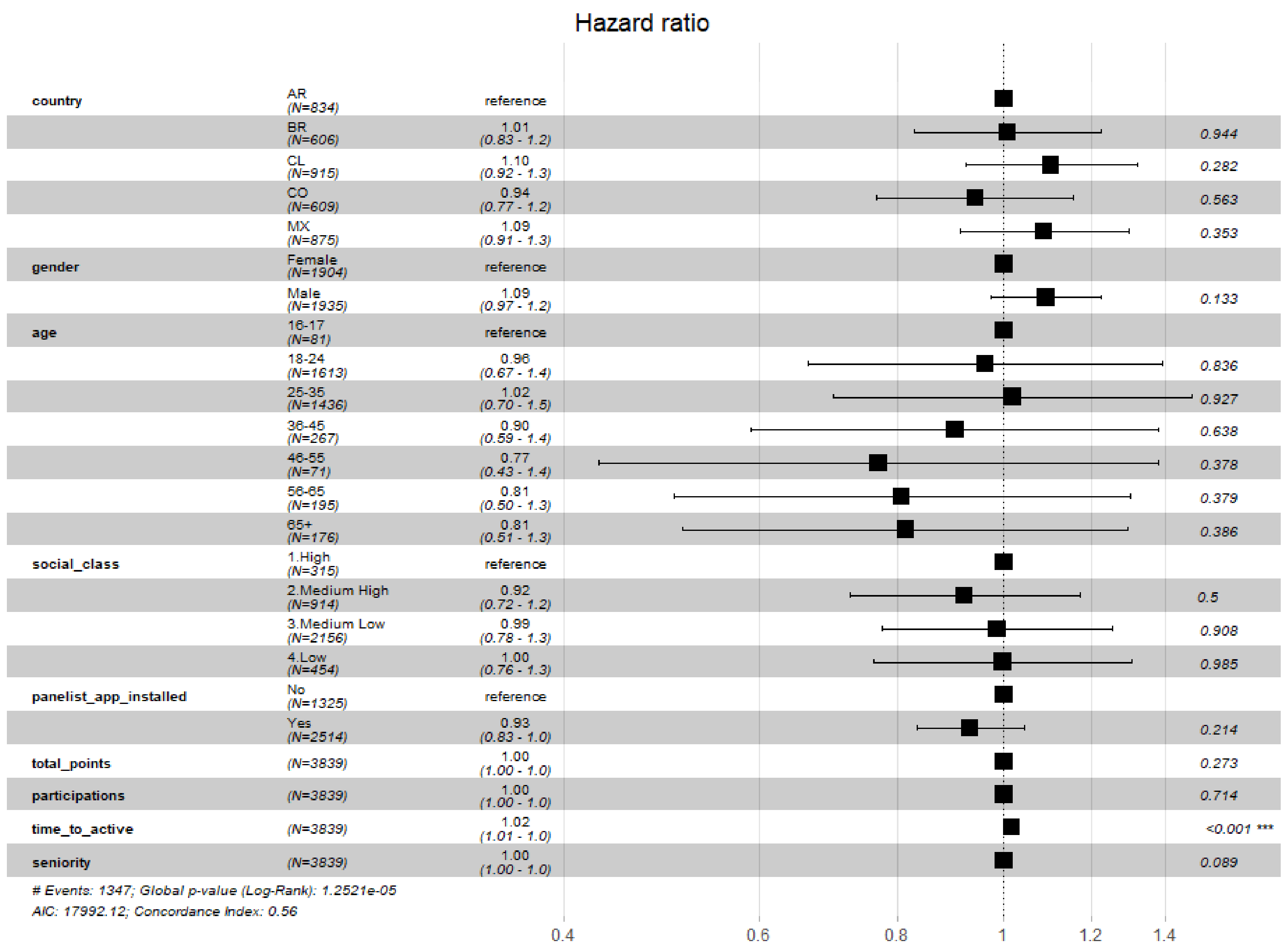


Figure 24: Cox Proportional-Hazards Model results.

4.4 Results and Discussion

The main conclusion of this chapter is that the predictors which are available at this moment, although reasonably good for predicting conversion of a panelists to an active Meterezized panelist, are not good enough in predicting retention.

Retention is a crucial point for the company strategy for the Meter Panel. If a panelist is not retained for a long time the 100 points given in exchange for the meter installation will not be paid back and the company will loose money.

Therefore, the action taken by the company which will not be evaluated here was to build a survey focused on two basic aspects: Internet data sharing opinions and issues; the panelists' experiences with the Premium proposal invitation, installation and sharing data.

This survey was organized as an experiment where different groups - (1) panelists who did not accept the Premium proposal or have ignored it; (2) panelists who have accepted the Premium proposal but were not retained after one month from becoming active; (3) panelists who have accepted the Premium proposal and were still active one month after the invitation - were compared.

The main object of this survey was to evaluate possible new variable that could later be used for predicting retention and also improving the predictions of conversion to active.

5 Conclusion

As mentioned before, this thesis aimed to approach the problem at hand holistically, applying different statistical analysis for each of the three main parts of this problem: (1) predicting conversion to active; (2) understanding life trajectories in the Meter Panel and (3) understanding and predicting retention of activated panelists.

Most of the efforts were put on the predictive model developed in chapter 2. Chapter 3 brings insights about the life trajectories and allowed checking if the statuses transition rules were working properly. Moreover, panelists were clustered into different groups given proximity between their sequence patterns allowing the planning of specific communication strategies addressed to each group. Finally, retention of panelists converted into active was analyzed. The need for better predictors was recognized leading to actions as the planning of a survey experiment to collect information from panelists on their opinions about sharing Internet navigation data and related issues.

A Appendix: Data Description

Three applications were developed in this thesis, therefore three data sets were used. These data sets are related in the following way:

1. The predictive model assigned probabilities of accepting the Premium proposal to all the panelists in the Access panel which were never invited before to become Premium. Data set 1 has 421,166 observations.
2. A subset containing 10,041 of the panelists in data set 1 was selected and these panelists were invited to become Premium. Among those 9,488 were still active in the Access Panel when this analysis was run and qualified for it. The life trajectories of these panelists during the first 96 days from the invitation were analyzed.
3. A subset of 3,635 panelists in data set 2 which have accepted the Premium invitations and became Meterized was selected. Among those 1,347 became INACTIVE during the first 60 days from the first active day. Moreover 225 cases were right censored since their survival/active time was longer than 60 day. The remaining 2,063 observations were also deleted since their survival time was unknown at the moment that this analysis was run. Survival curves and a Cox-Proportional Hazards Model is developed for the first 60 days from the invitation.

Next the variables constituting each data set will be listed. Noted that many variables are repeated in one, two or all three models.

1. Predictive Model Variables
   - *get_to_active*: equals 1 if the panelist has shared Internet navigation data (get to active) and 0 otherwise (has never got to active). It is the response variable for this model.
   - *country*: 13 countries - Spain (ES), Brazil (BR), Portugal (PT), Colombia (CO), Mexico (MX), Chile (CL), Argentine (AR), Peru (PE), Great Britain (GB), France (FR), Germany (DE), United States (US) and Italy (IT).
   - *gender*: Women = 0 and Men = 1.
   - *age*: in years from 16y to 98y with a mean age of 37y.
   - *ses_standard*[7]: standardized social class with the same five levels for each country, "unknown" = 0, "High" = 1, "MediumHigh" = 2, "MediumLow" = 3, "Low" = 4.

[7]For each country the official local definition of social class or information on income is measured on the panelists and later a standardization developed by the company is applied.

- *app*: equals 1 if a panelist has the company mobile application installed in one of his/her devices and 0 otherwise. This is an important variable since it is a strong indicator of engagement of the panelist with the company as well as an extra communication channel between both parts.
- *seniority*: measures in days how old a panelist is in the Access Panel.
- *recruitment*: equals 1 if the panelist was invited through a flag process and 1 if invited by the Direct Meter process.
- *invitations*: how many invitations to participate in surveys a panelist had from the first day in the Access Panel until the day of invitation.
- *participations*: how many invitations were converted into real participations. A panelist who has many participations is an engaged one and will be reminded of the extra points earned at each participation thus increasing the probability of retention of this panelist as a Meterized one.
- *invitations_30*: how many invitations to surveys in the 30 days before Premium proposal. This predictor intends to capture the momentus engagement of a panelist. For example, if a panelist who usually do many participations goes on vacations and stop answering to the survey invitations, it might be better not to offer the Premium proposal since there is a chance the panelist will miss it.
- *participations_30*: how many participations in the 30 days before Premium proposal. This plays a role altogether with the predictor `invitations_30`.
- *points*: how many points a panelist had when invited to become Premium.
- *exchanged_points*: how many points a panelist used to buy things before the Premium invitation.
- *n_exchanges*: how many times the panelists bought things in the company e-commerce.
- A list of 15 *Internet activities* organized as yes or no questions ("flight-purchase", "onlinebanking", "shopping", "casualonlinegaming", "socialmedia", "gambling", "downloaduploadphotos", "downloaduploadaudio", "downloadupload-films", "downloaduploadvideos", "jobsearch", "informationsearch", "onlinecourses", "playonlinegames", "watchvideos").
- *inv_part_01*: the ratio between *invitations* and *participations*.
- *inv_part_02*: the ratio between *invitations_30* and *participations_30*.
- *glm_demographics*: *get_to_active* $\sim$ *country* + *gender* + *age* + *ses_standard*. The variable is constituted by the predicted probabilities issued by a logistic model built using a specific subset of predictors with focus on demographics.

- *glm_panel*: *get_to_active* $\sim$ *app* + *seniority* + *recruitment*. The variable is constituted by the predicted probabilities issued by a logistic model built using a specific subset of predictors with focus on panelist features.
- *glm_inv_part*: *get_to_active* $\sim$ *invitations* + *participations* + *invitations_30* + *participations_30*. The variable is constituted by the predicted probabilities issued by a logistic model built using a specific subset of predictors with focus on the historic of invitations and participations to surveys in the Access Panel.
- *glm_points*: *get_to_active* $\sim$ *points* + *exchanged_points* + *n_exchanges*. The variable is constituted by the predicted probabilities issued by a logistic model built using a specific subset of predictors with focus on the points profile of each panelist in the Access Panel.

2. Cluster model based on status sequences dissimilarities

- *code_panelist*: An ID for identifying each panelist.
- *status*: The status of the panelist at each day during the first 96 days from the invitation.
- *timestamp*: The counting of days.

3. Regression tree based on on status sequences dissimilarities

- *country*: 13 countries - Spain (ES), Brazil (BR), Portugal (PT), Colombia (CO), Mexico (MX), Chile (CL), Argentine (AR), Peru (PE), Great Britain (GB), France (FR), Germany (DE), United States (US) and Italy (IT).
- *gender*: Women = 0 and Men = 1.
- *age*: in years from 16y to 98y with a mean age of 37y.
- *ses_standard*: standardized social class with the same five levels for each country, "unknown" = 0, "High" = 1, "MediumHigh" = 2, "MediumLow" = 3, "Low" = 4.
- *app*: equals 1 if a panelist has the company mobile application installed in one of his/her devices and 0 otherwise.

4. Cox-Proportional Hazards Model

- *country*: 13 countries - Spain (ES), Brazil (BR), Portugal (PT), Colombia (CO), Mexico (MX), Chile (CL), Argentine (AR), Peru (PE), Great Britain (GB), France (FR), Germany (DE), United States (US) and Italy (IT).
- *gender*: Women = 0 and Men = 1.

- *`age`*: in years from 16y to 98y with a mean age of 37y.
- *`ses_standard`*: standardized social class with the same five levels for each country, “unknown” = 0, “High” = 1, “MediumHigh” = 2, “MediumLow” = 3, “Low” = 4.
- *`app`*: equals 1 if a panelist has the company mobile application installed in one of his/her devices and 0 otherwise.
- *`total_points`*: Total number of points acumulated by the panelists at the moment he or she became an active Meterized panelist.
- *`participations`*: Cumulative number of participations in surveys in the Access Panel at the moment he or she became an active Meterized panelist.
- *`time_to_active`*: How many days to become an active Meterized panelist from the Premium invitation day.
- *`seniority`*: measures in days how old a panelist is in the Access Panel.

References

(2018). Cox model assumptions. http://www.sthda.com/english/wiki/cox-model-assumptions.

(2018). Cox proportional-hazards model. http://www.sthda.com/english/wiki/cox-proportional-hazards-model.

Abbott, A., Forrest, J. (1986). Optimal matching methods for historical sequences. *The Journal of Interdisciplinary History*, *16*(3), 471–494.

Anderson, J., Bernstein, L., Pike, M. (1982). Approximate confidence intervals for probabilities of survival and quantiles in life-table analysis. *Biometrics*, pp. 407–416.

Bland, J. M., Altman, D. G. (2004). The logrank test. *Bmj*, *328*(7447), 1073.

Blastland, M., Dilnot, A. W. (2008). *The tiger that isn't: seeing through a world of numbers*. Profile books.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*(2), 123–140, URL https://doi.org/10.1007/BF00058655.

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5–32, URL https://doi.org/10.1023/A:1010933404324.

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

Breiman, L., et al. (2001). Statistical modeling: The two cultures (with comments and a rejoinder by the author). *Statistical science*, *16*(3), 199–231.

Brick, J. M. (2011). The future of survey sampling. *Public Opinion Quarterly*, *75*(5), 872–888.

Brodersen, K. H., Ong, C. S., Stephan, K. E., Buhmann, J. M. (2010). The balanced accuracy and its posterior distribution. Em: *Pattern recognition (ICPR), 2010 20th international conference on*, IEEE, pp. 3121–3124.

Brownlee, J. (2014). *Discover Feature Engineering, How to Engineer Features and How to Get Good at It*. URL https://machinelearningmastery.com/discover-feature-engineering-how-to-engineer-features-and-how-to-get-good-at-it/s.

Callegaro, M., DiSogra, C. (2008). Computing response metrics for online panels. *Public opinion quarterly*, *72*(5), 1008–1032.

Chen, T., Guestrin, C. (2016). Xgboost: A scalable tree boosting system. Em: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pp. 785–794, URL http://doi.acm.org/10.1145/2939672.2939785.

Chen, T., He, T., Benesty, M., Khotilovich, V., Tang, Y. (2018). *xgboost: Extreme Gradient Boosting*. URL https://CRAN.R-project.org/package=xgboost, R package version 0.6.4.1.

Cleves, M., Gould, W., Gould, W. W., Gutierrez, R., Marchenko, Y. (2008). *An introduction to survival analysis using Stata*. Stata press.

Codd, E. F. (1970). A relational model of data for large shared data banks. *Communications of the ACM*, *13*(6), 377–387.

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and psychological measurement*, *20*(1), 37–46.

Cox, D. R. (1992). Regression models and life-tables. Em: *Breakthroughs in statistics*, Springer, pp. 527–541.

Diez, D. (2013). Survival analysis in R. *OpenIntro org.*

Elandt-Johnson, R. C., Johnson, N. L. (1980). *Survival models and data analysis*, vol 74. John Wiley & Sons.

Fawcett, T. (2006). An introduction to ROC analysis. *Pattern recognition letters*, *27*(8), 861–874.

Freund, Y., Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *J Comput Syst Sci*, *55*(1), 119–139, URL https://doi.org/10.1006/jcss.1997.1504.

Friedman, J., Hastie, T., Tibshirani, R., et al. (2000). Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *The Annals of Statistics*, *28*(2), 337–407.

Friedman, J., Hastie, T., Tibshirani, R. (2001). *The elements of statistical learning*, vol 1. Springer series in statistics New York, NY, USA:.

Friedman, J. H. (2000). Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, *29*, 1189–1232.

Friedman, J. H. (2002). Stochastic gradient boosting. *Computational Statistics & Data Analysis*, *38*(4), 367–378.

Gabadinho, A., Ritschard, G., Müller, N. S., Studer, M. (2011). Analyzing and visualizing state sequences in R with TraMineR. *Journal of Statistical Software*, *40*(4), 1–37.

Hahn, G. J., Doganaksoy, N. (2012). *A career in statistics: Beyond the numbers*. John Wiley & Sons.

He, H., Ma, Y. (2013). *Imbalanced learning: foundations, algorithms, and applications*. John Wiley & Sons.

Hosmer Jr, D. W., Lemeshow, S., Sturdivant, R. X. (2013). *Applied logistic regression*, vol 398. John Wiley & Sons.

James, G., Witten, D., Hastie, T., Tibshirani, R. (2013). *An introduction to statistical learning*, vol 112. Springer.

Johnson, R., Zhang, T. (2014). Learning nonlinear functions using regularized greedy forest. *IEEE transactions on pattern analysis and machine intelligence*, *36*(5), 942–954.

Kaplan, E. L., Meier, P. (1958). Nonparametric estimation from incomplete observations. *Journal of the American statistical association*, *53*(282), 457–481.

Kassambara, A. (2017). Practical guide to cluster analysis in r. *CreateSpace: North Charleston, SC, USA*.

Kassambara, A., Kosinski, M. (2018). *survminer: Drawing Survival Curves using 'ggplot2'*. URL https://CRAN.R-project.org/package=survminer, R package version 0.4.2.

Kearns, M. (1988). Thoughts on hypothesis boosting. *Unpublished manuscript (Machine Learning class project, December 1988)*, URL https://www.cis.upenn.edu/ mkearns/-papers/boostnote.pdf.

Krippendorff, K. (1970). Bivariate agreement coefficients for reliability of data. *Sociological methodology*, *2*, 139–150.

Kuhn, M., Johnson, K. (2013). *Applied predictive modeling*, vol 26. Springer.

Kuhn, M., Wing, J., Weston, S., Williams, A., Keefer, C., Engelhardt, A., Cooper, T., Mayer, Z., Kenkel, B., the R Core Team, Benesty, M., Lescarbeau, R., Ziem, A., Scrucca, L., Tang, Y., Candan, C., Hunt., T. (2018). *caret: Classification and Regression Training*. URL https://CRAN.R-project.org/package=caret, R package version 6.0-79.

Laurae (2018). xgboost: "hi i'm gamma. what can i do for you?"—and the tuning of regularization. https://medium.com/data-design/xgboost-hi-im-gamma-what-can-i-do-for-you-and-the-tuning-of-regularization-a42ea17e6ab6, [Online; accessed 19-July-2018].

Lundberg, S. M., Erion, G. G., Lee, S. I. (2018). Consistent individualized feature attribution for tree ensembles. *arXiv preprint arXiv:180203888*.

Maechler, M., Rousseeuw, P., Struyf, A., Hubert, M., Hornik, K. (2018). *cluster: Cluster Analysis Basics and Extensions*. R package version 2.0.7-1 — For new features, see the 'Changelog' file (in the package source).

McCullagh, P. (2002). What is a statistical model? *Annals of Statistics*, pp. 1225–1267.

McQuivey, J. (2013). *Digital disruption: Unleashing the next wave of innovation*. Forrester Research, Incorporated.

Murtagh, F., Legendre, P. (2011). Ward's hierarchical clustering method: clustering criterion and agglomerative algorithm. *arXiv preprint arXiv:11116285*.

Ng, A. Y. (2004). Feature selection, L1 vs. L2 regularization, and rotational invariance. Em: *Proceedings of the twenty-first international conference on Machine learning*, ACM, p 78.

Nielsen, D. (2016). Tree boosting with xgboost - why does xgboost win "every" machine learning competition? Master's Thesis, NTNU: Norwegian University of Science and Technology.

Painsky, A., Wornell, G. W. (2018). On the universality of the logistic loss function. *arXiv preprint arXiv:180503804*.

Pontius Jr, R. G., Millones, M. (2011). Death to kappa: birth of quantity disagreement and allocation disagreement for accuracy assessment. *International Journal of Remote Sensing*, *32*(15), 4407–4429.

R Core Team (2018). *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria, URL https://www.R-project.org/.

Reitermanov'a, Z. (2010). Data splitting. *WDS'10 Proceedings of Contributed Papers, Part I, 31–36*, charles University, Faculty of Mathematics and Physics, Prague, Czech Republic.

Rosenfield, G. H., Fitzpatrick-Lins, K. (1986). A coefficient of agreement as a measure of thematic classification accuracy. *Photogrammetric engineering and remote sensing*, *52*(2), 223–227.

Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, *3*(3), 210–229.

Schapire, R. E. (1990). The strength of weak learnability. *Machine Learning*, *5*, 197–227, URL https://doi.org/10.1007/BF00116037.

Sing, T., Sander, O., Beerenwinkel, N., Lengauer, T. (2005). Rocr: visualizing classifier performance in R. *Bioinformatics*, *21*(20), 7881, URL http://rocr.bioinf.mpi-sb.mpg.de.

Stone, H. S. (1971). *Introduction to computer organization and data structures.* McGraw-Hill, Inc..

Studer, M., Ritschard, G. (2016). What matters in differences between life trajectories: a comparative review of sequence dissimilarity measures. *Journal of the Royal Statistical Society, Series A*, *179*(2), 481–511.

Studer, M., Ritschard, G., Gabadinho, A., Müller, N. S. (2011). Discrepancy analysis of state sequences. *Sociological Methods & Research*, *40*(3), 471–510.

Therneau, T. M. (2015). *A Package for Survival Analysis in S.* URL https://CRAN.R-project.org/package=survival, version 2.38.

Wikipedia (2018). Taylor series — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Taylor%20seriesoldid=849947131, [Online; accessed 19-July-2018].

XGBoost-Developers (2018). Xgboost parameters description. https://xgboost.readthedocs.io/en/latest/
parameter.htmlparameters-in-r-package, accessed: 2018-07-31.