ERASMUS UNIVERSITY ROTTERDAM
Erasmus School of Economics

Bachelor Thesis Econometrics and Operations Research

Predicting and Explaining the Success of Bank Telemarketing

Name student: Julian van Erk
Student ID number: 471545je

Supervisor: Mikhail Zhelonkin
Second assessor: Utku Karaca

Date final version: July 5, 2020

Abstract

In this thesis, I study different methods that can predict and explain the success of bank telemarketing. I compare several methods of data mining, class imbalance sampling and variable selection on their predictive performance. For this, I use real data from the UCI Machine Learning Repository (Dua & Graff, 2019) on bank telemarketing campaigns of a Portuguese bank. This is the closest available dataset to the data analysed by Moro et al. (2014). XGBoost, in combination with no class imbalance technique and variable selection via group lasso with ridge regularization, turns out to be the best performing combination of methods. I apply sensitivity analysis and rule extraction on this method and find that managers can use this information to create a more profitable call strategy.

Contents

1 Introduction

In this thesis, I study methods that can predict and explain the success of bank telemarketing. My focus is on the following research question: *Which data preprocessing techniques and data mining models perform best in predicting the success of bank marketing? And how can one help managers to understand the best performing model better?* To answer this question, I compare data mining, class imbalance sampling and variable selection methods. Subsequently, I apply sensitivity analysis and rule extraction with the best performing combination of methods.

Due to the high level of competitiveness of the banking sector, marketing is an important tool for gaining new clients. This marketing takes place via mass marketing and direct marketing (Miguéis et al., 2017). Mass marketing has proven to be inefficient, such that there has been a shift from mass marketing to direct marketing (Su et al., 2006). According to Miguéis et al. (2017), direct marketing is now the most important way to acquire new customers for banks. Direct marketing is also important for interaction with existing customers. A popular direct marketing method is telemarketing, where long term deposits are sold by phone. Lau et al. (2004) describe that data mining (DM) techniques can be used to describe and predict customer behaviour in this telemarketing setting. By developing better DM techniques, one can help managers of banks to make marketing campaigns more efficient and achieve better target selections. For example, Moro et al. (2014) use DM techniques and find that they can filter potential customers in such a way that 79% of successful contacts are reached with only half of the calls.

To answer the research questions, I use real data on bank telemarketing campaign of a Portuguese bank during the period May 2008 to November 2010. The data consists of information on phone calls where people were asked if they want to subscribe to a bank term deposit. This dataset has been published by the University of California at the UCI Machine Learning Repository (Dua & Graff, 2019) and is the closest available dataset to the data analysed by Moro et al. (2014).

This thesis extends the research done by Moro et al. (2014). In addition to the DM methods used by Moro et al. (2014), I apply two additional DM methods: extreme gradient boosting or XGBoost and group lasso with ridge regularization. Furthermore, I use two class imbalance sampling techniques: the synthetic minority oversampling technique and Tomek links. I also use two variable selection methods: one using a genetic algorithm and the other using the group lasso with ridge regularization.

I find that XGBoost with no class imbalance technique and variable selection via group lasso with ride regularization is the best performing combination of methods. This combination has a superior predictive performance relative to the methods used in Moro et al. (2014), which enables telemarketing managers to improve target selection. I also find that DM methods using regularization or more flexible functions outperform others. Most DM methods do not profit from applying class imbalance techniques. Applying variable selection gives similar results as no variable selection while using fewer variables. I show that sensitivity analysis and rule extraction can be used to create a more profitable strategy.

In Section 2 I discuss relevant literature on the subject of bank telemarketing. Section 3 describes relevant characteristics of the used data. I explain DM, class imbalance and variable selection methods in Section 4. In this section, I also explain how to apply sensitivity analysis and rule extraction. I present the results in Section 5 and conclude my research in Section 6.

2 Literature

Lau et al. (2004) are the first to describe the potential usefulness of DM in bank telemarketing, but do not apply DM. Moro et al. (2012) use data from past telemarketing campaigns to make future long term deposit campaigns more efficient and use three DM methods for this: support vector machine,

naive Bayes and a decision tree. Out of these methods, the support vector machine outperforms the other methods, based on the area under the receiver operating characteristic curve (AUC) and the area under the lift cumulative curve (ALIFT), as described in Hanley & McNeil (1982) and Witten et al. (2016) in Chapter 5, respectively. In a follow-up study, Moro et al. (2014) compare decision trees, logistic regression, neural networks and support vector machines. In this study, Moro et al. (2014) show that the neural network method outperforms all other methods in predicting the success of bank telemarketing. Miguéis et al. (2017) find that a random forest method outperforms neural network and support vector machine based methods. Elsalamony (2014) compares a decision tree, neural network, naive Bayes and logistic regression method, and concludes that the decision tree is the best performing method, although this is based solely on the training set. This thesis extends the current literature by applying two new DM methods to the bank telemarketing dataset. First, I apply a gradient boosting technique with regularization named XGBoost (Chen & Guestrin, 2016), using sequential model-based optimization or SMBO (Hutter et al., 2011) for determining the hyperparameters. The second new method is a variant of the group lasso for logistic regression (Meier et al., 2008).

In a follow-up paper, Moro et al. (2015) achieve better results when variable extraction takes place. For this purpose, they have added customer lifetime value variables. In particular, Moro et al. (2015) use recency, frequency and monetary (RFM) characteristics to extract variables. In another follow-up paper, Moro et al. (2018) use domain experts to find additional variables and apply variable selection, which results in even better performance of the DM methods. Variable selection has been applied more often. Moro et al. (2014) use a semi-automatic approach to reduce the number of variables from 150 to 22, and this approach outperforms the approach without variable selection or with a forward selection procedure. Vajiramedhin & Suebsing (2014) use Pearson's correlation to select variables, based on the hypothesis that appropriate variables are uncorrelated and strongly correlated with the response variable (Hall & Smith, 1999). The authors combine this with a decision tree and find that variable selection improves predictive performance. Barraza et al. (2019) compare a mutual information and sensitivity analysis approach for variable selection using logistic regression. They find that both methods improve predictive performance, although both methods select other variables. I extend the literature by using the group lasso for logistic regression (Meier et al., 2008) and a genetic algorithm (Goldberg & Holland, 1988) as variable selection techniques.

According to Miguéis et al. (2017), the imbalance of the bank telemarketing data is a major problem, leading to models with poor predictive performance. Therefore, they apply two class imbalance sampling techniques: EasyEnsemble (Liu et al., 2008) and synthetic minority oversampling technique or SMOTE (Chawla et al., 2002). They use this in combination with random forests, neural networks, support vector machines and logistic regression. The authors find that EasyEnsemble works best for the random forests, while SMOTE gives the best performance for the other methods. As I do not use random forests, I apply SMOTE. I also use Tomek links (Tomek, 1976).

3 Data

The dataset I use has been published by the University of California at the UCI Machine Learning Repository (Dua & Graff, 2019) and is the closest available dataset to the data analysed by Moro et al. (2014). The dataset contains real data on bank telemarketing campaigns of a Portuguese bank. These campaigns consist of information on phone calls where people were asked if they want to subscribe to a bank term deposit. The bank does not filter clients. The phone call is considered a success if a term deposit is made. In total there are 41188 observations from May 2008 to November 2010. There are 12 duplicate rows which I remove, such that there remain 41176 observations.

The dataset contains 20 variables to predict the outcome of the call. These variables contain information on the bank client, social/economic attributes, the call itself and previous contacts. Table 1

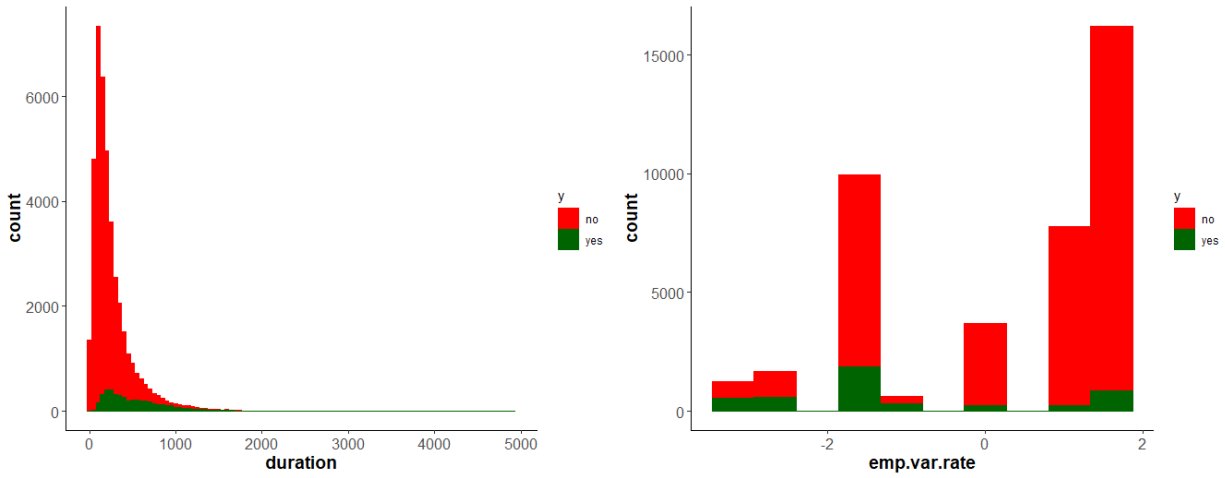
Table 1: Description and type of the variables from the bank telemarketing dataset from the UCI Machine Learning Repository (Dua & Graff, 2019).

Variable	Description	Type
y	Indicates whether client has subscribed to a term deposit	Categorical
age	Age of the client	Numeric
job	Job of the client	Categorical
marital	Marital status of the client	Categorical
education	Education of the client	Categorical
default	Indicates whether client has credit in default	Categorical
housing	Indicates whether client has a housing loan	Categorical
loan	Indicates whether client has a personal loan	Categorical
contact	Type of call contact communication	Categorical
month	Month of call	Categorical
day_of_week	Day of call	Categorical
duration	Duration of call in seconds	Numeric
campaign	Number of contacts made during current campaign for this client	Numeric
pdays	Days since last contact previous campaign for this client	Numeric
previous	Number of contacts for this client before current campaign	Numeric
poutcome	Outcome previous campaign for this client	Categorical
emp.var.rate	Employment variation rate (quarterly indicator)	Numeric
cons.price.idx	Consumer price index (monthly indicator)	Numeric
cons.conf.idx	Consumer confidence index (monthly indicator)	Numeric
euribor3m	Euribor 3 month interest rate (daily indicator)	Numeric
nr.employed	Number of employed citizens in Portugal	Numeric

shows the description and type of these variables and the output variable y. Note that *duration* is not known before a call is made and therefore my predictive models can only be considered as benchmarks for a realistic model. I choose to keep *duration* such that I can do inference on all variables.

I split the data into a training set and a test set. I use the training set to build the models and the test set to evaluate the models. The training set contains the first 40000 observations (May 2008 to June 2010), and the test set consists of the last 1176 observations (June 2010 to November 2010).

Only 4639 (11.27%) observations of the dataset are successes. Thus, the dataset is imbalanced. Many classification methods only work well if the dataset is balanced (Tan et al., 2018, Chapter 6) because they try to minimize the number of errors during the classification. As a result, classifiers often try to improve performance over the majority class, such that they do not detect minority class



(a) (b)

Figure 1: Two histograms describing the distribution of (a) duration in seconds of the call and (b) employment variation rate in per cent. Every bar is split into the number of successes (yes) and failures (no).

Table 2: Correlation between the economic variables. All correlations are significant with p-value $p < 0.05$.

	emp.var.rate	*cons.price.idx*	*cons.conf.idx*	*euribor3m*	*nr.employed*
emp.var.rate	1				
cons.price.idx	0.775	1			
cons.conf.idx	0.196	0.059	1		
euribor3m	0.972	0.688	0.278	1	
nr.employed	0.907	0.522	0.101	0.945	1

instances effectively. To solve this problem, I use several sampling techniques.

Because the dataset is imbalanced, the fraction of correctly classified instances, the accuracy, is not an appropriate performance measure. When I classify all instances of the dataset as a failure, I have an accuracy of 88.73%. Therefore, I use performance measures that take class imbalance into account.

To obtain a first image of the effect of a few variables on the probability of success, consider Figure 1. Figure 1 displays two histograms, describing the distribution of *duration* and *emp.var.rate*. Moreover, the histograms show the number of successes and failures for a specific interval of the distribution. It can be seen that the relative number of successes is higher for high values of *duration* and low values of *emp.var.rate*. Therefore, these variables are two potentially important variables for describing and predicting the success of a call.

In Table 2 one can see that the economic variables have a high correlation with one another. This multicollinearity can be a problem for certain classification methods as estimating the isolated effect of a variable is more difficult in case of multicollinearity (Cameron & Trivedi, 2005, Chapter 10). Moreover, some variables may not influence the dependent variable. These variables create noise in the model and could affect the performance of the classifier negatively (Tan et al., 2018, Chapter 2). Therefore, I use variable selection to select the most useful variables.

Categorical variables are dummy coded. Thus, a variable with n levels has $n - 1$ columns in the design matrix. Dependent variable y is equal to 1 if the call is a success and 0 otherwise.

4 Methodology

I use several techniques to explain and predict the success of bank telemarketing. In Section 4.1, I explain the procedure for answering the research question. Subsequently, in Section 4.2, I discuss the performance measures I use. Section 4.3 discusses the used DM methods. I explain several class imbalance techniques in Section 4.4, and Section 4.5 discusses the used variable selecting methods.

4.1 Procedure

Figure 2 shows the procedure I follow to answer the research question. I start with creating three sets of variables, using no variable selection (NO), a genetic algorithm (GA) and group lasso with ridge regularization (GLA). In the following steps, I combine the variable selection with class imbalance techniques: no sampling (NO), synthetic minority oversampling technique (SMOTE), Tomek links (TL) and a combination of SMOTE and TL (SMOTETL).

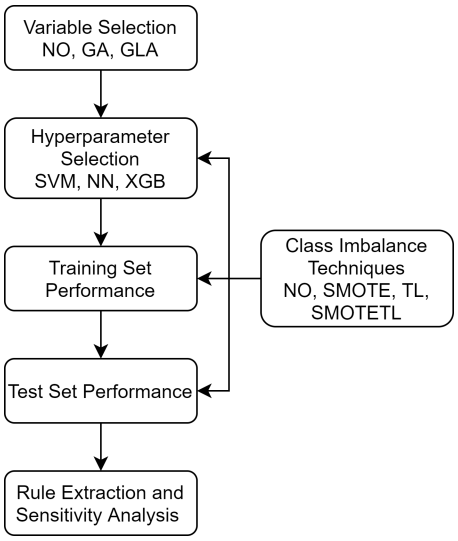


Figure 2: Procedure for answering the research question.

The combination of these preprocessing techniques results in a modified training set. I subsequently use this modified training set to build a predictive model with DM methods. The DM methods I use are logistic regression (LR), group lasso with ridge regularization (GLA), support vector machine (SVM), neural network (NN), decision tree (DT) and XGBoost (XGB).

SVM, NN, XGB and GLA have hyperparameters that need to be determined before training. I determine the hyperparameters of SVM, NN and XGB for every combination of preprocessing techniques. This way of hyperparameter selection is too time-consuming for GLA. Therefore, The hyperparameters of GLA are determined in the variable selection phase and are thus independent of used preprocessing techniques.

Using the selected hyperparameters, I test the performance of the methods on the training set. I do this by dividing the training set into an internal training and validation set for 20 distinct runs. For every run, the internal training set consists of a random sample of $\frac{2}{3}$ of the observations and the validation set of the other $\frac{1}{3}$. I use the same holdout scheme for every combination of methods. I test the performance by calculating the average AUC and ALIFT, using a Mann-Whitney test to check for statistical significance, as described by Wackerly et al. (2014) in Chapter 15.

I also evaluate the models on the test set. I do this in the same way as Moro et al. (2014) by using a rolling window with a length of 20000 observations. After predicting ten instances, I update the training set, replacing the ten oldest observations with the ten newest. I make predictions using every combination of DM, class imbalance and variable selection methods. I evaluate these predictions by calculating the AUC and ALIFT. The combination of preprocessing techniques that contains the DM method with the highest AUC is evaluated in more depth. I do this by plotting the ROC and lift cumulative curves of all DM methods belonging to this combination.

As in Moro et al. (2014), I apply sensitivity analysis and rule extraction on the combination of methods that has the highest AUC value.

4.2 Performance Measures

As concluded in the data section, the accuracy is not an appropriate performance measure in this application. Two measures that are invariant to class distribution are the true positive rate (TPR) and the false positive rate (FPR), defined as:

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{1}$$

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}, \tag{2}$$

where TP is the number of correctly predicted positives, FP the number of incorrectly predicted negatives, FN the number of incorrectly predicted positives and TN the number of correctly predicted negatives, as described by Tan et al. (2018) in Chapter 6. For all $D \in [0, 1]$ the TPR and FPR can be calculated by classifying an instance as positive if $\Pr(y_i = 1|x_i) \geq D$. By plotting the TPR against the FPR for all values of D, a receiver operating characteristic (ROC) curve is created (Tan et al., 2018, Chapter 6). If the ROC curve of one method dominates the curve of another method, one can say that this method outperforms the other (Tan et al., 2018, Chapter 6). Calculating the area under the ROC curve results in one of the most used performance measures with imbalanced data, the area under the curve (AUC) (Martens et al., 2011). This measure also has the advantage of being invariant to class imbalance (Tan et al., 2018, Chapter 6). An AUC of 1 means an ideal method, while a method with an AUC of 0.5 is equivalent to a random classifier.

Moro et al. (2014) also use another tool to measure performance, the lift cumulative curve. This curve is created by ordering the test set observations by the estimated probability of success, from high to low. Subsequently, the ordered population sample is plotted against the cumulative percentage of

captured successes. The area under this curve is called the ALIFT, as described by Witten et al. (2016) in Chapter 5. Lift analysis can determine how many successes would be captured if, for example, one were only allowed to call half of the customers. A perfect model has an ALIFT equal to $1-0.5p$, where p is the fraction of successes in the test set relative to the total number of instances. The maximum ALIFT of the test set is equal to 0.746, while a random classifier has an ALIFT of 0.5.

4.3 Data Mining Methods

In this section, I discuss the DM methods I use. First, I describe the logistic regression and group lasso with ridge regularization method. Subsequently, I describe the support vector machine and the neural network method. Finally, I explain two tree-based methods: decision tree and XGBoost.

4.3.1 Logistic Regression

The first method I use is logistic regression (LR). LR calculates probability of success for call i, given data x_i as follows:

$$\Pr(y_i = 1|x_i) = \frac{\exp(\beta_0 + \sum_{g=1}^{G} \beta_g x_{i,g})}{1 + \exp(\beta_0 + \sum_{g=1}^{G} \beta_g x_{i,g})}, \tag{3}$$

where $x_i = (x_{i,1}, \ldots, x_{i,G})$ and $\beta = (\beta_0, \ldots, \beta_G)'$. $x_{i,g}$ is the vector of values corresponding to variable g for call i and β_g is the corresponding parameter vector for variable g. Furthermore, β_0 is an intercept. The parameters can be estimated by maximizing the folowing log-likelihood function

$$l(\beta) = \sum_{i=1}^{N} [y_i \log\{\Pr(y_i = 1|x_i)\} + \{1 - y_i\} \log\{1 - \Pr(y_i = 1|x_i)\}], \tag{4}$$

where N is the number of calls the method is trained on, and y_i is the actual outcome (Franses & Paap, 2001, Chapter 4). This can be done with maximum likelihood estimation, as described by Franses & Paap (2001) in Chapter 4.

An advantage of this method is its easy interpretation. Each variable has a parameter that can be used to analyse the relationship between the variables and the outcomes. Also, LR is relatively easy and fast to implement. The simplicity of the method also has disadvantages, namely that it cannot describe complex nonlinear relations (Moro et al., 2014). Furthermore, LR does not make a trade-off between training performance and model complexity, such that applying LR easily leads to overfitting (Tan et al., 2018, Chapter 6).

4.3.2 Group Lasso with Ridge Regularization

Our second DM method is based on group lasso, first proposed by Meier et al. (2008). This method tries to overcome the overfitting problem of LR. The probability of success is calculated in the same way as LR, see (3). The difference between these two methods is that the parameters are trained using a different optimization function. The optimization function contains regularization terms, which can be used to prevent overfitting (Bishop, 2006, Chapter 1). According to Bühlmann & Van De Geer (2011), the group lasso performs well in both prediction and variable selection.

Before fitting, the design matrix X should first be standarized to $\tilde{X}$ such that $\frac{1}{N}\tilde{X}_g'\tilde{X}_g = I$, where $\tilde{X}_g$ denotes the columns of $\tilde{X}$ belonging to variable $g = 1, 2, \ldots, G$. This standardization is needed to handle correlation between different variables (Simon & Tibshirani, 2012). As shown by Breheny & Huang (2015), this can be achieved by the linear tranformation

$$\tilde{X}_g = X_g Q_g A_g^{-\frac{1}{2}}, \text{ for } g = 1, 2, \ldots, G, \tag{5}$$

where Q_g is an orthonormal matrix of the eigenvectors of $\frac{1}{N}X_g'X_g$ and A_g is a diagonal matrix that contains the eigenvalues of $\frac{1}{N}X_g'X_g$.

Let $\tilde{\beta} = (\tilde{\beta}_0, \tilde{\beta}_1, \ldots, \tilde{\beta}_G)'$, where $\tilde{\beta}_g$ is the coefficient vector for variable $g = 1, 2, \ldots, G$ in the transformed scale and $\tilde{\beta}_0$ is an intercept. Furthermore, let $\tilde{\beta}_{feat}$ be the coefficients of the variables in the transformed space without intercept, such that $\tilde{\beta} = (\tilde{\beta}_0, \tilde{\beta}_{feat}')'$. One can subsequently estimate $\tilde{\beta}$ by minimizing

$$S_{\lambda,\alpha}(\tilde{\beta}) = -l(\tilde{\beta}) + \lambda\alpha \left(\sum_{g=1}^{G} \sqrt{df_g} \left\| \tilde{\beta}_g \right\|_2 \right) + \frac{1}{2}\lambda(1-\alpha) \left\| \tilde{\beta}_{feat} \right\|_2^2, \tag{6}$$

where $l(\tilde{\beta})$ is equal to the original log-likelihood in (4), df_g is the number of parameters for group g, $\|\|_2$ is the L2 norm, $\lambda \geq 0$ determines the penalty, and $\alpha \in (0, 1]$ determines the mix between group lasso and rigde regularization. Note that the intercept coefficient is not penalized. If $\alpha = 1$, (6) simplifies to the objective function of group lasso (Meier et al., 2008). For a particular value of λ and α one can estimate parameters $\tilde{\beta}$ using the group descent algorithm, as described in Breheny & Huang (2015). Furthermore, the parameters can be transformed back to the original scale using

$$\beta_g = Q_g A_g^{-\frac{1}{2}} \tilde{\beta}_g, \text{ for } g = 1, 2, \ldots, G. \tag{7}$$

Equation (6) consists of two different penalties. The left penalty applies a lasso penalty (Tibshirani, 1996) on the L2 norm of every group. In this way, the parameters of $\tilde{\beta}_g$ and therefore β_g are either all zero or all nonzero. The other penalty is a ridge penalty for individual coefficients. In this way, also less important levels of categorical variables shrink. Moreover, in Section 3 I concluded that the economic variables have a high correlation. By adding a ridge penalty, coefficients of highly correlated variables shrink toward each other (Hastie et al., 2009, Chapter 3).

To determine α and λ, I follow the same procedure as Friedman et al. (2010). I use k-fold cross-validation and split the training data into 10 equally sized groups. Moreover, the positive instances are equally distributed among the groups. I subsequently train the model 10 times, using 9 of the 10 groups. The other group is used for validation. I search hyperparameter α under the values $\alpha \in \{0.001, 0.05, 0.10, \ldots, 0.95, 1\}$. For λ, I search under 100 different values, evenly spaced on the log-scale and $\lambda_{max} = 1000\lambda_{min}$. Moreover, λ_{max} depends on the data and α, as described by Friedman et al. (2010).

For every α, λ combination and for every fold, I train the model parameters β and determine the deviance over the validation set. The deviance is equal to $-2l(\beta)$, where $l(\beta)$ is the same as in (4). In this way, I can calculate the mean and standard deviation of the deviance for every α, λ combination. Instead of selecting the α, λ combination with the minimum deviance, I again follow the procedure described by Friedman et al. (2010). For every α I choose λ_{1SE}, which is the largest λ such that the deviation is within one standard deviation of λ_{min} that minimizes the deviance. Due to the larger λ, the model more restrictive while having an objective comparable to that of λ_{min}. I subsequently choose the α, λ_{1SE} combination with the minimum deviance.

4.3.3 Support Vector Machine

The support vector machine (SVM) tries to create an optimal hyperplane to separate classes. This hyperplane can be created in the variable space, but to create non-linear decision boundaries one can transform the data with a function $\varphi(...)$. This gives a hyperplane in the transformed space of the form $h(x) = \mu + \nu'\varphi(x)$, where x is a vector of input variables, and μ and ν parameters of the hyperplane. The margin of this hyperplane, which is the sum of the distances from the hyperplane to the nearest point of the class belonging to that side, should be as large as possible and is equal to $\frac{1}{||\nu||}$ (Tan et al., 2018, Chapter 6). Because class distributions often overlap, I use a soft margin where some training

set data points can be misclassified. According to Bishop (2006), the described hyperplane can be found by solving

$$\min_{\mu,\nu,\xi_i} \frac{||\nu||^2}{2} + C\sum_{i=1}^{N}\xi_i, \tag{8}$$

$$\text{s.t. } y_i\{\mu + \nu'\varphi(x_i)\} \geq 1 - \xi_i, \tag{9}$$

$$\xi_i \geq 0, \tag{10}$$

where ξ_i is a slack variable for observation i and C is the penalty for observations on the wrong side of the decision boundary. This slack variable is equivalent to the hinge loss function (Hastie et al., 2009, Chapter 12). An important notion here is that SVM uses coding scheme $y_i \in \{-1, 1\}$, where y_i is 1 if the call is a success and -1 otherwise. I standardize the input variables first, such that I can use the heuristics of $C = 3$ (Cortez, 2010). This problem can be solved by rewriting it to the dual problem, using Lagrange multipliers κ_i:

$$\max_{\kappa_i} \sum_{i=1}^{N}\kappa_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\kappa_i\kappa_j y_i y_j\langle\varphi(x_i), \varphi(x_j)\rangle, \tag{11}$$

$$\text{s.t.} \sum_{i=1}^{N}\kappa_i y_i = 0, \tag{12}$$

$$0 \leq \kappa_i \leq C, \tag{13}$$

where $\langle\varphi(x_i), \varphi(x_j)\rangle$ is the inner product between $\varphi(x_i)$ and $\varphi(x_j)$, as described in Tan et al. (2018) Chapter 6.

To overcome the problem of specifying transformation function φ, I use the kernel trick (Tan et al., 2018, Chapter 6). Like Moro et al. (2014), I use a Gaussian kernel with hyperparameter ψ: $\langle\varphi(x), \varphi(x')\rangle = K(x, x') = \exp(-\psi||x - x'||^2)$. The dual problem can be solved efficiently with sequential minimal optimization (SMO), as described by Platt (1998). Solving the dual problem gives the estimated hyperplane (Hastie et al., 2009, Chapter 12)

$$\hat{h}(x) = \sum_{i=1}^{N}\hat{\kappa}_i y_i K(x_i, x) + \hat{\mu}. \tag{14}$$

With just this function, it is not possible to calculate probabilities. Platt (1999) proposes a transformation of $\hat{h}(x)$ with the sigmoid function, such that the probability of success becomes

$$\Pr(y_i = 1|x_i) = \frac{1}{1 + \exp\{A\hat{h}(x_i) + B\}}, \tag{15}$$

where A en B are parameters that can to be estimated using a regularized maximum likelihood problem, as described by Lin et al. (2007).

To determine hyperparameter ψ, I first divide the training set into an internal training and validation set for 20 distinct runs. For every run, the internal training set consists of $\frac{2}{3}$ of the observations and the validation set of the other $\frac{1}{3}$. Subsequently, I train the model on the internal training set for every $\psi \in 2^k : \ k \in \{-15, -11.4, -7.8, -4.2, -0.6, 3\}$, using the same holdout scheme for every ψ. I subsequently calculate the mean AUC of the validation sets and select the hyperparameter which results in the highest mean AUC.

Hastie et al. (2009) state that SVM performs well in real classification problems and that the SVM can be seen as an L2 norm regularizer of the hinge loss function, which prevents overfitting. Another advantage of SVM is that it can be formulated as a convex optimization problem when using the hinge loss function, allowing SMO to find a global optimum (Tan et al., 2018, Chapter 6). The DT and NN methods employ a greedy strategy and often find a local optimum. A major disadvantage of SVM is the large computation time when training the model (Tan et al., 2018, Chapter 6). The model is also difficult to interpret due to its complexity.

4.3.4 Neural Network

The neural network (NN) method I use is a multilayer perceptron or feedforward neural network and has the same architecture as Moro et al. (2014). The architecture of this NN consists of an input, a hidden and an output layer, as described in Hastie et al. (2009). Each of these layers contains nodes. The input layer nodes consist of the standardized input variables. There is one output node, which represents the probability of success. In between are H hidden nodes, which can be seen as latent variables, enabling to build complex variables that can help to distinguish classes (Tan et al., 2018, Chapter 6).

Each node has an activation value, which consists of a linear combination of all outputs of the preceding nodes transformed by an activation function. As in Moro et al. (2014), I use the logistic function as activation function. Using this function, the activation value of a node t in layer l is calculated as follows:

$$a_t^l = \sigma\left(\sum_{s=1}^{S} v_{ts}^l a_s^{l-1} + b_t^l\right), \tag{16}$$

where $\sigma(x)$ is the logistic or sigmoid function, $s = 1, \ldots, S$ are the nodes of the previous layer and v^l, b^l are respectively weight and bias parameters for layer l (Bishop, 2006, Chapter 5). The nodes in the first layer are equal to the input variables, thus $a_t^1 = x_t$. Given that our NN design has one hidden layer and H hidden nodes, I can combine the different layers such that I obtain one function for calculating the probability of success:

$$\Pr(y_i = 1|x_i) = \sigma\left\{\sum_{t=1}^{H} v_{1t}^2 \sigma\left(\sum_{s=1}^{O} v_{ts}^1 x_{i,s} + b_t^1\right) + b_1^2\right\}, \tag{17}$$

where $x_{i,s}$ is the s^{th} column of the design matrix for call i and O is equal to the number of columns in the design matrix. To train this model, I optimize the conditional maximum likelihood function. This is the same function as in (4), but with the probabilities as defined in (17). The parameters v, b are updated 100 times using the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm, described by Nocedal & Wright (2006) in Chapter 6. Derivatives are calculated using the backpropagation technique (Werbos & John, 1974).

To determine hyperparameter H, I use the same grid search approach as ψ for SVM. I search under ranges $H \in \{0, 2, 4, 6, 8, 10, 12\}$.

Multi-layer NNs with hidden layers are universal approximators, making them expressive and able to make complex decision boundaries (Tan et al., 2018, Chapter 6; Hastie et al., 2009, Chapter 11). Therefore, NNs can be seen as a solution to the non-flexibility of the LR method. Learning a NN model, on the other hand, can be difficult because the problem is not convex, which means one can get stuck at a local minimum. I partly solve this problem by training an ensemble of seven models with different starting values and give as output the average of these seven models, as proposed in Hastie et al. (2009) in Chapter 11. Another disadvantage is that the model is difficult to interpret due to its complexity.

4.3.5 Decision Tree

A decision tree (DT) consists of nodes and branches and is a hierarchical structure of questions and possible answers. A DT starts at a root node, where the first test takes place. Subsequently, one follows the branch associated with the outcome of the test and ends up at an internal node or a terminal node. At an internal node, another test takes place. A terminal node determines the predicted (probability of a particular) class.

To build the DT I use the Classification and regression trees (CART) algorithm, developed by Breiman et al. (1984). This algorithm generates binary DTs. The splitting is based on an impurity measure called the Gini index:

$$\text{Gini index} = 1 - \sum_c \{p_c(h)\}^2, \tag{18}$$

where $p_c(t)$ is the frequency of observations of class $c = 0, 1$ at a node h (Tan et al., 2018, Chapter 3). A lower impurity of the child nodes weighted by the number of observations denotes a better split. For each node, the best split among all variables has to be determined. When the selected split does not improve the parent node's impurity enough, it becomes a terminal node. I set this complexity parameter equal to 0.01. There are also other stopping conditions to prevent overfitting. For example, there must be at least 20 observations in a node to be able to split and at least 7 in a terminal node for the split to continue. Furthermore, the maximum depth of the tree is 30 and no splitting is done if all observations in a node have the same predictors. For more details of the CART algorithm, I refer to Breiman et al. (1984).

The biggest advantage of a DT is its easy interpretation. One of the disadvantages is that DTs are often inaccurate because they overfit on the training data (Tan et al., 2018, Chapter 6). In Moro et al. (2014) the DT model has one of the worst performances.

4.3.6 XGBoost

One way to overcome the problems of the DT is to build an ensemble of DTs. The main idea is to combine the results of an ensemble of weak learners, which often results in a better performance than individual learners (Tan et al., 2018, Chapter 6). Using an ensemble of size K, one can combine the output $f_k(x_i)$ of individual trees (Chen & Guestrin, 2016):

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), \tag{19}$$

$$f_k(x_i) = w_{q(x_i)},\ w \in \mathbb{R}^T,\ q : \mathbb{R}^G \to \{1, 2, \ldots, T\}, \tag{20}$$

where q is the structure of a CART tree (Breiman et al., 1984), T is the number of leaf nodes, and w is a vector of leaf weights. The combined output is used to calculate the probability of success in the following way:

$$\Pr(y_i = 1|x_i) = \frac{e^{\hat{y}_i}}{1 + e^{\hat{y}_i}}. \tag{21}$$

There are several ways to build the ensemble of trees, such as bagging (Breiman, 1996), gradient boosting (Friedman, 2001), and random forest (Breiman, 2001). Boosting and random forest generally outperform bagging methods (Hastie et al., 2009, Chapter 15). A very promising gradient boosting technique is extreme gradient boosting or XGBoost (XGB) (Chen & Guestrin, 2016). XGB has won many classification competitions and performs well on a wide range of problems (Chen & Guestrin, 2016). Therefore, I use an implementation of this method to built the tree ensemble.

I build tree ensemble $\tau = (f_1, f_2, \ldots, f_K)$ by minimizing the following objective function, which consists of a loss and regularization part:

$$\min_{\tau}\ L(\tau) = \sum_{i=1}^{N} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k), \tag{22}$$

$$\Omega(f_k) = \gamma T + \frac{1}{2}\delta \sum_{j=1}^{T} w_j^2 + \omega \sum_{j=1}^{T} |w_j|, \tag{23}$$

where loss function $l(y_i, \hat{y}_i)$ is the same as in (4), but with probabilities calculated as in (21). Regularization part $\Omega(f_k)$ consists of several regularization terms that prevents overfitting: γ penalizes for the number of leaves and δ for the L2 norm of the weights of the leaf nodes. In contrast to Chen & Guestrin (2016), I also add a L1 regulation term on the leaf weights, which can set some leaf weights to zero.

Equation (22) contains functions as parameters and is therefore difficult to train. Instead of solving (22) directly, boosting trains trees in an additive manner. Each boosting iteration trains one tree f_k. When I write the predictive value at boosting iteration k as $\hat{y}_i^k$, then

$$\hat{y}_i^k = \hat{y}_i^{k-1} + f_k(x_i), \text{ for } k = 1, 2, \ldots, K, \tag{24}$$

with $\hat{y}_i^0 = 0$. Eventually, trees are grown similarly as the DT, using a different measure to determine the best split. Again, the trees only have binary splits and only if the split improves the objective function the split is made. In Appendix A I derive the exact splitting condition for objective function (22), following the same derivation steps as Chen & Guestrin (2016).

Chen & Guestrin (2016) use more regularization techniques to prevent overfitting, which I also implement. Shrinkage, introduced by Friedman (2002), reduces the contribution of each tree. After a tree is learned, one updates $\hat{y}_i^k$ by $\eta f_k(x_i)$, where η is a value between zero and one so that the influence of individual trees is reduced (Friedman, 2002). Other techniques are column sampling, row subsampling, maximum tree depth and allowing partitions to happen only if the weight of every child is at least a certain number.

Because of the many hyperparameters, XGB is difficult to tune. The grid search approach I use for NN and SVM results in too many function evaluations. A different approach is sequential model-based or Bayesian optimization (SMBO). Xia et al. (2017) use SMBO for tuning hyperparameters of XGB and finds in a classification application that it outperforms a grid search strategy. With SMBO I try to find optimal hyperparameters θ^* that maximize the AUC over $R = 20$ different holdout schemes:

$$\theta^* = \arg\min_{\theta} \; y(\theta) = \sum_{r=1}^{R} AUC_r(\theta), \tag{25}$$

where $AUC_r(\theta)$ is the AUC value of run r using hyperparameters θ. Hutter et al. (2011) describe the general procedure for SMBO. First one creates an initial design with F points: (θ_f, y_f) for $f = 1, \ldots, F$. Subsequently, one fits an approximate or surrogate function on the initial points. This surrogate function can be used to find a promising new point, which is added to the design.

There are different implementations of SMBO, using different ways of creating an initial design, creating surrogate functions and calculating promising points. I use the implementation of Bischl et al. (2017) because they find their implementation is superior to several other implementations in a wide range of applications. Their initial design is a Latin hypercube design (McKay et al., 1979) that maximizes the minimum distance between points. To build the surrogate functions, they use kriging or Gaussian process regression (Jones et al., 1998). The implementation of Bischl et al. (2017) proposes new points using the expected improvement. I use an initial design with size 100 and perform 100 iterations of the SMBO algorithm. For more details about the SMBO procedure, I refer to Bischl et al. (2017). Table 3 shows the hyperparameter ranges under which I search. These ranges are similar to the values in Xia et al. (2017).

Table 3: Hyperparameter range search values for XGB.

Hyperparameter	Values
K	$\{10, 20, 50, 100, 200, 500, 1000\}$
η	$[0.001, 0.20]$
γ	$[0, 10]$
max_depth	$\{1, 2, \ldots, 12\}$
min_child_weight	$\{0, 1, 2, 3, 4\}$
$subsample$	$[0.8, 1]$
$colsample\_by\_tree$	$[0.3, 1]$
δ	$[0, 10]$
ω	$[0, 5]$

4.4 Class Imbalance

As mentioned in Section 3, the dataset is imbalanced, which results in poor performance of many classification methods (Tan et al., 2018, Chapter 6). To solve this problem, I use class imbalance sampling techniques. These methods change the distribution of the training set, such that the classification methods are better able to identify the minority instances.

The first class imbalance method I use is synthetic minority oversampling technique or SMOTE, proposed by Chawla et al. (2002). For every trainset, I determine the number of positive and negative instances. Subsequently, I create $SM = [\frac{F}{P}] - 1$ new synthetic instances for every minority class instance in the training set, where P is the number of positive instances, F the number of negative instances and $[x]$ is the round function. In this way, the number of positive and negative instances is approximately equal.

I create a new synthetic instance for a minority class instance by first calculating the k nearest minority instance neighbours. To do this, I need the distance between every minority instance. I first rescale all numeric variables to a $[0, 1]$ scale and calculate the squared differences between the rescaled numeric variables. Chawla et al. (2002) propose to handle categorical values as follows: add the median of the standard deviation of all numerical variables for every different categorical variable. Finally, I take the square root, such that I obtain a Euclidean like distance. To summarize, if I let Mi be the set of training minority instances, Nu the set of numeric variables and Ca the set of categorical variables:

$$d(x_i, x_j) = \sqrt{\sum_{g \in Nu} \left(\frac{x_{i,g} - \min_k x_{k,g}}{\max_k x_{k,g} - \min_k x_{k,g}} - \frac{x_{j,g} - \min_k x_{k,g}}{\max_k x_{k,g} - \min_k x_{k,g}} \right)^2 + \sum_{g \in Ca} \{1(x_{i,g} \neq x_{j,g}) MSD\}^2}, \quad (26)$$

where $1()$ is the indicator function, $k \in Mi$ is the index over all minority instances in the training set, and MSD is the median of the standard deviations of all rescaled numeric variables over the minority class.

After determining the k nearest neighbours, I can calculate new instances for every point by drawing with replacement SM of the nearest neighbours. For every drawing, I create one synthetic instance. Every numerical variable value of this new instance is set by drawing a random number from a uniform distribution between the original point and the selected nearest neighbour. For every categorical variable, the level is set to the level of the original point or the nearest neighbour, each with a probability of 0.5.

As there is no current implementation in R of SMOTE that can handle categorical variables, I modified the source code of the IRIC package by Zhu et al. (2019).

Tomek links (TL) can be used as a data cleaning method and tries to remove noisy points (Tomek, 1976). According to Tomek (1976), two instances x_i and x_j form a Tomek link (x_i, x_j) if they are of different classes and there is no instance x_k, such that $d(x_i, x_k) < d(x_i, x_j)$ or $d(x_j, x_k) < d(x_i, x_j)$. I use the same distance measure as with SMOTE, with the only differences that I calculate $\min_k x_{k,l}$ and $\min_k x_{k,l}$ over all training instances and I compute MSD using all instances. This is because instances of both classes are part of a Tomek link. By removing the Tomek links, the overlap between classes is cleaned up, which can improve the classification performance (Batista et al., 2004).

Batista et al. (2004) compare different oversampling and undersampling techniques. They find that a combination of SMOTE and TL performs well in datasets with a small number of positive examples. This technique creates first synthetic minority instances using SMOTE and subsequently removes the Tomek links. I use this hybrid approach and call it SMOTETL.

4.5 Variable Selection

There are several reasons for applying variable selection: it makes the model easier to interpret, reduces the dimensionality, and it can improve the predictive performance by removing redundant or noise

variables (Hastie et al., 2009, Chapter 3). I use two variable selection techniques: genetic algorithm and group lasso with ridge regularization.

4.5.1 Genetic Algorithm

Variable selection has the goal of improving the performance of a model. Variable selection can thus be seen as a combinatorial optimization problem, where one maximizes a performance measure. The input of this problem is a binary vector representing the selected variables.

In this case, I maximize the AUC because it is invariant to class imbalance and classification costs (Tan et al., 2018, Chapter 6). I train a NN model on a random $\frac{2}{3}$ of the trainset and calculate the AUC on the other $\frac{1}{3}$. I do this for $R = 20$ distinct runs, using the same holdout scheme for every input. I choose the NN method because it is a universal approximator of any target function and because it is able to build complex variables from the input variables (Tan et al., 2018, Chapter 6). Furthermore, to lower the computation effort, I select only one value for the number of hidden nodes H, namely $[\frac{V}{2}]$, where V is the number of selected variables. This heuristic for the number of hidden layers tends to perform well in classification (Cortez, 2010). In summary, I try to solve the following problem:

$$\max_z \; g(z) = \sum_{r=1}^{R} AUC_r(z), \tag{27}$$

$$\text{s.t. } z \in \mathbb{B}^G, \tag{28}$$

where $AUC_r(z)$ is the AUC value of the r-th run using a NN model with $[\frac{1}{2}\sum_{g=1}^{G} z_g]$ hidden layers using variables represented by a binary vector z.

One way to solve this problem is evaluating $f(z)$ every possible value of z. The problem with this approach is that it is too time-consuming, as it involves training a NN model $2^{20}R$ times. As the objective function is discontinuous and thus nondifferential, it is impossible to use continuous optimization techniques. Furthermore, there may be several local optima. According to Liu & Motoda (1998), genetic algorithms can be an effective technique for these kinds of problems, especially in the context of variable selection.

I use the genetic algorithm described by Goldberg (1989) in Chapter 1. The parameter settings are the same as in Scrucca (2013). The algorithm starts by creating a random population of vectors z and calculating the corresponding $g(z)$. In the next generation, a population with higher values of $g(z)$ is wanted. To achieve this, there is first a selection procedure that draws with replacement a sample of parents from the population, such that better fits have a greater chance to be selected. To calculate the probability of selection, I use the linear rank approach (Blickle & Thiele, 1996) because this procedure performs well if the objective function values are roughly the same among the population (Blickle & Thiele, 1996), which is the case in this application. After selection, crossover and mutation take place. Crossover swaps some elements between two parents, such that the characteristics of better fits are mixed in the next generation. Every parent belongs to one pair of parents, and the probability of mixing is 0.8. Mutation changes randomly one element of a single parent and makes the algorithm more robust (Goldberg, 1989, Chapter 1). For every parent, the probability of a mutation is 0.1. The new population is created by adding the best solution of the last iteration to the modified parents. I repeat this procedure 50 times, and I use an initial population size of 25. In Appendix B one can find the specific implementation of the algorithm.

4.5.2 Group Lasso with Ridge Regularization

In Section 4.3.2 I noted that for every variable the coefficient vector either consists of all zeros or nonzeros. Thus, it has the attractive property that it selects at a group level, such that it selects

only whole categorical variables (Meier et al., 2008). Regular lasso (Tibshirani, 1996) can only select columns of the design matrix, which makes it inadequate to handle categorical variables that are dummy coded. According to Bühlmann & Van De Geer (2011), the group lasso also performs well in variable selection.

4.6 Sensitivity Analysis and Rule Extraction

As in Moro et al. (2014), I use rule extraction and sensitivity analysis to obtain more intuition about complex models. According to Martens & Provost (2014), opening black-box models that use difficult to interpret methods such as SVM and NN helps to explain to humans how these models make decisions. It can also help to make the models more accepted by managers (Martens & Provost, 2014). I apply rule extraction by learning a DT on predictions of the best performing combination of methods (Setiono, 2003), using the same procedure as in Section 4.3.5, but with a smaller complexity parameter. This complexity parameter is set equal to 0.001, such that there is a low error in describing the complex black-box model. The predictions of all our DM methods are probabilities. Thus before fitting the DT, I first round the probabilities to 1 if $P(y_i = 1|x_i) \geq 0.5$ and to 0 otherwise.

Sensitivity analysis can be applied by varying the input of a variable over its domain and examining the model responses. Cortez & Embrechts (2013) describe several sensitivity analysis techniques based on the data-based sensitivity analysis (DSA) algorithm. I use several of these techniques. I calculate the input relevance and visually show which variables are most important with an importance bar plot. I also calculate the average impact of a variable and show this by creating a variable effect characteristic (VEC) curve.

5 Results

I this section, I discuss the results of the used methods. First, I look at which variables and hyperparameters are selected. I subsequently evaluate the performance of the methods on the training and test set. Finally, I apply sensitivity analysis and rule extraction. I use R to obtain the results.

5.1 Variable and Hyperparameter Selection

One can find the selected variables of GA and GLA in Appendix C. Figure 3 shows the mean, median and maximum objective value of the population at every iteration of GA. The first 11 iterations the algorithm finds better solutions, but after iteration 11 there is no further improvement. The mean AUC value has a few drops. These drops happen when mutation results in the discarding of *duration* by one of the population members. GA selects 13 of the 20 variables, of which most are economic variables or variables about the call itself. GLA selects more variables than GA and also selects variables about the called client. *housing* is the only variable that is not selected by both selection methods. Thus, this variable is not important for modelling the success of bank telemarketing. There are also 10 variables both techniques select. Two examples are *duration* and *emp.var.rate*, of which I had seen in the data section that these are important variables.

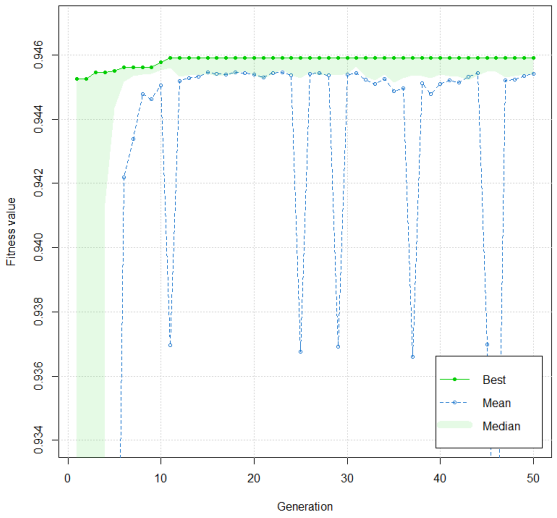


Figure 3: Maximum, mean and median AUC value of the population for every iteration of the genetic algoritm.

One can find the selected hyperparameters of the GLA, SVM, NN and XGB DM methods in Appendix D.

5.2 Training Set Performance

Table 4 reports the performance measure values of the methods on the training set. The best results are often from the XGB DM method. The NN method, which was the best performing method in Moro et al. (2014), does also perform well and is the best method in case GLA combined with TL is used. As in Moro et al. (2014), the DT is the worst performing method in both the AUC and ALIFT.

The class imbalance techniques have a different impact on the performance of different DM methods. The performance of the DT method is influenced the most. Applying SMOTE and SMOTETL increases the AUC and ALIFT value approximately 4 and 5 percentage points, respectively. On the other hand, applying TL worsens the performance of the DT. Another observation from Table 4 is that the GLA DM method also benefits from SMOTE and SMOTETL, although this improvement is relatively small compared to the improvement of the DT. The performance of the other DM methods is not affected by the class imbalance techniques.

The variable selection methods also have a different impact on different DM methods. As with the class imbalance techniques, variable selection affects the DT the most and has a positive influence on the performance. On the other hand, the DT is less affected by variable selection methods than class imbalance methods. For the other DM methods, the variable selection methods do not affect the performance strongly. Thus, simpler models with fewer variables can achieve the same performance. Overall, the GA selection method results in a slightly better performance than the GLA selection.

Table 4: AUC and ALIFT values of the methods on the training set. DM methods that are significantly better than other DM methods using the same combination of preprocessing techniques at 95 per cent confidence level in bold.

		NO						GA						GLA					
		LR	DT	SVM	NN	GLA	XGB	LR	DT	SVM	NN	GLA	XGB	LR	DT	SVM	NN	GLA	XGB
NO	AUC	0.933	0.856	0.937	0.943	0.933	**0.948**	0.933	0.859	0.933	0.946	0.932	**0.949**	0.933	0.855	0.937	0.945	0.933	0.946
	ALIFT	0.890	0.777	0.893	0.899	0.889	**0.902**	0.889	0.781	0.889	0.901	0.888	**0.904**	0.890	0.776	0.893	0.900	0.889	0.900
SMOTE	AUC	0.935	0.883	0.937	0.944	0.936	**0.948**	0.934	0.898	0.938	0.944	0.935	**0.948**	0.933	0.893	0.937	**0.945**	0.936	0.943
	ALIFT	0.891	0.828	0.893	0.899	0.892	**0.902**	0.890	0.846	0.893	0.899	0.891	**0.901**	0.889	0.840	0.893	**0.900**	0.891	0.898
TL	AUC	0.932	0.841	0.935	0.942	0.931	**0.948**	0.932	0.847	0.936	0.944	0.932	**0.947**	0.932	0.845	0.935	0.942	0.931	**0.947**
	ALIFT	0.889	0.761	0.891	0.898	0.888	**0.903**	0.888	0.770	0.892	0.898	0.888	**0.902**	0.888	0.763	0.891	0.897	0.887	**0.901**
SMOTE TL	AUC	0.933	0.882	0.937	0.943	0.935	**0.947**	0.934	0.896	0.938	0.943	0.935	**0.947**	0.933	0.888	0.937	0.944	0.936	0.944
	ALIFT	0.889	0.827	0.893	0.898	0.891	**0.901**	0.890	0.844	0.894	0.898	0.891	**0.901**	0.889	0.834	0.892	0.899	0.892	0.899

5.3 Test Set Performance

Table 5 shows the performance measure values of the methods on the test set. The AUC and ALIFT values are lower for the test set than for the training set. This can be explained by the fact that variable and hyperparameter selection are tuned for the training set. Furthermore, the data characteristics that explain the success of bank telemarketing can change over time. The ALIFT values are also lower because there are relatively more positive observations in the test set than in the train set.

As with the training set, the XGB method has overall the highest performance measure values. The AUC and ALIFT are on average 0.5 and 0.2 percentage point higher than the second-best NN method. GLA as a DM method outperforms the non-regularized LR, with an average increase of 4 and 2 percentage points for the AUC an ALIFT measure respectively. The NN method, which uses a more flexible function to calculate probabilities, improves the performance even more.

From Table 5, it can be seen that the class imbalance techniques do not increase the AUC and ALIFT values for all DM methods. The DT and NN methods benefit from TL, but the SMOTE

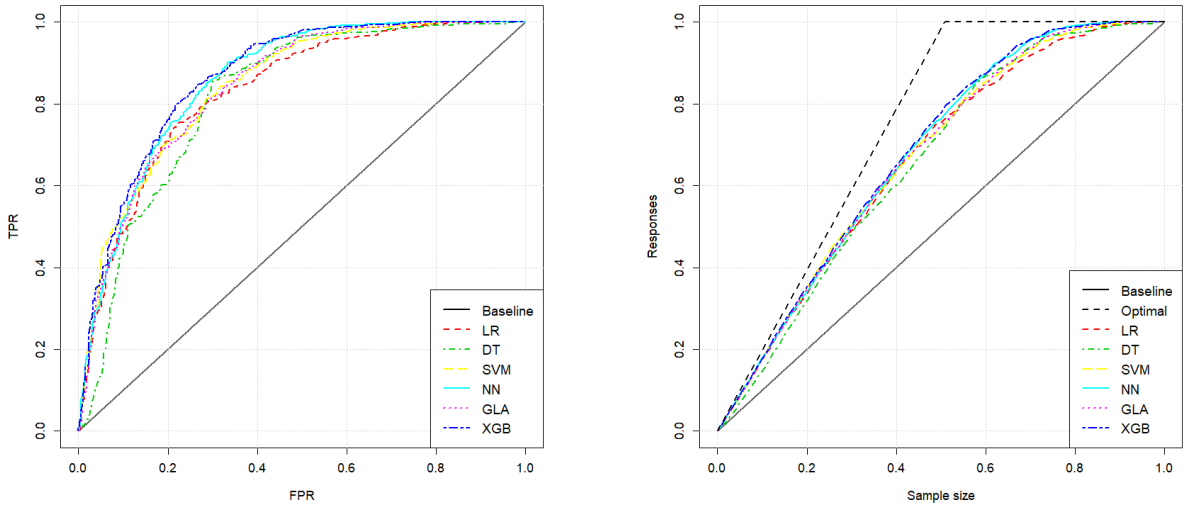
Table 5: AUC and ALIFT values of the methods on the test set. Highest values per combination of preprocessing techniques in bold.

		NO						GA						GLA					
		LR	DT	SVM	NN	GLA	XGB	LR	DT	SVM	NN	GLA	XGB	LR	DT	SVM	NN	GLA	XGB
NO	AUC	0.780	0.818	0.847	0.858	0.843	**0.866**	0.825	0.818	0.841	0.858	0.846	**0.864**	0.831	0.818	0.844	0.858	0.843	**0.867**
	ALIFT	0.638	0.656	0.671	0.676	0.669	**0.680**	0.660	0.656	0.668	0.676	0.670	**0.679**	0.663	0.656	0.669	0.676	0.669	**0.680**
SMOTE	AUC	0.783	0.754	0.843	0.853	0.842	**0.862**	0.820	0.746	0.853	0.855	0.846	**0.859**	0.815	0.763	0.845	0.854	0.842	**0.864**
	ALIFT	0.640	0.625	0.669	0.674	0.668	**0.678**	0.658	0.621	0.673	0.675	0.671	**0.676**	0.655	0.629	0.670	0.674	0.669	**0.679**
TL	AUC	0.788	0.823	0.841	0.861	0.838	**0.866**	0.814	0.823	0.842	0.859	0.843	**0.864**	0.830	0.822	0.842	0.860	0.839	**0.863**
	ALIFT	0.642	0.659	0.668	0.678	0.666	**0.680**	0.654	0.659	0.668	0.677	0.669	**0.679**	0.662	0.658	0.668	0.677	0.667	**0.679**
SMOTE TL	AUC	0.782	0.779	0.842	**0.863**	0.842	0.854	0.807	0.746	0.853	**0.855**	0.847	0.854	0.803	0.776	0.844	0.852	0.842	**0.864**
	ALIFT	0.639	0.637	0.668	**0.679**	0.668	0.674	0.651	0.621	0.674	**0.675**	0.671	0.674	0.649	0.636	0.670	0.673	0.668	**0.679**

and SMOTETL techniques worsen the performance of these methods. This is in contrast to the performance of the DT in the training set, where the SMOTE and SMOTETL techniques outperform the other class imbalance methods. An explanation for this behaviour is that the DT overfits on the training set. The method determines the decision rules too much on the specific characteristics of the artificially created successes, instead of taking out the general characteristics. The DT method is thus more affected by the changing characteristics over time. Another interesting observation is that class imbalance improves the results for the LR method when there is no variable selection, but it worsens it if there is variable selection. The SVM method only improves using the SMOTE and SMOTETL method if GA variable selection is applied. Other methods are slightly worse off from applying class imbalance techniques.

LR benefits most from variable selection, improving the AUC and ALIFT on average by 3 and 2 percentage points when applying GA and 4 and 2 percentage points when applying GLA. The GLA DM method benefits from GA variable selection, while the performance is similar to no variable selection if GLA variable selection is used. Overall, it depends on the DM and class imbalance technique which variable selection method performs best. GA and GLA do not consistently outperform each other.

Overall, the class imbalance and variable selection methods affect the DM methods with no or



(a) ROC curves

(b) Lift cumulative curves

Figure 4: ROC and lift cumulative curve of the six DM methods that use variable selection from GLA and no class imbalance technique. Both graphs also include a baseline curve. The lift cumulative curve also contains an optimal line that describes a situation where all success are assigned a higher probability than the non-successes.

little regularization the most.

The best performing combination of methods uses XGB with the variables determined by GLA, while not applying any class imbalance technique. The NN model without preprocessing techniques is the most similar method to the best method used by Moro et al. (2014). Thus, the best performing method increases the AUC by 0.9 percentage point and the ALIFT by 0.4. Figure 4 displays the ROC and lift cumulative curve of all DM methods using the variable selection of GLA and no class imbalance technique.

It can be seen from Figure 4a that the ROC curve of XGB dominates all others for most FPR values. Between FPR values 0.58 and 0.75, the ROC curve of NN dominates that of XGB. The SVM ROC curve dominates the XGB ROC curve for FPR values between 0.05 and 0.08. Therefore, one cannot say that the XGB DM method is better than SVM and NN. The ROC curves of GLA, LR and DT are always dominated by the curve of XGB, such that one can conclude that XGB outperforms these methods.

The ROC curve can also help to determine which method to choose for specific situations. If one wants to make as little unnecessary calls as possible while capturing a relatively large number of successes, then a method that performs well with low FPR values should be chosen. On the other hand, if one wants to capture all successful calls even though many calls are unnecessary, a method that has high TPR values for high FPR values should be chosen. For example, the SVM method performs well for low FPR values but does not perform well for high FPR values. SVM is thus an efficient method if only a small part of the people can be called, but it is not an appropriate method to capture all successes. The XGB method can be used for both applications as it has both high TPR values for low and high FPR values.

Figure 4b depicts the lift cumulative curves for the DM methods. The optimal line describes a situation where a DM method orders the observations in such a way that all success are assigned a higher probability than the non-successes. The XGB method curve dominates the other curves for most sample size values. The SVM line dominates the XGB line for sample size values between 0.22 en 0.32, but for higher sample sizes SVM is one of the worst-performing methods. This is in correspondence with our earlier observation that SVM is efficient in capturing the most obvious successes, but is not an appropriate method for capturing almost all successes.

The lift cumulative curve also visualizes an important application of DM methods: target selection. If a bank is only able to call 50% of the clients, it can capture 77.1% of the successes using the XGB method. In this situation, one can thus capture 27.1 percentage points more than by randomly choosing 50% of the clients to call.

5.4 Sensitivity Analysis and Rule Extraction

I now apply sensitivity analysis and rule extraction on the best performing combination of methods of the test set, which is the XGB method using the GLA variable selection and no class imbalance technique.

Figure 5 portrays the relative input importance of the variables. The most important variable is *duration*, with an importance of 45%. This in accordance with my findings in the data section that a call with a short duration is less likely to be a success. A short duration corresponds to people hanging up because they are not interested in the offer. Many economic variables are also important. The number of employed people, the employment variation rate and the consumer confidence index have a high importance. It can also be seen that *pdays*, the number of days since the last contact of the previous campaign, is important in explaining the success of bank telemarketing.

The fitted DT on the outputs of the best combination of methods has a mean absolute error of 0.013, such that the DT describes the outputs of the best method well. The corresponding tree has a depth of 13. Figure 6 shows the first six levels of this tree.

This tree again shows the importance of *duration*. This variable determines not only the split of the root node but also in nodes at lower decision levels. While *cons.price.idx* is not one of the most important variables overall, if the duration of the call is longer than 835 seconds the next decision rule is based on this variable. From this, it can be concluded that the consumer price index is an important variable for relatively long calls.

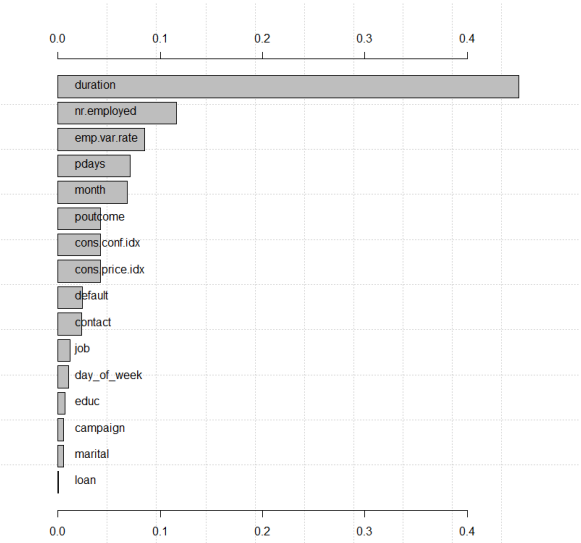


Figure 5: Relative input variable importance fraction for the XGB method using GLA variable selection and no class imbalance technique.

Figure 7 displays two plots with the VEC curves of several important variables. Again, one can see the high influence *duration* has on the probability of success. If the duration of the call is zero, the probability of a call is zero. This corresponds to cases where nobody answers the phone. The probability of success decreases on average when the duration of a call is too long. Long calls correspond to people doubting whether they should subscribe to a bank term deposit. These people want more information and need more time to decide than early convinced clients.

The VEC curves of the two economic variables decrease on average when their values increase. A bank term deposit is thus more likely when the number of people employed is low or when there is a large drop in unemployment. If an economy is in a recession, people act more carefully with their money. People save money for bad times and are thus more likely to subscribe to a bank term deposit. The used data contains observation during the financial crisis of 2008. The probability of success is also dependent on the month the call is made and is on average highest in November.

Telemarketing managers can use the retrieved information from sensitivity analysis and rule extraction by controlling and responding to the variable values. Some variables can be directly controlled, such as the day until the next call. If the highest probability of success is achieved after 12 days, managers can implement a policy that takes this into account. Other variables, such as the economic variables, cannot be modified by the manager, but the manager can still respond to them. For example, the manager knows that the probability of success increases in case of an economic crisis. The manager can act on this by hiring additional agents.

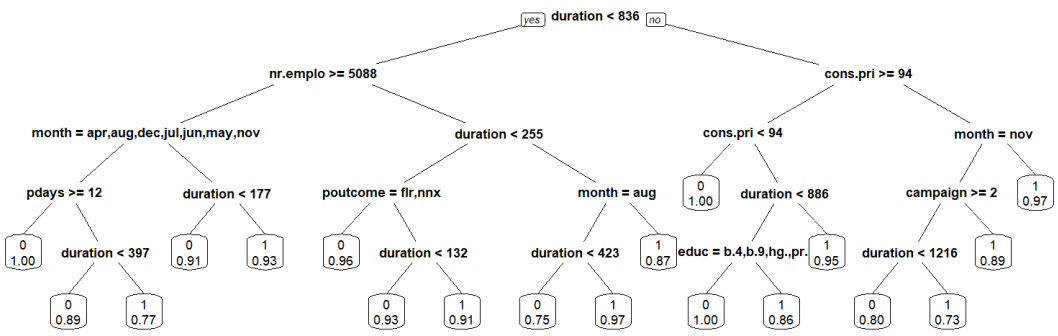


Figure 6: DT fitted on the best combination of methods: XGB with no class imbalance technique and GLA variable selection. Leaf nodes give probability of majority class (0 is failure, 1 is success).

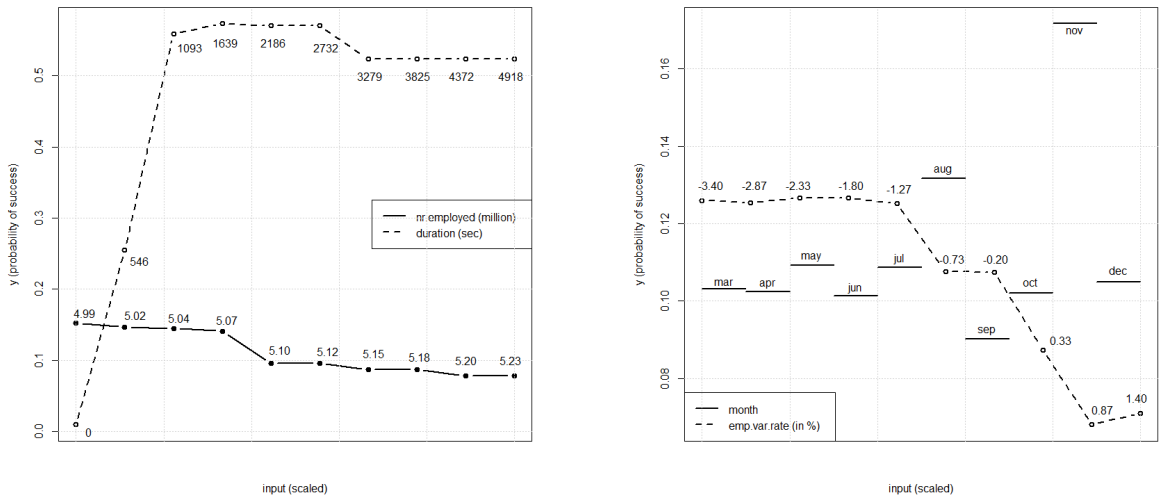


Figure 7: VEC curves showing the global influence of *nr.employed*, *duration*, *month* and *emp.var.rate* on the probability of success.

6 Conclusion and Discussion

6.1 Conclusion

In this paper, I studied the prediction and explanation of the success of bank telemarketing. The research question was *'Which data preprocessing techniques and data mining models perform best in predicting the success of bank marketing? And how can one help managers to understand the best performing model better?'*. To answer these research questions, I used real data on a bank telemarketing campaign from the UCI Machine Learning Repository (Dua & Graff, 2019), which is the closest dataset to data analysed by Moro et al. (2014). To predict the success of bank telemarketing, I used several combinations of DM, variable selection and class imbalance sampling methods. With the help of rule extraction and sensitivity analysis, I analysed the best performing combination of methods.

The DM method based on extreme gradient boosting or XGBoost, in combination with no class imbalance and variable selection via group lasso with ridge regularization was the best performing combination of methods. Furthermore, I determined the hyperparameters of XGBoost by sequential model-based optimization. This combination of methods outperformed the best method of Moro et al. (2014). Thus, our combination of methods enables telemarketing managers to make better predictions of the success of bank telemarketing.

More regularized DM methods perform better than methods with no or little regularization. This was, among other things, shown by applying a regularized version of logistic regression, namely the group lasso with ridge regularization. The neural network, which applies a more flexible function to calculate probabilities, also outperforms the logistic regression.

DM methods with no or little regularization are affected most by the preprocessing methods. After applying variable selection, less regularized methods performed better, while other DM method often got similar results. Most DM methods do not profit from applying class imbalance techniques.

I found that managers can use information from sensitivity analysis and rule extraction to understand the best performing method better. I also described how these managers could use these techniques to create a more profitable strategy.

6.2 Discussion

A shortcoming of this paper is that I include the duration of the call in the DM methods. Because the duration of the call is not known beforehand, my predictive models are not realistic.

Another shortcoming is my implementation of the synthetic minority oversampling technique (SMOTE). I set the oversampling percentage such that the training set consisted of approximately of the same number of positive and negative instances. A better approach would have been to test the performance of different percentages using cross-validation, as done in Chawla et al. (2002).

One can also discuss whether the used performance measure, the area under the receiver operating characteristic curve (AUC), is a valid performance measure. Hand (2009) states that the AUC gives misleading results if the ROC curves cross different models. The plotted ROC curves showed that this is indeed often the case. Furthermore, according to Hand (2009), the AUC uses different scales while comparing the performance of different classifiers. On the other hand, Ferri et al. (2011) do provide a coherent interpretation when optimal thresholds are considered.

For future research, one can evaluate the performance of the described methods also on the dataset of Moro et al. (2018). This dataset not only contains some strong predictors of Moro et al. (2014) that were not available but also customer lifetime variables and domain expert variables. Furthermore, a realistic predictive model can be build by discarding the duration variable and adding these strong predictors.

Since regularization and more flexible functions both improved the performance of the logistic regression, it would be interesting how a combination would perform. One could, for example, apply a neural network with regularization by learning the parameters with a loss function that has an L1 or L2 penalty on the coefficients.

References

Barraza, N., Moro, S., Ferreyra, M., & de la Peña, A. (2019). Mutual information and sensitivity analysis for feature selection in customer targeting: A comparative study. *Journal of Information Science*, *45*(1), 53–67.

Batista, G. E., Prati, R. C., & Monard, M. C. (2004). A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD Explorations Newsletter*, *6*(1), 20–29.

Bischl, B., Richter, J., Bossek, J., Horn, D., Thomas, J., & Lang, M. (2017). mlrMBO: A modular framework for model-based optimization of expensive black-box functions. *arXiv preprint arXiv:1703.03373*.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. New York, NY: Springer.

Blickle, T., & Thiele, L. (1996). A comparison of selection schemes used in evolutionary algorithms. *Evolutionary Computation*, *4*(4), 361–394.

Breheny, P., & Huang, J. (2015). Group descent algorithms for nonconvex penalized linear and logistic regression models with grouped predictors. *Statistics and Computing*, *25*(2), 173–187.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*(2), 123–140.

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

Breiman, L., Friedman, J. H., Stone, C. J., & Olshen, R. A. (1984). *Classification and regression trees*. Boca Raton, FL: CRC Press.

Bühlmann, P., & Van De Geer, S. (2011). *Statistics for high-dimensional data: methods, theory and applications*. Berlin, Heidelberg, Germany: Springer.

Cameron, A. C., & Trivedi, P. K. (2005). *Microeconometrics: methods and applications*. Cambridge, UK: Cambridge University Press.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, *16*, 321–357.

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In B. Krishnapuram, M. Shah, A. J. Smola, C. C. Aggarwal, D. Shen, & R. Rastogi (Eds.), *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (pp. 785–794). San Francisco, CA: Association for Computing Machinery.

Cortez, P. (2010). Data mining with neural networks and support vector machines using the R/rminer tool. In P. Perner (Ed.), *Advances in data mining. Applications and theoretical aspects 10th industrial conference on data mining* (pp. 572–583). Berlin, Heidelberg, Germany: Springer.

Cortez, P., & Embrechts, M. J. (2013). Using sensitivity analysis and visualization techniques to open black box data mining models. *Information Sciences*, *225*, 1–17.

Dua, D., & Graff, C. (2019). *UCI machine learning repository.* Retrieved from `http://archive.ics.uci.edu/ml`

Elsalamony, H. A. (2014). Bank direct marketing analysis of data mining techniques. *International Journal of Computer Applications*, *85*(7), 12–22.

Ferri, C., Hernández-Orallo, J., & Flach, P. A. (2011). A coherent interpretation of AUC as a measure of aggregated classification performance. In T. S. Lise Getoor (Ed.), *Proceedings of the 28th international conference on machine learning* (pp. 657–664). Madison, WI: Omnipress.

Franses, P. H., & Paap, R. (2001). *Quantitative models in marketing research*. Cambridge, UK: Cambridge University Press.

Friedman, J. H. (2001). 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.

Friedman, J. H., Hastie, T., & Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, *33*(1), 1-22.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization and machine learning.* Boston, MA: Addison-Wesley.

Goldberg, D. E., & Holland, J. H. (1988). Genetic algorithms and machine learning. *Machine Learning*, *3*, 95-99.

Hall, M. A., & Smith, L. A. (1999). Feature selection for machine learning: Comparing a correlation-based filter approach to the wrapper. In A. N. Kumar & I. Russell (Eds.), *Proceedings of the 12th international Florida artificial intelligence research society conference* (pp. 235–239). St. Augustine, FL: AAAI Press.

Hand, D. J. (2009). Measuring classifier performance: A coherent alternative to the area under the ROC curve. *Machine Learning*, *77*(1), 103–123.

Hanley, J. A., & McNeil, B. J. (1982). The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, *143*(1), 29–36.

Hastie, T., Tibshirani, R., & Friedman, J. H. (2009). *The elements of statistical learning: Data mining, inference, and prediction.* New York, NY: Springer.

Hutter, F., Hoos, H. H., & Leyton-Brown, K. (2011). Sequential model-based optimization for general algorithm configuration. In C. Coello (Ed.), *Proceedings of the 5th international conference on learning and intelligent optimization* (pp. 507–523). Berlin, Heidelberg, Germany: Springer.

Jones, D. R., Schonlau, M., & Welch, W. J. (1998). Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, *13*(4), 455–492.

Lau, K.-N., Chow, H., & Liu, C. (2004). A database approach to cross selling in the banking industry: Practices, strategies and challenges. *Journal of Database Marketing & Customer Strategy Management*, *11*(3), 216–234.

Lin, H.-T., Lin, C.-J., & Weng, R. C. (2007). A note on platt's probabilistic outputs for support vector machines. *Machine Learning*, *68*(3), 267–276.

Liu, H., & Motoda, H. (1998). *Feature extraction, construction and selection: A data mining perspective.* Boston, MA: Springer.

Liu, X.-Y., Wu, J., & Zhou, Z.-H. (2008). Exploratory undersampling for class-imbalance learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *39*(2), 539–550.

Martens, D., & Provost, F. (2014). Explaining data-driven document classifications. *Mis Quarterly*, *38*(1), 73–100.

Martens, D., Vanthienen, J., Verbeke, W., & Baesens, B. (2011). Performance of classification models from a user perspective. *Decision Support Systems*, *51*(4), 782–793.

McKay, M. D., Beckman, R. J., & Conover, W. J. (1979). Comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, *21*(2), 239–245.

Meier, L., Van De Geer, S., & Bühlmann, P. (2008). The group lasso for logistic regression. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *70*(1), 53–71.

Miguéis, V. L., Camanho, A. S., & Borges, J. (2017). Predicting direct marketing response in banking: Comparison of class imbalance methods. *Service Business*, *11*(4), 831–849.

Moro, S., Cortez, P., & Laureano, R. (2012). Enhancing bank direct marketing through data mining. In *Proceedings of the 41st international conference of the european marketing academy* (pp. 1–8).

Lisbon, Portugal: European Marketing Academy.

Moro, S., Cortez, P., & Rita, P. (2014). A data-driven approach to predict the success of bank telemarketing. *Decision Support Systems*, *62*, 22–31.

Moro, S., Cortez, P., & Rita, P. (2015). Using customer lifetime value and neural networks to improve the prediction of bank deposit subscription in telemarketing campaigns. *Neural Computing and Applications*, *26*(1), 131–139.

Moro, S., Cortez, P., & Rita, P. (2018). A divide-and-conquer strategy using feature relevance and expert knowledge for enhancing a data mining approach to bank telemarketing. *Expert Systems*, *35*(3), e12253.

Nocedal, J., & Wright, S. J. (2006). *Numerical optimization*. New York, NY: Springer.

Platt, J. (1998). *Sequential minimal optimization: A fast algorithm for training support vector machines* (Tech. Rep. No. MSR-TR-98-14). Microsoft Research.

Platt, J. (1999). Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In A. Smola, P. Bartlett, B. Schölkopf, & D. Schuurmans (Eds.), *Advances in large margin classifiers* (pp. 61–74). Cambridge, MA: MIT Press.

R Core Team. (2020). R: A language and environment for statistical computing [Computer software manual]. Vienna, Austria. Retrieved from `https://www.R-project.org/`

Scrucca, L. (2013). GA: A package for genetic algorithms in R. *Journal of Statistical Software*, *53*(4), 1–37.

Setiono, R. (2003). Techniques for extracting classification and regression rules from artificial neural networks. In D. B. Fogel & C. J. Robinson (Eds.), *Computational intelligence: The experts speak* (pp. 99–113). New York, NY: IEEE.

Simon, N., & Tibshirani, R. (2012). Standardization and the group lasso penalty. *Statistica Sinica*, *22*(3), 983–1001.

Su, C.-T., Chen, Y.-H., & Sha, D. (2006). Linking innovative product development with customer knowledge: a data-mining approach. *Technovation*, *26*(7), 784–795.

Tan, P.-N., Steinbach, M., Karpatne, A., & Kumar, V. (2018). *Introduction to data mining*. New York, NY: Pearson.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *58*(1), 267–288.

Tomek, I. (1976). Two modifications of CNN. *IEEE Transactions on Systems, Man, and Cybernetics*, *7(2)*, 679-772.

Vajiramedhin, C., & Suebsing, A. (2014). Feature selection with data balancing for prediction of bank telemarketing. *Applied Mathematical Sciences*, *8*(114), 5667–5672.

Wackerly, D., Mendenhall, W., & Scheaffer, R. L. (2014). *Mathematical statistics with applications*. Belmont, CA: Thomson Brooks/Cole.

Werbos, P., & John, P. (1974). Beyond regression: New tools for prediction and analysis in the behavioral sciences. *PhD thesis, Harvard University*.

Witten, I. H., Frank, E., Hall, M. A., & Pal, C. J. (2016). *Data mining: Practical machine learning tools and techniques*. San Francisco, CA: Morgan Kaufmann.

Xia, Y., Liu, C., Li, Y., & Liu, N. (2017). A boosted decision tree approach using bayesian hyper-parameter optimization for credit scoring. *Expert Systems with Applications*, *78*, 225–241.

Zhu, B., Gao, Z., Zhao, J., & vanden Broucke, S. K. (2019). IRIC: An R library for binary imbalanced classification. *SoftwareX*, *10*, 100341.

A XGBoost Splitting Condition

The derivation of the objective function with L1 regularization follows the same steps as in Chen & Guestrin (2016). To find the best f_k as defined in (24), one wants to optimize the objective function at iteration k:

$$L^k(f_k) = \sum_{i=1}^{N} l\{y_i, \hat{y}_i^{k-1} + f_k(x_i)\} + \Omega(f_k) + \sum_{l=1}^{k-1} \Omega(f_l),$$

Loss function $l\{y_i, \hat{y}_i^{k-1} + f_k(x_i)\}$ does not have a nice form in case of logistic regression, so instead I take the second Taylor approximation as in Chen & Guestrin (2016):

$$L^k(f_k) \simeq \sum_{i=1}^{N} \left\{ l(y_i, \hat{y}_i^{-1}) + g_i f_k(x_i) + \frac{1}{2} h_i f_k^2(x_i) \right\} + \Omega(f_k) + \sum_{l=1}^{k-1} \Omega(f_l),$$

with

$$g_i = \frac{\partial l(y_i, \hat{y}_i^{k-1})}{\partial \hat{y}_i^{k-1}},$$
$$h_i = \frac{\partial^2 l(y_i, \hat{y}_i^{k-1})}{(\partial \hat{y}_i^{k-1})^2}.$$

Because I need to minimize this function, I can leave out all constants, such that I obtain

$$\tilde{L}^k(f_k) = \sum_{i=1}^{N} \left\{ g_i f_k(x_i) + \frac{1}{2} h_i f_k^2(x_i) \right\} + \Omega(f_k),$$

Using the definition of $\Omega(f_k)$ of (23) and that $f(x_i) = w_{q(x_i)}$ according to (20):

$$\begin{aligned}
\tilde{L}^k(f_k) &= \sum_{i=1}^{N} \left\{ g_i f_k(x_i) + \frac{1}{2} h_i f_k^2(x_i) \right\} + \gamma T + \frac{1}{2}\delta \sum_{j=1}^{T} w_j^2 + \omega \sum_{j=1}^{T} |w_j| \\
&= \sum_{i=1}^{N} \left\{ g_i w_{q(x_i)} + \frac{1}{2} h_i w_{q(x_i)}^2(x_i) \right\} + \gamma T + \frac{1}{2}\delta \sum_{j=1}^{T} w_j^2 + \omega \sum_{j=1}^{T} |w_j| \\
&= \sum_{j=1}^{T} \left\{ \left(\sum_{i \in I_j} g_i \right) w_j + \frac{1}{2} \left(\sum_{i \in I_j} h_i \right) w_j^2 \right\} + \gamma T + \frac{1}{2}\delta \sum_{j=1}^{T} w_j^2 + \omega \sum_{j=1}^{T} |w_j|,
\end{aligned}$$

where $I_j = \{i | q(x_i) = j\}$. I now define $G_j = \sum_{i \in I_j} g_i$ and $H_j = \sum_{i \in I_j} h_i$, like Chen & Guestrin (2016). Then

$$\begin{aligned}
\tilde{L}^k(f_k) &= \sum_{j=1}^{T} \left(G_j w_j + \frac{1}{2} H_j w_j^2 \right) + \gamma T + \frac{1}{2}\delta \sum_{j=1}^{T} w_j^2 + \omega \sum_{j=1}^{T} |w_j| \\
&= \sum_{j=1}^{T} \left\{ G_j w_j + \omega |w_j| + \frac{1}{2}(H_j + \delta) w_j^2 \right\} + \gamma T.
\end{aligned}$$

If one now fixes the tree structure q, it is possible to calculate the optimale value of w_j^* by solving

$$0 = \frac{\partial \tilde{L}^k}{\partial w_j} = \begin{cases} G_j + \omega + (H_j + \delta) w_j & \text{if} \quad w_j > 0, \\ G_j - \omega + (H_j + \delta) w_j & \text{if} \quad w_j < 0. \end{cases}$$

Solving for w_j gives

$$w_j^* = \begin{cases} -\frac{G_j + \omega}{H_j + \delta} & \text{if} \quad G_j < -\omega, \\ -\frac{G_j - \omega}{H_j + \delta} & \text{if} \quad G_j > \omega, \\ 0 & \text{if} \quad -\omega \leq G_j \leq \omega. \end{cases}$$

If I now define

$$B(G_j, \omega) = \begin{cases} G_j + \omega & \text{if} \quad G_j < -\omega, \\ G_j - \omega & \text{if} \quad G_j > \omega, \\ 0 & \text{if} \quad -\omega \leq G_j \leq \omega, \end{cases}$$

then

$$w_j^* = \frac{-B(G_j, \omega)}{H_j + \delta}.$$

Filling in w_j^* in the objective function for a fixed tree stucture q gives:

$$\begin{aligned} \tilde{L}^{k*}(q) &= \sum_{j=1}^{T} \left\{ G_j \frac{-B(G_j, \omega)}{H_j + \delta} + \omega \left| \frac{B(G_j, \omega)}{H_j + \delta} \right| + \frac{B(G_j, \omega)^2}{2(H_j + \delta)} \right\} + \gamma T \\ &= \sum_{j=1}^{T} \left\{ \frac{-G_j B(G_j, \omega) + \frac{1}{2} B(G_j, \omega)^2}{H_j + \delta} + \omega \left| \frac{B(G_j, \omega)}{H_j + \delta} \right| \right\} + \gamma T. \end{aligned}$$

So, for every tree structure one can calculate the optimal objective value. As it is computationally hard to look at all possibile tree structures, XGBoost uses a greedy algoritm that uses this objective function to determine the best binary split (Chen & Guestrin, 2016). To simplify notation I first define:

$$S(G_j, H_j, \omega, \delta) = \frac{-G_j B(G_j, \omega) + \frac{1}{2} B(G_j, \omega)^2}{H_j + \delta} + \omega \left| \frac{B(G_j, \omega)}{H_j + \delta} \right|,$$

such that the gain or loss reduction from a split is calculated as

$$gain = S(G_L + G_R, H_L + H_R, \omega, \delta) - S(G_L, H_L, \omega, \delta) - S(G_R, H_R, \omega, \delta) - \gamma,$$

where G_L and G_R are the sum of the first order gradients of the instances at the left and right side respectively, and H_L and H_R are the sum of the second order gradients of the instances at the left and right side respectively.

B Algorithm for the Genetic Algorithm

Algorithm 1: Genetic algorithm for variable selection.

Result: Binary vector z_{best} containing best variable subset found
Set population size N; set number of iterations K; crossover probability c; mutation rate m;
Create random initial population $z^0 = (z_1^0, z_2^0, \ldots, z_N^0)$; calculate $f^0 = \{f(z_1^0), f(z_2^0), \ldots, f(z_N^0)\}$;
$length \leftarrow length(z_1^0)$;
Calculate rank vector $rank$, where $rank_i = j$ if $f(z_i^0)$ is the j-th lowest value of f^0;
Calculate probability vector p^0, where $p_i^0 = \frac{2rank_i}{(N+1)N}$, for $i = 1, \ldots, N$;
$z_{best} = \arg\max_{z \in z^0} f(z)$;
for $k \leftarrow 1$ **to** K **do**
    `// Selection`
    Initialize reproducing population r;
    for $l \leftarrow 1$ **to** $N-1$ **do**
        select one element *parent* from $(z_1^{k-1}, z_2^{k-1}, \ldots, z_N^{k-1})$ using probabilities $(p_1^{k-1}, p_2^{k-1}, \ldots, p_N^{k-1})$;
        $r_l \leftarrow parent$;
    end
    `// Crossover`
    Create set R where every element consists of 2 random elements of r, without repetition, such that $|R| = \lfloor \frac{|r|}{2} \rfloor$;
    for *each* $(r_l, r_m) \in R$ **do**
        Draw $U_1 \sim Unif(0,1)$;
        if $U_1 < c$ **then**
            Draw integer $T_1 = \lceil (length - 1)U_2 \rceil,\ U_2 \sim Unif(0,1)$;
            for $v \leftarrow T_1 + 1$ **to** $length$ **do**
                $r_{l,v} \leftarrow r_{m,v}$;
                $r_{m,v} \leftarrow r_{l,v}$;
            end
    end
    `// Mutation`
    for $l \leftarrow 1$ **to** $N-1$ **do**
        Draw $U_3 \sim Unif(0,1)$;
        if $U_3 < m$ **then**
            Draw integer $T_2 = \lceil lengthU_4 \rceil,\ U_4 \sim Unif(0,1)$;
            for $v \leftarrow 1$ **to** $length$ **do**
                if $v = T_2$ **then**
                    if $r_{l,v} = 1$ **then**
                        $r_{l,v} \leftarrow 0$;
                    else
                        $r_{l,v} \leftarrow 1$;
            end
    end
    `// Update best solution and population`
    $(z_1^k, z_2^k, \ldots, z_N^k) = (z_{best}, r_1, \ldots, r_{N-1})$;
    $f^k = \{f(z_1^k), f(z_2^k), \ldots, f(z_N^k)\}$;
    `// update probabilities`
    Recalculate rank vector $rank$, where $rank_i = j$ if $f(z_i^k)$ is the j-th lowest value of f^k;
    Calculate probability vector p^k, where $p_i^k = \frac{2rank_i}{(N+1)N}$, for $i = 1, \ldots, N$;
    `// Update` z_{best}
    $z_{best} = \arg\max_{z \in z^k} f(z)$;
end

C Variable Selection

Table 6: Selected variables after variable selection. ✓ means the variable is selected.

Variable	Genetic Algorithm	Group Lasso with Ridge
age	✓	
job		✓
marital		✓
education		✓
default	✓	✓
housing		
loan		✓
contact	✓	✓
month	✓	✓
day_of_week		✓
duration	✓	✓
campaign	✓	✓
pdays	✓	✓
previous	✓	
poutcome	✓	✓
emp.var.rate	✓	✓
cons.price.idx		✓
cons.conf.idx	✓	✓
euribor3m	✓	
nr.employed	✓	✓

D Hyperparameters

Selected parameters for GLA as DM method are $\lambda = 0.003947097$ and $\alpha = 0.30$. I use these hyperparameters for every class imbalance and variable selection combination. One can find the hyperparameters of SVM, NN and XGB in Table 7.

Table 7: Hyperparameters of the SVM, NN and XGB method for every class imbalance and variable selection combination.

	NO	SMOTE	TL	SMOTETL
NO	**SVM**	**SVM**	**SVM**	**SVM**
	$\psi = 2^{-7.8}$	$\psi = 2^{-7.8}$	$\psi = 2^{-7.8}$	$\psi = 2^{-7.8}$
	NN	**NN**	**NN**	**NN**
	$H = 8$	$H = 8$	$H = 8$	$H = 8$
	XGB	**XGB**	**XGB**	**XGB**
	$K = 500$	$K = 1000$	$K = 200$	$K = 1000$
	$\eta = 0.0218$	$\eta = 0.0698$	$\eta = 0.0673$	$\eta = 0.0964$
	$\gamma = 4.95$	$\gamma = 4.18$	$\gamma = 1.72$	$\gamma = 3.63$
	$max\_depth = 6$	$max\_depth = 6$	$max\_depth = 6$	$max\_depth = 6$
	$min\_child\_weight = 4$	$min\_child\_weight = 1$	$min\_child\_weight = 3$	$min\_child\_weight = 2$
	$subsample = 0.885$	$subsample = 0.886$	$subsample = 1$	$subsample = 0.916$
	$colsample\_by\_tree = 0.481$	$colsample\_by\_tree = 0.426$	$colsample\_by\_tree = 0.602$	$colsample\_by\_tree = 0.309$
	$\delta = 7.72$	$\delta = 2.55$	$\delta = 9.34$	$\delta = 1.78$
	$\omega = 0.299$	$\omega = 3.85$	$\omega = 4.09$	$\omega = 3.78$
GA	**SVM**	**SVM**	**SVM**	**SVM**
	$\psi = 2^{-7.8}$	$\psi = 2^{-7.8}$	$\psi = 2^{-7.8}$	$\psi = 2^{-7.8}$
	NN	**NN**	**NN**	**NN**
	$H = 8$	$H = 8$	$H = 8$	$H = 6$
	XGB	**XGB**	**XGB**	**XGB**
	$K = 1000$	$K = 200$	$K = 500$	$K = 200$
	$\eta = 0.0108$	$\eta = 0.121$	$\eta = 0.0445$	$\eta = 0.17$
	$\gamma = 0.776$	$\gamma = 4.24$	$\gamma = 2.28$	$\gamma = 3.6$
	$max\_depth = 6$	$max\_depth = 7$	$max\_depth = 5$	$max\_depth = 7$
	$min\_child\_weight = 0$	$min\_child\_weight = 1$	$min\_child\_weight = 1$	$min\_child\_weight = 3$
	$subsample = 0.892$	$subsample = 0.997$	$subsample = 0.918$	$subsample = 0.81$
	$colsample\_by\_tree = 0.6$	$colsample\_by\_tree = 0.452$	$colsample\_by\_tree = 0.417$	$colsample\_by\_tree = 0.353$
	$\delta = 4.52$	$\delta = 1.04$	$\delta = 9.87$	$\delta = 1.73$
	$\omega = 1.74$	$\omega = 4.67$	$\omega = 4.8$	$\omega = 3.58$
GLA	**SVM**	**SVM**	**SVM**	**SVM**
	$\psi = 2^{-7.8}$	$\psi = 2^{-7.8}$	$\psi = 2^{-7.8}$	$\psi = 2^{-7.8}$
	NN	**NN**	**NN**	**NN**
	$H = 8$	$H = 6$	$H = 8$	$H = 6$
	XGB	**XGB**	**XGB**	**XGB**
	$K = 1000$	$K = 500$	$K = 200$	$K = 100$
	$\eta = 0.0122$	$\eta = 0.0219$	$\eta = 0.0523$	$\eta = 0.118$
	$\gamma = 3.49$	$\gamma = 1.84$	$\gamma = 4.9$	$\gamma = 4.21$
	$max\_depth = 7$	$max\_depth = 6$	$max\_depth = 6$	$max\_depth = 7$
	$min\_child\_weight = 1$	$min\_child\_weight = 4$	$min\_child\_weight = 0$	$min\_child\_weight = 3$
	$subsample = 0.812$	$subsample = 0.902$	$subsample = 0.809$	$subsample = 0.814$
	$colsample\_by\_tree = 0.375$	$colsample\_by\_tree = 0.334$	$colsample\_by\_tree = 0.46$	$colsample\_by\_tree = 0.354$
	$\delta = 7.51$	$\delta = 9.98$	$\delta = 8.92$	$\delta = 0.112$
	$\omega = 3.78$	$\omega = 3.31$	$\omega = 1.83$	$\omega = 4.77$