Bank Telemarketing Effectiveness prediction

Masters Project Report submitted in partial fulfilment of the requirements
For the award of the degree of

Master of Science
By

Saikat Kar
Roll No – C91/STS/201023

Under the supervision of
Prof. Sugata Sen Roy
Prof. Manisha Pal

Department of Statistics
University of Calcutta
July 13, 2022

Abstract

Nowadays, marketing spending in the banking industry is massive, meaning that it is essential for banks to optimize marketing strategies and improve effectiveness. Understanding customers' need leads to more effective marketing plans, smarter product designs and greater customer satisfaction. Using dataset from here ; as data pre-processing we have used One-hot encoding, normalizing data and nextly as feature selection stage we used Boruta (Wrapper approach) algorithm. In this thesis, we have used well known machine learning methods Lasso logistic regression, decision trees, Random Forest, Xg-Boost. As this dataset is highly class-imbalanced, so we have used Sensitivity (major), Precision (minor) and AUC of Precision-Recall Curve. Based on these, we have found Xg-Boost model as best performing classification model with Sensitivity 89.44% and AUC 46.05%. Next, from point of view of financial institution like bank they need information like to get as much as possible yes response by targeting small percentage of customers i.e. by effective targeting. That is done by cumulative Gain and Lift Chart. Next, we have analysed characteristics of those top targeted customers in subscribing term-deposit by DALEX.

Acknowledgements

I would like to thank my supervisors, Prof. Sugata Sen Roy and Prof. Manisha Pal, for their guidance and insightful discussions. I am also very grateful to have been a part of the Department of Statistics and learn from amazing professors there.

I want to thank my family for providing me the support and encouragement to pursue this project. The support of my friends at the University of Calcutta has been invaluable and has kept me going through the challenges I was faced with during this work.

1. INTRODUCTION

The bank provides financial services/products such as savings accounts, current accounts, debit cards, etc. to its customers. In order to increase its overall revenue, the bank conducts various marketing campaigns for its financial products such as credit cards, term deposits, loans, etc. These campaigns are intended for the bank's existing customers. However, the marketing campaigns need to be cost-efficient so that the bank not only increases their overall revenues but also the total profit.

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 [1]. The increasingly vast number of marketing campaigns over time has reduced its effect on the general public [2]. According to [1] , 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. [3] 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. Such direct campaigns can be enhanced through the use of Business Intelligence (BI) and Data Mining (DM) techniques.

Research question consists of following questions: *Which data pre-processing 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 the research questions, I use real data on bank telemarketing campaign of a Portuguese bank during the period May 2008 to November 2010 [here]. 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 [4] and is the closest available dataset to the data analysed by [5] .

This thesis extends the research done by [5]. In addition to the DM methods used by [5], we have applied two additional DM methods: extreme gradient boosting or Xg-Boost and Lasso-logistic. We also used two variable selection methods: one using a genetic algorithm and Boruta (Wrapper algorithm).

In Section 2, we discuss relevant literature on the subject of bank telemarketing. Section 3 describes relevant characteristics of the used data. We explained 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 REVIEW

This section explains the previous research work which have been already done in classification using ML techniques. The data which is used in this study work is the data of customers of a Portuguese banking institution. The similar data set has been used in [6]. In [6], the aim of this study was to find the model that can increase the success rate of telemarketing for the bank. The statistical techniques of data mining which have been used in their research are Support Vector Machine (SVM), Decision Tree (DT) and Naive Bayes. The performance of these models was checked through the Receiver Operator Characteristics (ROC) curve (detail of ROC curve is given in section 5). Among all these statistical techniques, SVM comes up with the most efficient results. Regarding attributes, Call duration was the most relevant feature which states that longer calls tend increase the success rate. After that month of contact, number of contacts, days since last contact, last contact result and first contact duration attributes respectively.

In [5], objective of the study was to predict the success of bank telemarketing. Data set which they used in their research was consists of 150 attributes and is complete data set of the period 2008 to 2013. They compare the 4 data mining models i.e. Logistic Regression (LR), Decision Tree, Support Vector Machine and Neural Network (NN). The best result was obtained by the neural network while decision trees discloses that probability of success in inbound calls are greater.

Statistical learning algorithms have successfully been used in many research problems for classification. For example [7] conducted a

research to find out the fault diagnosis system for reciprocating compressors. Reciprocating compressors are extensively used in petroleum industry. Data was taken from Oil Corporation (5 years operational data) and uses the Support Vector Machine to analyse it. They come up with the results that SVM accurately predicts the 80% right classification to find the potential faults in compressor.

Similarly, [8] did a research in medical field for diagnosing the urological dysfunctions using SVM. In this research data was collected from the 381 patients who are suffering from a number of urological dysfunctions to check the overall worth of decision support system. The fivefold cross validation has been utilized for the robustness. The outputs of this study describe that for the purpose of classification SVM attained the accuracy of 84.25%.

[9] utilized the machine learning in decision support system. In their research they introduce the air traffic management for the future which can manage the flight schedule and flow of air traffic professionally. Their system involves many decision makers and utilized it with the neural network. They require such system which can make the optimal decision in the critical situation. Their simulation studies prove that system which is based on neural network is performed more efficiently than the current air traffic system.

Another research by [10] the machine learning methods are used in time series for rainfall prediction. Data was derived from the 42 cities including climatic features. They tried Support vector regression, NN, and k nearest neighbours. After performing these methods they come up with the results that machine learning methods have predictive accuracy.

[11] used the machine learning in field of radiology. They used it for the neurological disease diagnosis images, medical image segmentation and MRI images. They come-up with the results that machine learning identifies the complex patterns. It also helps the radiologists to make right decisions. Furthermore, they suggest that development of technology in machine learning is an asset for long term in the field of radiology.

3. DATA

This section illustrates the data and information of all the variables. Data set which is utilized for this research has been taken from University of California, Irvine (UCI) machine learning repository website https://archive.ics.uci.edu/ml/datasets/bank+marketing which is openly available for the public for research purpose. 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.

The dataset contains 20 variables to predict the outcome of the call. These variables contain information on the bank client, social or economic attributes, the call itself and previous contacts. Table1 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.

4. OBJECTIVE

The primary objective is to increase efficiency of telemarketing campaign.

Main objectives are:

- to find a model that can explain success of a contact, i.e. if the client subscribes the deposit. Such model can increase campaign efficiency by identifying the main characteristics that affect success, helping in a better management of the available resources (e.g. human effort, phone calls, time) and selection of a high quality and affordable set of potential buying customers. This is done by building a efficient model and making efficient in the sense that it will minimise the wrong prediction of customers who actually bought term-deposit, but predicted as non-buyers. Here, we are less keen about customers who didn't actually bought and also their predictions.

- Finding out the customer segments, using data for customers, who subscribed to term deposit. This helps to identify the profile of a

customer, who is more likely to acquire the product and develop more targeted campaigns.

- Next, from point of view of financial institution like bank they need information like to get as much as possible yes response by targeting small percentage of customers i.e. by effective targeting. That is done by cumulative Gain and Lift Chart.

- Next, we have analysed characteristics of those top targeted customers in subscribing term-deposit by DALEX.

5. DATA PRE-PROCESSING AND FEATURE SELECTION

In this paper, we develop well known ML model's prediction algorithm, namely Lasso-logistic, Random-forest, Xg-Boost, to predict whether client will subscribe the term-deposit. First, we deal with features and labels in the initial dataset, and divide the whole experimental dataset into training set and test set. Second, we use a new feature selection method Boruta algorithm to select the experimental features. After that, the hyper-parameter tuning of classification models are tuned. Finally, we use evaluation indicators to evaluate the trained model on the test set. The details of the main steps are described in the following sections.

5.1 Data Pre-processing

For data mining and machine learning, data pre-processing often accounts for about 70% of the workload, and the quality of the data set determines the success of the experiment [12] . Before constructing the model, plenty of time and efforts are spent in processing the data to ensure the validity of the results.

The dataset is derived from bank telemarketing campaign of a Portuguese bank during the period May 2008 to November 2010 [here]. The data consists of information on phone calls where people were asked if they want to subscribe to a bank term deposit.

The duration is not known before a call is performed. Also, after the end of the call response is obviously known. Thus, this input

should only be included for benchmark purposes and should be discarded if the intention is to have a realistic predictive model.

5.1.1 One-hot encoding

Xg-Boost is only suitable for processing numeric vectors, it is necessary to convert other forms of data into numeric vectors. For the considered problem, we need to convert the feature attributes into numerical values. To this end, we use the one-hot encoding method, which is a process by which feature attributes are converted into a form suitable for machine learning algorithms to do a better job in prediction. To be precise, suppose that there are n samples, each of which has a discrete feature with m attribute values. The one-hot code uses the m-bit register to binary encode the state of m attribute values, and for each sample, only one bit is active at any time. By one-hot encoding each sample, this feature is transformed into a sparse n*m matrix. In this paper, we use the one-hot code to convert the data types of the six features, say 'job', 'marital', 'education', 'default', 'housing' & 'loan'. Let's say 'marital' has 3 levels and it puts binary numeric values (1 to that respective category and 0 for others). In terms of the one-hot code, a sparse matrix depicted in Table 2 is obtained.

Table 2: Sparse matrix of 'marital'

	divorced	**married**	**single**
Sample 1	0	1	0
Sample 2	0	0	1
Sample 3	1	0	0
Sample 4	1	0	0
Sample 5	0	1	0

Table 1: Description and type of the variables from the bank telemarketing dataset from the UCI Machine Learning Repository

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

5.1.2 Discretization method

Data discretization is a method of converting attributes values of continuous data into a finite set of intervals with minimum data loss. In contrast, data binarization is used to transform the continuous and discrete attributes into binary attributes. In this data 'pdays' contains some values from [0, 30) and one value as 999 meaning client was not previously contacted. Now, it's very hard to proceed for modelling and interpretation with this kind of data and need for discretization. Thus, after discretization by interval method we get intervals as [0, 9) [9, 18) [18, 27] and 999 as 'Not Contacted'.

5.2 Feature Selection and Approaches

Feature selection is the procedure of narrowing down a subset of variables, attributes or features which can be used within the predictive modelling process. It plays vital role in the machine learning, specifically if the problem is higher dimensional. As high dimension data may consist of irrelevant variables therefore it is important to do variable selection, as pointed out in [13] . It's objective is to find the small subset instead. [14] Mentioned that feature

selection method is better than the method typical weights averaging which are mostly used in the implemented of all the recent Machine Learning libraries. According to [15] the precision of accuracy in the data can be attained only by using the appropriate features.

Process of trimming down a large data-frame, so that we can identify the factors which largely affect the target variable.

Disadvantages of dealing with overlarge feature sets:

- One is purely technical — dealing with large feature sets slows down algorithms, takes too many resources and is simply inconvenient.
- Another is even more important — many machine learning algorithms exhibit a decrease of accuracy when the number of variables is significantly higher than optimal https://www.sciencedirect.com/science/article/pii/S000437029700043X .Therefore selection of the small (possibly minimal) feature set giving best possible classification results is desirable for practical reasons. This problem, known as minimal optimal problem.

5.2.1 Boruta method

Boruta is an algorithm in the field of machine learning, and more specifically, a feature selection algorithm. The aim of the algorithm as presented in this https://www.jstatsoft.org/article/view/v036i11 to find all relevant features. It's a wrapper algorithm around the random-forest classification algorithm. It works in an iterative manner, and in each iteration the aim is to remove features according to Z-score.

Shadow attributes are pseudo features that are added to the information system, and produced by taking existing features from the original data-set and shuffling the values of those features between the original samples.

The Boruta algorithm consists of following steps:

1. Extend the information system by adding copies of all variables (the information system is always extended by at least 5 shadow attributes, even if the number of attributes in the original set is lower than 5).

2. Shuffle the added attributes to remove their correlations with the response.

3. Run a random forest classifier on the extended information system and gather the Z scores computed.

4. Find the maximum Z score among shadow attributes (MZSA), and then assign a hit to every attribute that scored better than MZSA.

5. For each attribute with undetermined importance perform a two-sided test of equality with the MZSA.

6. Deem the attributes which have importance significantly lower than MZSA as 'unimportant' and permanently remove them from the information system.

7. Deem the attributes which have importance significantly higher than MZSA as 'important'.

8. Remove all shadow attributes.

9. Repeat the procedure until the importance is assigned for all the attributes, or the algorithm has reached the previously set limit of the random forest runs.

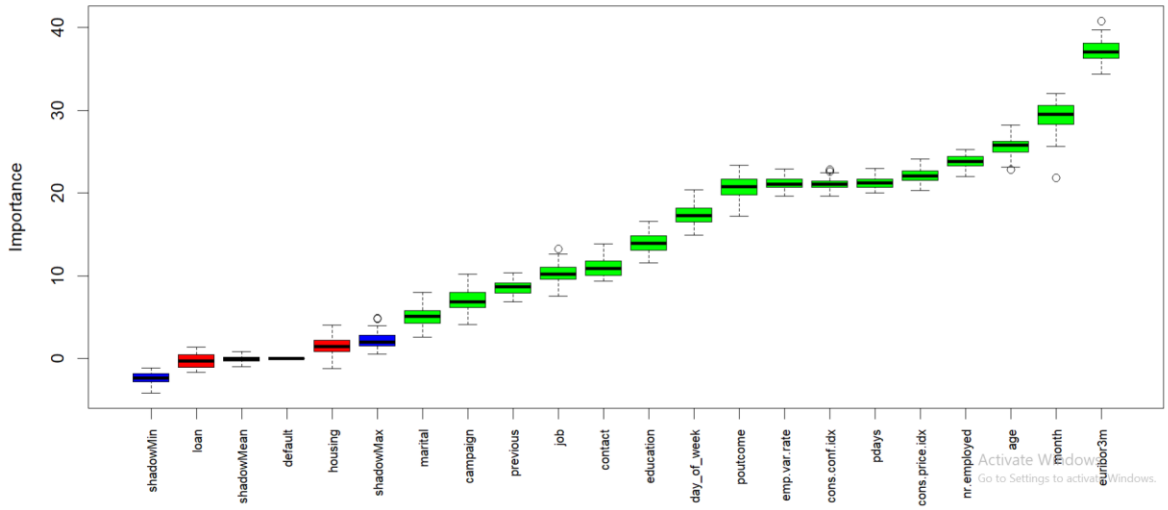


Figure 1.1: Boruta result plot for train data. Blue boxplots correspond to minimal, average and maximum Z score of a shadow attribute. Red and green boxplots represent Z scores of respectively rejected and confirmed attributes.

So, here "loan", "default" and "housing" are unimportant predictors and rest are important.

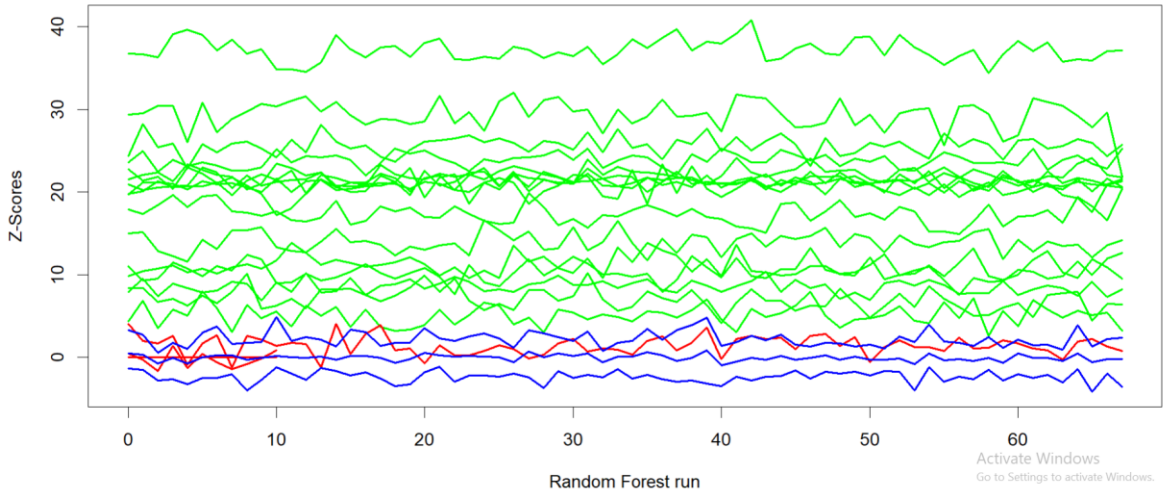


Figure 1.2: Z score evolution during Boruta run. Green lines correspond to confirmed attributes, red to rejected ones and blue to respectively minimal, average and maximal shadow attribute importance.

The result can be seen on Figure 2. One may notice that consecutive removal of random noise increases the Z score of important attributes and improves their separation from the unimportant ones; one of them is even ‘pulled’ out of the group of unimportant attributes just after the first initial round. Also, on certain occasions, unimportant attributes may achieve a higher Z score than the most important shadow attribute, and this is the reason why we need multiple random forest runs to arrive at a statistically significant decision.

6. DATA EXPLORATION

6.1 Response variable

First we wish to start off by getting an idea of how two levels of response variable are distributed in data.

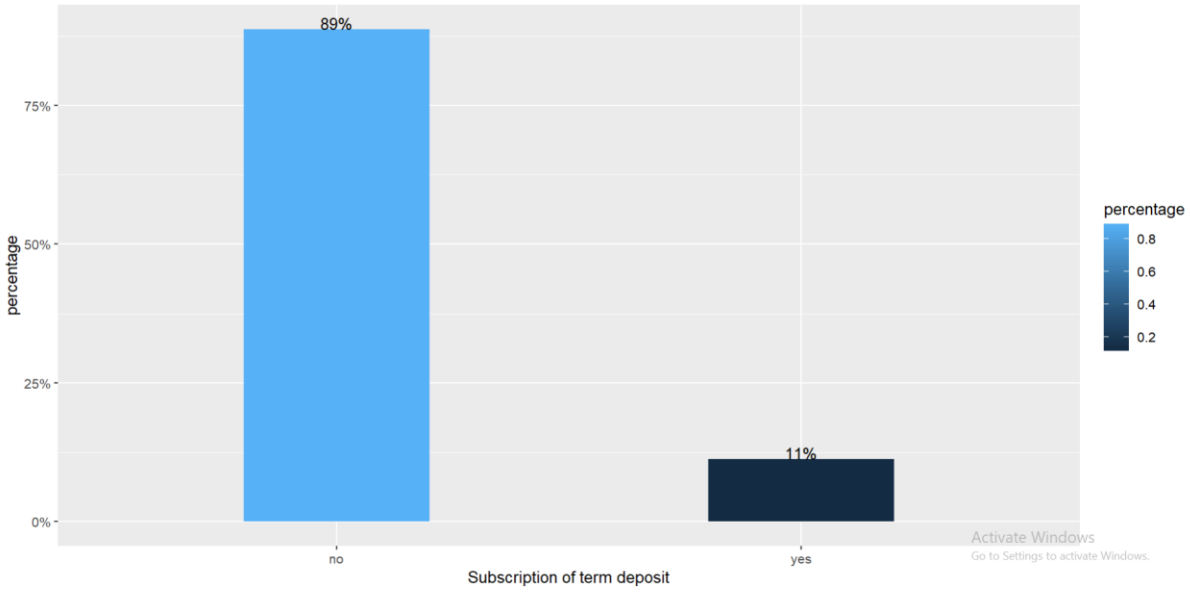


Fig 2.1: Percentage of term deposit subscription

From fig 3.1, we see that there is not evenly distributed between two classes, rather response = no is about 8 times than response = yes. So, clearly the data is imbalanced.

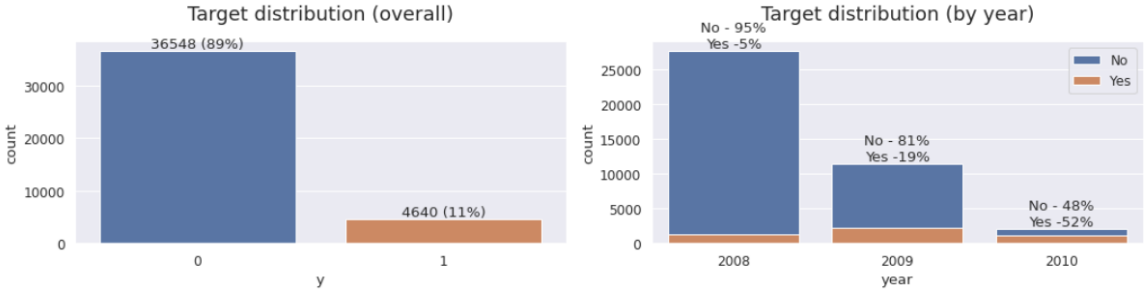


Fig 2.2: Percentage of term deposit subscription by year

Here, we are adding year **ordered by date (from May 2008 to November 2010)** as an auxiliary variable to find insights of this as well as socio economic context predictors. Here, in 2010 proportion of yes getting unusually high compared to 2008 , 2009 and this is because of euribor 3 month rate, employment variation rate explained in later.

Next we want to see is there presence of correlation between continuous variables, which can be found by following plot:

Fig 2.3: Correlation b/w continuous variables

- Correlation are high/moderately high between
  i) Employment variation rate with consumer price index, euribor 3month rate, no of employees.
  ii) Euribor 3month rate with no of employees, consumer price index.

Histograms

We construct histograms for each variables to get preliminary idea of distributions.

Some preliminary observations are:

- Multiple of the variables are positively skewed, with smaller values of the data occurring very frequently and larger values appearing much fewer times in comparison. These variables include Age (in year), duration (in seconds), campaign, previous.
- Variables like pdays appear to be negatively skewed – higher values of the data appear to be more frequently.
- Employment variation rate, consumer price index, consumer confidence index, Euribor 3month rate and number of employees all have multimodal distributions.

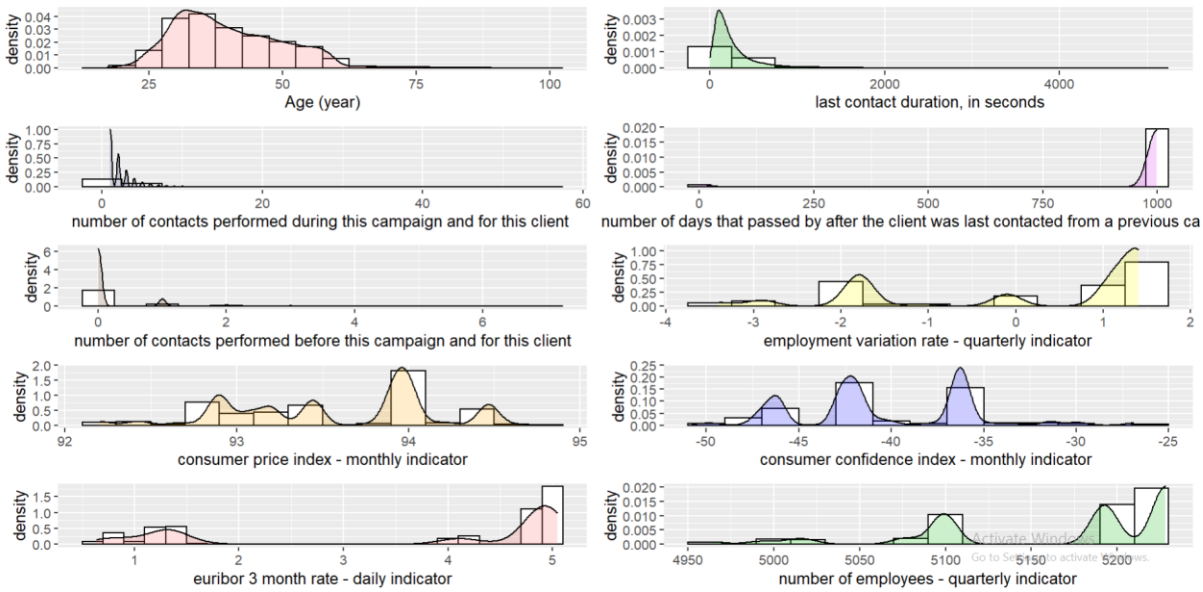


Fig 2.4: Histograms of numeric predictors

Feature Analysis

6.2 Bank Client variables

1. Age

Analyzing distribution and 'Yes' proportion of 'Age' variable.

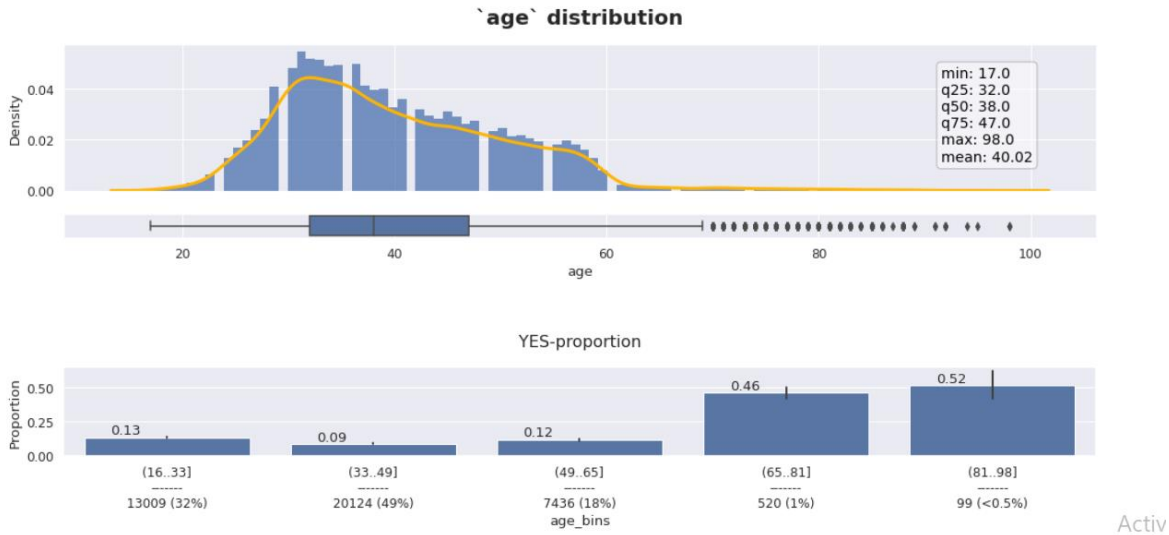


Fig 2.5: Histogram and distribution over age intervals

Here, Higher age-group above 65, are more likely to subscribe to term deposit and these age groups are very less targeted, whereas less than 65 age groups are very highly targeted and getting very less 'yes' response. Thus, leading to overall small proportion of 'Yes' response.

Fig 2.6: Distribution of Age by month and year

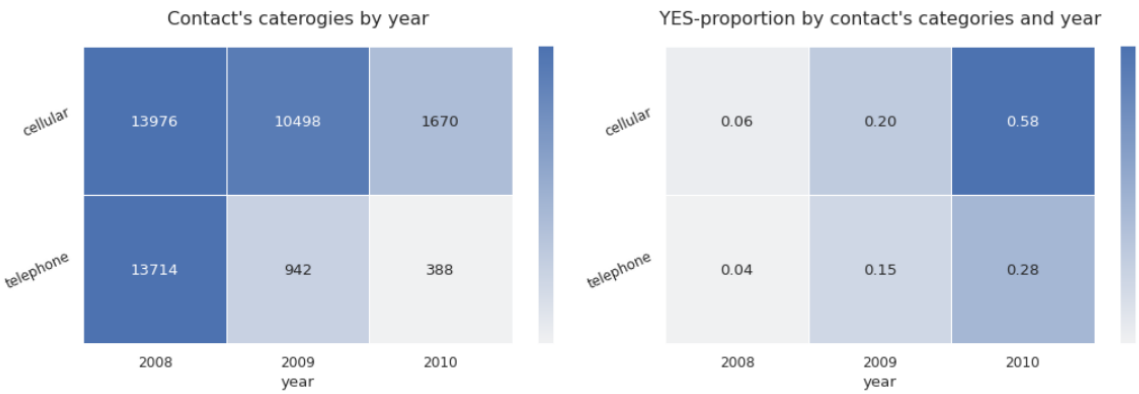
Most significantly high response can be found in October month of 2008(63%), followed by in December, November and March of 2007 we got 51%,47% and 45% of 'yes' response; Whereas in May and march of 2010 we got 58% and 57% 'yes' Reponses respectively. Some interesting hidden economic facts are there for this kind of behavior of clients, which we will see later when analyzing economic predictors. Interestingly, throughout 2010 year for all month they got high 'yes' responses.

6.3 Related last contact variables

2. Contact

Analyzing distribution and 'Yes' proportion of 'Contact' variable.

Fig 2.7: Distribution of contact



Pretty big difference in 2010 between cellular and telephone.So, focusing on year 2010 as follows:

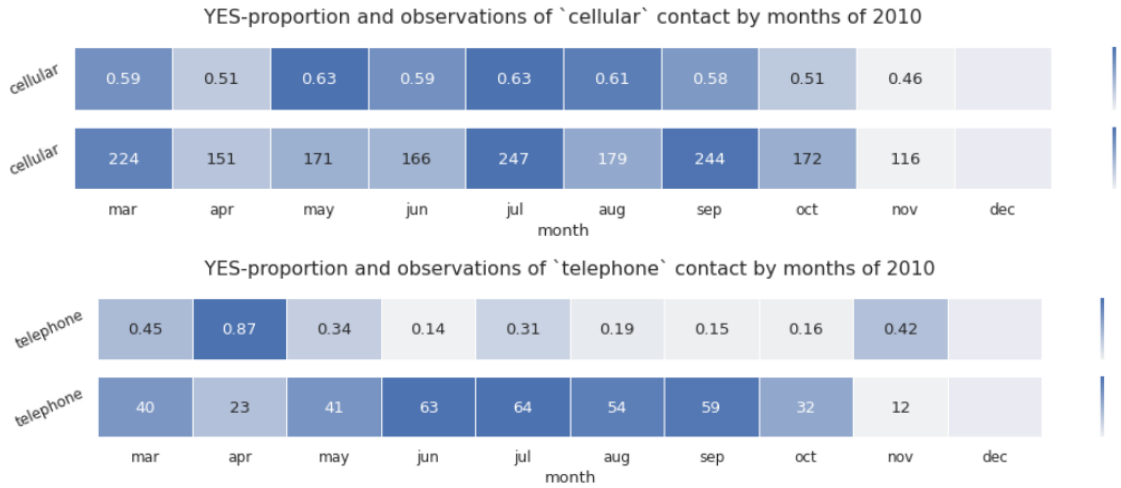


Fig 2.8

We've got an anomaly high YES-proportions for a cellular type of contact in 2010. The reason for this is unknown so far(may be for some reason sales or service of repairing of cellular in April month,2010 has decreased .

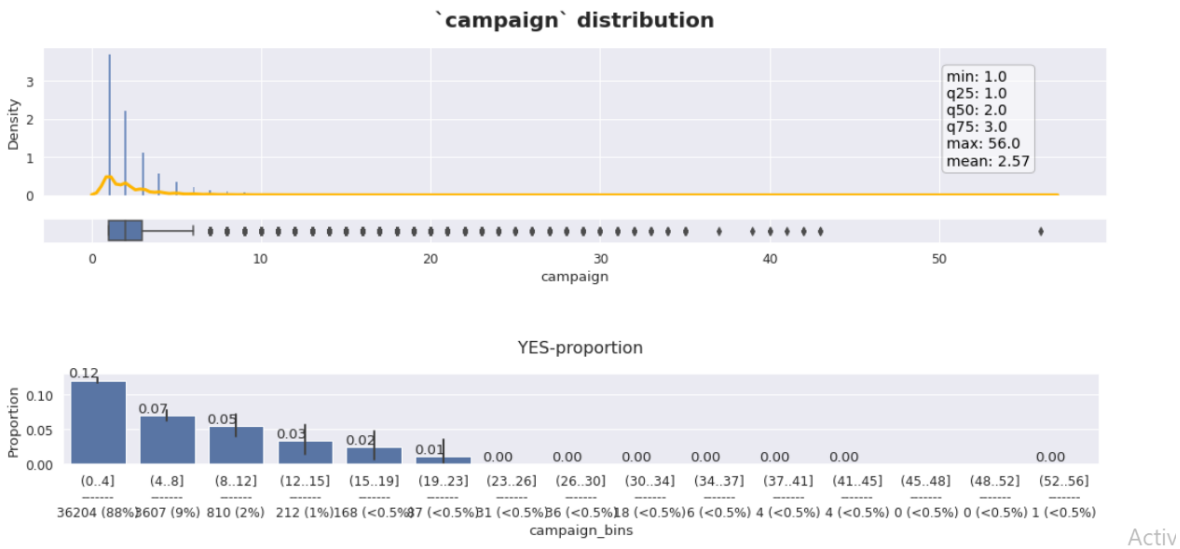
Most people were contacted by the cellular (≈≈63%)

- People which were contacted via cellular are a bit likely to agree to subscribe a bank term deposit. But just a bit:
  - 2008: 0.06 vs 0.04
  - 2009: 0.20 vs 0.15
  - 2010: 0.58 vs 0.28 (anomaly results regarding 2008-09)

6.4 Other Attributes

3. Campaign

Analyzing distribution and 'Yes' proportion of 'Campaign' variable.



Let's take a closer look at the first bin (from 1 to 4 contacts) and try to see if YES-proportion is affected by the number of contacts during this campaign:

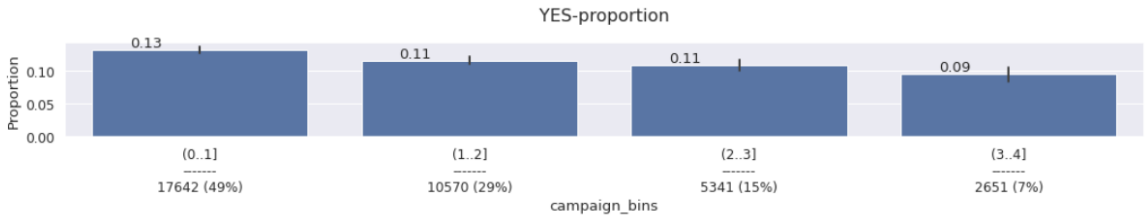


Fig 2.9: Distribution of campaign and of yes proportion of campaign

The plot above shows that the more contacts is the less YES-answers, BUT on the other hand every time we get additional positive answers from people who didn't agree previously .Yes, the proportion is becoming smaller and smaller, hence it's still there.

- Nearly 49% were contacted only once.
- 88% of people were contacted less than 5 times.
- More contacts lead to less YES-answers, but it's still effective.

4. pdays

Analyzing distribution and 'Yes' proportion of 'pdays' variable. Here, pdays=999 means not previously contacted and here denoted by -1.

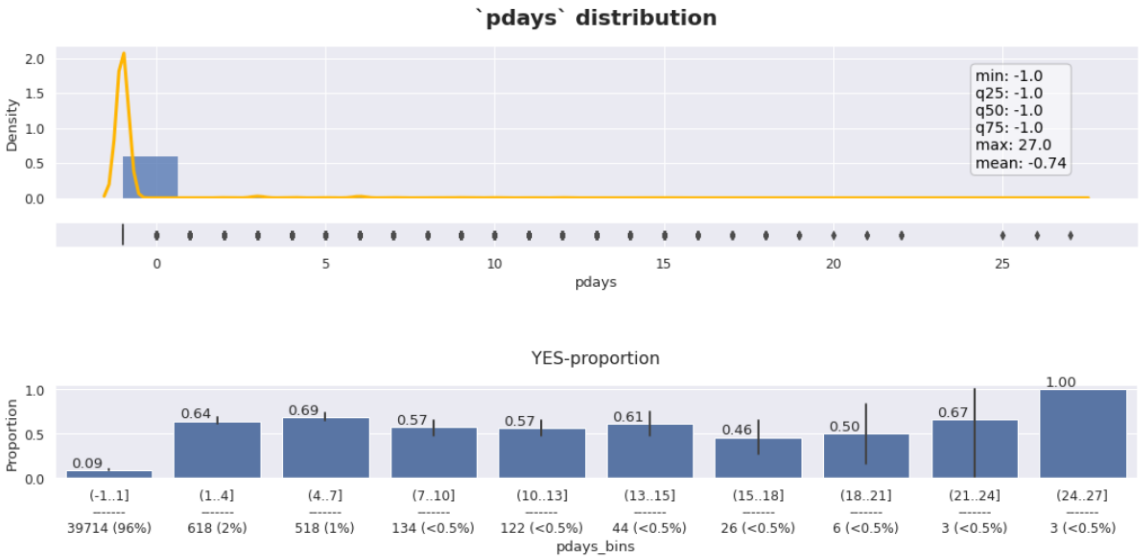


Fig 2.10: Distribution of pdays and of yes proportion of pdays

- Around 96% were not contacted from the last campaign.
- The chances to get YES are pretty high within the first 2 weeks (up to 14-15 days); further we have high variance and small data.
- People, who was not being previously contacted, have the lowest probability to say YES (0.09).

5. Previous

Analyzing distribution and 'Yes' proportion of 'Previous' variable.

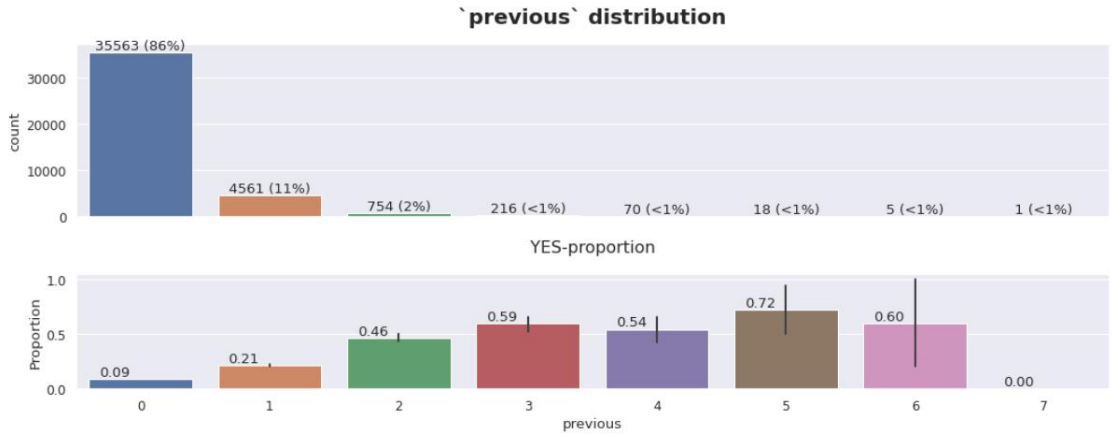


Fig 2.11: Distribution of previous and of yes proportion of previous

- Around 86% were not contacted before this campaign.

- It seems like making contacts before a campaign is much better than do not do them at all (for numbers bigger than 3 we have a small number of observations).
- Furthermore, we can suppose that more previous contacts mean more chances to get YES, but only to some degree (approx. 3 to 5).

6. Poutcome

Analyzing distribution and 'Yes' proportion of 'Poutcome variable.

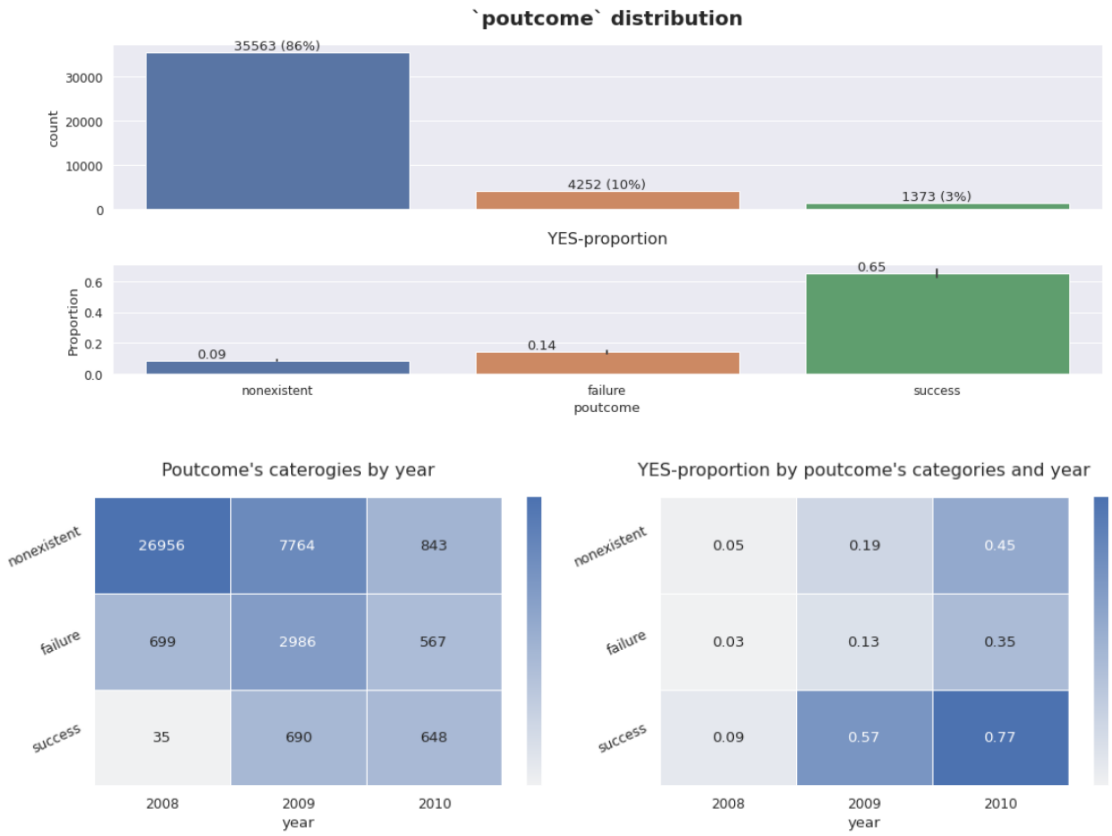


Fig 2.12: Distribution of poutcome and of yes proportion of poutcome

"Success -> Success" clients: 0.65*1373 = 2.2%

"Failure-> Success" clients: 0.14*4212 = 1.4%

- 86% did not participate in the previous marketing campaign, therefore we don't have any previous outcomes for them.
- About 2.2% of clients answered YES during both previous and current campaigns.
- About 1.4% of clients answered YES for the current campaign, while having NO with a previous one.

- If we have a positive answer with a previous campaign, then we more likely to get YES answer during the current one.
- If we have a negative answer with a previous campaign, then we less likely to get YES answer, and it is even lower probability for those whose results are non-existent.

6.5 Social and Economic context attributes

7. Employment Variation Rate

Figure displays histogram, describing the distribution of 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 low values of emp.var.rate.

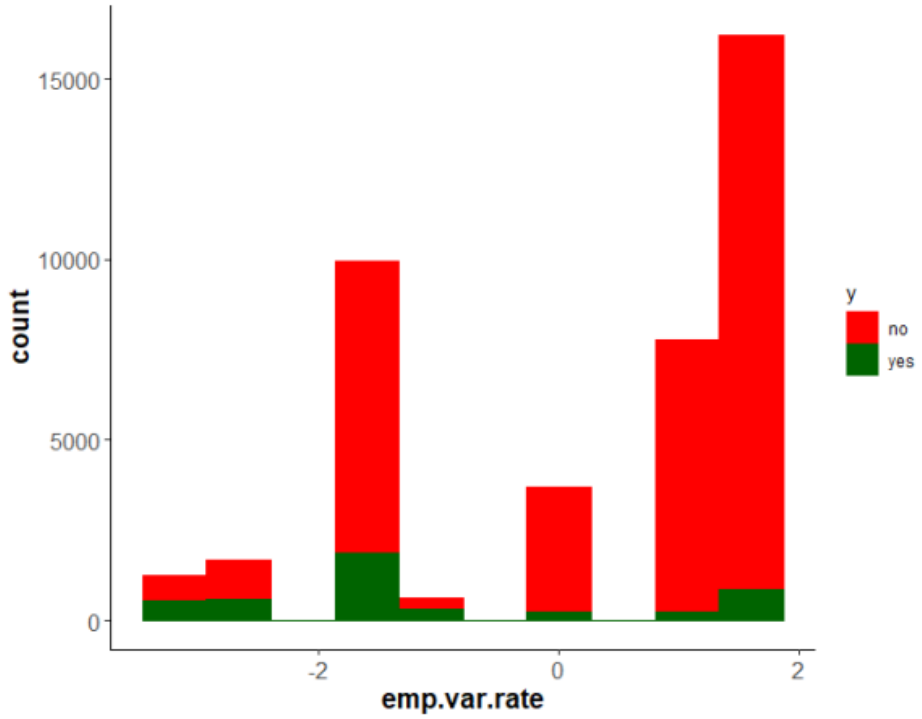


Fig 2.13 : Distribution of Employment variation rate

Here, we are going to analyse these features by year.

In year 2008 and 2009 employment variation rate is highly variable, unstable, whereas in 2010 this is stable and proportion of getting yes response in 2010 is more or less higher throughout all months than 2008 and 2009(except in October 2008).

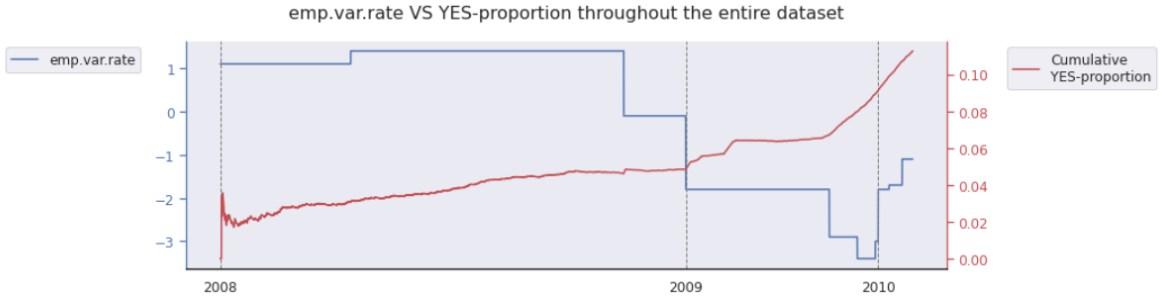


Fig 2.14: distribution of employment variation rate by year

As you see, it is a light negative correlation. That means, the higher employment variation rate - the lower YES-proportion we got, and vice versa. And that seems quite reasonable: if the employment rate is highly various, people are not sure about their future and tend to make cautious decisions - so they give YES-answer less frequent. But we should be careful here and recall that correlation is not causation.

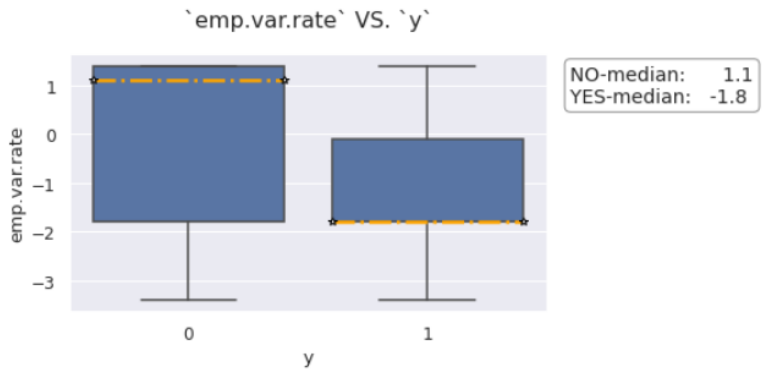


Fig 2.15

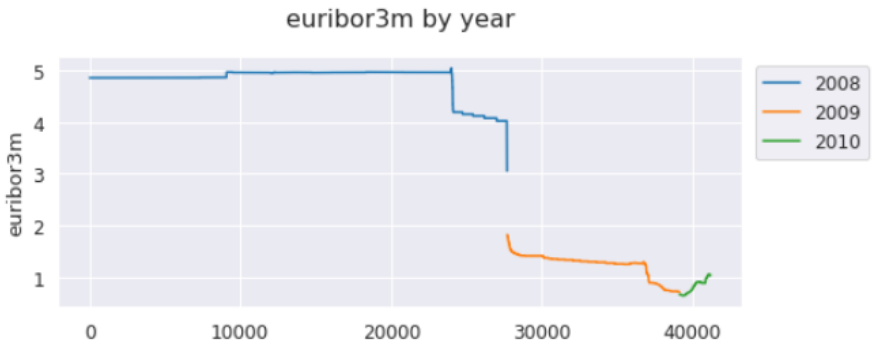
Obviously, there is a huge difference between medians, and that's why this feature will be really useful for classification.

Year	Variation of employment variation rate	Number of observations
2008	0.244666	67.2
2009	0.329827	27.8
2010	0.090704	5.0

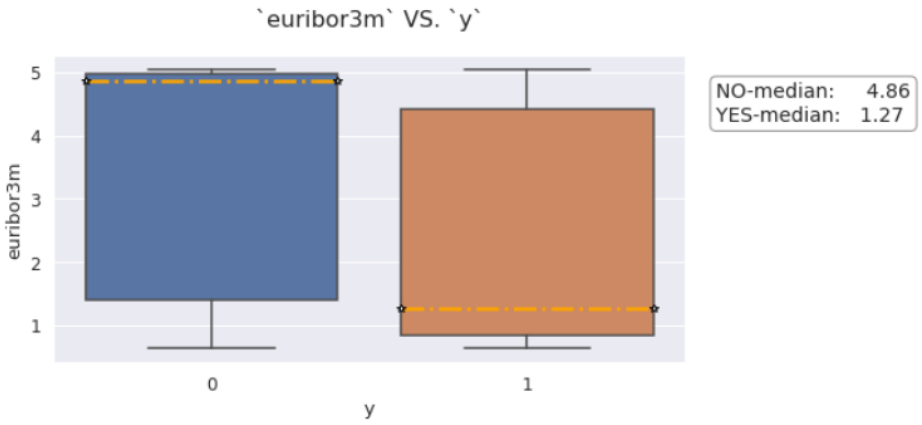
Both 2008 and 2009 have a higher variation of employment variation rate and that's about 95% of all observations. As we all know, 2008 and 2009 were the years of the global financial crisis, so those 95% were obtained during a quite hard time for economy.

8. Euribor 3 month rate – daily indicator

Euribor - The Euro Interbank Offered Rate is a daily reference rate, published by the European Money Markets Institute, based on the averaged interest rates at which Eurozone banks offer to lend unsecured funds to other banks in the euro wholesale money market. Prior to 2015, the rate was published by the European Banking Federation.



- 2008: during that year a euribor3m was stable and dropped at the end.
- 2009: the euribor3m continued to gradually decrease, but much slowly than in the last months of 2008, and there was another drop at the end of the year.
- 2010: the euribor3m was growing.



Huge difference between medians for these two categories. Thus, it is useful for classification.

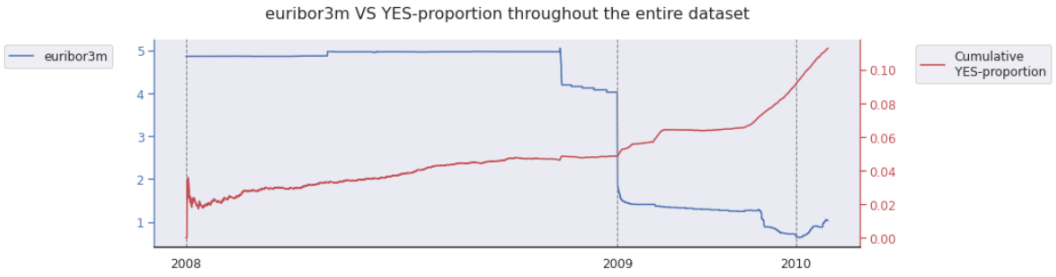


Fig 2.16: Distribution of Euribor 3 month rate by year

Here, if reference rate is going to be low at which rate European bank offer to lend unsecured funds, thus they can lend more money at low rate and offer to open term deposit at high interest rate. Thus, proportion of 'yes' response is getting higher when this rate is going to be low.

7. MODELING STRATEGIES

7.1 Lasso-Logistic regression

The main objective of lasso is to find and identify the feature subset whose coefficients are non-zero [16]. Lasso is very powerful technique in regression and prevent the model for overfitting by simplify the function. One of the main benefit of using this technique is that it can be used for the purpose of best subset selection. In this way, it skipped the extraneous variables from the model and comes up with only that variables which are model relevant. It is simple technique which is based on regression and works with significance of p-value for the selection of best subset [17]. It becomes popular for the regression and classification problems with various independent variables.

Here, we have data (x_i, y_i) ; i=1(1)N where, $x_i = (x_{i1}, x_{i2}, \dots, x_{ip})^T$ re the predictors and y_i are the response.

Here, we are going to predict term deposit subscription using given all 16 predictors (excluding housing, loan, and default by Boruta Method).

y = $\begin{cases} 1 \text{ ; if subscribed} \\ 0 \text{ ; if not subscribed} \end{cases}$; so y is a binary variable

Here, are some following choices for fitting the data.

1. Logistic Regression
2. Lasso – logistic Regression
3. Ridge – logistic regression

First model that we'll build is a parametric model and since our target variable is binary we'll use Logistic regression.

Here, we are performing Lasso-logistic regression as this will give us a subset of predictors (unlike ridge logistic regression).

Split data into train and test after dropping the duration variable since this attribute highly affects the output target (e.g., if duration=0 then y='no').

First, we are fitting logistic regression model on all predictors; model is as follows:

$$\text{logit}(p) = \beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k + \gamma_{11} M_{mar} + \cdots + \gamma_{19} M_{nov} + \cdots + \gamma_{l1} \text{Poutcome}_{success} + \gamma_{l2} \text{Poutcome}_{failure}$$

where $(x_1, \ldots, x_k)$ represents all continuous predictors such as age, campaign, previous, employment variation rate , consumer price index , consumer confidence index, euribor 3 month rate, number of employees. Here, k=8

and M denotes last contact month of year and having 10 levels as march, april , …,December.

Also, $M_{mar} = \begin{cases} 1 \text{ ; if month} = \text{march} \\ 0 \text{ ; otherwise} \end{cases}$

Similarly other dummy variables are created for all other 8 levels . For last level don't need to create any dummy variable.

Similarly, from dataset description, we see that for all 12 categorical variables having respective levels and for each of variable (n-1) dummy predictors have been created if they have n levels. Here, l= 12.

Two advantages of doing lasso-logistic over simple logistic regression:

- Shrinking the regression coefficients towards zero.
- Coefficient estimates can significantly reduce their variance (with the cost of slightly higher bias).

In logistic regression, Maximum likelihood estimation procedure estimates $\beta_0, \beta_1, \ldots, \beta_k$ which is getting by maximizing the likelihood

$\frac{\delta l}{\delta \underset{\sim}{\beta}} = 0$ Where $l = \log L = \frac{y\theta - b(\theta)}{a(\phi)} + c(y, \phi)$;

$\theta = \ln\left(\frac{\pi}{1-\pi}\right), b(\theta) = \ln(1 + e^{\theta}), \phi = 1$ as $y \sim \text{bin}(1, \pi)$

In Lasso-logistic regression, the lasso coefficient estimates solve the problem

$$\hat{\beta} = \arg\min_{\underset{\sim}{\beta}} [L\{\underset{\sim}{y}, f(\underset{\sim}{\beta}, \underset{\sim}{X})\} + \lambda(\underset{\sim}{\beta})] \quad \text{--------} (1)$$

$$\text{where } \lambda(\underset{\sim}{\beta}) = \lambda . \left|\left|\underset{\sim}{\beta}\right|\right|_1 = \lambda . \sum_{j=1}^{p} |\beta_j| \text{ have to minimise subject to s}$$

and L(.)is loss function ; $\underset{\sim}{y}$ is the vector of observed values of dependent variable and $f\left(\underset{\sim}{\beta}, \underset{\sim}{X}\right)$ is the corresponding vector of the model's predictions computed for model coefficients $\underset{\sim}{\beta}$ and the matrix $\underset{\sim}{X}$ of values of explanatory variables for the observation from training dataset.

- In this case(unlike ridge regression) $\underset{\sim}{\beta}, \underset{\sim}{\sigma^2}$ have to be obtained by using numerical optimization procedure.
- For our binary classification problem, for natural choice is Bernoulli distribution and resulting loss function is,

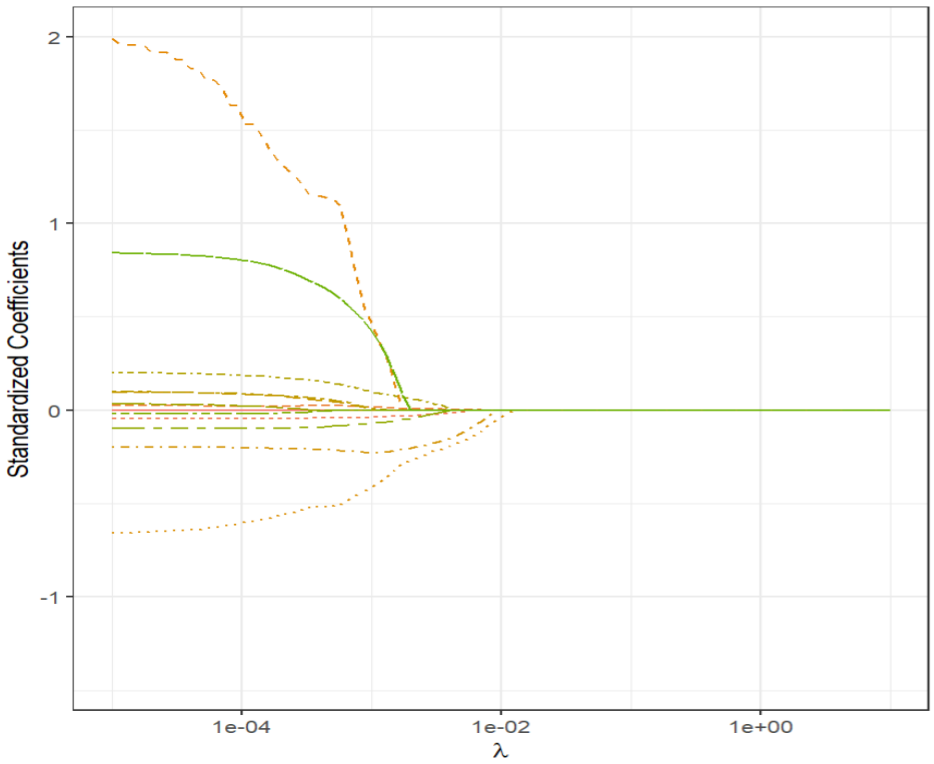
$$L\left(\underset{\sim}{y}, \underset{\sim}{p}\right) = -\frac{1}{n}\sum_{i=1}^{n}\{y_i x_i' \underset{\sim}{\beta} - \ln(1 + \exp(x_i' \underset{\sim}{\beta})\} \text{ --------- (2)}$$

Where $\ln\left(\frac{p_i}{1-p_i}\right) = x_i' \underset{\sim}{\beta}$

Where y_i is equal to 0 or 1 in case of "no response" and "response" (or "failure" and "success"), and p_i is the probability of y_i being equal to 1

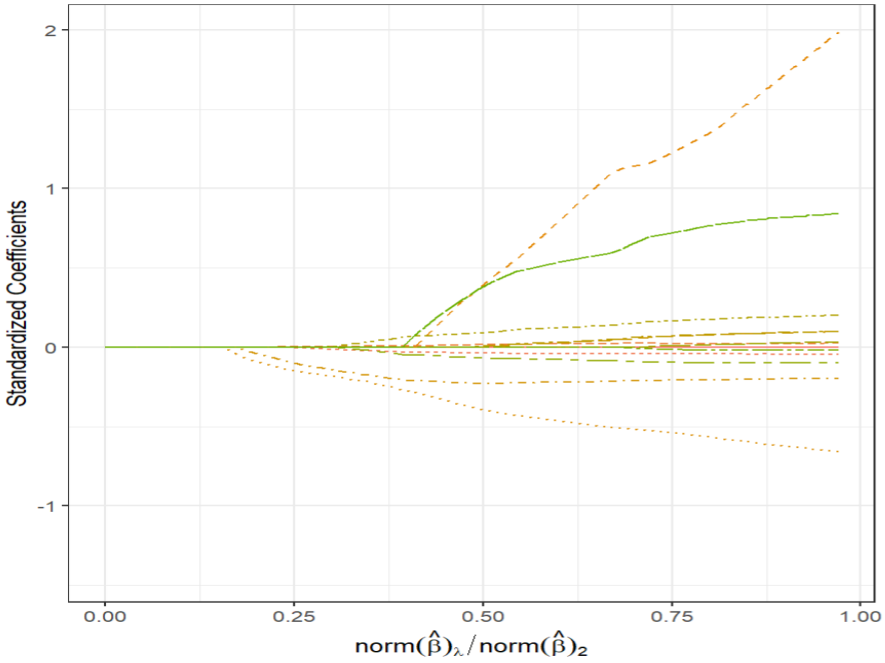
Shrinkage Path

The following figure show the change of estimated coefficients with respect to the change of the penalty parameter (log (λ)), which is the shrinkage path.



In above Figure, the lasso regression coefficient estimates for the data set are displayed. Each curve corresponds to the lasso regression coefficient estimate for one of the 16 variables, plotted as a function of λ. For example, the line represents the large lasso regression estimate for the poutcome-success, month-mar and pdays-not contacted coefficient, as λ is varied. At the extreme left-hand side of the plot, λ is essentially zero, and so the corresponding lasso coefficient estimates are the same as the usual least squares estimates. But as λ increases, the lasso coefficient estimates shrink towards zero. When λ is extremely large, then all of the lasso coefficient estimates are basically zero; this corresponds to the null model that contains no predictors. In this plot, the poutcome-success, month-mar and pdays-not contacted variables are tend to have by far the largest coefficient estimates.

Clearly, the shrinkage effect are strong with λ. Depending on the choice of λ, some of the coefficients are exactly equal to zero. As the λ increases, the estimated coefficients are shrunk exactly to zero.



Standardized lasso regression coefficients are displayed as a function of $\frac{\left|\left|\hat{\beta}_\lambda^S\right|\right|_1}{\left|\left|\hat{\beta}\right|\right|_1}$

Here, $\lambda = 0$ => lasso estimates are same as least square estimates => $\frac{\left|\left|\hat{\beta}_\lambda^S\right|\right|_1}{\left|\left|\hat{\beta}\right|\right|_1} = 1$

Here, $\lambda = \infty$ => lasso estimates are vector of 0's => $\frac{\left|\left|\hat{\beta}_\lambda^S\right|\right|_1}{\left|\left|\hat{\beta}\right|\right|_1} = 0$

Here, x axis represents amount that the lasso regression coefficient estimates have been shrunken towards zero. Here, from right to left we see that poutcome-success, month-mar are very slowly shrunken towards zero i.e. they are very important predictors; followed by horsepower predictor.

Model Selection

Here λ is the hyper-parameter for lasso-logistic model. Here, for lasso–logistic regression; we have chosen to implement the function over a grid of values ranging from $\lambda = 10^{10}$ to $\lambda = 10^{-2}$, essentially covering from the null model containing only the intercept, to the ML fit.

Here, by fitting the model we get $\text{lamda}_{\min} = 0.01$ for which binomial deviance is lowest.

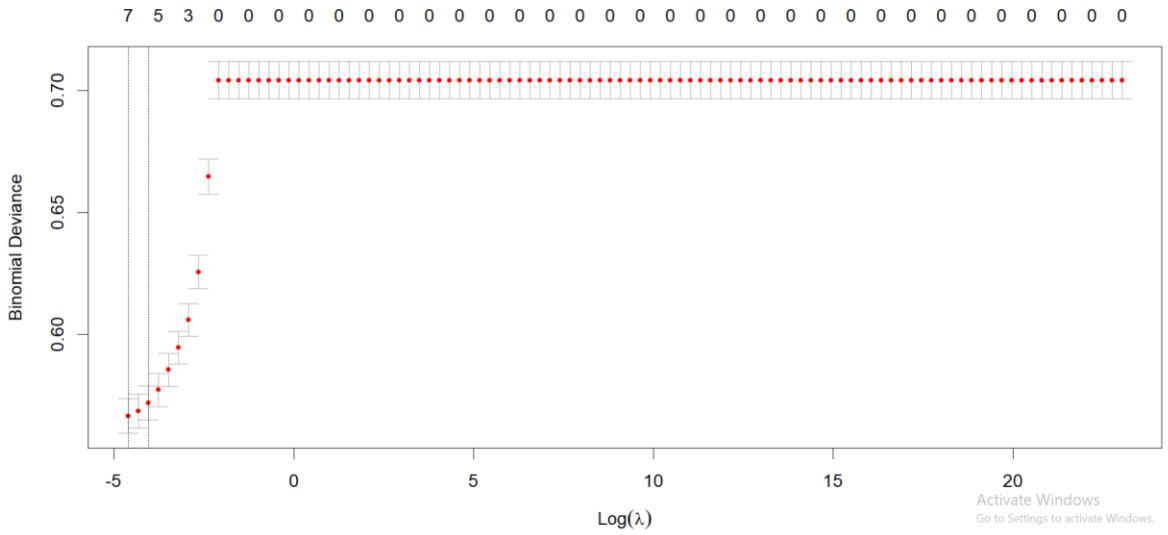


Fig : Binomial deviance against log(λ)

From fig, we have to choose that λ which minimizes binomial deviance i.e. to choose λ between those two vertical lines approximately from e^{-4} to e^{-3} . Here $\lambda = e^{-4.6} = 0.01$.

In the above plot, the first dotted line is that λ value for which the minimal deviance is achieved, but it is likely that same deviance with different λ's. The second dotted line is the largest λ for which the deviance is within one standard error of the minimal deviance .

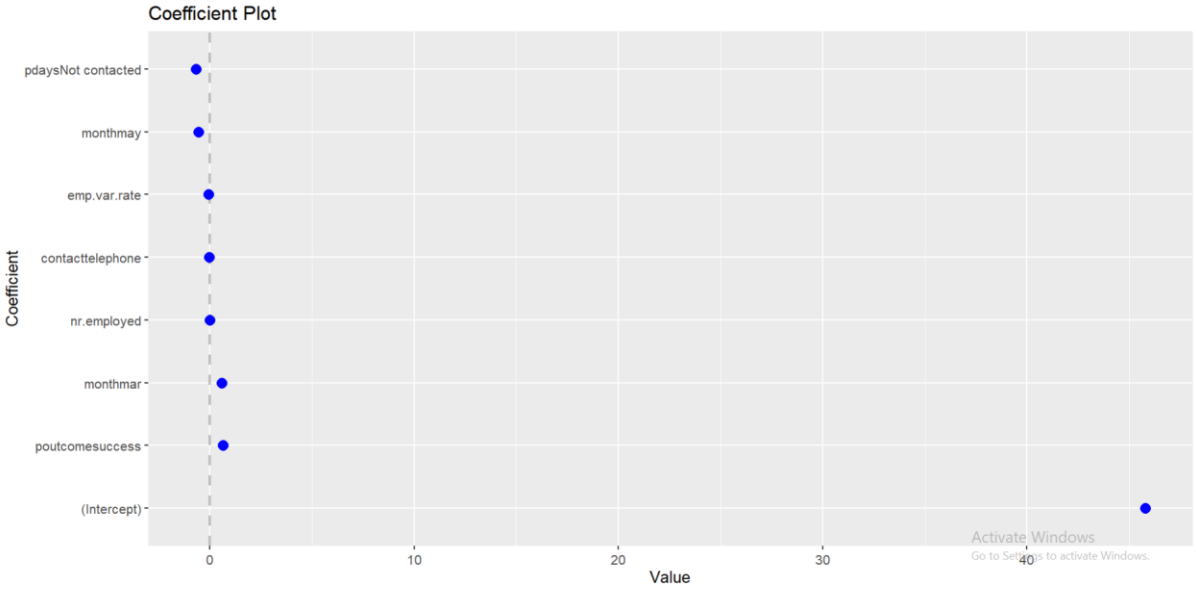
Here, in the top of the plot showing number of non-zero coefficient of predictors for different λ's.

Coefficient estimates:

We get following estimates by fitting procedure (1)

Variable name	Coefficient
Intercept	-45.791
Contact-telephone	-0.036
Month-mar	0.570
Month-may	-0.555
Pdays-not contacted	-0.679
Poutcome-success	0.652
Emp.var.rate	-0.055
Nr.employed	-0.009

Plot of coefficients:



So, now model becomes

$$logit(p) = 45.79 - 0.036C + 0.57M_{mar} - 0.555M_{may} - 0.68Pdays_{Not\ contacted} + 0.65Poutcome_{success} - 0.0547E - 0.009N$$

Where, p = probability of term deposit subscription

$$C = \begin{cases} 1\ ; \text{if contacted through telephone} \\ 0\ ; \text{ if contacted through cellular} \end{cases};$$

$$M_{mar} = \begin{cases} 1; contact\ month\ is\ march \\ 0\ ; otherwise \end{cases};$$

$$M_{may} = \begin{cases} 1; contact\ month\ is\ may \\ 0\ ; otherwise \end{cases};$$

$$\text{Pdays}_{\text{Not contacted}} = \begin{cases} 1\ ; \text{not contacted before this campaign after contacted from last campaign} \\ 0\ ;\ \text{O. w.} \end{cases}$$

$$\text{Poutcome}_{\text{success}} = \begin{cases} 1\ ; \text{outcome of previous campaign was success} \\ 0\ ; \text{O. w.} \end{cases},$$

E = Employment variation rate and N = Number of employees

Interpretation:

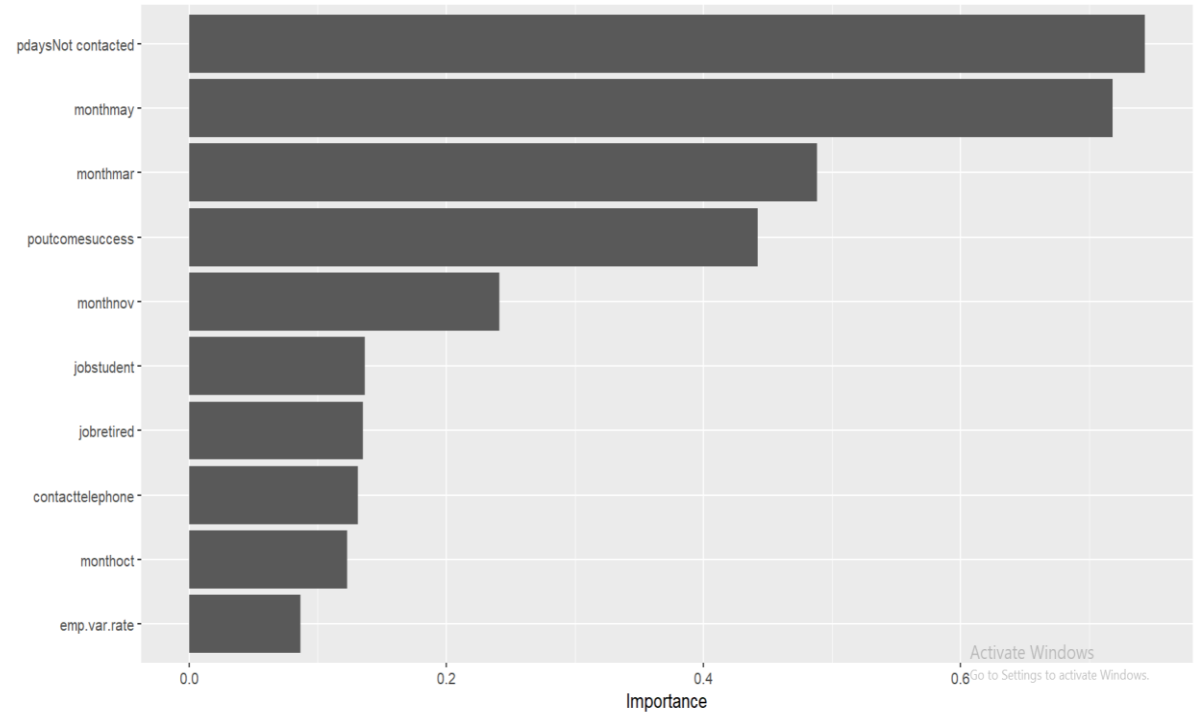
- Holding other predictors at fixed value , odds of getting term deposit subscription if communication type is telephone over odds of getting term deposit subscription if communication type is cellular is exp(-0.036054275) = 0.96 . In terms of percentage, if through telephone contacted we get approx 4% less “yes” response.
- Holding other predictors at fixed value , odds of getting term deposit subscription if last contact month is march over odds of getting term deposit subscription if last contact month is any other except march is exp(0.570403931) = 1.7689 ≈ 1.77 . In terms of percentage, if contacted in March month , there is 77% more chance of getting term deposit subscription.
- Holding other predictors at fixed value , odds of getting term deposit subscription if last contact month is may over odds of getting term deposit subscription if last contact month is any other except may is exp(-0.555443601) = 0.57. In terms of percentage, if contacted in May month , there is 43% less chance of getting term deposit subscription.
- Holding other predictors at fixed value , odds of getting term deposit subscription if not contacted in this running campaign over odds of getting term deposit subscription if contacted any number of times after last campaign is exp(-0.679852911) = 0.51. In terms of percentage, if not contacted in campaign, there is 49% less chance of getting term deposit subscription.
- Holding other predictors at fixed value , odds of getting term deposit subscription if contacted those clients who think previous campaign was success over odds of getting term deposit subscription if contacted those

clients who think previous campaign was not success is exp(0.651689017) = 1.92. In terms of percentage, if contacted those clients who think previous campaign was success,there is 92% more chance of getting term deposit subscription.

- The coefficient of emp.var.rate says that, keeping other predictors at a fixed value, we will see 5% decrease in the odds of getting term deposit subscription for one unit increase in employment variation rate since exp(-0.054705617)≈ 0.95.
- The coefficient of nr.employed says that, keeping other predictors at a fixed value, we will see 1% decrease in the odds of getting term deposit subscription for one unit increase in number of employees since exp(-0.009162675)≈ 0.99.

Variable Importance plot

Similar to (generalized) linear models, the absolute value of coefficients are returned for a model. It is important that features be standardized prior to fitting the model.



Here, pdaysNot contacted, monthmay is carrying about 70% importance for the model, followed by monthmar and poutcome success by approx 45%. So, for next campaign,

- Target those individuals who have not contacted in previous campaigns.

- In month march and may should be targeted to achieve higher success.
- Target those individuals who have marked previous campaign as success

Prediction

Here, we have created a model using weights for each class of response "yes" and "no".

As, recall is more important than precision we are giving much weight to "yes" class(minority class) .

Here, weight for each class is $1 - \frac{\#\ of\ class\ members}{\#\ of\ total\ members}$

Class	Weight
Yes	0.887
No	0.113
Sum	1

Now, using this weight we have built the model; and get predicted probabilities on the test dataset. Using 0.5(default) as threshold we are getting predicted class and confusion matrix as follows.

N.T : As, here we have used class-weights for giving importance on recall; thus no need to change threshold.

Now, here we are only interested on those ,which model predicted correctly those clients who have actually subscribed term deposit. That means, we want to reduce the number of misclassified no of persons who actually subscribed term deposit but predicted as non-subscribed.

	Actual	
Predicted	0	1
0	TN(=True Negeative)	FN(=False Negeative)
1	FP(=False Positive)	TP(=True Positive)

Now, we need to find all these numbers such that followings are satisfied.

- Sensitivity(TPR/Recall) = P[Test = 1 | actual =1] = $\frac{\mathrm{TP}}{\mathrm{TP+FN}}$ represents out of total actual yes responses model predicts how many yes responses correctly.

- Precision(+ve Predicted value) = P[Actual=1|Test=1] = $\frac{TP}{TP+FP}$ represents out of total predicted yes, how many are actually yes response.
  Problems:
- Firstly , as actual yes reponse is very small amount(for imbalanced data) we want to small no of wrongly predicted as non-subscriber where as they are actually subscriber.
- Secondly, out of total predicted yes which model predicts we also need to reduce wrongly predicted as non-subscriber . It's important from the point of view of cost and time. Because if model predicts wrongly those clients as subsciber ; then from bank telecommunication happen for next campaign and actually they will get non-subscriber. So, there time and cost both are damaging.
  But, we get first point i.e. main focus is to minimize mainly FN ; so it's more imp than reducing FP. So, here Recall is more important than precision.

	Actual	
Predicted	0	1
0	9350	540
1	1614	852

Recall/Sensitivity = 61.21%

Precision = 34.56%

Precision-Recall Curve is More Informative than ROC in Imbalanced Data:

Accuracy is misleading:

The failure of accuracy as a metric on imbalanced data is well-known. Consider the case of a dataset with the ratio of 1 positive per 100 negatives. On this task, a model that predicts all cases to be negative yields an accuracy of 99%. This model, nevertheless, is a dummy classifier that always predicts the majority class.

1. AUROC is overly optimistic:

AUROC is threshold-invariant. We do not have to decide what threshold separating positive and negative classes should be in order to calculate the

metric. This means even when the two prediction error costs (i.e. false positive and false negative) are unequal, the metric is indifferent to it. However, when the negative class is more prevalent but there is low value in true-negative predictions, the ROC can provide an overly optimistic measurement of a model's performance. For example, consider the case of a dataset which has 10 positives and 100,000 negatives. We have 2 models:

- Model A: predicts 900 positives, in which 9 of them are true positives
- Model B: predicts 90 positives, in which 9 of them are true positives

Obviously, Model B has better performance. Although they both predict the same number of positives, Model B does not output as many false positives. In other words, Model B is more "precise". However, consider the ROC analysis of the two models, which measures true positive rate (TPR) against false positive rate (FPR):

- Model A: TPR = 9/10 = 0.9 and FPR = (900–9)/100,000 = 0.00891
- Model B: TPR = 9/10 = 0.9 and FPR = (90–9)/100,000 = 0.00081

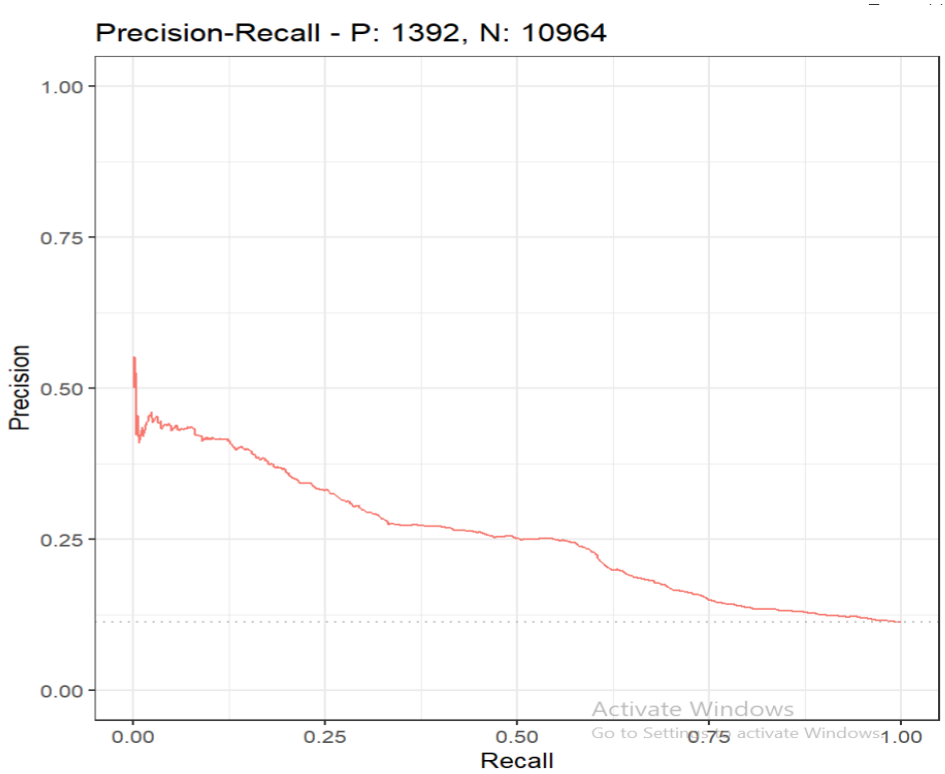
TPR is, as expected, exactly the same between both models. On the other hand, since the number of negatives largely dominates that of positives, the difference of FPR between both models (0.00891–0.00081 = 0.0081) is lost in the sense that it can be rounded to almost 0. In other words, a large change in the number of false positives resulted in a tiny change in the FPR and thereby, ROC is unable to reflect the superior performance of Model B in the context that true negatives are not relevant to the problem.

Precision-Recall curve is more informative:

In contrast, the Precision-Recall (PR) curve is specifically tailored for the detection of rare events and is the metric that should be used when the positive class is of more interest than the negative one. Because precision and recall don't consider true negatives, the PR curve is not affected by the data imbalance. Back to the example above:

- Model A: recall = TPR = 0.9 and precision = 9/900 = 0.01
- Model B: recall = TPR = 0.9 and precision = 9/90 = 0.1

Clearly, PR analysis is more informative compared to the ROC analysis above.



Area under curve of precision-recall curve is 0.439

This curve gives approx 75% recall at 12.5% precision, i.e. to get all actual "yes" responses we have to take a high cost of more time and money.

We will compare this with PR curves of other models.

Gains and Lift Charts:

- Gain or lift is a measure of the effectiveness of a classification model calculated as the ratio between the results obtained with and without the model.
- Gain and lift charts are visual aids for evaluating performance of classification models. However, In contrast to the confusion matrix on the whole population , gain or lift chart evaluates model performance in a portion of the population.

- The greater the area between the lift curve and the baseline, the better the model.

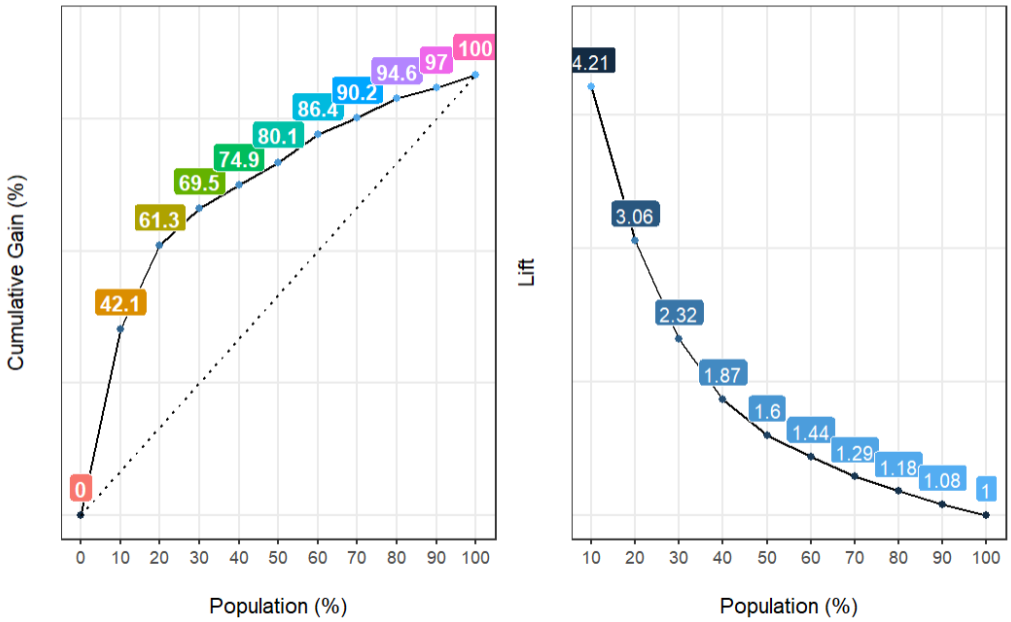
Cumulative Gains Chart:

- The y-axis shows the percentage of positive responses.This is a percentage of total possible positive responses(1392 as the overall response rate shows)
- The x-axis shows the percentage of customers contacted, which is a fraction of 12,356 total customers.
- Baseline(overall response rate): If we contact X% of customers, then we will receive X% of the total positive responses.
- gain implies how much positive cases are catched if the cut point to define the positive class is set to the column "Score Point"

Percentage population	Total Customers Contacted	Positive Responses	Gain($= \frac{\text{pos responses}}{\text{sum(pos responses)}}$)	Score. point
10	1236	586	42.1	0.735
20	2471	853	61.3	0.501
30	3707	967	69.5	0.445
40	4942	1043	74.9	0.344
50	6178	1115	80.1	0.327
60	7414	1203	86.4	0.311
70	8649	1255	90.2	0.290
80	9885	1317	94.6	0.249
90	11120	1350	97	0.215
100	12356	1392	100.00	0.105

- *Interpretation*:

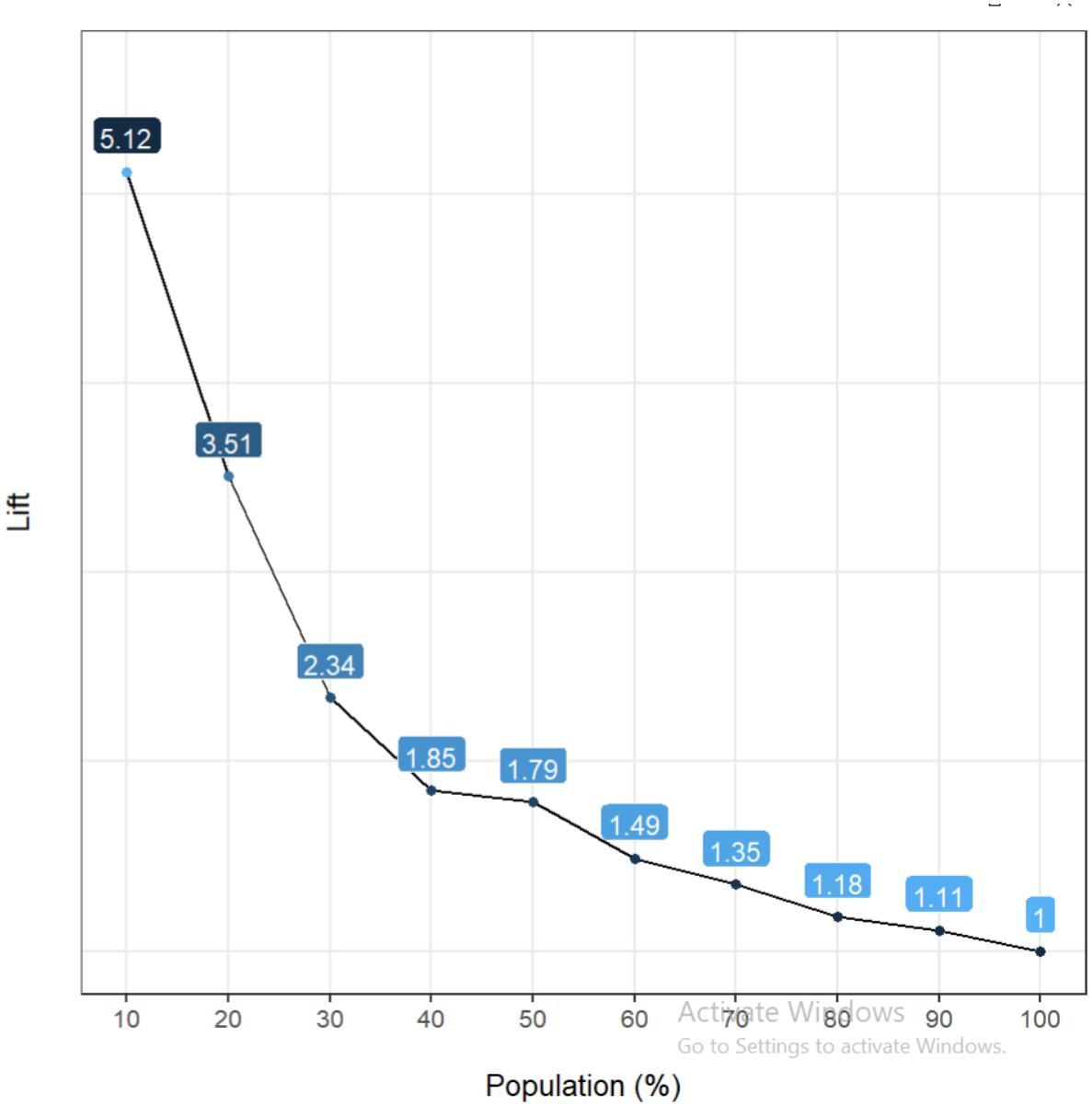
First row telling, the first 10% of the population, ordered by score, collects 51.22% of total positive cases. For example, if bank performs top 10% of most potential customers, then they are going to receive 51.22% positive responses.



- We will examine, what are the important features that leading top 10% population to get 42.10% positive responses by DALEX. It'll be easy to target those customers having these specific characteristics.
- Lift Curve:

Using the predictions of the response model, calculating the percentage of positive responses for the percent of customers contacted and map these points to create the lift curve.

- Calculating the points on the lift curve by determining the ratio between the result predicted by our model and the result using no model.
- For contacting 10% of customers, using no model we should get 10% of the positive responders and using the given model we should get 51.2% of responders.The y-value of the lift curve at 10% is 51.2/10=5.12



- The lift chart shows how much more likely we are to receive responses than if we contact a random sample of customers.

Evaluation:

We can assess the value of predicted model by using the model to score a set of customers and then contacting them in this order.

The actual response rates are recorded for each cut-off point, such as the first 10% contacted, the first 20% contacted etc.

We create cumulative gains and lift charts using the actual response rates to see how much the predicted model would have helped this situation.

The information can be used to determine wheather we should use this model or one similar to it.

Now, if we want to compare several models, a quick metric is to see if the gain at the beginning of the population(10-30%) is higher.

As a result, the model with higher gain at the beginning will have captured more information from the data.

Now, we want to target top 10% potential customers and want to see which top predictors are responsible for predicting their highest probability of successs. (using LIME function)

Interactive Studio for Lasso-logistic model:

- https://rpubs.com/Saikat173/914208

7.2 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 [18]. This algorithm generates binary DTs. The splitting is based on an impurity measure called the Gini index:

$$\text{Gini Index} = 1 - \sum_c \{p_c(h)\}^2$$

Where $\{p_c(t)$ is the frequency of observations of class c = 0, 1 at a node h [19]. 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 [18].

Terminologies:

Root Nodes – It is the node present at the beginning of a decision tree from this node the population starts dividing according to various features.

Decision Nodes – the nodes we get after splitting the root nodes are called Decision Node.

Leaf Nodes – the nodes where further splitting is not possible are called leaf nodes or terminal nodes.

Sub-tree – just like a small portion of a graph is called sub-graph similarly a sub-section of this decision tree is called sub-tree.

Pruning – is nothing but cutting down some nodes to stop overfitting.

Questions: How to select root node based on what criterion?

To answer this, we need to define Entropy; which is uncertainty or measure of disorder.

$$E(S) = -p_+ \log_2 p_+ - p_- \log_2 p_-$$

Where, p_+ = probability of positive class
p_- = probability of negative class
S = subset of training data

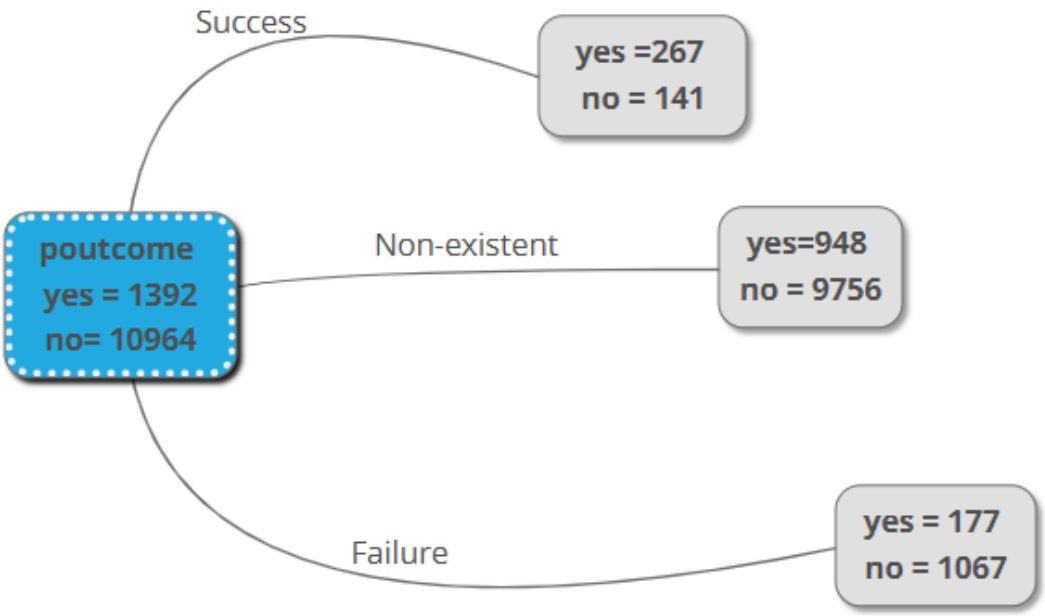
Among all p predictors , there is m_1 categorical and m_2 continuous predictors. Let's say, i[th] categorical predictor has a_i levels ; i=1(1) m_1 , $m_1 + m_2 = p$

Now, within Categorical Variables:

To find answer of above question, we have to measure with Information Gain;

which is defined as, $E(Y) - E(Y|X)$

Now, let's understand this for selected 2 features "poutcome" , "loan".



Now, calculating entropy

E(parent) = $-\left(\frac{1392}{12356}\right)\log_2\left(\frac{1392}{12356}\right) - \left(\frac{10964}{12356}\right)\log_2\left(\frac{10964}{12356}\right) = \ 0.5079$

E(parent |success) = $-\left(\frac{267}{408}\right)\log_2\left(\frac{267}{408}\right) - \left(\frac{141}{408}\right)\log_2\left(\frac{141}{408}\right) = \ 0.9301$

E(parent |non-existent) = $-\left(\frac{948}{10704}\right)\log_2\left(\frac{948}{10704}\right) - \left(\frac{9756}{10704}\right)\log_2\left(\frac{9756}{10704}\right) = 0.4312$

E(parent |failure) =$-\left(\frac{177}{1244}\right)\log_2\left(\frac{177}{1244}\right) - \left(\frac{1067}{1244}\right)\log_2\left(\frac{1067}{1244}\right) = 0.5902$

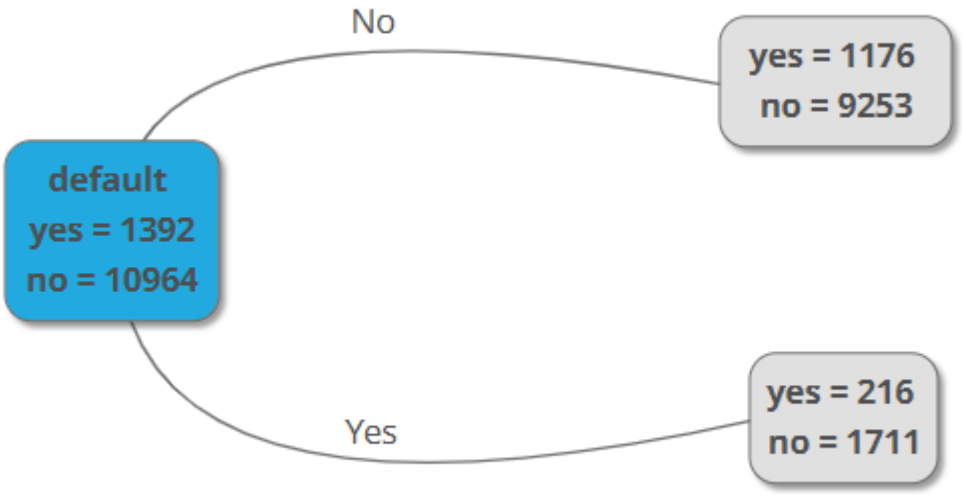
To see weighted average entropy of each node,

E(parent|poutcome) =

$\left(\frac{408}{12356}\right) * 0.9301 + \left(\frac{10704}{12356}\right) * 0.4312 + \left(\frac{1244}{12356}\right) * 0.5902 = 0.4638$

For poutcome, Information Gain = 0.5079-0.4638 = 0.0441

we can say that the entropy(impurity) of the dataset will decrease by 0.04 if we make “poutcome” as our root node.



Similarly, we can compute E(Parent|default) = 0.5078

So, for default , Information Gain = 0.0001

We now see that the “poutcome” feature gives more reduction which is 0.04 more than the “loan” feature. Hence we will select the feature which has the highest information gain and then split the node based on that feature.

In this example “poutcome” will be our root node and we’ll do the same for sub-nodes.

Now, if along with poutcome and loan this two categorical features, one continuous predictor “nr.employed”. then, to find based on which values to split, it considers all possible splits, ordering the samples by the feature value and calculating the information gain (or other criterion) improvement for each split. It then chooses the cut-off value to achieve the split in the ordered samples.

In practice, some optimization occurs, ignoring constant features, using approximated criterion improvements, etc.

Using this technique, we get the following decision tree for our dataset with 16 features.

For our given problem, it is to be a Classification Tree as it is used to predict a qualitative response.

- For classification tree, we predict each observation belongs to the most commonly occurring class of training observation in the region to which it belongs.
- We are interested also in the class proportions among the training observations that fall into that region.

Algorithm of building Classification tree:

1. Use recursive binary splitting to grow a large tree on the training data, stopping only when each terminal node has fewer than some minimum number of observations.
2. Apply cost complexity pruning to the large tree in order to obtain a sequence of best subtrees, as a function of α.
3. Use K-fold cross-validation to choose α. That is, divide the training observations into K folds. For each k = 1,...,K:

(a) Repeat Steps 1 and 2 on all but the kth fold of the training data.

(b) Evaluate the miss-classification error on the data in the left-out kth fold, as a function of α. Average the results for each value of α, and pick α to minimize the average error.

4. Return the subtree from Step 2 that corresponds to the chosen value of α.

For each value of α there corresponds a subtree $T \subset T_0$ such that

$$\sum_{m=1}^{|T|} \sum_{i:x_i \in R_m} (y_i \neq \hat{y}_{R_m}) + \alpha|T| \text{ ---------(1)}$$

is as small as possible. Here, $|T|$ indicates the number of terminal nodes of the tree T, R_m is the rectangle (i.e. the subset of predictor space)

corresponding to the m^{th} terminal node, and $\hat{y}_{R_m}$ is the most commonly occurring class of training observations in the region to which it belongs associated with R_m.

$\sum_{i:x_i \in R_m}(y_i \neq \hat{y}_{R_m})$ denoting misclassification error rate of m^{th} terminal node.

classification error is not sufficiently sensitive for tree-growing, and in practice two other measures are preferable.

The Gini index is defined by = $\sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$

Cross-entropy is defined by = $-\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}$

$\hat{p}_{mk}$ represents the proportion of training observations in the m^{th} region that are from k^{th} class.

- Gini Index, entropy will take on a value near zero if the $\hat{p}_{mk}$ are all near zero or near one. Therefore, like the Gini index, the entropy will take on a small value if the m^{th} node is pure.

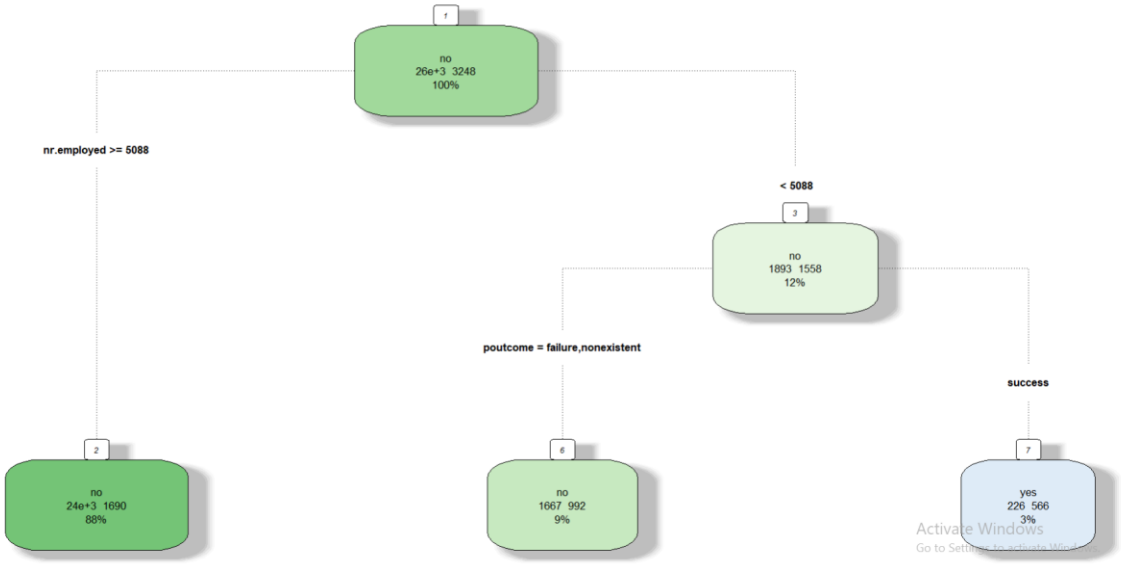
So, instead of mis-classification error rate we can use Gini Index, Entropy for tree pruning.

Fitting Classification Trees:

For our data,

There are response variable "y" for 28,832 clients presented term deposit subscription. Yes indicates term deposit subscribed and No indicates term deposit not subscribed.

- Here splitting for each node purity , we used Gini Index(default; but we will change splitting criterion when pruning and see whether giving best split or not).



Observation:

Start at root node(top of the graph)

- First splitting by the variable “nr.employed” and criterion is nr.employed $\geq$ 5088 ; this variable giving best split which is determined by Gini Index.
- At the top node ;
  firstly it’s showing “no” which is determined by majority of category of the response variable present at initial splitting.
  Secondly, it’s showing number of rows of “no” and “yes”.
  Thirdly, it’s showing it’s taking full 28,832 rows when initial splitting. That’s why 100% showing.
- Node 2 showing when nr.employed $\geq$ 5088 satisfied, it producing left child node and also this node’s majority vote is towards “no” class. Additionally, in this terminal node they use 88% rows of full dataset.
- Node 3 showing when nr.employed $<$ 5088 satisfied, it producing right child node and also this node’s majority vote is towards “no” class.
  # of “yes” response = 1558
  # of “no” response = 1893
  Additionally, in this node they use only 12% rows of full dataset.

- Then, again it's splitting by the variable "poutcome" = failure, non-existent as poutcome having these values giving second statistically significant variable for splitting measured by Gini Index.

Node 6 having majority vote towards "no" class and using 9% of full dataset.

Node 7 having majority vote towards "yes" class and using only 3% of full dataset.

Note:

1. When nr.employed < 5088 and poutcome = success is satisfied;
   Only 3% people have been targeted and getting success.
   So, for next campaign, employees should target this segment when in financial institution, number of employees(quarterly) < 5088 and those customers who said previous campaign was success.(i.e. poutcome = success)

Output:

```
Classification tree

Variables actually used in tree construction:

[1] "nr.employed" "poutcome"

Number of terminal nodes:  3

Residual mean deviance:  0.5676 = 16360 / 28829

Misclassification error rate: 0.1009 = 2908 / 28832
```

Above summary output giving variables that are used as internal nodes, number of terminal nodes and (training) error rate.

Training error rate is 10% ; the deviance reported in the output of summary() is given by

$$-2\sum_m \sum_k n_{mk} log\hat{p}_{mk} \quad \text{----------(2)}$$

where n_{mk} is the number of observations in the m^{th} terminal node that belong to the k^{th} class. A small deviance indicates a tree that provides a good fit to the (training) data. The residual mean deviance reported is simply the deviance divided by $n - |T_0|$, which in this case is $28832 - 3 = 28829$.

In order to properly evaluate the performance of a classification tree on these data, we must estimate the test error rather than simply computing the training error.

	Actual	
Predicted	No	Yes
No	10,877	1154
yes	87	238

Thus, miss-class error = 10.09%

Recall = 17.10%

Precision = 26.77%

From the point of view of stated reasons in Page-18, this model isn't good and need for pruning.

Pruning:

Step 1:

Building two decision trees using splitting criterion = Gini index or Information value and see whether it's giving better F_3 score (accuracy) on test dataset.

But, here both splitting criterion giving same result. So, we can proceed any of two model.

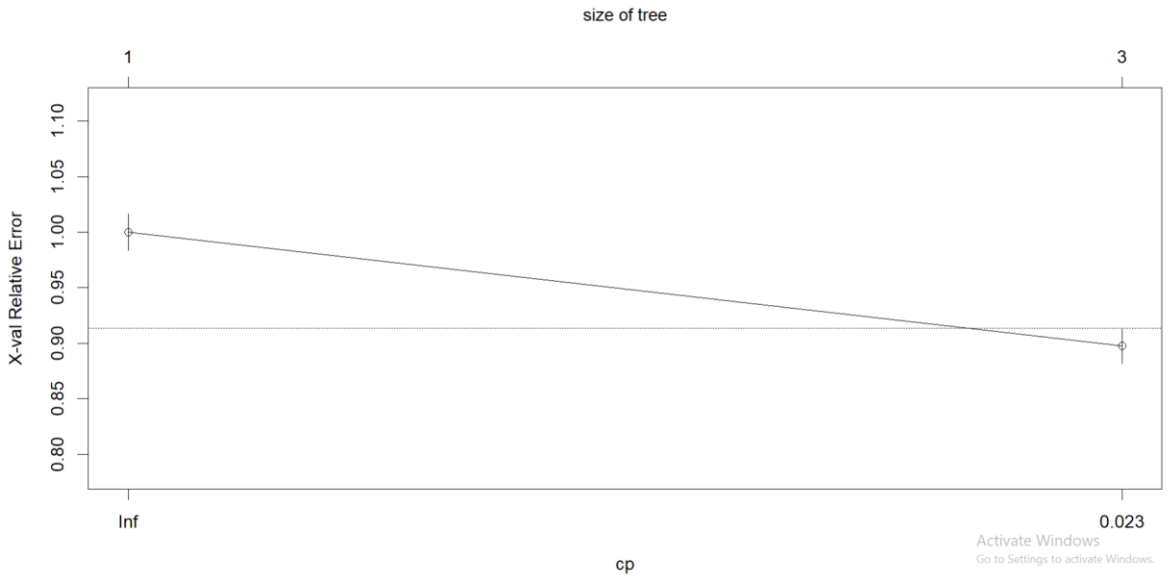
Step2:

We finding that cost complexity parameter(α) value for which (1) is giving small .

CP	rel error
0.0523399	1.0000000
0.0100000	0.8953202

Here, taking cp=0.01 for which the line cuts the horizontal line and building model again considering this cp.

But also using this cp value we get same model, same F_3 score (accuracy) on test dataset and same tree plot as before.



Step 3:

Next, we search for hyper parameter tuning through grid search method.

Three hyper-parameters :

i) min-split: the minimum number of observations that must exist in a node in order for a split to be attempted.

ii) Max_depth: it indicates how deep the decision tree. The deeper the tree, the more splits it has & it captures more information about the data. Generally, maximum value of max_depth implies overfit of the data.

iii) Cost Complexity parameter: The tuning parameter α controls a trade-off between the subtree's complexity and its fit to the training data. When α = 0, then the subtree T will simply equal T_0, because then (1) just measures the training error. However, as α increases, there is a price to pay for having a tree with many terminal nodes, and so the quantity (1) will tend to be minimized for a smaller subtree.

Now, we choose min-split a sequence taking from 1 to 1500 increasing by 75, max_depth from 1 to 5 increasing by 1 and cp from 0 to 0.3 increasing by 0.05.

Now, we get 660 tree model fit and for each tree we perform testing on the test dataset. Extracting best model and prediction on test dataset, we get following:

	Actual	
Predicted	No	Yes
No	8564	558
Yes	2400	834

Mis-class error = 0.2394

Recall = 59.91%

Precision = 25.79%

Thus, than unpruned decision tree, pruned decision tree giving better recall values and precision also. But compared to lasso-logistic not performing better than that. So, one decision tree failed to capture important scenario . So, we need to do modelling for Random-Forest.

Disadvantage:

(i)Decision tree suffers from high variance.

Bootstrap is a general-purpose procedure for reducing the variance of a statistical learning method.

7.3 Bagging

Bootstrap, aggregation or bagging is a general-purpose procedure for reducing the variance of a statistical learning method. Hence a natural way to reduce the variance and increase the test set accuracy of a statistical learning method is to take many training sets from the population, build a separate prediction model using each training set, and average the resulting predictions. In other words, we could calculate $\hat{f}^1(x), \hat{f}^2(x), \ldots., \hat{f}^B(x)$; predictions from the individual base learners using B separate training sets, and average them in order to obtain a single low-variance statistical learning model, given by

$$\hat{f}_{avg}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^b(x) ;$$

x is the record for which we want to generate a prediction

Of course, this is not practical because we generally do not have access to multiple training sets. Instead, we can bootstrap, by taking repeated samples from the (single) training data set. In this approach we generate B different bootstrapped training data sets. We then train our method on the b^{th}

bootstrapped training set in order to get $\hat{f}^{*b}(x)$, and finally average all the predictions, to obtain

$$\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{*b}(x)$$

This is called bagging.

.Here, we fitting classification tree, where number of variables tried at each split, m = p.

```
> model_bag = randomForest(x=train[,-20],y=train[,20],
+                          xtest=test[,-20],ytest= test[,20],
+                          mtry=19,ntrees=500)
> model_bag

Call:
 randomForest(x = train[, -20], y = train[, 20], xtest = test[,      -20], ytest = test[,
 20], mtry = 19, ntrees = 500)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 19

        OOB estimate of  error rate: 10.84%
Confusion matrix:
    1     0 class.error
1 983  2265  0.69735222
0 861 24723  0.03365385
               Test set error rate: 11.01%
Confusion matrix:
    1     0 class.error
1 393   999  0.71767241
0 361 10603  0.03292594
> |
```

Interpretation:

- This is a classification problem and total 19 variables have been tried at each split of creating node (Here is one disadvantage).
- Here, for i^{th} row, those trees are taken where i^{th} row isn't taken in bootstrap sample for training purpose .So, for those trees for this row it gives prediction and by taking B/3 major voting class is predicted. So, here 10.84% sample of training data is giving wrong prediction.
- Within class 1(actual) , 30.27% is correctly predicted and within class 0(actual) 96.64% is correctly predicted.

Here, Out of Bag (OOB) error is the estimation of the test error without performing Cross-validation or validation set approach.

Reason: Recall that the key to bagging is that trees are repeatedly fit to bootstrapped subsets of the observations. One can show that on average, each bagged tree makes use of around two-thirds of the observations. The

remaining one-third of the observations not used to fit a given bagged tree are referred to as the out-of-bag (OOB) observations. We can predict the response for the i^{th} observation using each of the trees in which that observation was OOB. This will yield around $\frac{B}{3}$ predictions for the ith observation. In order to obtain a single prediction for the i^{th} observation, we can average these predicted responses (if regression is the goal) or can take a majority vote (if classification is the goal). This leads to a single OOB prediction for the i^{th} observation. An OOB prediction can be obtained in this way for each of the n observations, from which the overall OOB classification error (for a classification problem) can be computed. The resulting OOB error is a valid estimate of the test error for the bagged model, since the response for each observation is predicted using only the trees that were not fit using that observation.

Now, prediction on test data using this model; we get following:

Miss-classification error = $\frac{999+361}{12356} = 11.01\%$

N.T: Miss-classification error decreases rapidly compared to decision tree and also gives better prediction on test data. Also, OOB error(=10.85%) is close to actual test error (=11.01%). This is because decision tree suffers from high variance but for bagging taking average of prediction of each tree(once trees are fully grown) it reduces variance.

Disadvantage of Bagging:

- The need for random forest surfaced after discovering that the bagging algorithm results in correlated trees when faced with a data set having strong predictors. Unfortunately, averaging several highly correlated trees doesn't lead to a large reduction in variance compared to random forest.
- Let's say a data set has a **very strong predictor**, along with other **moderately** strong predictors. In bagging, a tree grown every time would consider the **very strong predictor** at its root node, thereby resulting in trees similar to each other.
- The **main difference between** random forest and bagging is that random forest considers only a subset of predictors at a split. This results in trees with different predictors at top split, thereby resulting in **decorrelated trees** and more reliable average output. That's why we say random forest is robust to correlated predictors.

7.4 Random-Forest

Random Forest is non-parametric statistical technique and depends on decision trees. It utilizes the idea of aggregation since they need to require few conditions on the model that is produced using the observed data [20]. Random forest algorithm is efficient and work for both regression as well as classification problems. RF is set of the classification trees which is also called decision tree. It comprises with leaf nodes and other explanatory variables which are exist on the other intermediate node. By gathering the leaf nodes and explanatory variables that refers to be class variables which is also called the decision variables or predictor variables [21]. The principle on which RF works is to join numerous binary decision trees on the base of various samples generated by bootstrap which can be gathered from the learning sample or by selecting the each node randomly as a subset of the independent variables [22]. These are some properties of random forest according to [21].

Building many **decision trees** results in a **forest**. A random forest works the following way:

- First, it uses the Bootstrap (Aggregating) algorithm to create random samples. Given a data set D1 (n rows and p columns), it creates a new dataset (D2) by sampling $\frac{2}{3}$ cases at random with replacement from the original data. About 1/3 of the rows from D1 are left out, known as Out of Bag (OOB) samples.
- Then, the model trains on D2. OOB sample is used to determine unbiased estimate of the error.
- Out of p columns, P << p columns are selected at each node in the data set. The P columns are selected at random. Usually, the default choice of P is $\sqrt{p}$ for classification tree. Based on Information gain, then choose root node for each tree.
- Unlike a tree, no pruning takes place in random forest; i.e, each tree is grown fully. In decision trees, pruning is a method to avoid overfitting. Pruning means selecting a subtree that leads to the lowest test error rate. We can use cross validation to determine the test error rate of a subtree.
- Several trees are grown and the final prediction is obtained by major voting.

First, we fitted random-forest with m=$\sqrt{p}$ (default; later we will check whether changing m improves OOB error or not)

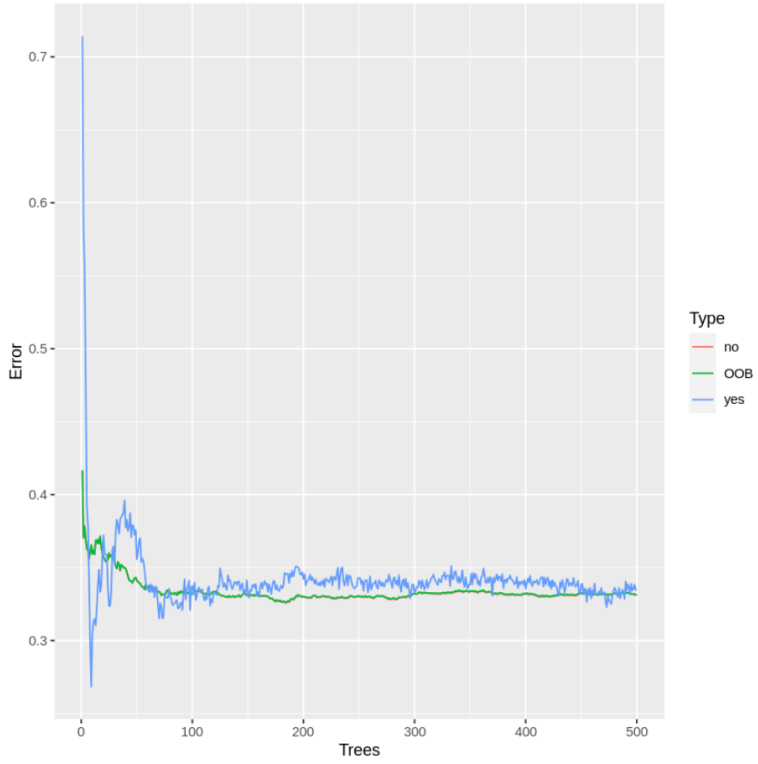
```
Call:
 randomForest(x = train[, -17], y = train[, 17], xtest = test[,      -17], ytest = test[, 17], ntree = 500, weights = model_weights)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 4

        OOB estimate of  error rate: 33.15%
Confusion matrix:
       no  yes class.error
no  17109 8475   0.3312617
yes   524 1041   0.3348243
                Test set error rate: 31.81%
Confusion matrix:
      no  yes class.error
no  7403 3561   0.3247902
yes  369 1023   0.2650862
```

No of variables tried at each split m = $\sqrt{20}$= 4.47($\approx$ 4)

Also, Number of trees to built = 500 (default)

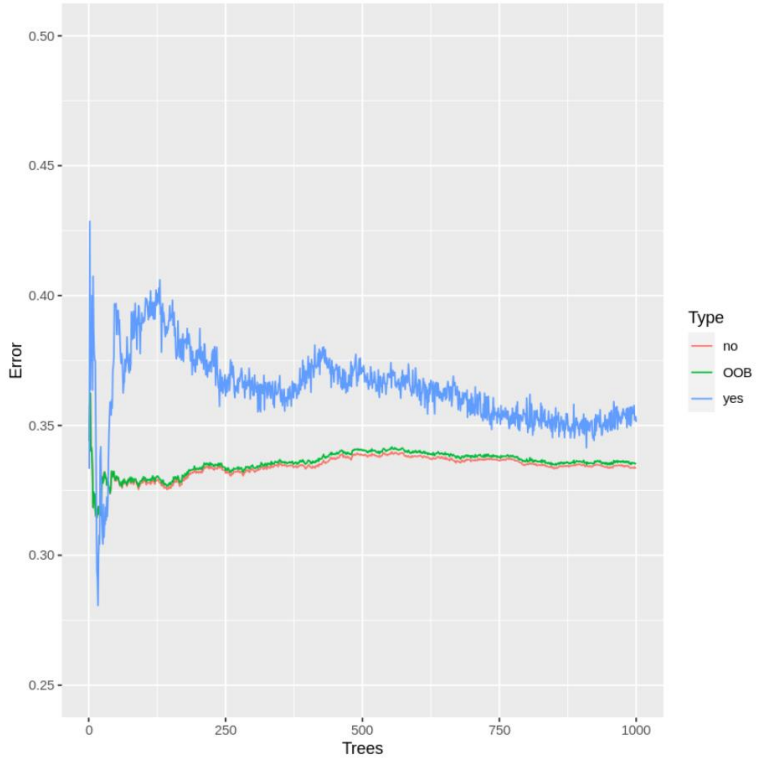
Here, Error rate for ‘yes’ class is 33.48% which is less than that for Bagged trees. But, within class 1 model is wrongly predicting much as class 0 ; thus class error is high. From point of view of importance of recall we have to reduce this by changing mtry (=m).

For 500 trees,



Till 500 trees, error rate of ‘yes’ class not stable , have to check whether improving for 1000 trees or not and other errors becoming flat but up.

Checking whether error rate of class “1” decreases more with increasing number of trees.



So, there is improvement of errors of all from increasing 500 to 1000 trees. So, Changing number of trees from 500 to 1000.

Now, trying different values of mtry to get min class 1 error (as recall is imp)

0.0971330765939238 · 0.200501253132832 · 0.275732217573222 · 0.352195945945946 · 0.396631578947368 · 0.415891800507185 · 0.44645107097858 · 0.461085401766933 · 0.447231833910035 · 0.465606190885641 · 0.458404074702886 · 0.463197969543147 · 0.454545454545455 · 0.467297762478485 · 0.470155709342561 · 0.45964316057774 · 0 · 0 · 0 · 0

Here, we get mtry=1 giving min class1 error.

Now, trying different node size values to get min class 1 error.

```
set.seed(345)
node_size = seq(1,1000,75)
err_cls_1 = vector(length=length(node_size))
for(i in 1:length(node_size)){
  temp.model = randomForest(train$y~.,data=train,mtry=mtry_best,ntree=1000,nodesize=node_size[i],
                            importance=TRUE,weights = model_weights)
  err_cls_1[i] = temp.model$confusion[2,3]
}
err_cls_1
```

0.0917235494880546 · 0.0748299319727891 · 0.0743589743589743 · 0.0694087403598972 · 0.0705236486486487 · 0.0745131244707875 · 0.0659386812263755 · 0.0518423307626392 · 0.0622649394066026 · 0.0665807560137457 · 0.0605670103092784 · 0.0585756426464391 · 0.0526762956669499 · 0.0632644453816955

```
node_size_best = node_size[which.min(err_cls_1)]
node_size_best
```

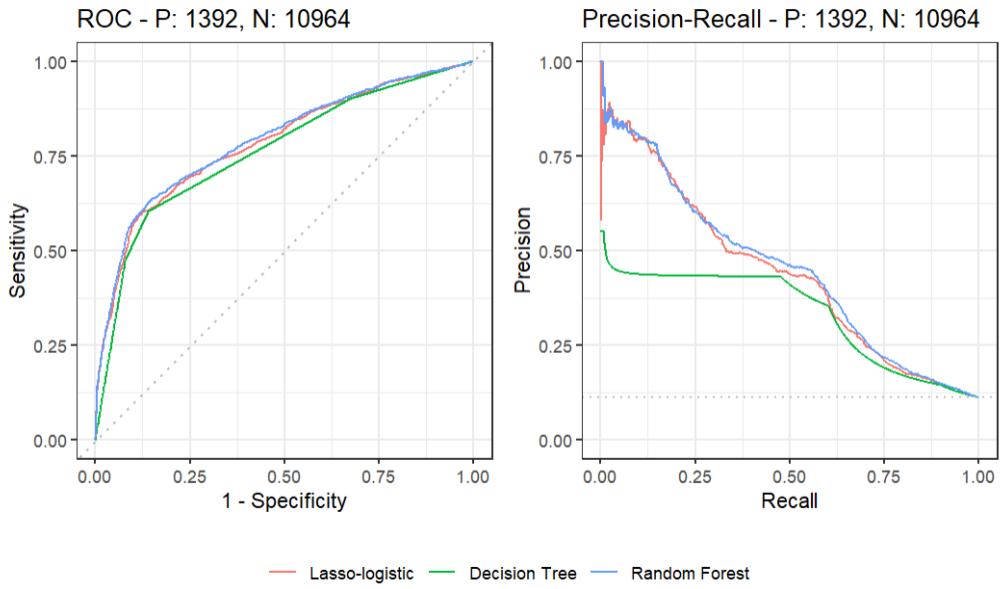
526

Now, getting node-size=526 as here, we finding that node-size for which error of class 1 is minimum when building total 1000 trees, mtry=1.

```
Call:
 randomForest(x = train[, -17], y = train[, 17], xtest = test[,      -17], ytest = test[, 17], mtry = 1, weights = model_weights,
nodesize = 526, importance = TRUE, ntrees = 1000)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 1

        OOB estimate of  error rate: 77.45%
Confusion matrix:
      no   yes class.error
no  4666 20918  0.81762039
yes   91  1452  0.05897602
                Test set error rate: 72.43%
Confusion matrix:
      no  yes class.error
no  2072 8892  0.81101788
yes   58 1334  0.04166667
```

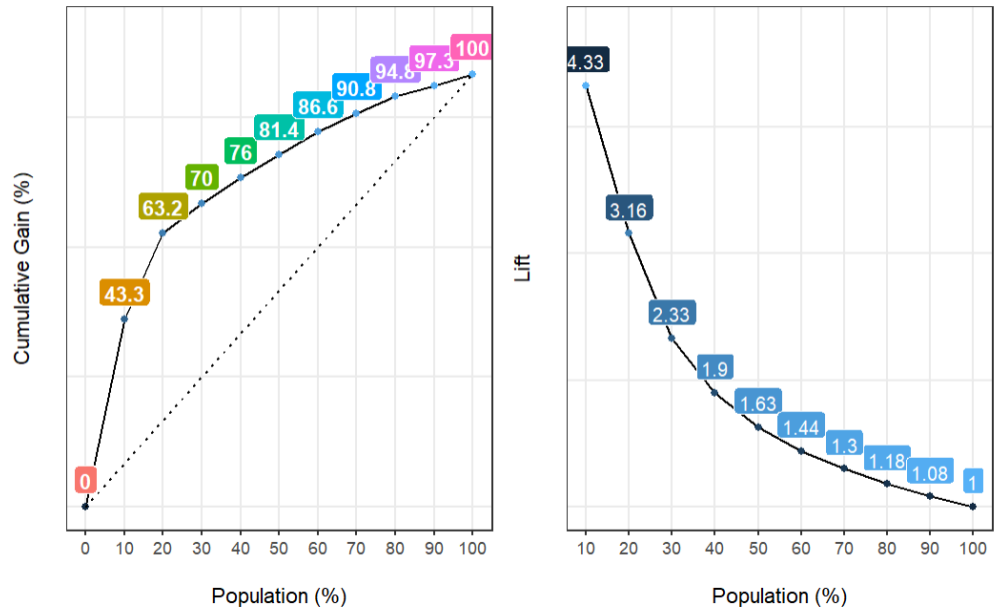
- Here, class1 error rate for train dataset is approx. 5.90% which is very small than that class1 error of Bagged ,decision tree and unpruned random forest model.
- Also, OOB error rate for train dataset is estimate of Test error rate.

Now, building Precision-Recall curve of Random-forest model along with PR curve of Lasso-logistic model, decision tree model.



	AUC
Lasso-logistic	0.4385
Decision Tree	0.3324
Random-Forest	0.4496

So, here among these three model Random-forest performing well. Also let's see for Random-Forest model, how Cumulative Gain chart behaving for targeting top percentage of clients.



Now, if we compare between lasso-logistic and Random-forest model by Cumulative Gain Chart, we see that

Sample size	Lasso-logistic	Random-Forest
10%	42.1%	43.3%
20%	61.3%	63.2%
30%	69.5%	70%
40%	74.9%	76%
50%	80.1%	81.4%
60%	86.4%	86.6%
70%	90.2%	90.8%
80%	94.6%	94.8%
90%	97%	97.3%
100%	100.00%	100%

Thus Random-Forest giving better result when targeting from 10%-50% clients. After that, percentage of getting 'yes' response for both model is almost same. Thus, targeting small percentage of clients and getting higher response is obtained from Random-Forest model.

Interactive Studio for Random-forest model:

https://rpubs.com/Saikat173/911280

7.5 XG-Boost

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 [24]. Using an ensemble of size K, one can combine the output $f_k(x_i)$ of individual trees [25]:

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

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

Where q is the structure of a CART tree [18], 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:

$$P[y_i = 1|x_i] = \frac{e^{\widehat{y_i}}}{1+e^{\widehat{y_i}}}$$

There are several ways to build the ensemble of trees, such as bagging [26] , gradient boosting [27] , and random forest [27]. Boosting and random forest generally outperform bagging methods [28] . A very promising gradient boosting technique is extreme gradient boosting or XG-Boost (XGB) [25]. XGB has won many classification competitions and performs well on a wide range of problems [25]. Therefore, I use an implementation of this method to built the tree ensemble. I build tree ensemble $\tau = (f_1, f_2, \dots, f_k)$ by minimizing the following objective function, which consists of a loss and regularization part:

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

$$\Omega(f_k) = \gamma T + \frac{1}{2}\delta \sum_{j=1}^{T} \omega_j^2 + \omega \sum_{j=1}^{|T|} |\omega_j| \qquad (2)$$

Where loss function is $l(y_i, \hat{y}_i)$. 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 [25] , I also add a L1 regulation term on the leaf weights, which can set some leaf weights to zero.

Equation (1) contains functions as parameters and is therefore difficult to train. Instead of solving (1) 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

$$\hat{y}_i^{\,k} = \hat{y}_i^{\,k-1} + f_k(x_i); \text{ for k=1,…,K}$$

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. Here we derive the exact splitting condition for objective function (1), following the same derivation steps as [25].

[25] Use more regularization techniques to prevent overfitting, which I also implement. Shrinkage, introduced by [13] , 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 [13]. 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 hyper parameters, XGB is difficult to tune. A different approach is sequential model-based or Bayesian optimization (SMBO). [29] Use SMBO for tuning hyper parameters of XGB and finds in a classification application that it outperforms a grid search strategy. With SMBO I try to find optimal hyper parameters θ^* that maximize the AUC over R = 20 different holdout schemes:

$$\theta^* = argmin_\theta \, y(\theta) = \sum_{r=1}^{R} AUC_r(\theta) \,,$$

$AUC_r(\theta)$ Is the AUC value of run r using hyper parameters θ. [30] describe the general procedure for SMBO. First one creates an initial design with F points: (θ_f, y_f) for f = 1,…., 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 [31] 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 [32] 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 [31] . Table 3 shows the hyper parameter ranges under which I search. These ranges are similar to the values in [29].

Hyper parameter	Values
K	{10,20,50,100,200,500,1000}
η	[0.001,0.20]
γ	[0,10]
Max_depth	{1,2,…..,12}
Min_child_weight	{0,1,2,3,4}
Subsample	[0.8,1]
Colsample_bytree	[0.3,1]
δ	[0,10]
ω	[0,5]

- Parameter for tree booster:

Eta [default 0.3: learning rate]

Scale the contribution of each tree by a factor 0 < eta < 1. Used to prevent over-fitting by making the boosting process more conservative. Low eta value means more robust to overfitting.

Range : [0,1]

gamma [default = 0: min split loss]

Minimum loss reduction required to make a further partition on a leaf node of the tree.

Range : [0,∞)

Max_depth [default=6]

maximum depth of a tree

Range: [0,∞)

Min_child_weight [default=1]

minimum sum of instance weight (hessian) needed in a child. If the tree partition step results in a leaf node with the sum of instance weight less than min_child_weight , then the building process will give up further partitioning. The larger, the more conservative algorithm.

Range: [0,∞)

Subsample [default =1]

sub-sample ratio of the training instance. Setting it to 0.5 means that xg-boost randomly collected half of the data instances to grow trees and this will prevent over-fitting. It makes computation shorter (because less data to analyse).

Range: (0,1)

Colsample_bytree [default=1]

sub-sample ratio of columns when constructing each tree.

N-rounds: number of rounds of iteration.

After performing randomized search, we get following best hyper parameters:

```
Tuning parameter 'nrounds' was held constant at a value of 100
Tuning
 parameter 'colsample_bytree' was held constant at a value of 1
Tuning
 parameter 'subsample' was held constant at a value of 0.7
ROC was used to select the optimal model using the largest value.
The final values used for the model were nrounds = 100, max_depth = 3, eta =
 0.1, gamma = 0, colsample_bytree = 1, min_child_weight = 5 and subsample = 0.7.
```

Prediction on Test data

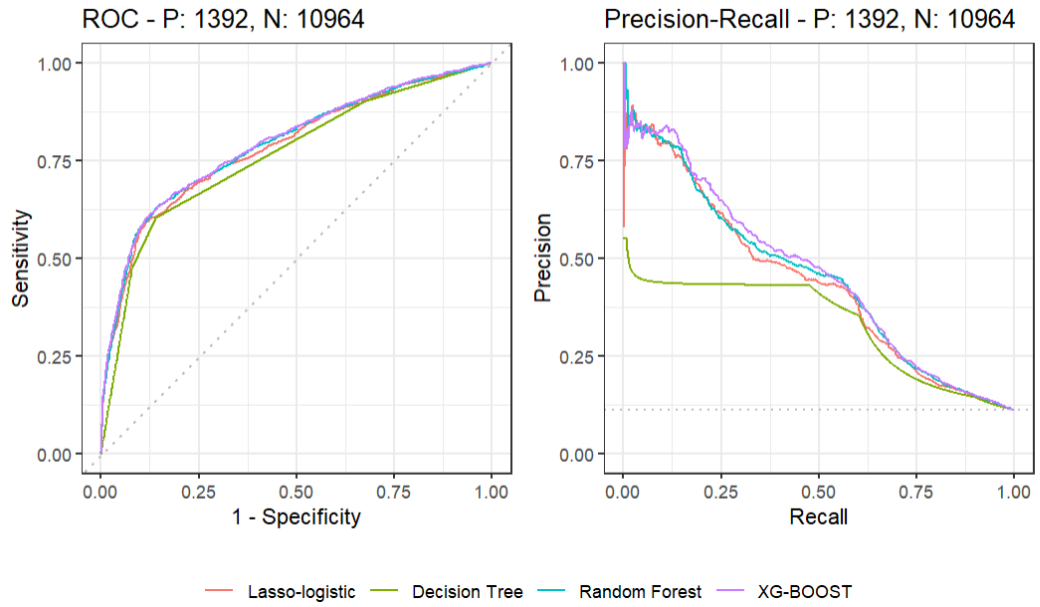
```
Confusion Matrix and Statistics

          Reference
Prediction  yes   no
       yes 1245 6861
       no   147 4103

               Accuracy : 0.4328
                 95% CI : (0.4241, 0.4416)
    No Information Rate : 0.8873
    P-Value [Acc > NIR] : 1

                  Kappa : 0.0865

 Mcnemar's Test P-Value : <0.0000000000000002

            Sensitivity : 0.8944
            Specificity : 0.3742
         Pos Pred Value : 0.1536
         Neg Pred Value : 0.9654
             Prevalence : 0.1127
         Detection Rate : 0.1008
   Detection Prevalence : 0.6560
      Balanced Accuracy : 0.6343

       'Positive' Class : yes
```

8. Comparison of Models

So, Comparison of Sensitivities, Positive Predictive values of these models we get:

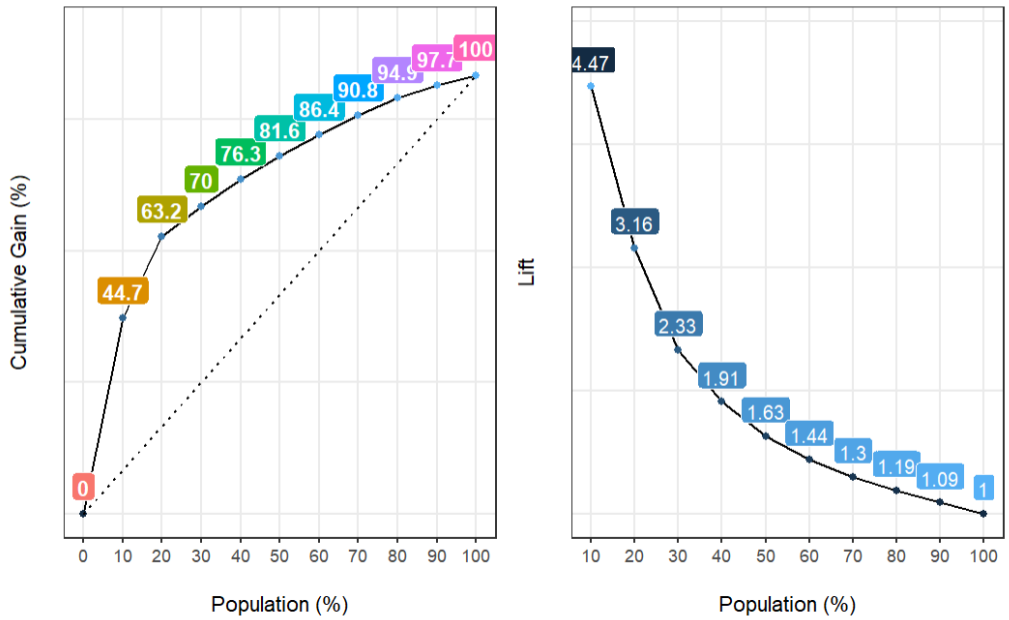
	Lasso-logistic	Decision tree	Xg-boost	Random-Forest
Sensitivity	0.6128	0.6034	0.8944	**0.9332**
Pos Predicted Values	0.3413	**0.3538**	0.1536	0.1364

In terms of Sensitivities and Positive Predicted Values Random-Forest and Decision trees performing well. But, to find optimum model we have to look at PR curve and compare their AUC values.



	AUC
Xg-Boost	0.4605
Random-Forest	0.4496
Lasso-logistic	0.4385
Decision Tree	0.3324

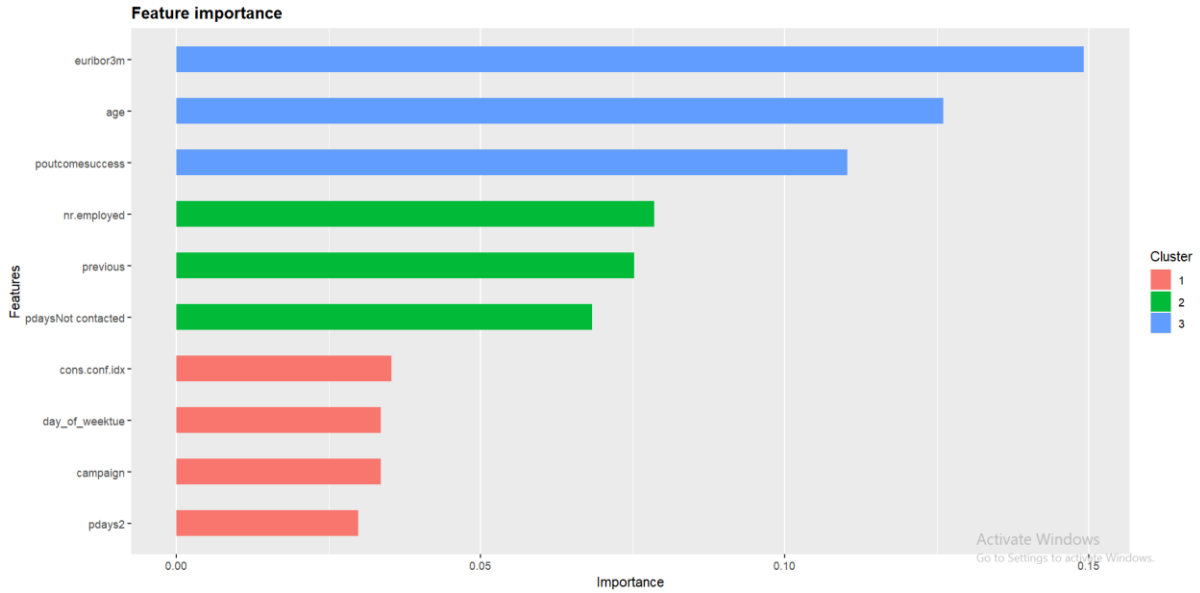
Thus, comparing AUC values we get Xg-Boost performing best for successfully predicting high ‘yes’ responses and now, comparing all these models to see how they able to extract high responses by targeting small percentage of population.



Sample size	Lasso-logistic	Random-Forest	Xg-Boost
10%	42.1%	43.3%	44.7%
20%	61.3%	63.2%	63.2%
30%	69.5%	70%	70%
40%	74.9%	76%	76.3%
50%	80.1%	81.4%	81.6%
60%	86.4%	86.6%	86.4%
70%	90.2%	90.8%	90.8%
80%	94.6%	94.8%	94.9%
90%	97%	97.3%	97.7%
100%	100%	100%	100%

So, we see that if bank targets top 10%-50% clients and in that case Xg-Boost model giving best results for getting yes response compared to all models. From 60%-100% Xg-Boost model close to Random-Forest model (except at 60%) but higher than Lasso-logistic model. So, it's better to target small percentage of client and getting higher Reponses (50% population targeting most favourable in consideration of time and money).

Variable Importance



So, As Xg-Boost is the best model for prediction and explaining the success, the above graph explaining variable's global importance. Here, Euribor 3 month rate is highest followed by Age and poutcome (if having value as yes) are the top 3 predictors of explaining the prediction by Xg-Boost model. Next, Number of Employed, previous and pdays (if having value as Not Contacted) are the next set of predictor's almost same percentage of explaining. Least four are consumer confidence index, weekday (having value as Tuesday) , campaign and pdays(having values in the class [9,18)).

References

[1] Migu´eis, 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.

[2] 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.

[3] 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.

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

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

[6] Moro, S., Laureano, R. & Cortez, P. (2011), Using data mining for bank direct marketing: An application of the crisp-dm methodology, in 'Proceedings of European Simulation and Modelling Conference-ESM'2011', Eurosis, pp. 117–121.

[7] Qi, G., Zhu, Z., Erqinhu, K., Chen, Y., Chai, Y. & Sun, J. (2018), 'Fault-diagnosis for reciprocating compressors using big data and machine learning', Simulation Modelling Practice and Theory 80, 104–127.

[8] Gil, D. & Johnsson, M. (2010), 'Using support vector machines in diagnoses of urological dysfunctions', Expert Systems with Applications 37(6), 4713–4718.

[9] Nogami, J., Nakasuka, S. & Tanabe, T. (1996), 'Real-time decision support for air traffic management, utilizing machine learning', Control Engineering Practice 4(8), 1129–1141.

[10] Cramer, S., Kampouridis, M., Freitas, A. A. & Alexandridis, A. K. (2017), 'An extensive evaluation of seven machine learning methods for rainfall prediction in weather derivatives', Expert Systems with Applications 85, 169–181.

[11] Wang, S. & Summers, R. M. (2012), 'Machine learning and radiology', Medical image analysis 16(5), 933–951.

[12] Miksovsky P, Matousek K, Kouba Z (2002) Data pre-processing support for data mining. In: IEEE Int. Conf. Syst. Man Cybern. IEEE, Yasmine Hammamet, Tunisia

[13] Friedman, J., Hastie, T. & Tibshirani, R. (2001), The elements of statistical learning, Vol. 1, Springer series in statistics New York.

[14] Izetta, J., Verdes, P. F. & Granitto, P. M. (2017), 'Improved multiclass feature selection via list combination', Expert Systems with Applications 88, 205–216.

[15] Trambaiolli, L. R., Biazoli, C. E., Balardin, J. B., Hoexter, M. Q. & Sato, J. R. (2017), 'The relevance of feature selection methods to the classification of obsessive-compulsive disorder based on volumetric measures', Journal of affective disorders 222, 49–56.

[16] Zhang, Z., Tian, Y., Bai, L., Xiahou, J. & Hancock, E. (2017), 'High-order covariate interacted lasso for feature selection', Pattern Recognition Letters 87, 139–146.

[17] Bardsley, W. E., Vetrova, V. & Liu, S. (2015), 'Toward creating simpler hydrological models: A lasso subset selection approach', Environmental Modelling & Software 72, 33–43.

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

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

[20] Genuer, R., Poggi, J.-M. & Tuleau-Malot, C. (2015), 'Vsurf: An r package for variable selection using random forests.', R Journal 7(2).

[21] Jaiswal, J. K. & Samikannu, R. (2017), Application of random forest algorithm on feature subset selection and classification and regression, in 'Computing and Communication Technologies (WCCCT), 2017 World Congress on', IEEE, pp. 65–68.

[22] Genuer, R., Poggi, J.-M. & Tuleau-Malot, C. (2010), 'Variable selection using random forests', Pattern Recognition Letters 31(14), 2225–2236.

[23] https://dpmartin42.github.io/posts/r/imbalanced-classes-part-2

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

[25] 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.

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

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

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

[29] Xia, Y., Liu, C., Li, Y., & Liu, N. (2017). A boosted decision tree approach using bayesian hyperparameter optimization for credit scoring. Expert Systems with Applications, 78 , 225–241.

[30] 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.

[31] 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 .

[32] 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.

Code

Lasso logistic regression

Required libraries

```
library(caret)
library(glmnet)
```

I. Model Building

```
#spliting data into train test 70/30
set.seed(1234)
trainIndex<-createDataPartition(data1$y,p=0.7,times = 1,list = F)
train<-data1[trainIndex,-c(5:7,11)]
test<-data1[-trainIndex,-c(5:7,11)]
levels(train$y)
x_full_data = model.matrix(y~.,data=data1[,-c(5:7,11)])[,-1]
# Dummy code categorical predictor variables of train dataset
x_train <- model.matrix(y~., data = train)[,-1]
# Convert the outcome (class) to a numerical variable
y_train <- train$y
# Dummy code categorical predictor variables of train dataset
x_test <- model.matrix(y~., data = test)[,-1]
# Convert the outcome (class) to a numerical variable
y_test <- test$y
# Create model weights (they sum to one)
model_weights <- ifelse(y_train == "yes",
                        1-(table(y_train)[2]/length(y_train)),
                        1-(table(y_train)[1]/length(y_train)))
# Find the best lambda using cross-validation
set.seed(123)
cv.lasso <- cv.glmnet(x_train, factor(y_train), alpha = 1, family = "binomial",nfolds =
20,lambda=10^seq(10,-2,length=100),type.measure = "deviance",keep = TRUE)
```

1. Shrinkage Path

```
windowsFonts(script=windowsFont("Script MT Bold"))
grid = 10^seq(1,-5,length=100)
lasso.mod = glmnet(x=x_train,y=y_train, family = "binomial",
                   alpha=1,lambda = grid)

#Extract the coef values and transform them in a long, tidy form suitable for ggplot2
beta=coef(lasso.mod)
tmp <- as.data.frame(as.matrix(beta))
tmp$coef <- row.names(tmp)
```

```
tmp <- reshape::melt(tmp, id = "coef")
tmp$variable <- as.numeric(gsub("s", "", tmp$variable))
tmp$lambda <- lasso.mod$lambda[tmp$variable+1] # extract the lambda values
tmp$norm <- apply(abs(beta[-1,]), 2, sum)[tmp$variable+1] # compute L1 norm
tmp$col = rep((c(rep("#B3B3B3",20),"#A6D854",rep("#B3B3B3",4),"#66C2A5",
        "#FC8D62",rep("#B3B3B3",10),
        "#FFD92F",rep("#B3B3B3",2),"#8DA0CB","#E5C494",rep("#B3B3B3",3),
        "#E78AC3")),100)
tmp$lty = rep((c(rep("1",20),"2",rep("1",4),"3",
            "4",rep("1",10),
            "5",rep("1",2),"6","1",rep("1",3), "2")),100)

#Plot with ggplot2
ggplot(tmp[tmp$coef != "(Intercept)",], aes(lambda, value, color = coef, linetype = coef)) +
  geom_line() +
  scale_x_log10() +
  xlab(expression(lambda)) +
  ylab("Standardized Coefficients") +
  guides(color = guide_legend(title = ""),
       linetype = guide_legend(title = "")) +
  theme_bw() +
  theme(legend.key.width = unit(3,"lines"))
lasso.ls = glmnet(x_train,y_train,lambda = 0,alpha = 1,family=binomial)
coef_ls = coef(lasso.ls)
beta_l1_norm_ls = sum(abs(coef_ls[-1]))
tmp$amount_shr = tmp$norm/beta_l1_norm_ls
#Plot with ggplot
# x11(width = 13/2.54, height = 9/2.54)
ggplot(tmp[tmp$coef != "(Intercept)",], aes(amount_shr, value, color = coef, linetype =
coef)) +
  geom_line() +
xlab(expression(norm(hat(beta))[lambda]/norm(hat(beta))[2],family="script")) +
  ylab("Standardized Coefficients") +
  guides(color = guide_legend(title = ""),
       linetype = guide_legend(title = "")) +
  theme_bw() + theme(legend.key.width = unit(3,"lines"))
```

2. Plot of Binomial deviance & best λ

```
plot(cv.lasso)
bestlam = cv.lasso$lambda.min
bestlam
```

3. Final model

```
model <- glmnet(x_train, y_train,weights=model_weights, alpha = 1, family =
"binomial",lambda = cv.lasso$lambda.min)
coef(model)
```

```
library(coefplot)
coefplot(model,sort = "magnitude",decreasing = "TRUE")
```

4. Prediction & Confusion matrix

```
probabilities <- model %>% predict(newx = x_test,type="response")
predicted.classes <- ifelse(probabilities >  0.5, "yes", "no")
tab = confusionMatrix(reference=factor(y_test),data=factor(predicted.classes),
positive = "yes")
tab
```

II. Evaluation Metrics

1. Variable Importance Measure

```
library(vip)
vip(model)
# Compute model-specific VI scores
imp_lasso=vi(model)
imp_lasso=imp_lasso[,c(1,2)]
imp_lasso[,2] = imp_lasso[,2]*100
imp_lasso
Precision Recall Curve
```

2. Precision-Recall curve

```
#train data

library(precrec)
preds_lasso_tr <- predict(model, newx = x_train, type="response")
preds_lasso_tr = as.data.frame(preds_lasso_tr,row.names = FALSE)
names(preds_lasso_tr)= "yes"
preds_lasso_tr$no = 1-preds_lasso_tr[,1]
preds_lasso_tr = preds_lasso_tr[c("no","yes")]
lasso.mm_tr = mmdata(scores = preds_lasso_tr,labels = train$y,
            modnames = c("Lasso-logistic"),dsids=c(1,2),posclass = "yes",
            mode="rocprc")
eval_tr = evalmod(lasso.mm_tr)
autoplot(eval_tr)
# on test data
preds_lasso_ts <- predict(model, newx = x_test, type="response")
preds_lasso_ts = as.data.frame(preds_lasso_ts,row.names = FALSE)
names(preds_lasso_ts)= "yes"
preds_lasso_ts$no = 1-preds_lasso_ts[,1]
preds_lasso_ts = preds_lasso_ts[c("no","yes")]
lasso.mm_ts = mmdata(scores = preds_lasso_ts,labels = test$y,
            modnames = c("Lasso-logistic"),dsids=c(1,2),posclass = "yes",
            mode="rocprc")
```

```
eval_ts = evalmod(lasso.mm_ts)
autoplot(eval_ts)
```

3. Cumulative Gain & lift chart

```
library(funModeling)
test$y  = ifelse(test$y=="no",0,1)
preds_lasso_ts$actual <- test$y
# Plot the gain and lift curve
gain_lift(data=preds_lasso_ts, score='yes', target='actual')
```

III. DALEX Dashboard

```
# Interactive dashboard DALEX for explaining how much changing each #predictor for one
unit change and also how important predictors are #contributing in predictions for high
probabilistic individual
library(modelStudio)
library(DALEX)
# create an explainer for the model
colnames(test)
y.test = as.numeric(y.test)
explainer <- DALEX::explain(model=model,
                            data = x_test, y = y.test,verbose= TRUE,precalculate = TRUE,
type = "classification",label = "Lasso-logistic")
preds_lasso_ts1 = preds_lasso_ts[order(preds_lasso_ts$yes,decreasing = TRUE),]
a = rownames(head(preds_lasso_ts1,10))
rownames(test[c(11731,11725,11798,12285,11786,11807,11775,12288,12308,11753),])
new_row =
x_full_data[c(39148,39135,39349,40942,39327,39390,39292,40969,41032,39227),]
rownames(new_row) =
c("high1","high2","high3","high4","high5","high6","high7","high8","high9","high10")
modelStudio(explainer,new_observation = new_row,N=200,B=5)
```

Random Forest

```
#Alternatively using weights
library(randomForest)
library(dplyr) # for data manipulation
library(caret) # for model-building
library(purrr) # for functional programming (map)
library(pROC) # for AUC calculations
library(PRROC) #for preision-recall curve
library(ggplot2)
```

I. Model Building

```
test_roc <- function(model, data) {
  roc(data$y, predict(model, data, type = "prob")[, "yes"])  }
# Create model weights (they sum to one)
model_weights <- ifelse(train$y == "yes", (1/table(train$y)[2]) * 1.5, (1/table(train$y)[1]) *
0.5)
#model building by randomForest function
model_rf = randomForest(x=train[,-17],y=train[,17],
                        xtest=test[,-17],ytest= test[,17],
                        ntree=500,weights = model_weights)
model_rf
```

1. Plot of Error rate

a. Optimal number of trees

```
oob.error.data = data.frame(
  Trees = rep(1:nrow(model_rf$err.rate),times=3),
  Type = rep(c("OOB","no","yes"),each = nrow(model_rf$err.rate)),
  Error = c(model_rf$err.rate[,"OOB"],
          model_rf$err.rate[,"no"],
          model_rf$err.rate[,"yes"]))
ggplot(data=oob.error.data, aes(x=Trees,y=Error))+geom_line(aes(color=Type))

model_rf1 = randomForest(x=train[,-17],y=train[,17],
                        xtest=test[,-17],ytest= test[,17],
                        ntree=1000,weights = model_weights)
model_rf1
oob.error.data1 = data.frame(
  Trees = rep(1:nrow(model_rf1$err.rate),times=3),
  Type = rep(c("OOB","no","yes"),each = nrow(model_rf1$err.rate)),
  Error = c(model_rf1$err.rate[,"OOB"],
          model_rf1$err.rate[,"no"],
          model_rf1$err.rate[,"yes"]))
ggplot(data=oob.error.data1, aes(x=Trees,y=Error))+geom_line(aes(color=Type)) + ylim(0.25
,0.50)
#stabilised, so we will take 1000 trees
```

2. Optimal number of mtry

```
set.seed(123)
oob.values = vector(length=20)
for(i in 1:16){
  temp.model = randomForest(train$y~.,data = train,mtry=i,ntree=1000,weights = model_we
ights)
```

```
  oob.values[i] = temp.model$err.rate[nrow(temp.model$err.rate),3]
}
oob.values

mtry_best = which.min(oob.values[1:16])
mtry_best
```

3. Optimal number of node-size

```
set.seed(345)
node_size = seq(1,1000,75)
err_cls_1 = vector(length=length(node_size))
for(i in 1:length(node_size)){
  temp.model = randomForest(train$y~.,data=train,mtry=mtry_best,ntree=1000,nodesize=n
ode_size[i],
                            importance=TRUE,weights = model_weights)
  err_cls_1[i] = temp.model$confusion[2,3]
}
err_cls_1
node_size_best = node_size[which.min(err_cls_1)]
node_size_best
model_rf_final = randomForest(x=train[,-17],y=train[,17],
                              xtest=test[,-17],ytest= test[,17],
                              mtry=1,ntrees=1000,weights = model_weights,nodesize = 526,
                              importance = TRUE)
model_rf_final

#Model building by caret package
# Set up control function for training
set.seed(123)
ctrl_rf <- trainControl(method ="cv",number = 10, #repeats = 5, summaryFunction =
prSummary,savePredictions = TRUE, classProbs = TRUE, verboseIter = F)
```

4. Final model

```
# Build weighted model
weighted_fit <- train(y~.,data=train, method = "ranger",weights =
model_weights,importance = 'impurity', metric = "AUC",trControl = ctrl_rf)
weighted_fit
weighted_fit %>%  test_roc(data = test) %>% auc()
```

5. Predictions & Consusion Matrix

```
#Make predictions to expose class labels
test$y = ifelse(test$y == "0","no","yes")
test$y = as.factor(test$y)
preds <- predict(weighted_fit, newdata=test, type="raw")
```

```
predicted <- cbind(data.frame(class_preds=preds), test)
# Create a confusion matrix object
cm <- caret::confusionMatrix(predicted$class_preds, predicted$y,positive="yes")
print(cm)
```

II. Evaluation Metrics

1. Variable Importance Plot

```
library(caret)
plot(varImp(weighted_fit))
```

2. Precision-Recall curve

```
#on train data
#Comparison(Precision-recall curve)
#Precision recall curve(logistic and decision tree)
library(precrec)
preds_lasso_tr <- predict(model, newx = x_train, type="response")
preds_lasso_tr = as.data.frame(preds_lasso_tr,row.names = FALSE)
names(preds_lasso_tr)= "yes"
preds_lasso_tr$no = 1-preds_lasso_tr[,1]
preds_lasso_tr = preds_lasso_tr[c("no","yes")]
preds_dt_tr <- predict(weighted_dt, newdata=train, type="prob")
preds_rf_tr = predict(weighted_fit,newdata=train,type="prob")
xgb.mm_tr = mmdata(scores =
join_scores(preds_lasso_tr[,2],preds_dt_tr[,2],preds_rf_tr[,2]),
            labels = train$y,
            modnames = c("Lasso-logistic","Decision Tree","Random Forest"),
            dsids= c(1,2,3),posclass = "yes",
            mode="rocprc")

eval_tr = evalmod(xgb.mm_tr)
autoplot(eval_tr)
#on test data
preds_lasso_ts <- predict(model, newx = x_test, type="response")
preds_lasso_ts = as.data.frame(preds_lasso_ts,row.names = FALSE)
names(preds_lasso_ts)= "yes"
preds_lasso_ts$no = 1-preds_lasso_ts[,1]
preds_lasso_ts = preds_lasso_ts[c("no","yes")]
preds_dt_ts <- predict(weighted_dt, newdata=test, type="prob")
preds_rf_ts <- predict(weighted_fit, newdata=test, type="prob")
xgb.mm_ts = mmdata(scores =
join_scores(preds_lasso_ts[,2],preds_dt_ts[,2],preds_rf_ts[,2])
            ,labels = test$y,
            modnames = c("Lasso-logistic","Decision Tree","Random Forest"),
            dsids= c(1,2,3),posclass = "yes",
            mode="rocprc")
```

```
eval_ts = evalmod(xgb.mm_ts)
autoplot(eval_ts)
```

3. Cumulative Gain & lift chart

```
#Cumulative Gain & lift chart
library(funModeling)
test$y  = ifelse(test$y=="no",0,1)
preds_rf_ts$actual <- test$y
# Plot the gain and lift curve
gain_lift(data=preds_rf_ts, score='yes', target='actual')
```

XG-BOOST

I. Model Building

```
library(snow)

library(parallel)

library(doParallel)

library(xgboost)

library(Ckmeans.1d.dp)

label=as.numeric(as.factor(train$y))

neg=sum(label==1)

pos=sum(label==2)

print(neg/pos)

test_roc <- function(model, data) {

  roc(data$y,

     predict(model, data, type = "prob")[, "yes"])

}

dtest=sparse.model.matrix(y~.-1, data = data.frame(test))

dtrain=sparse.model.matrix(y~.-1, data=data.frame(train))
```

1. Parameter Tuning

```
xgb.grid=expand.grid(nrounds=100, eta=seq(0.1, 1, 0.2), max_depth=c(3, 5, 10), gamma = 0,
subsample = 0.7, min_child_weight = c(1, 3, 5), colsample_bytree = 1)

myCl=makeCluster(detectCores()-1)
```

```
registerDoParallel(myCl)

xgb.control=trainControl(method = "cv",number = 10, verboseIter = TRUE, classProbs = TRUE,
allowParallel = TRUE,summaryFunction = prSummary)
```

2. Final Model

```
xgb.train = train(y~.,data=train, trControl = xgb.control, #tuneGrid = xgb.grid, metric="AUC",method
= "xgbTree",scale_pos_weight=(neg/pos)*3)
```

II. Evaluation Metrics

1. Variable Importance Plot

```
#Variable Importance Plot

library(caret)

plot(varImp(xgb.train))
```

2. Precision-Recall curve

```
#on train data

library(precrec)

preds_lasso_tr <- predict(model, newx = x_train, type="response")

preds_lasso_tr = as.data.frame(preds_lasso_tr,row.names = FALSE)

names(preds_lasso_tr)= "yes"

preds_lasso_tr$no = 1-preds_lasso_tr[,1]

preds_lasso_tr = preds_lasso_tr[c("no","yes")]

preds_dt_tr <- predict(weighted_dt, newdata=train, type="prob")

preds_rf_tr = predict(weighted_fit,newdata=train,type="prob")

preds_xgb_tr = predict(xgb.train,newdata=train,type="prob")

xgb.mm_tr = mmdata(scores =
join_scores(preds_lasso_tr[,2],preds_dt_tr[,2],preds_rf_tr[,2],preds_xgb_tr[,1]),

                   labels = train$y,

                   modnames = c("Lasso-logistic","Decision Tree","Random Forest","XG-BOOST"),

                   dsids= c(1,2,3,4),posclass = "yes",

                   mode="rocprc")


eval_tr = evalmod(xgb.mm_tr)

autoplot(eval_tr)

#on test data
```

```
preds_lasso_ts <- predict(model, newx = x_test, type="response")

preds_lasso_ts = as.data.frame(preds_lasso_ts,row.names = FALSE)

names(preds_lasso_ts)= "yes"

preds_lasso_ts$no = 1-preds_lasso_ts[,1]

preds_lasso_ts = preds_lasso_ts[c("no","yes")]

preds_dt_ts <- predict(weighted_dt, newdata=test, type="prob")

preds_rf_ts <- predict(weighted_fit, newdata=test, type="prob")

preds_xgb_ts <- predict(xgb.train, newdata=test, type="prob")

xgb.mm_ts = mmdata(scores =
join_scores(preds_lasso_ts[,2],preds_dt_ts[,2],preds_rf_ts[,2],preds_xgb_ts[,1])

            ,labels = test$y,

            modnames = c("Lasso-logistic","Decision Tree","Random Forest","XG-BOOST"),

            dsids= c(1,2,3,4),posclass = "yes",

            mode="rocprc")


eval_ts = evalmod(xgb.mm_ts)

autoplot(eval_ts)
```

3. Cumulative Gain & lift chart

```
#Cumulative Gain & lift chart

library(funModeling)

test$y  = ifelse(test$y=="no",0,1)

preds_xgb_ts$actual <- test$y

# Plot the gain and lift curve

gain_lift(data=preds_xgb_ts, score='yes', target='actual')
```

9. Conclusion and Discussion

In this paper, I studied the prediction and explanation of the success of bank telemarketing. The research question was 'Which data pre-processing 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 https://archive.ics.uci.edu/ml/datasets/bank+marketing, which is the closest dataset to data analysed by [5]. To predict the success of bank telemarketing, I used variable selection, several combinations of DM. 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 XG-Boost, in combination with Boruta method (feature selection method) was the best performing combination of methods. Furthermore, I determined the hyper parameters of XG-Boost by sequential model-based optimization. This combination of methods outperformed the best method of [5]. Thus, our combination of methods enables telemarketing managers to make better predictions of the success of bank telemarketing.

Among all the DM methods, XG-Boost having AUC 0.4605 of PR curve and also from Cumulative gain chart we see that from 10%-50% the chance of getting percentage of yes response is high compared to other DM models by targeting small percentage of clients. Now, from the point of view of Variable Importance, Euribor 3 month rate, Age and Those clients who marked previous campaign as success are highly important predictors for the model. Also, Number of Employees, Number of contacts performed before this campaign, Number of days that passed by after the client was last contacted from a previous campaign are second cluster of important predictors for the model.

Key findings

I. Customers are most likely to subscribe to term deposit when the condition of the economy is more favourable while there is a decline in the prices of goods and services.
II. If reference rate is going to be low at which rate European bank offer to lend unsecured funds, thus they can lend more money at low rate and

offer to open term deposit at high interest rate. Thus, proportion of 'yes' response is getting higher when this rate is going to be low.

III. Those list of clients who marked previous campaign as 'success' should be targeted more for this tele-marketing Campaign.

IV. Taking into account of cost and time, bank can target small (like 50% clients) with above characteristics to get high percentage of yes response from them.

Limitations

When analysing the performance of a classifier, it is imperative to note that no classifier is universally acceptable as being the best at handling all forms of data mining tasks. Different algorithms respond to certain types of attributes differently and the best performing classifier for a particular dataset could be worst when dealing with an entirely different application domain. Therefore, the possibility of the generalization of the findings of this research to other application domain cannot be evaluated.