Predicting term deposit subscribers for bank

Jaydeep Chakraborty

Data Science Project

June 23, 2020

Outline

Predicting likely subscribers of term deposit product

- A Portuguese bank conducts direct marketing campaign to sell term deposits
- Past data of all such campaigns are provided at customer level
- The data consists of customer demographics, product holding details at the bank and previous campaign history details. Alongwith these information we also know if the customer responded successfully to the campaign and subscribed the term deposit product or not
- Bank spends a lot of money to conduct the direct marketing campaigns. Currently the campaigns are executed on all prospects. The bank is keen to reduce the marketing spend by figuring out the likely subscribers from the future prospects
- We are keen to build a data science solution to help the bank. By considering the past customer data and responses we need to come up with a model that allows us to get the probability to subscribe for any future prospects. By focusing resources only on the highly likely subscribers, the bank will be able to save substantial money and increase customer satisfaction.
- We will be building a classification model to solve this business challenge
- Data Source - The data can be downloaded from UCI Machine Learning Repository
- Citation - [Moro et al., 2014] S. Moro, P. Cortez and P. Rita. A Data-Driven Approach to Predict the Success of Bank Telemarketing. Decision Support Systems, Elsevier, 62:22-31, June 2014

Outline

Steps involved in building the classification model

1. Environment set up

- All required packages for data manipulation, visualization, model building & pdf authoring is installed & loaded
- Separate R file for all user defined functions is loaded in the memory

2. Data load

- A single dataset of 45K observations is downloaded & imported
- Data Dictionary is also created by extracting relevant details from a text file
- Data audit is performed. No data cleaning is required

3. Data exploration

- Uni-variate analysis is performed separately on all categorical & continuous variables. This helps us in realizing if any variable needs an outlier or missing value treatment or if any variable transformation like log or grouping is required
- Bi-variate analysis is then performed only on the categorical variables so that we know the appropriate dummy variables that we need to create

4. Data preparation

- Appropriate dummy variables (1-hot encoding) are created
- Log transformation is performed on continuous variables with large range of values & highly skewed histogram
- Outliers are identified among continuous variables
- Grouped variables are created
- Removed multicollinear variables using VIF test
- CHAID is used to create derived variables to find cohorts with higher percentage of responders
- Multiple datasets with different percentage of training & validation data were created

Steps involved in building the classification model (cont.)

5. Model selection

- Models were built using Logistic Regression, Random Forest & Gradient Boosting Model. All model iterations were performed on training dataset with 90% observations and outlier treatment. This helped us select the best model iteration basis Accuracy and F1.
- New iterations were then done on training datasets with 85% & 80% observations using the selected modeling technique with same model parameters to check if a similar model will perform better on different splits of training & validation

6. Best model detailing

- Lift chart is built to estimate the benefit from the model
- Important variables are identified to help the business realize the key levers that can increase the response rate

Key aspects of the solution approach

- Multiple training & validation split, multiple modeling techniques & multiple iterations are performed to come up with the best performing model
- Overall response rate is low. Hence F1 statistic is considered along with accuracy to compare model performance
- Derived variables are created using CHAID to define rules for identifying cohorts with high percentage of responders
- Variable transformations are done to improve their effectiveness in the model
- Dummy variables are created by combining levels (of a categorical variable) with similar response rate
- The project report is built using RMarkdown and binb package to enable pdf output with slides. Slides makes it easy to navigate and understand the report.

Facilitating easy understanding & execution of code

- The file InternalFunctions.R contains all the functions created by me. Each function allows us to implement a particular step of the solution. Several of these functions are called multiple times in the main code. This allows us to keep the main code simple and modular.
- All important datasets, models, predicted rating & rmse are provided separately. In case one doesn't want to execute the entire code due to time constraint or machine limitations, one can simply load these files and see the outputs themselves. All the saved objects have detailed & intuitive names for easy understanding.
- Detailed comments are provided in the report as well as the code to enable ease of understanding.
- Variable naming is kept uniform and intuitive to enable easy understanding throughout the program.

Navigating the project folder

- Report.pdf -Report
- Report.Rmd -R Markdown file for creating the report
- MainCode.R -Main code for building classification model
- InternalFunctions.R -All functions created by me. These functions will be needed in main code
- SavedObjects -All the saved datasets, model objects & model predictions.
- All other files -Are used for report generation

Outline

Environment set up

1. Defining list of all required packages

```r
required.packages.data.manipulation <- c('Hmisc','data.table','plyr','tidyverse','pander')
required.packages.visualization <- c('RColorBrewer','ggplot2','gridExtra')
required.packages.model <- c('car','caret','party','pROC','h2o')
required.packages.authoring <- c('rmarkdown','binb')
required.packages <- c(required.packages.data.manipulation,
                       required.packages.visualization,
                       required.packages.model,
                       required.packages.authoring)
```

2. Installing required packages if needed

```r
packages.to.install <- required.packages[which(!required.packages %in% installed.packages()[,1])]
if(length(packages.to.install)>0) {
  cat('Following packages will be installed:\n', packages.to.install)
  install.packages(packages.to.install)
  packages.to.install <- required.packages[which(!required.packages %in% installed.packages()[,1])]
}
if(length(packages.to.install)>0) cat('Failed to install:\n', packages.to.install) else
  print('All required packages are installed.')
```

3. Loading in memory

```r
#Loading required packages in memory
sapply(required.packages, require, character.only = TRUE)

# Loading user defined functions created to make the code modular & easy to understand
source('InternalFunctions.R')
```

Data import

1. Importing bank marketing dataset from UCI Machine Learning Repository

```
# data.load function will load the data from the UCI link
dt <- data.load('https://archive.ics.uci.edu/ml/machine-learning-databases/00222/bank.zip')
```

2. Viewing top 5 rows

```
pander(head(dt,5), style='simple', split.table = 80, caption = 'Top 5 rows')
```

Table 1: Top 5 rows (continued below)

age	job	marital	education	default	balance	housing	loan
58	management	married	tertiary	no	2143	yes	no
44	technician	single	secondary	no	29	yes	no
33	entrepreneur	married	secondary	no	2	yes	yes
47	blue-collar	married	unknown	no	1506	yes	no
33	unknown	single	unknown	no	1	no	no

contact	day	month	duration	campaign	pdays	previous	poutcome	y
unknown	5	may	261	1	-1	0	unknown	no
unknown	5	may	151	1	-1	0	unknown	no
unknown	5	may	76	1	-1	0	unknown	no
unknown	5	may	92	1	-1	0	unknown	no
unknown	5	may	198	1	-1	0	unknown	no

Data import (cont.)

3. Data dictionary

```
# Creates the data dictionary from the file - 'bank-names.txt'
show.data.dictionary()
```

number	Variable	Description
1	age	(numeric)
2	job	type of job (categorical: admin.,unknown,unemployed,management,housemaid,entrepreneur,student,blue:c
3	marital	marital status (categorical: married,divorced,single; note: divorced means divorced or widowed)
4	education	unknown,secondary,primary,tertiary)
5	default	has credit in default? (binary: yes,no)
6	balance	average yearly balance, in euros (numeric)
7	housing	has housing loan? (binary: yes,no)
8	loan	has personal loan? (binary: yes,no)
9	contact	contact communication type (categorical: unknown,telephone,cellular)
10	day	last contact day of the month (numeric)
11	month	last contact month of year (categorical: jan, feb, mar, ..., nov, dec)
12	duration	last contact duration, in seconds (numeric)
13	campaign	number of contacts performed during this campaign and for this client (numeric, includes last contac
14	pdays	number of days that passed by after the client was last contacted from a previous campaign (numeric,
15	previous	number of contacts performed before this campaign and for this client (numeric)
16	poutcome	outcome of the previous marketing campaign (categorical: unknown,other,failure,success)
17	y	has the client subscribed a term deposit? (binary: yes,no)

Data audit

Performing basic audit on data loaded

```
# Provides datatypes, descriptive statistics & missing value count for each column
data.audit(dt)
```

Key insights from data audit:

- Overall data has been loaded properly
- There are no missing values
- Data type of each variable is correct
- In the dependent variable ‘y’ we have 11.7% 'yes' in both training & validation dataset
- Outliers might be possible in variables - pdays, previous, campaign, duration, balance
- In variable ‘contact’ we see that 6.4% observations are ‘telephone’ while 28.8% is ‘unknown’ and rest is ‘cellular’. It is possible that all the unknown are actually telephone. We will consider making only 1 dummy variable for ‘cellular’
- All the character variables needs dummy variable (one-hot encoding) creation
- Variable 'day' although considered as integer, will need to be considered as factor
- Apart from ‘day’ variable transformation, no other data preparation is required. We will perform dummy creation post bi-variate analysis

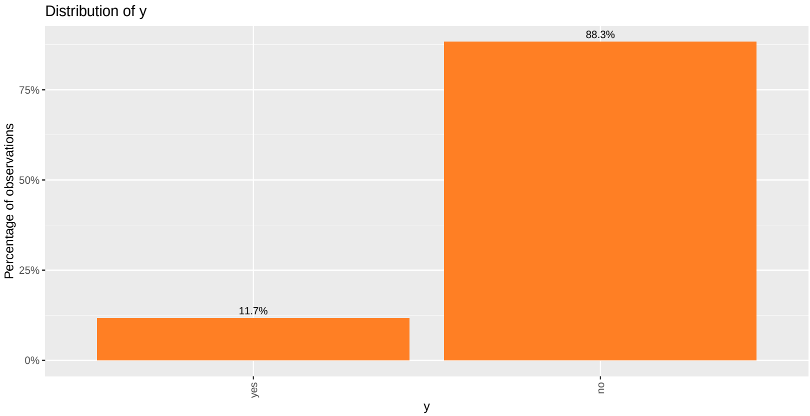
Outline

Uni-variate analysis of categorical variables

univ.categ function performs uni-variate analysis on categorical data

Exploring y or our dependent variable, that we wish to predict

```
univ.categ(dt, 'y', acc = 0.1)
```

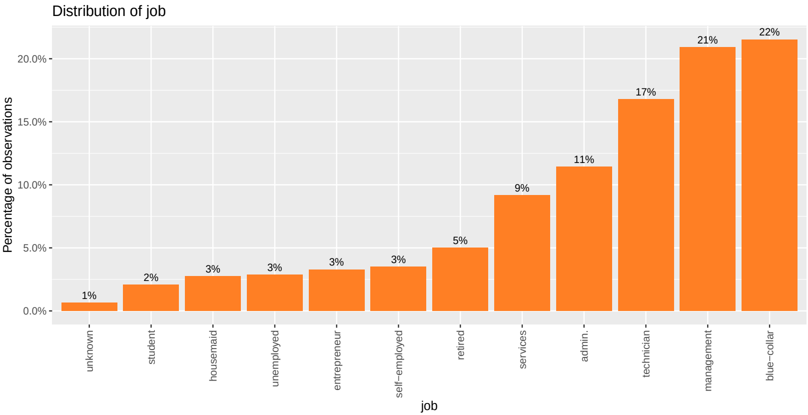


12% of customers have taken term deposit

Uni-variate analysis of categorical variables (cont.)

Variable - job

```
univ.categ(dt, 'job')
```

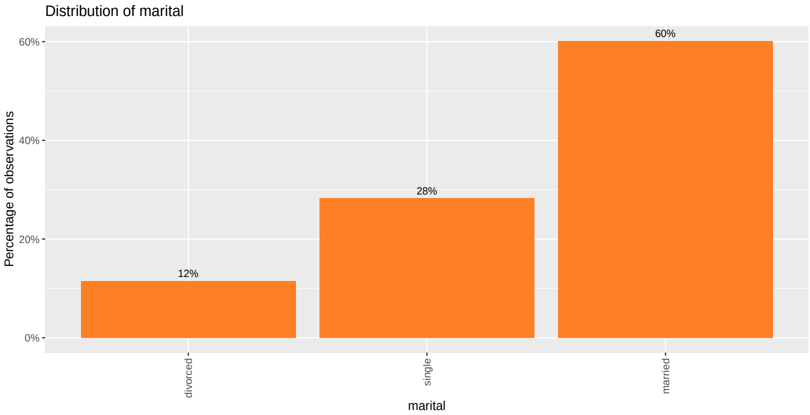


Technician, management & blue-collar comprises of 60% of all job types

Uni-variate analysis of categorical variables (cont.)

Variable - marital

```
univ.categ(dt, 'marital')
```

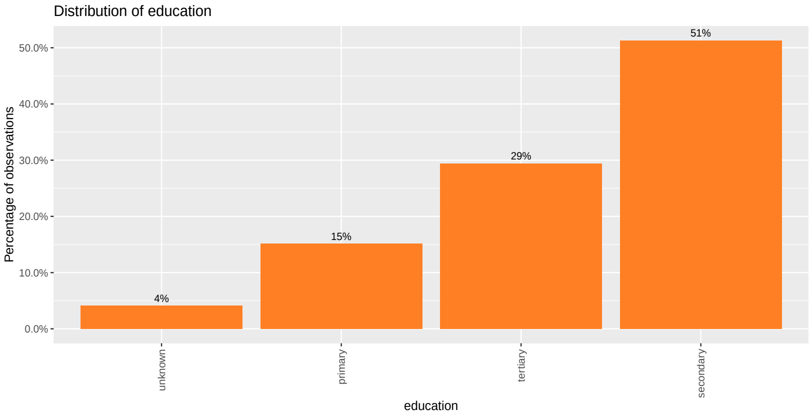


Majority of customers are married

Uni-variate analysis of categorical variables (cont.)

Variable - education

```
univ.categ(dt, 'education')
```

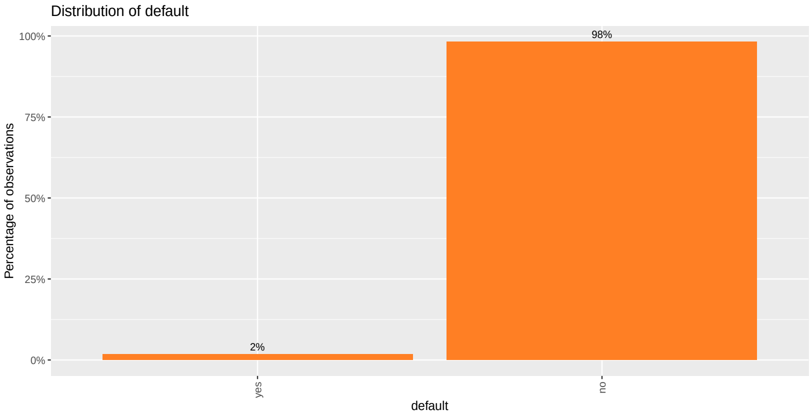


Majority of customers have completed secondary education. There are 4% unknown.

Uni-variate analysis of categorical variables (cont.)

Variable - default

```
univ.categ(dt, 'default')
```

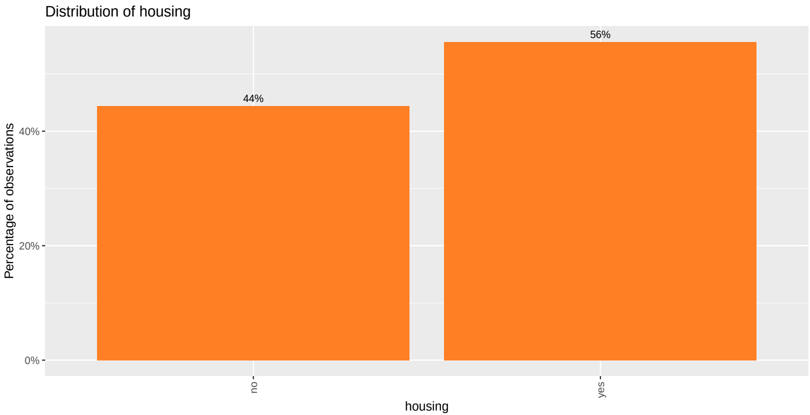


Mostly there is only 1 value. 2% of customers have credit in default.

Uni-variate analysis of categorical variables (cont.)

Variable - housing

```
univ.categ(dt, 'housing')
```

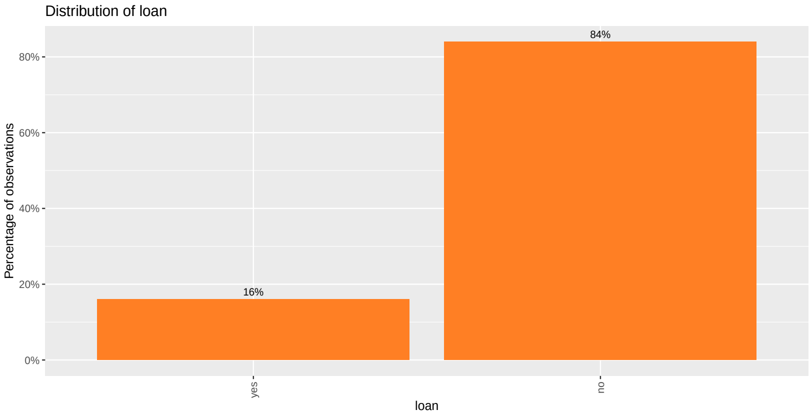


A little more than half of all customers have taken a housing loan

Uni-variate analysis of categorical variables (cont.)

Variable - loan

```
univ.categ(dt, 'loan')
```

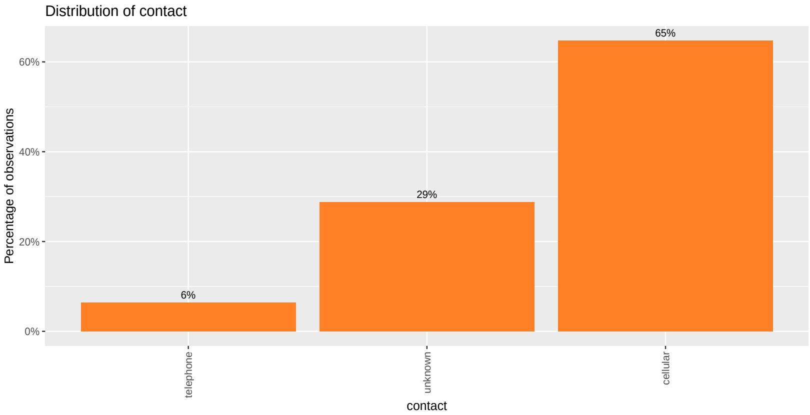


High majority of customers do not have a personal loan

Uni-variate analysis of categorical variables (cont.)

Variable - contact

```
univ.categ(dt, 'contact')
```

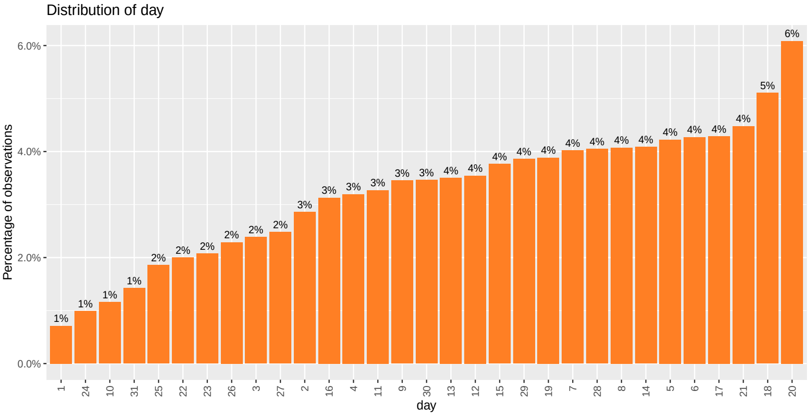


Majority of customers can be contacted using mobile. Unknowns are quite higher than telephone.

Uni-variate analysis of categorical variables (cont.)

Variable - day

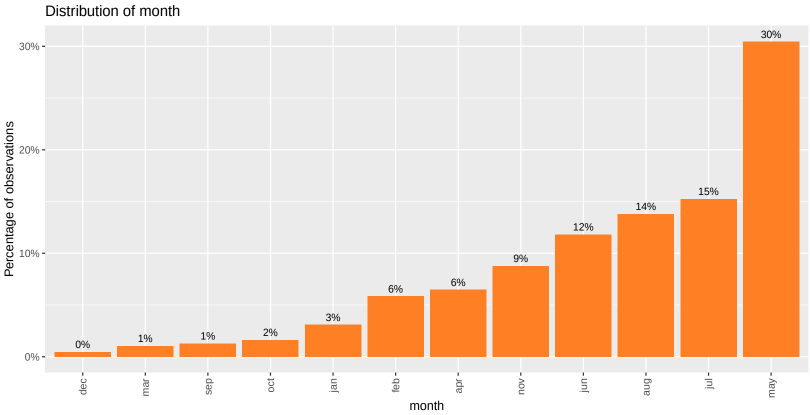
```
univ.categ(dt, 'day')
```



Uni-variate analysis of categorical variables (cont.)

Variable - month

```
univ.categ(dt, 'month')
```

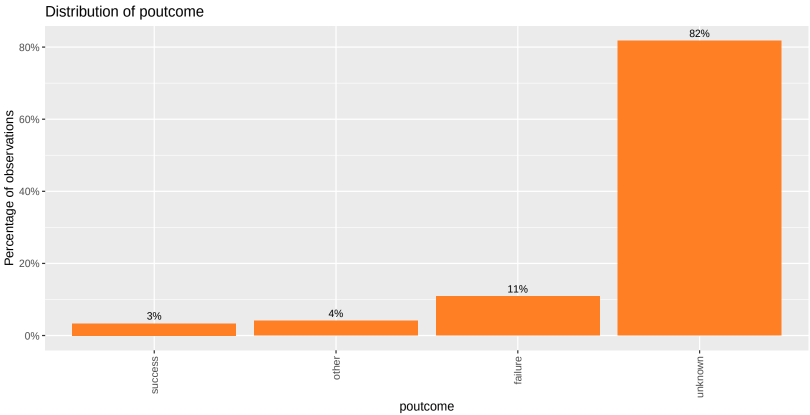


Almost one third of customers are contacted in May

Uni-variate analysis of categorical variables (cont.)

Variable - poutcome

```
univ.categ(dt, 'poutcome')
```



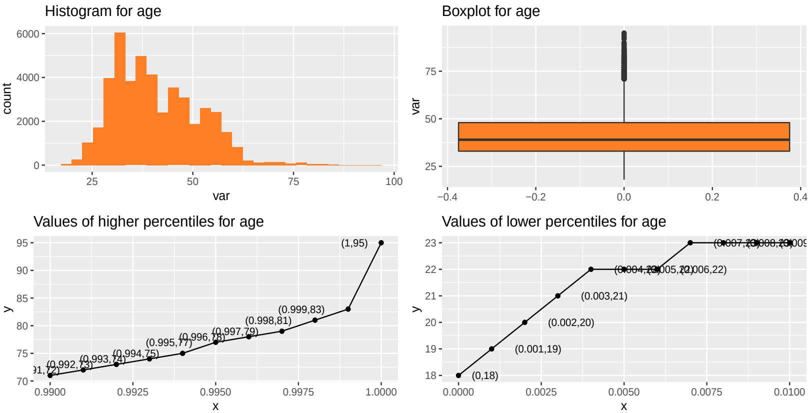
Status of previous campaigns are unknown for 82% of cases. Only 3% have resulted in success earlier

Uni-variate analysis of continous variables

univ.cont function performs uni-variate analysis on continuous data

Variable - age

```
univ.cont(dt, 'age')
```

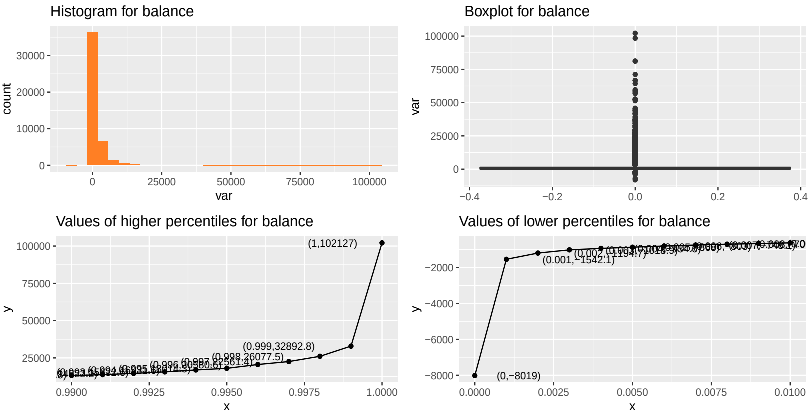


Higher outliers present. Outlier Cutoff 99.9%ile = 83 years will be considered later

Uni-variate analysis of continous variables (cont.)

Variable - balance

```
univ.cont(dt, 'balance')
```

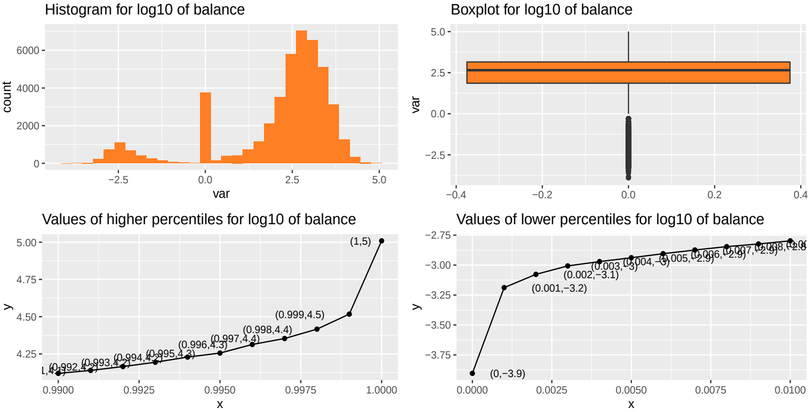


Lets perform log transformation as the range is very high and histogram is skewed

Uni-variate analysis of continous variables (cont.)

Variable - We will consider the log transformation of balance

```
univ.cont(dt, 'balance', log.transform = T)
```

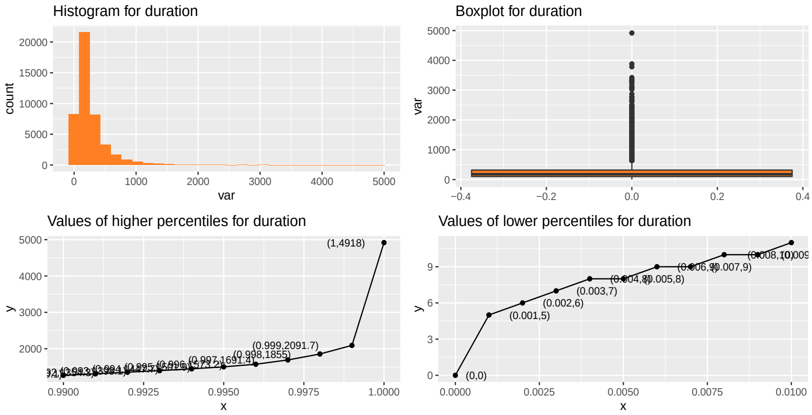


- Note that for log transformation, values = 0 are retained as 0 and for negative values -log10(abs(x)) is used
- We will create a derived variable with the following levels: 1: less than -2.5 , 2: between -2.5 to 0, 3: equal to 0, 4: between 0 and 2.5, 5: greater than 2.5

Uni-variate analysis of continous variables (cont.)

Variable - duration

```
univ.cont(dt, 'duration')
```

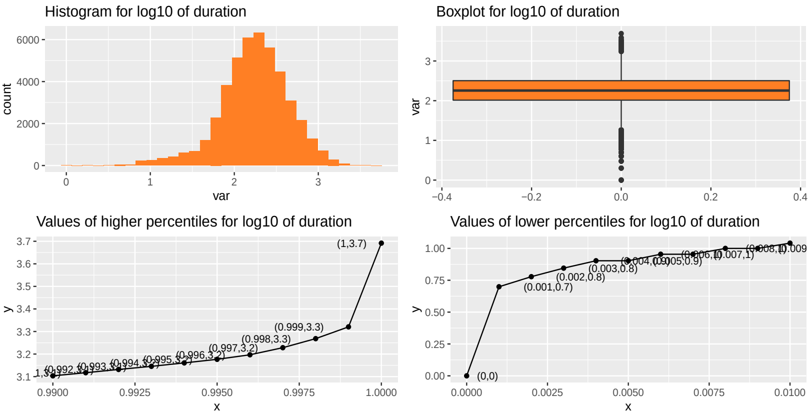


Lets perform log transformation as the range is very high and histogram is skewed

Uni-variate analysis of continous variables (cont.)

Variable - We will consider the log transformation of duration

```
univ.cont(dt, 'duration', log.transform = T)
```

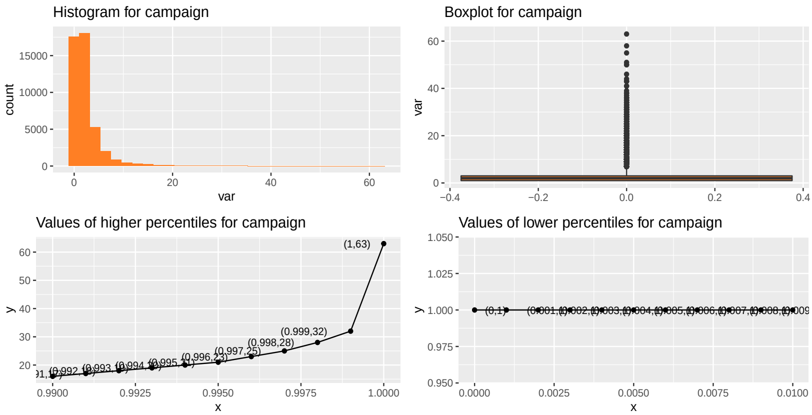


- We will consider log transformation and also outlier treatment
- Values of log transformation more than 3.3 or less than 0.7 will be considered as outliers

Uni-variate analysis of continous variables (cont.)

Variable - campaign

```
univ.cont(dt, 'campaign')
```

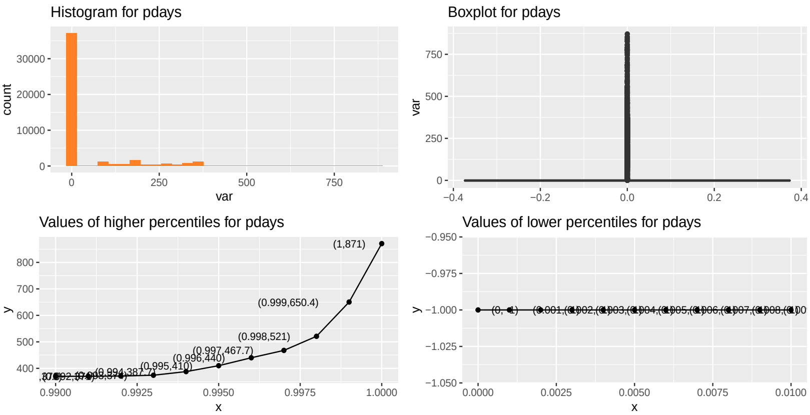


Majority of values is 0. We will consider this variable as is with higher outlier treatment. Higher outlier cutoff will be 32.

Uni-variate analysis of continous variables (cont.)

Variable - pdays

```
univ.cont(dt, 'pdays')
```

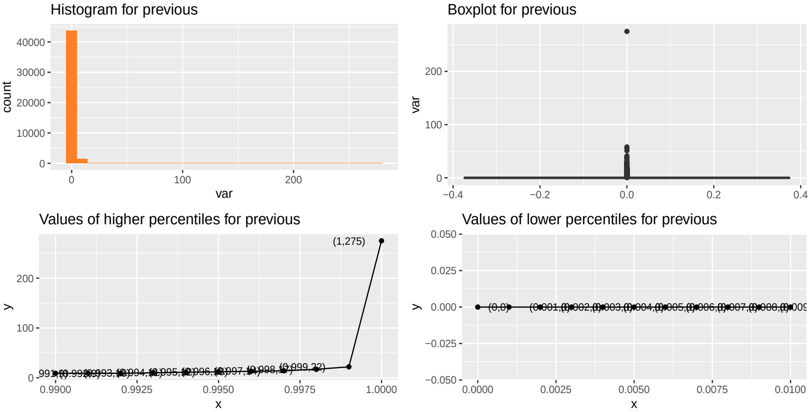


Majority of values is 0. We will consider this variable as is with higher outlier treatment. Higher outlier cutoff will be 637.

Uni-variate analysis of continous variables (cont.)

Variable - previous

```
univ.cont(dt, 'previous')
```



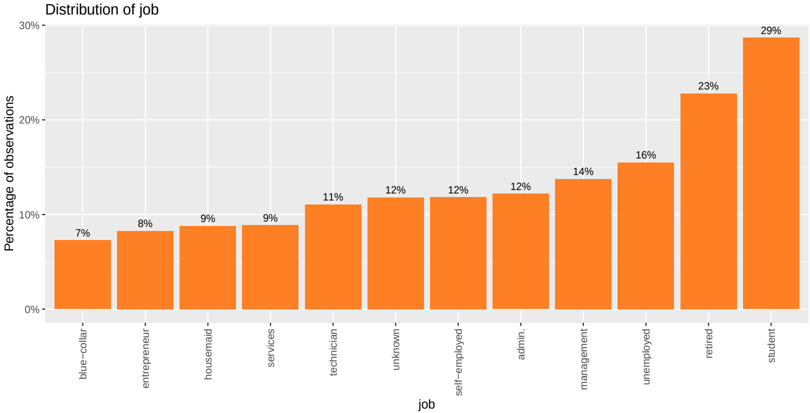
Majority of values is 0. We will consider this variable as is with higher outlier treatment. Higher outlier cutoff will be 22.

Bi-variate analysis of categorical variables

biv.categ function helps in exploring relationship of categorical variables with dependent variable y

Variable - job

```
biv.categ(dt, 'job')
```

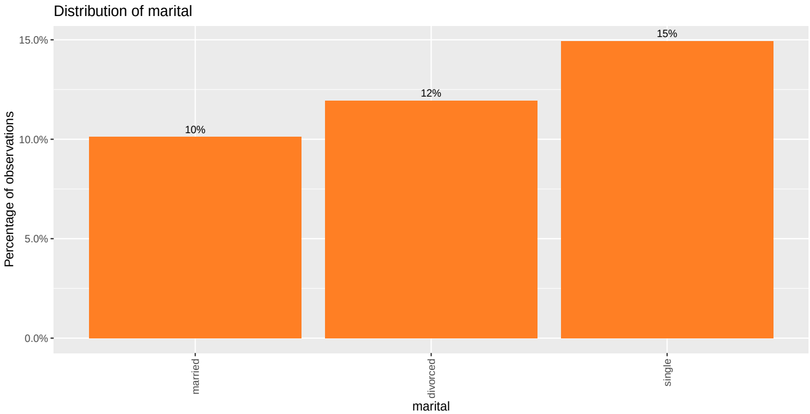


4 dummy variables will be created for job type - student, retired, unemployed, technician or self-employed or admin or unknown or management.

Bi-variate analysis of categorical variables (cont.)

Variable - marital

```
biv.categ(dt, 'marital')
```

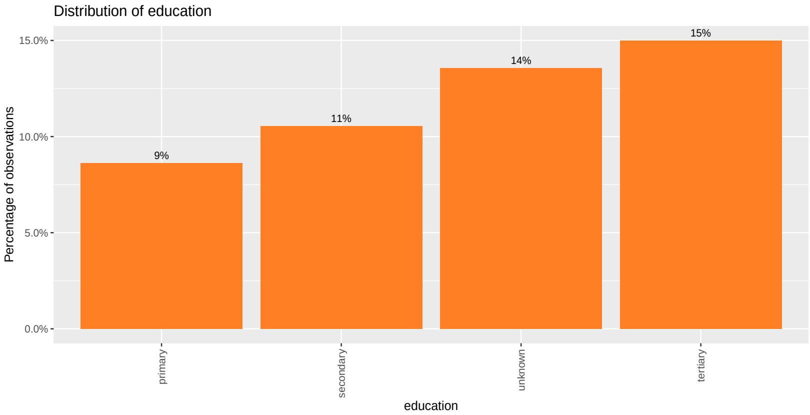


2 dummy variables will be created for marital status - single, divorced.

Bi-variate analysis of categorical variables (cont.)

Variable - education

```
biv.categ(dt, 'education')
```

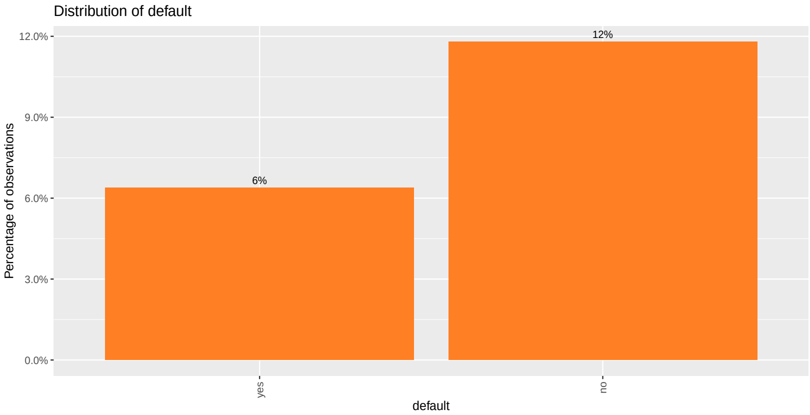


3 dummy variables will be created for education type - tertiary, unknown , secondary.

Bi-variate analysis of categorical variables (cont.)

Variable - default

```
biv.categ(dt, 'default')
```

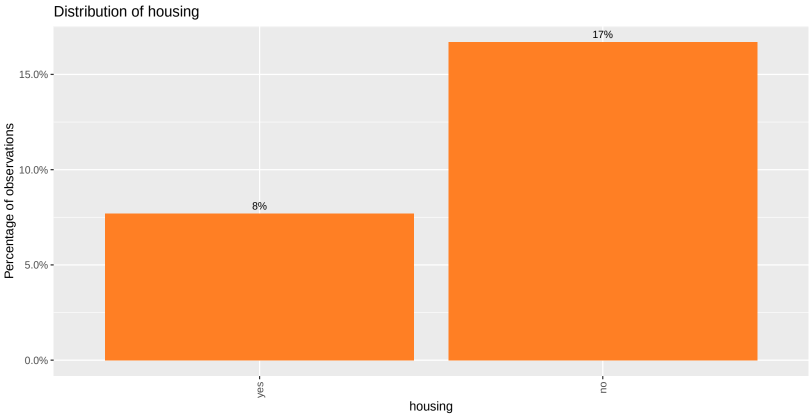


1 dummy variable will be created for default - no.

Bi-variate analysis of categorical variables (cont.)

Variable - housing

```
biv.categ(dt, 'housing')
```

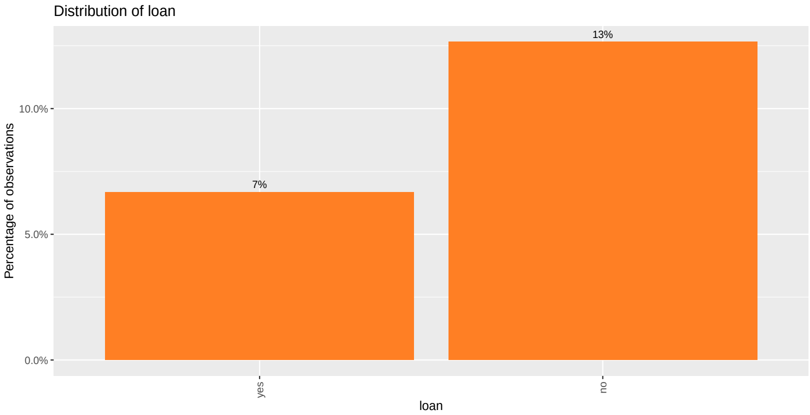


1 dummy variable will be created for housing loan - no.

Bi-variate analysis of categorical variables (cont.)

Variable - loan

```
biv.categ(dt, 'loan')
```

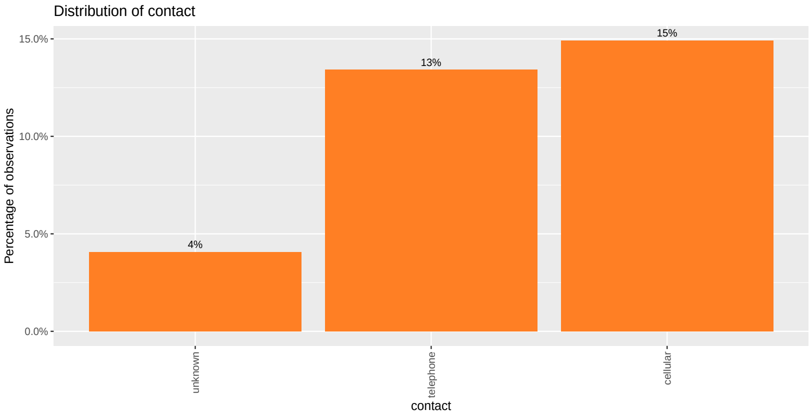


1 dummy variable will be created for personal loan - no.

Bi-variate analysis of categorical variables (cont.)

Variable - contact

```
biv.categ(dt, 'contact')
```

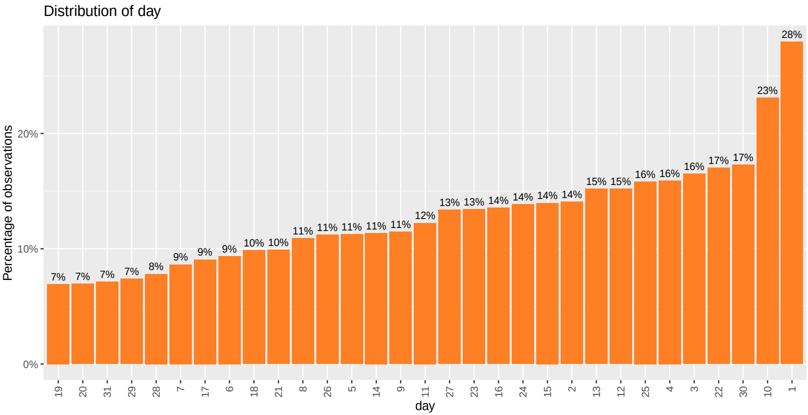


2 dummy variables will be created for contact type - cellular, unknown.

Bi-variate analysis of categorical variables (cont.)

Variable - day

```
biv.categ(dt, 'day')
```

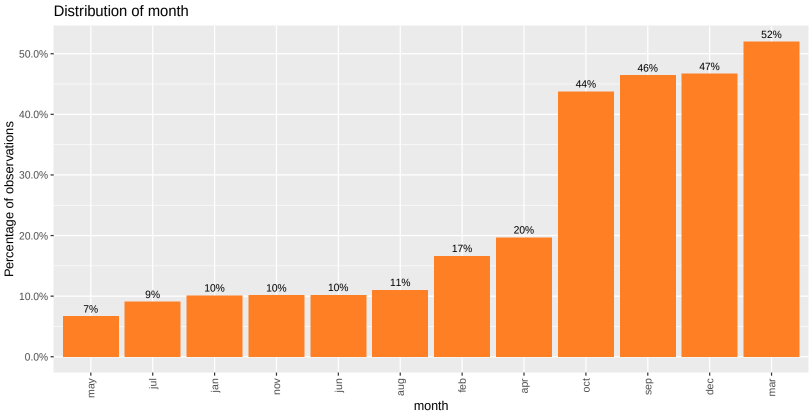


5 dummy variables will be created for day - 1, 10, 30 or 3 or 22 or 25 or 4, 12 or 13 or 2 or 24 or 27 or 15 or 23 or 16, 11 or 9 or 26 or 5 or 14 or 8.

Bi-variate analysis of categorical variables (cont.)

Variable - month

```
biv.categ(dt, 'month')
```

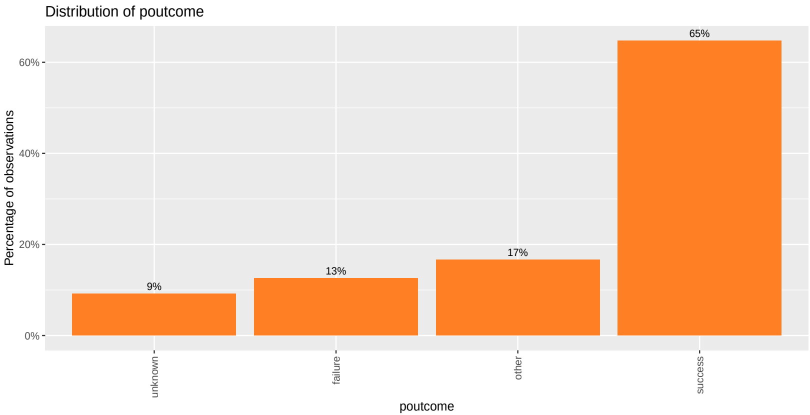


4 dummy variables will be created for month - mar, oct or dec or sep, feb, apr.

Bi-variate analysis of categorical variables (cont.)

Variable - poutcome

```
biv.categ(dt, 'poutcome')
```



3 dummy variables will be created for previous outcome - success, other, failure.

Outline

Dummy variable creation

```
# Dummy creation for - job
dt <- dt %>% mutate(d.job.1 = ifelse(job == 'student',1,0),
                    d.job.2 = ifelse(job == 'retired',1,0),
                    d.job.3 = ifelse(job == 'unemployed',1,0),
                    d.job.4 = ifelse(job %in% c('technician' , 'self-employed','admin.','unknown','management'),1,0))

# Dummy creation for - marital
dt <- dt %>% mutate(d.marital.1 = ifelse(marital == 'single',1,0),
                    d.marital.2 = ifelse(marital == 'divorced',1,0))

# Dummy creation for - education
dt <- dt %>% mutate(d.education.1 = ifelse(education == 'tertiary',1,0),
                    d.education.2 = ifelse(education == 'unknown',1,0),
                    d.education.3 = ifelse(education == 'secondary',1,0))

# Dummy creation for - default
dt <- dt %>% mutate(d.default.1 = ifelse(default == 'no',1,0))

# Dummy creation for - housing
dt <- dt %>% mutate(d.housing.1 = ifelse(housing == 'no',1,0))

# Dummy creation for - loan
dt <- dt %>% mutate(d.loan.1 = ifelse(loan == 'no',1,0))

# Dummy creation for - contact
dt <- dt %>% mutate(d.contact.1 = ifelse(contact == 'cellular',1,0),
                    d.contact.2 = ifelse(contact == 'unknown',1,0))

# Dummy creation for - day
dt <- dt %>% mutate(d.day.1 = ifelse(day == 1,1,0),
                    d.day.2 = ifelse(day == 10,1,0),
                    d.day.3 = ifelse(day %in% c(30,3,22,25,4),1,0),
                    d.day.4 = ifelse(day %in% c(12,13,2, 24,27,15,23,16),1,0),
                    d.day.5 = ifelse(day %in% c(11,9,26,5,14,8),1,0))
```

Dummy variable creation (cont.)

```r
# Dummy creation for - month
dt <- dt %>% mutate(d.month.1 = ifelse(month == 'mar',1,0),
                    d.month.2 = ifelse(month %in% c('oct','dec','sep'),1,0),
                    d.month.3 = ifelse(month == 'feb',1,0),
                    d.month.4 = ifelse(month == 'apr',1,0))

# Dummy creation for - poutcome
dt <- dt %>% mutate(d.poutcome.1 = ifelse(poutcome == 'success',1,0),
                    d.poutcome.2 = ifelse(poutcome == 'other',1,0),
                    d.poutcome.3 = ifelse(poutcome == 'failure',1,0))

# Converting dependent variable to 0 & 1
dt$y <- ifelse(dt$y=='yes',1,0)

# Removing categorical variables whose dummy variables has been created
dt <- dt %>% select(-c(2:5,7:11,16))
```

Creating log transformation variables for continous variables with high range and skewed histogram

```
# Log transformation for balance
dt$l.balance <- ifelse(dt$balance ==0, 0, ifelse(dt$balance<0, -log10(abs(dt$balance)), log10(dt$balance)))

# Log transformation for duration
dt$l.duration <- ifelse(dt$duration ==0, 0, ifelse(dt$duration<0, -log10(abs(dt$duration)), log10(dt$duration)))

# Removing the variables whose log transformation is done
dt <- dt[,-c(2,3)]
```

Creating a flag for all outliers

```
dt$is.outlier <- F
dt <- dt %>% mutate(is.outlier = ifelse(age>=83,T,is.outlier)) # 63 outliers
dt <- dt %>% mutate(is.outlier = ifelse(l.duration>=3.3 | l.duration<=0.7,T,is.outlier)) # 120 outliers
dt <- dt %>% mutate(is.outlier = ifelse(campaign>=32,T,is.outlier)) # 47 outliers
dt <- dt %>% mutate(is.outlier = ifelse(pdays>=637,T,is.outlier)) # 41 outliers
dt <- dt %>% mutate(is.outlier = ifelse(previous>=22,T,is.outlier)) # 49 outliers

# Percentage of observations detected as outliers
100*prop.table(table(dt$is.outlier))
```

0.7% observations detected as outliers.

Creating grouped variables

```
# Creating grouped variable for log of balance
dt <- dt %>% mutate(g.l.balance = ifelse(l.balance>-2.5,
                                         ifelse(l.balance==0,3,
                                                ifelse(l.balance>0,ifelse(l.balance>2.5,5,4),2)),1))
```

Remove multicollinear variables using VIF test

```
# lm.out <- lm(y~.-g.l.balance, data = dt)
# sort(vif(lm.out))
#
# lm.out <- lm(y~.-g.l.balance -pdays, data = dt)
# sort(vif(lm.out))
#
# lm.out <- lm(y~.-g.l.balance -pdays-d.contact.2, data = dt)
# sort(vif(lm.out))

lm.out <- lm(y~.-g.l.balance -pdays-d.contact.2-d.education.1, data = dt)
sort(vif(lm.out))
```

```
##     d.month.1       d.day.2       d.day.1      d.loan.1   d.default.1
##      1.025393      1.026909      1.027161      1.040848      1.058073
##    l.duration       d.job.3   d.marital.2 d.education.2     d.month.4
##      1.063211      1.069683      1.070815      1.071968      1.079566
## d.education.3    is.outlier     d.month.3      campaign     l.balance
##      1.084609      1.087843      1.090458      1.094578      1.103479
##     d.month.2       d.job.1  d.poutcome.1       d.day.3   d.housing.1
##      1.110004      1.154824      1.167849      1.189916      1.190005
##   d.contact.1  d.poutcome.2       d.day.5       d.job.4       d.day.4
##      1.199559      1.217238      1.222179      1.222314      1.231004
##   d.marital.1  d.poutcome.3       d.job.2      previous           age
##      1.333049      1.344148      1.377001      1.477469      1.650979
```

All VIF values are now less than 2.

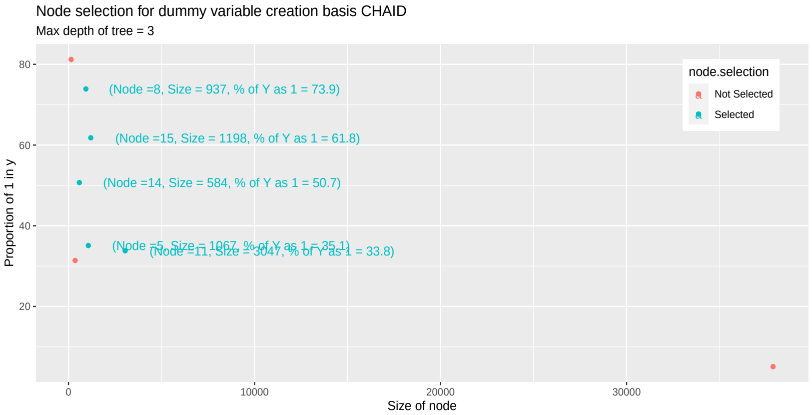
```
# Removing variables with high VIF. Creating a backup of dt, before we do this.
dt <- dt[, -c(3,19,12)]
```

We will keep g.l.balance. When building model we will always try only one between l.balance & g.l.balance

We will build a decision tree to find cohorts where there is a higher probability to find y = 1

- We want to find cohorts where y=1 for at least 12% of cases & minimum size of cohort is 500
- Multiple iterations are done. The final selected iteration is shown here

```
m.chaid3 <- ctree(y ~ ., data = dt, controls = ctree_control(testtype = "Univariate",maxdepth = 3))
plot.chaid(dt,m.chaid3,S = 500, P = 11.7, D= 3)
```



We will go ahead with m.chaid3 and create 5 dummy variables as they cover maximum count of y =1 with highest proportion

```
# Adding nodes to dt
dt$chaid.node <- predict(m.chaid3, newdata = dt, type="node")
# Creating dummy variables for node = 14, 8,5,15, 11
dt <- dt %>% mutate(node.14 = ifelse(chaid.node == 14, 1,0))
dt <- dt %>% mutate(node.8 = ifelse(chaid.node == 8, 1,0))
dt <- dt %>% mutate(node.5 = ifelse(chaid.node == 5, 1,0))
dt <- dt %>% mutate(node.15 = ifelse(chaid.node == 15, 1,0))
dt <- dt %>% mutate(node.11 = ifelse(chaid.node == 11, 1,0))
# Dropping node variable
dt$chaid.node <- NULL
```

5 new dummy variables are added to the dataset. These derived variables might enrich the model.

Creating final datasets for model building

- We will create 6 sets of training & validation datasets:
- 1. Training dataset with 90% observations and outlier treatment
- 2. Training dataset with 90% observations without outlier treatment
- 3. Training dataset with 85% observations and outlier treatment
- 4. Training dataset with 85% observations without outlier treatment
- 5. Training dataset with 80% observations and outlier treatment
- 6. Training dataset with 80% observations without outlier treatment

```
train.n.validation <- data.split(dt, train.percentage = 0.9, outlier.treatment = T)
train90t <- train.n.validation$train
validation90t <- train.n.validation$validation

train.n.validation <- data.split(dt, train.percentage = 0.9, outlier.treatment = F)
train90 <- train.n.validation$train
validation90 <- train.n.validation$validation

train.n.validation <- data.split(dt, train.percentage = 0.85, outlier.treatment = T)
train85t <- train.n.validation$train
validation85t <- train.n.validation$validation

train.n.validation <- data.split(dt, train.percentage = 0.85, outlier.treatment = F)
train85 <- train.n.validation$train
validation85 <- train.n.validation$validation

train.n.validation <- data.split(dt, train.percentage = 0.8, outlier.treatment = T)
train80t <- train.n.validation$train
validation80t <- train.n.validation$validation

train.n.validation <- data.split(dt, train.percentage = 0.8, outlier.treatment = F)
train80 <- train.n.validation$train
validation80 <- train.n.validation$validation

rm(train.n.validation)
```

- We will run several models on first set of training & validation i.e. Training dataset with 90% observations and outlier treatment
- The modeling technique selected from above basis F1 accuracy metric (on validation) will be applied on all other sets of training & validation
- The best model will be then selected basis F1 accuracy metric on validation

Outline

Logistic Regression model (Training dataset - 90% observation & outlier treated)

Creating a table to store model iteration results

```
model.results <- data.frame(SNo = integer(), ModelType = character(), Model.Parameters = character(),
                            Accuracy.train = double(), Accuracy.validation = double(), F1.train = double(),
                            F1.validation = double(), stringsAsFactors=F)
```

Model iterations

```
m.log.reg <- glm(y~. -g.l.balance , data=train90t, family=binomial())
summary(m.log.reg)

m.log.reg <- glm(y~. -g.l.balance -d.default.1, data=train90t, family=binomial())
summary(m.log.reg)

m.log.reg <- glm(y~. -g.l.balance -d.default.1 -d.poutcome.3, data=train90t, family=binomial())
summary(m.log.reg)

m.log.reg <- glm(y~. -g.l.balance -d.default.1 -d.poutcome.3 -age, data=train90t, family=binomial())
summary(m.log.reg)

m.log.reg <- glm(y~. -g.l.balance -d.default.1 -d.poutcome.3 -age - d.education.2, data=train90t,
family=binomial())
summary(m.log.reg)
```

Logistic Regression model

Final Model Iteration - All variables in the model are significant

```
m.log.reg <- glm(y~. -g.l.balance -d.default.1 -d.poutcome.3 -age - d.education.2 - d.job.3, data=train90t,
                 family=binomial())
summary(m.log.reg)
```

```
##
## Call:
## glm(formula = y ~ . - g.l.balance - d.default.1 - d.poutcome.3 -
##     age - d.education.2 - d.job.3, family = binomial(), data = train90t)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.8761  -0.3568  -0.2017  -0.1021   3.9757
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -13.32140    0.27255 -48.876  < 2e-16 ***
## campaign       -0.06785    0.01064  -6.376 1.82e-10 ***
## previous        0.04809    0.01151   4.177 2.96e-05 ***
## d.job.1         0.80798    0.11059   7.306 2.75e-13 ***
## d.job.2         0.53185    0.08236   6.457 1.07e-10 ***
## d.job.4         0.26465    0.04435   5.967 2.42e-09 ***
## d.marital.1     0.29793    0.04479   6.651 2.91e-11 ***
## d.marital.2     0.22327    0.06307   3.540 0.000400 ***
## d.education.3 -0.07950    0.03967  -2.004 0.045069 *
## d.housing.1     0.74411    0.04261  17.464  < 2e-16 ***
## d.loan.1        0.40012    0.06234   6.419 1.37e-10 ***
## d.contact.1     0.81212    0.05624  14.441  < 2e-16 ***
## d.day.1         0.61541    0.17546   3.507 0.000452 ***
## d.day.2         0.85204    0.14844   5.740 9.47e-09 ***
## d.day.3         0.51135    0.05997   8.526  < 2e-16 ***
## d.day.4         0.54406    0.05173  10.517  < 2e-16 ***
## d.day.5         0.29382    0.05471   5.370 7.86e-08 ***
## d.month.1       2.43215    0.12356  19.684  < 2e-16 ***
## d.month.2       0.35163    0.13148   2.674 0.007484 **
## d.month.3       0.38579    0.07534   5.121 3.04e-07 ***
## d.month.4       0.68342    0.06702  10.197  < 2e-16 ***
## d.poutcome.1    2.12334    0.12312  17.246  < 2e-16 ***
## d.poutcome.2    0.27674    0.09216   3.003 0.002673 **
## l.balance       0.09182    0.01396   6.579 4.75e-11 ***
```

Logistic Regression model

Confusion matrix on training data

```
pred.log.reg.t <- get.predictions(m.log.reg, train90t, train90t)
cm.t <- confusion.matrix(train90t$y, pred.log.reg.t)
```

```
##       Predicted
## Actual     0     1
##      0 29189  6545
##      1   682  3985
```

```
pander(cm.t, style='simple', split.table = 80)
```

Metric	Value
True Negative	29189
False Positive	6545
False Negative	682
True Positive	3985
Sensitivity	0.8539
Specificity	0.8168
Recall	0.8539
Precision	0.3784
Accuracy	0.8211
F1	0.5244

Logistic Regression model

Confusion matrix on validation data

```
pred.log.reg <- get.predictions(m.log.reg, train90t, validation90t)
cm.v <- confusion.matrix(validation90t$y, pred.log.reg)
```

```
##       Predicted
## Actual    0    1
##      0 3204  746
##      1   83  457
```

```
pander(cm.v, style='simple', split.table = 80)
```

Metric	Value
True Negative	3204
False Positive	746
False Negative	83
True Positive	457
Sensitivity	0.8463
Specificity	0.8111
Recall	0.8463
Precision	0.3799
Accuracy	0.8154
F1	0.5244

Adding model performance in result table

```
model.results <- add.model.result(1,'Logistic Regression', 'NA', cm.t, cm.v, model.results)
```

Random Forest model (Training dataset - 90% observation & outlier treated)

Model iterations - Only the best iteration is shown here

```
h2o.init(nthreads=-1, max_mem_size = "10G")  # initializes with all available threads and 10Gb memory
h2o.removeAll() # frees up the memory

# Getting the input data ready
train <- train90t
train$y <- as.factor(train$y)

# Model run
m.rf2 <- h2o.randomForest(y = "y", training_frame = as.h2o(train), ntrees = 50, max_depth=15, nfolds = 3,
                          seed = 1,keep_cross_validation_predictions = TRUE, fold_assignment = "Random")

# Confusion matrix on training data
pred.rf2.t <- h2o.predict(m.rf2, as.h2o(train90t))
cm.t <- confusion.matrix(train90t$y, as.data.frame(pred.rf2.t$predict)$predict)

# Confusion matrix on validation data
pred.rf2 <- h2o.predict(m.rf2, as.h2o(validation90t))
cm.v <- confusion.matrix(validation90t$y, as.data.frame(pred.rf2$predict)$predict)

# Adding model performance in result table
model.results <- add.model.result(3,'Random Forest', 'ntrees = 50, max_depth=15', cm.t, cm.v, model.results)
```

Random Forest model

Performance on training data

```
pander(cm.t, style='simple', split.table = 80)
```

Metric	Value
True Negative	33716
False Positive	2018
False Negative	835
True Positive	3832
Sensitivity	0.8211
Specificity	0.9435
Recall	0.8211
Precision	0.655
Accuracy	0.9294
F1	0.7287

Performance on validation data

```
pander(cm.v, style='simple', split.table = 80)
```

Metric	Value
True Negative	3601
False Positive	349
False Negative	165
True Positive	375
Sensitivity	0.6944
Specificity	0.9116
Recall	0.6944
Precision	0.518
Accuracy	0.8855
F1	0.5934

Gradient Boosting model (Training dataset - 90% observation & outlier treated)

Model iterations - Only the best iteration is shown here

```r
h2o.init(nthreads=-1, max_mem_size = "10G")  # initializes with all available threads and 10Gb memory
h2o.removeAll() # frees up the memory

# Getting the input data ready
train <- train90t
train$y <- as.factor(train$y)

# Model run
m.rf2 <- h2o.randomForest(y = "y", training_frame = as.h2o(train), ntrees = 50, max_depth=15, nfolds = 3,
                          seed = 1,keep_cross_validation_predictions = TRUE, fold_assignment = "Random")

m.gbm8 <- h2o.gbm(y = "y", training_frame = as.h2o(train), ntrees = 55, max_depth=5, nfolds = 3,
                  seed = 1,keep_cross_validation_predictions = TRUE, fold_assignment = "Random")

m.ensemble <- h2o.stackedEnsemble(y = "y",training_frame = as.h2o(train),
                                  base_models = list(m.gbm8@model_id, m.rf2@model_id))

# Confusion matrix on training data
pred.ensemble.t <- h2o.predict(m.ensemble, as.h2o(train90t))
cm.t <- confusion.matrix(train90t$y, as.data.frame(pred.ensemble.t$predict)$predict)

# Confusion matrix on validation data
pred.ensemble <- h2o.predict(m.ensemble, as.h2o(validation90t))
cm.v <- confusion.matrix(validation90t$y, as.data.frame(pred.ensemble$predict)$predict)

# Adding model performance in result table
model.results <- add.model.result(19,'GBM', 'ntrees = 55, max_depth=5', cm.t, cm.v, model.results)
```

Gradient Boosting model

Performance on training data

```
pander(cm.t, style='simple', split.table = 80)
```

Metric	Value
True Negative	33472
False Positive	2262
False Negative	1519
True Positive	3148
Sensitivity	0.6745
Specificity	0.9367
Recall	0.6745
Precision	0.5819
Accuracy	0.9064
F1	0.6248

Performance on validation data

```
pander(cm.v, style='simple', split.table = 80)
```

Metric	Value
True Negative	3683
False Positive	267
False Negative	179
True Positive	361
Sensitivity	0.6685
Specificity	0.9324
Recall	0.6685
Precision	0.5748
Accuracy	0.9007
F1	0.6182

Ensemble model (Training dataset - 90% observation & outlier treated)

Model iteration - best Random Forest model & best GBM model are considered

```
h2o.init(nthreads=-1, max_mem_size = "10G")  # initializes with all available threads and 10Gb memory
h2o.removeAll() # frees up the memory

# Getting the input data ready
train <- train90t
train$y <- as.factor(train$y)

# Model run
m.gbm8 <- h2o.gbm(y = "y", training_frame = as.h2o(train), ntrees = 55, max_depth=5, nfolds = 3,
                  seed = 1,keep_cross_validation_predictions = TRUE, fold_assignment = "Random")

# Confusion matrix on training data
pred.gbm8.t <- h2o.predict(m.gbm8, as.h2o(train90t))
cm.t <- confusion.matrix(train90t$y, as.data.frame(pred.gbm8.t$predict)$predict)

# Confusion matrix on validation data
pred.gbm8 <- h2o.predict(m.gbm8, as.h2o(validation90t))
cm.v <- confusion.matrix(validation90t$y, as.data.frame(pred.gbm8$predict)$predict)

# Adding model performance in result table
model.results <- add.model.result(21,'GBM + RF', 'NA', cm.t, cm.v, model.results)
save(m.gbm8,pred.gbm8.t, pred.gbm8, file= 'best.model.rdata')
```

Ensemble model

Performance on training data

```r
pander(cm.t, style='simple', split.table = 80)
```

Metric	Value
True Negative	33472
False Positive	2262
False Negative	1519
True Positive	3148
Sensitivity	0.6745
Specificity	0.9367
Recall	0.6745
Precision	0.5819
Accuracy	0.9064
F1	0.6248

Performance on validation data

```r
pander(cm.v, style='simple', split.table = 80)
```

Metric	Value
True Negative	3683
False Positive	267
False Negative	179
True Positive	361
Sensitivity	0.6685
Specificity	0.9324
Recall	0.6685
Precision	0.5748
Accuracy	0.9007
F1	0.6182

Outline

Result comparison of all model iterations

All model iterations are shown here. Refer code for all iteration details.

```
load('model.results1.rdata')
mr <- model.results1 %>% arrange(desc(F1.validation))
pander(mr, style='simple', split.table = 160)
```

SNo	ModelType	Model.Parameters	Accuracy.train	Accuracy.validation	F1.train	F1.validation
19	GBM	ntrees = 55, max_depth=5	0.905	0.9	0.624	0.617
14	GBM	ntrees = 50, max_depth=5	0.903	0.898	0.621	0.616
20	GBM	ntrees = 60, max_depth=5	0.906	0.9	0.626	0.614
18	GBM	ntrees = 45, max_depth=5	0.903	0.898	0.62	0.613
21	GBM + RF	NA	0.924	0.89	0.713	0.604
15	GBM	ntrees = 100, max_depth=5	0.915	0.901	0.641	0.603
3	Random Forest	ntrees = 50, max_depth=15	0.938	0.889	0.758	0.601
11	Random Forest	ntrees = 500, max_depth=50	0.997	0.887	0.988	0.601
7	Random Forest	ntrees = 100, max_depth=10	0.902	0.887	0.639	0.6
10	Random Forest	ntrees = 100, max_depth=20	0.975	0.886	0.9	0.6
17	GBM	ntrees = 50, max_depth=6	0.914	0.899	0.644	0.6
4	Random Forest	ntrees = 50, max_depth=10	0.904	0.889	0.641	0.599
2	Random Forest	ntrees = 50, max_depth=20	0.974	0.885	0.895	0.596
16	GBM	ntrees = 50, max_depth=4	0.897	0.891	0.606	0.596
6	Random Forest	ntrees = 100, max_depth=5	0.888	0.887	0.577	0.589
5	Random Forest	ntrees = 50, max_depth=5	0.883	0.883	0.571	0.586
9	Random Forest	ntrees = 200, max_depth=5	0.89	0.887	0.578	0.582
13	GBM	ntrees = 50, max_depth=10	0.95	0.899	0.779	0.564
8	Random Forest	ntrees = 200, max_depth=3	0.873	0.869	0.558	0.561
1	Logistic Regression	NA	0.816	0.808	0.516	0.517
12	GBM	ntrees = 50, max_depth=15	0.984	0.892	0.927	0.516

GBM model with ntrees = 55 & max_depth=5 has given the best performance

Model iterations on other sets of training & validation data

GBM model with ntrees = 55 & max_depth=5 will be tried on other training datasets

Model iteration on 90% Training without outlier treatment

```
h2o.init(nthreads=-1, max_mem_size = "10G")
h2o.removeAll()
train <- train90
train$y <- as.factor(train$y)

m.gbm11 <- h2o.gbm(y = "y", training_frame = as.h2o(train), ntrees = 55, max_depth=5, nfolds = 3,
                   seed = 1,keep_cross_validation_predictions = TRUE, fold_assignment = "Random")

# Performance on training data
pred.gbm11.t <- h2o.predict(m.gbm11, as.h2o(train90))
cm.t <- confusion.matrix(train90$y, as.data.frame(pred.gbm11.t$predict)$predict)

# Performance on validation data
pred.gbm11 <- h2o.predict(m.gbm11, as.h2o(validation90))
cm.v <- confusion.matrix(validation90$y, as.data.frame(pred.gbm11$predict)$predict)

# Adding model results
model.results <- add.model.result(22,'90% Train without outlier treatement', 'GBM: ntrees = 55,
                                  max_depth=5', cm.t, cm.v, model.results)
```

GBM model with ntrees = 55 & max_depth=5 on Training (90%) data without outlier treatment

Performance on training data

```r
pander(cm.t[c(5:10),], style='simple', split.table = 80)
```

	Metric	Value
5	Sensitivity	0.6813
6	Specificity	0.9344
7	Recall	0.6813
8	Precision	0.581
9	Accuracy	0.9046
10	F1	0.6271

Performance on validation data

```r
pander(cm.v[c(5:10),], style='simple', split.table = 80)
```

	Metric	Value
5	Sensitivity	0.6606
6	Specificity	0.9319
7	Recall	0.6606
8	Precision	0.5456
9	Accuracy	0.902
10	F1	0.5976

- Model performance has reduced on dataset without outlier treatment, showing that outlier treatment is important
- We will hence, try the next iterations only on outlier treated data with 85% and 80% training

Model iterations on other sets of training & validation data

GBM model with ntrees = 55 & max_depth=5 will be tried on other training datasets

Model iteration on 85% Training with outlier treatment

```
h2o.init(nthreads=-1, max_mem_size = "10G")
h2o.removeAll()
train <- train90
train$y <- as.factor(train$y)

m.gbm12 <- h2o.gbm(y = "y", training_frame = as.h2o(train), ntrees = 55, max_depth=5, nfolds = 3,
                   seed = 1,keep_cross_validation_predictions = TRUE, fold_assignment = "Random")

# Performance on training data
pred.gbm12.t <- h2o.predict(m.gbm12, as.h2o(train85t))
cm.t <- confusion.matrix(train85t$y, as.data.frame(pred.gbm12.t$predict)$predict)

# Performance on validation data
pred.gbm12 <- h2o.predict(m.gbm12, as.h2o(validation85t))
cm.v <- confusion.matrix(validation85t$y, as.data.frame(pred.gbm12$predict)$predict)

# Adding model results
model.results <- add.model.result(23,'85% Train with outlier treatment', 'GBM: ntrees = 55, max_depth=5',
                                  cm.t, cm.v, model.results)
```

GBM model with ntrees = 55 & max_depth=5 on Training (85%) data with outlier treatment

Performance on training data

```r
pander(cm.t[c(5:10),], style='simple', split.table = 80)
```

	Metric	Value
5	Sensitivity	0.6771
6	Specificity	0.935
7	Recall	0.6771
8	Precision	0.5754
9	Accuracy	0.9053
10	F1	0.6221

Performance on validation data

```r
pander(cm.v[c(5:10),], style='simple', split.table = 80)
```

	Metric	Value
5	Sensitivity	0.6761
6	Specificity	0.9348
7	Recall	0.6761
8	Precision	0.588
9	Accuracy	0.9035
10	F1	0.629

- Model performance with 85% training is lower than 90% training data
- It is fair to assume that the model performance will not improve with 80% training data.

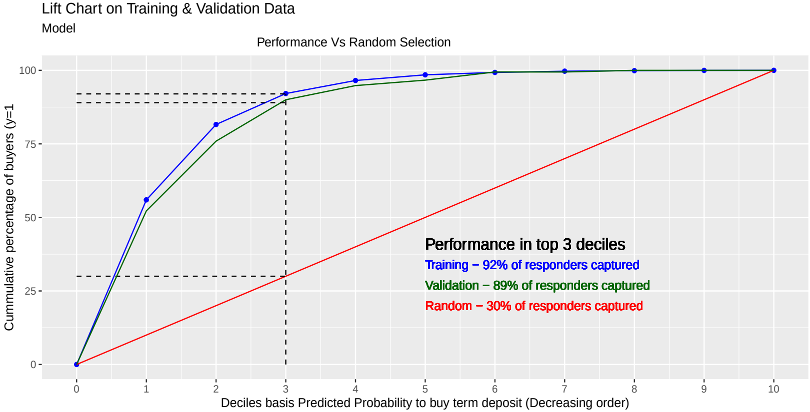
Hence we can select GBM with ntrees = 55 & max_depth=5 on 90% Training Data with outlier treatment as the best model

Lift Chart generation using final selected model on 90% training data with outlier treatment

```r
# m.gbm8 (GBM with ntrees = 55 & max_depth=5 has given the best performance
load('best.model.rdata')
# Lift chart on training data for best model
lift.chart.train <- generate.lift.chart.table(train90t$y, as.data.frame(pred.gbm8.t$p1)$p1)
# Lift chart on validation data for best model
lift.chart.validation <- generate.lift.chart.table(validation90t$y, as.data.frame(pred.gbm8$p1)$p1)
# Generating the lift chart to compare model performance on test & validation
lift.chart <- ggplot(data = lift.chart.train, aes(x=Segment, y = CummPercentY_1)) + geom_point(color = 'blue') +
  geom_line(color = 'blue') + geom_line(aes(y=CummRandom), color = 'red') +
  scale_x_continuous(name ="Deciles basis Predicted Probability to buy term deposit (Decreasing order)",
                     breaks=seq(0,10,1))+
  labs(y = 'Cummulative percentage of buyers (y=1') + ggtitle('Lift Chart on Training & Validation Data', 'Model
                                                          Performance Vs Random Selection')+
  geom_text(aes(x = 5, y=41, label = 'Performance in top 3 deciles'), size = 5, hjust = 0, color = 'black') +
  geom_text(aes(x = 5, y=34, label = 'Training - 92% of responders captured'), size = 4, hjust = 0, color = 'blue') +
  geom_text(aes(x = 5, y=27, label = 'Validation - 89% of responders captured'), size = 4, hjust = 0,
          color = 'darkgreen') +
  geom_text(aes(x = 5, y=20, label = 'Random - 30% of responders captured'), size = 4, hjust = 0, color = 'red') +
  geom_line(data = data.frame(x = c(3,3), y = c(0,92)), aes(x = x, y = y), linetype = "dashed") +
  geom_line(data = data.frame(x = c(0,3), y = c(92,92)), aes(x = x, y = y), linetype = "dashed") +
  geom_line(data = data.frame(x = c(0,3), y = c(30,30)), aes(x = x, y = y), linetype = "dashed") +
  geom_line(data = lift.chart.validation, aes(x=Segment, y = CummPercentY_1), color = 'darkgreen')+
  geom_line(data = data.frame(x = c(0,3), y = c(89,89)), aes(x = x, y = y), linetype = "dashed")
```

Lift Chart generation using final selected model on 90% training data with outlier treatment

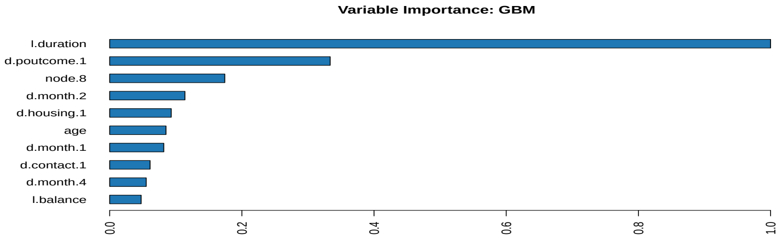
```
plot(lift.chart)
```



Key variables

Plot of top 10 most important variables, sorted in the order of relative variable importance. The most important variable is given a value 1.

```
h2o.varimp_plot(m.gbm8,num_of_features =10)
```



- l.duration -Last contact duration plays the most important role. Log transformation is considered.
- d.poutcome.1 - If outcome of previous marketing campaign was success
- node.8 - Derived variable from CHAID model
- d.month.2 - If last contact month was Sep or Oct or Dec
- d.housing.1 - Does not have housing loan
- age - Prospect age
- d.month.1 - If last contact month was Mar
- d.contact.1 - If communication type is Cellular
- d.month.4 - If last contact month was Apr
- l.balance - Average annual balance in Euro. Log transformation is considered.