Bank Marketing Data Analysis - Project Report

Bhusan Bathani(bbath2) Mathew Leung(wmleung2)
Vijayakumar Sitha Mohan(VS24)

Contents

```
library(knitr)      # web widget
library(tidyverse)  # data manipulation
library(caret)      # rocr analysis
library(ROCR)       # rocr analysis
library(kableExtra) # nice table html formating
library(gridExtra)  # arranging ggplot in grid
library(rpart)      # decision tree
library(rpart.plot) # decision tree plotting
library(caTools)    # split
library(lmtest)
library(randomForest)
library(ada)
library(boot)
library(e1071)      # caret depends on this
library(ada)
library(ROSE)
library(DMwR)
library(latticeExtra)
library(pROC)
```

Introduction

Our team selected a dataset of direct banking marketing of a Portuguese bank. The goal is to predict if the client subscribe to the bank term deposit. The dataset has client personal attributes, marketing campaign attributes, social and economic context attributes. The response variable (y) is binomial with value either 1 (subscribe) or 0 (not subscribe).

Our team is interested in various classification technique performance. We selected this dataset because it is a clean dataset for classification. It has been top ten most popular from the data repository that we

download from. We can focus on analyzing classification techniques rather then cleaning data. The URL to the dataset is http://archive.ics.uci.edu/ml/datasets/Bank+Marketing.

We picked the following three classification techniques for this project:

- Logistic Regression (It is covered in this course STAT420)
- Random Forest (Very popular technique in Ensemble learning with a number of weak classifier based on decision tree)
- Adaptive Boosting (More sophisticated technique in Ensemble learning with a number of weak classifier based on decision tree)

Methods

Understanding Data

- The Bank-Marketing dataset is downloaded from UCI Machine Learning Repository and the same is available at http://archive.ics.uci.edu/ml/datasets/Bank+Marketing.
- There were 4 datasets in it from which bank-additional.csv is used that has subset of 4,119 observations out of all available observations of 41,188 in the full dataset.
- 20 inputs ordered by date (from May 2008 to November 2010). There are 20 input variables and 1 output variable (desired target).
- The dataset has customer data, socio-economic data, telemarketing data and some other data. Some attributes are numerical, and some are categorical.
- There are 4 categories of data in the dataset.
  - Customer's demographic information - age,job,marital,education,default,housing,loan
  - Customer's financial data - default, hosing loan, personal loan
  - Campaign and Marketing data - contact,month,day_of_week,duration,campaign,pdays,previous,poutcome
  - Overall market and economic indicators - emp.var.rate,cons.price.idx,cons.conf.idx,euribor3m,nr.employed

Attribute Information:

Column.Name	Type	Description
age	Numeric	Client Age
job	Categorical	Type of Job
marital	Categorical	Client's marital status
education	Categorical	Client's education
default	Categorical	has credit in default?
housing	Categorical	has housing loan?
loan	Categorical	has personal loadn?
contact	string	contact communication type
day	Categorical	Day of last contact with client
month	Categorical	Month of last contact with client
duration	Numeric	last contact duration, in seconds
campaign	Numeric	number of contacts performed during this campaign and for this client
pdays	Numeric	no of days passed by after client was last contacted from a prev campaign
previous	Numeric	Number of client contacts performed before this campaign
poutcome	Categorical	outcome of the previous marketing campaign
emp.var.rate	Numeric	Quarterly employment variation rate
cons.price.idx	Numeric	Monthly consumer price index
cons.conf.idx	Numeric	Monthly consumer confidence index
euribor3m	Numeric	Daily euribor 3-month rate
nr.employed	Numeric	Quarterly number of employees
Term Deposit	Binary	has the client subscribed a term deposit

- Output variable (desired target) is **Term Deposit** which is categorical binary variable.

Data Validation Checking for Missing value

```
any(is.na(bank_df1))
```

```
## [1] FALSE
```

- There are 0 rows with missing (NA) value in any of the columns.

```
paste("There are ", sum(duplicated(bank_df1)), "Duplicate Row(s)")
```

Data Curation (cleaning , missing data, duplicate)

```
## [1] "There are  2 Duplicate Row(s)"
```

Remove Duplicates

```
bank_df = bank_df1 %>% distinct
```

Recode categorical attributes as factor variables

```
bank_df$y = ifelse(bank_df$y=='yes',1,0)
bank_df$y = as.factor(bank_df$y)
bank_df$job = as.factor(bank_df$job)
bank_df$education = as.factor(bank_df$education)
bank_df$marital = as.factor(bank_df$marital)
bank_df$default = as.factor(bank_df$default)
bank_df$housing = as.factor(bank_df$housing)
bank_df$loan = as.factor(bank_df$loan)
bank_df$contact = as.factor(bank_df$contact)
bank_df$month = as.factor(bank_df$month)
bank_df$poutcome = as.factor(bank_df$poutcome)
bank_df$day_of_week = as.factor(bank_df$day_of_week)
bank_df$default = as.factor(bank_df$default)
```

Verify Levels of Categorical attributes

```
levels(bank_df$job)
```

```
##  [1] "admin."        "blue-collar"   "entrepreneur"  "housemaid"
##  [5] "management"    "retired"       "self-employed" "services"
##  [9] "student"       "technician"    "unemployed"    "unknown"
```

```
levels(bank_df$marital)
```

```
## [1] "divorced" "married"  "single"   "unknown"
```

```
levels(bank_df$education)
```

```
## [1] "basic.4y"            "basic.6y"            "basic.9y"
## [4] "high.school"         "illiterate"          "professional.course"
## [7] "university.degree"   "unknown"
```

```
levels(bank_df$default)
```

```
## [1] "no"      "unknown" "yes"
```

```
levels(bank_df$housing)
```

```
## [1] "no"      "unknown" "yes"
```

```
levels(bank_df$loan)
```

```
## [1] "no"      "unknown" "yes"
```

```
levels(bank_df$contact)
```

```
## [1] "cellular"  "telephone"
```

```
levels(bank_df$month)
```

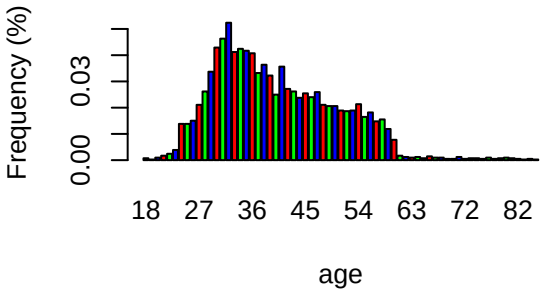
```
##  [1] "apr" "aug" "dec" "jul" "jun" "mar" "may" "nov" "oct" "sep"
```

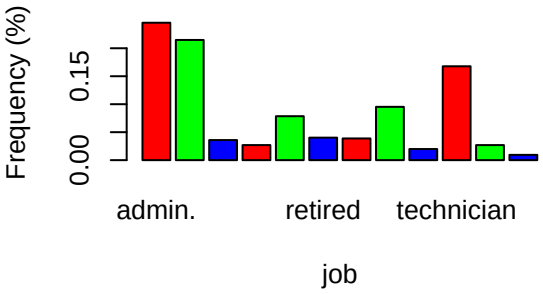
```
levels(bank_df$poutcome)
```

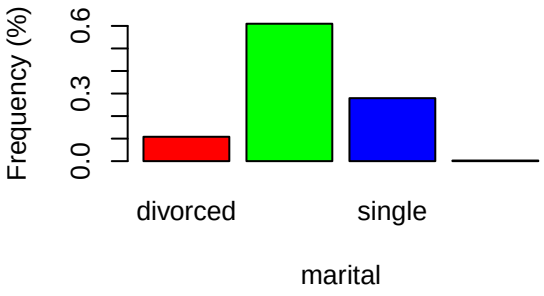
```
## [1] "failure"     "nonexistent" "success"
```

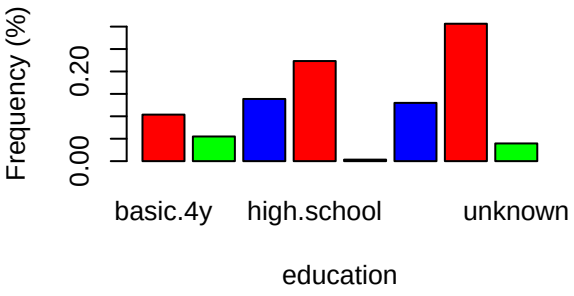
Data Analysis Let's see the levels and frequency of these data at higher level in histogram to have a sense of data distributed.

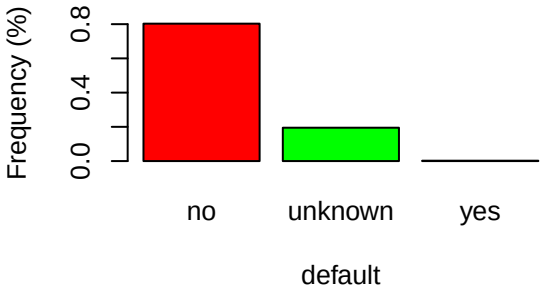
```
par(mfrow=c(2,2))
for(i in 1:21){
  barplot(prop.table(table(bank_df[,i])) ,
          xlab=names(bank_df[i]), ylab= "Frequency (%)" , col = rainbow(3), horiz = FALSE)
}
```

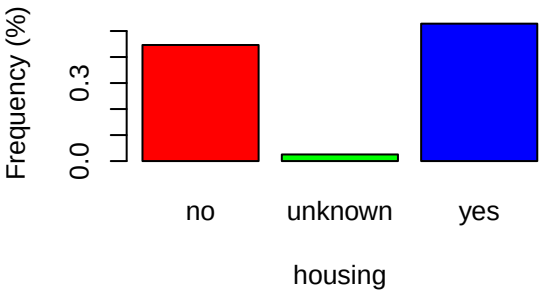


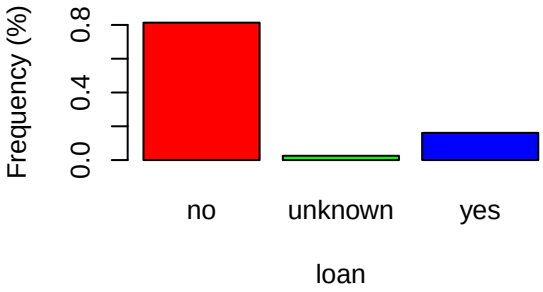


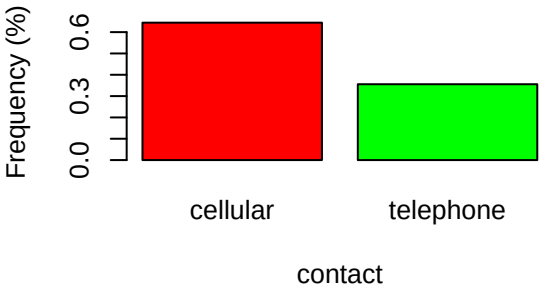


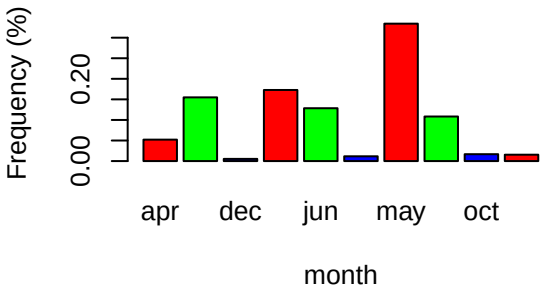


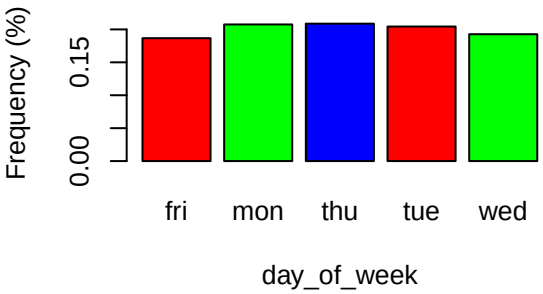

Frequency (%)
0.0
0.4
0.8
no
unknown
yes
default

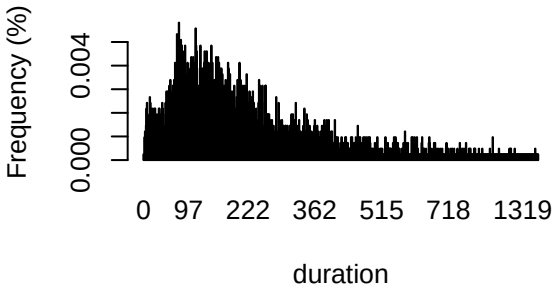

Frequency (%)
0.0
0.3
no
unknown
yes
housing

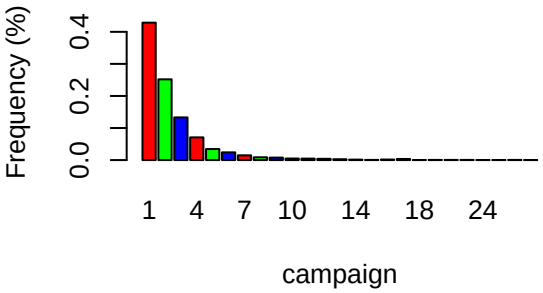

Frequency (%)
0.0
0.4
0.8
no
unknown
yes
loan

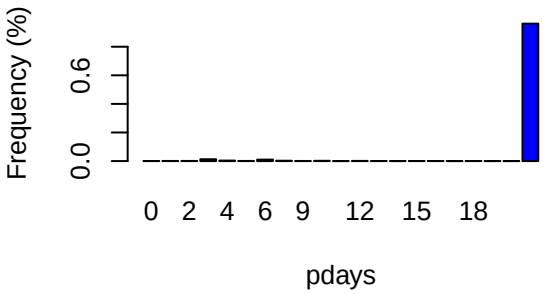

Frequency (%)
0.0
0.3
0.6
cellular
telephone
contact

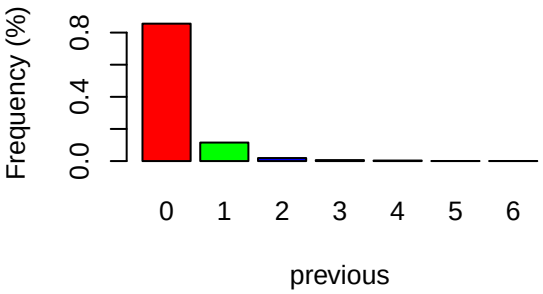

Frequency (%)
0.00
0.20
apr
dec
jun
may
oct
month

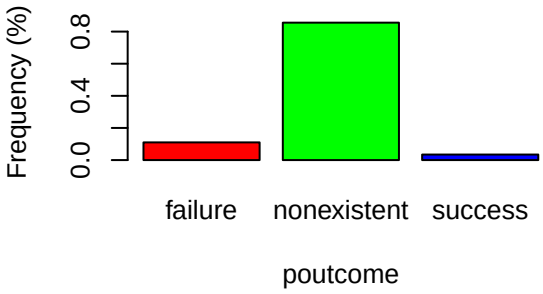

Frequency (%)
0.00
0.15
fri
mon
thu
tue
wed
day_of_week

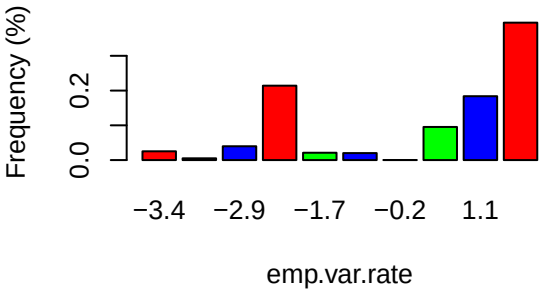

Frequency (%)
0.000
0.004
0
97
222
362
515
718
1319
duration

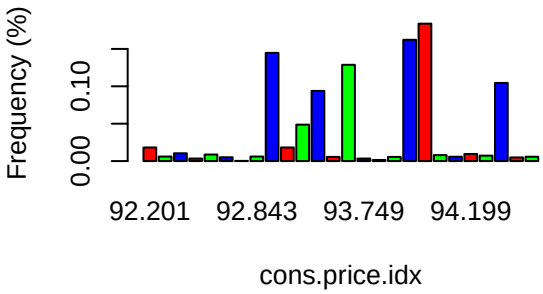

Frequency (%)
0.0
0.2
0.4
1
4
7
10
14
18
24
campaign

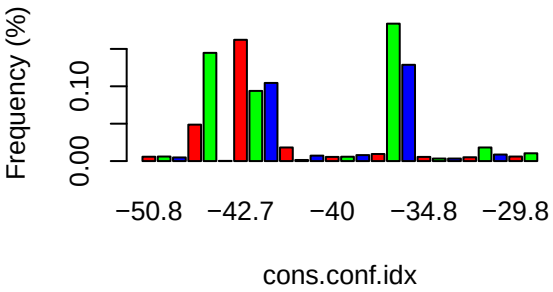

Frequency (%)
0.0
0.6
0 2 4 6 9 12 15 18
pdays

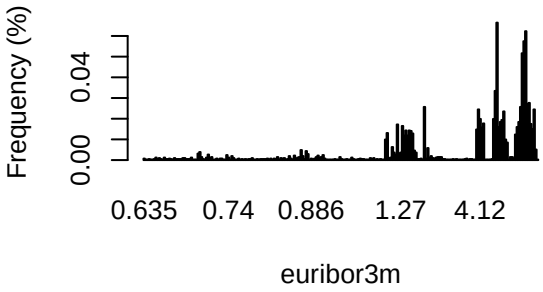

Frequency (%)
0.0
0.4
0.8
0 1 2 3 4 5 6
previous


Frequency (%)
0.0
0.4
0.8
failure
nonexistent
success
poutcome

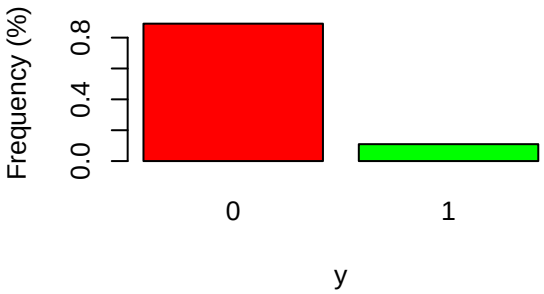

Frequency (%)
0.0
0.2
−3.4
−2.9
−1.7
−0.2
1.1
emp.var.rate


Frequency (%)
0.00
0.10
92.201
92.843
93.749
94.199
cons.price.idx


Frequency (%)
0.00
0.10
−50.8
−42.7
−40
−34.8
−29.8
cons.conf.idx

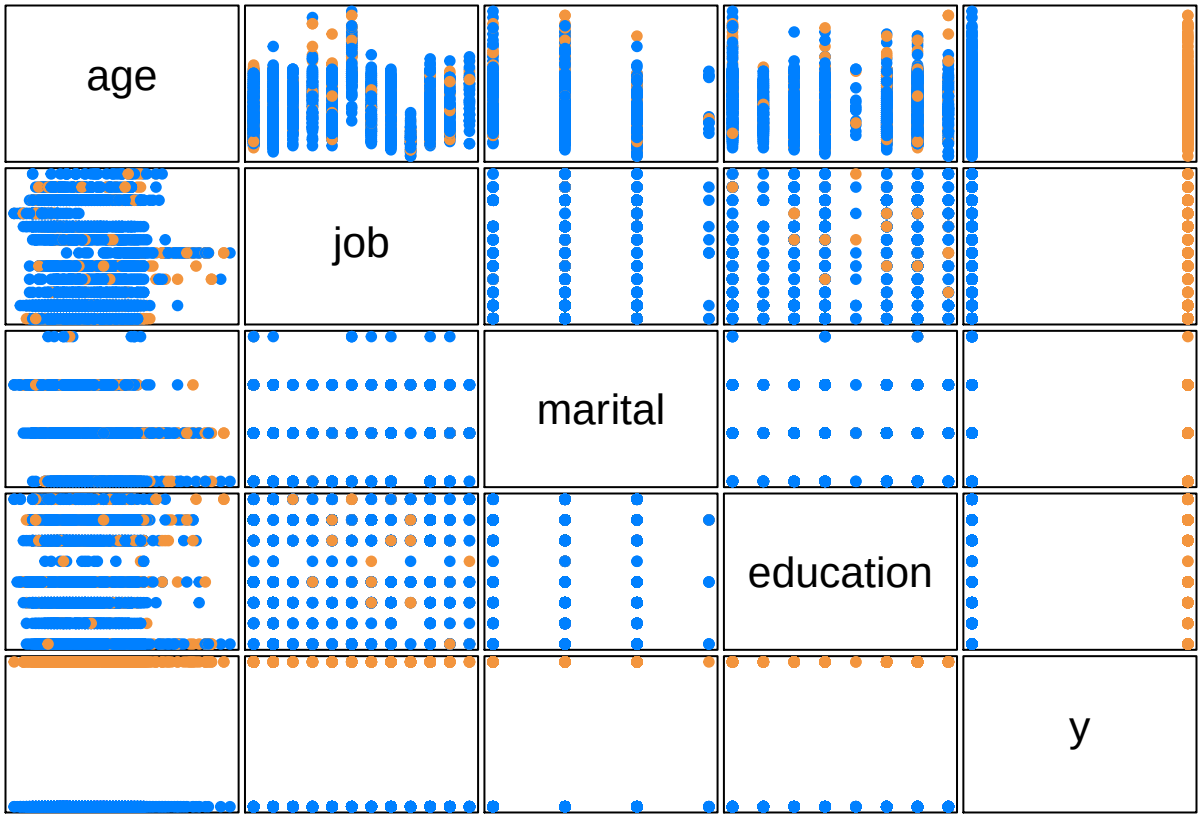

Frequency (%)
0.00
0.04
0.635
0.74
0.886
1.27
4.12
euribor3m


Frequency (%)
0.0
0.2
4963.6
5017.5
5099.1
5195.8
nr.employed



- The above barplots show frequency distribution of each attribute in the dataset.
- Let's analyze the dataset for any linearity between predictors and y using boxplot and pairs.

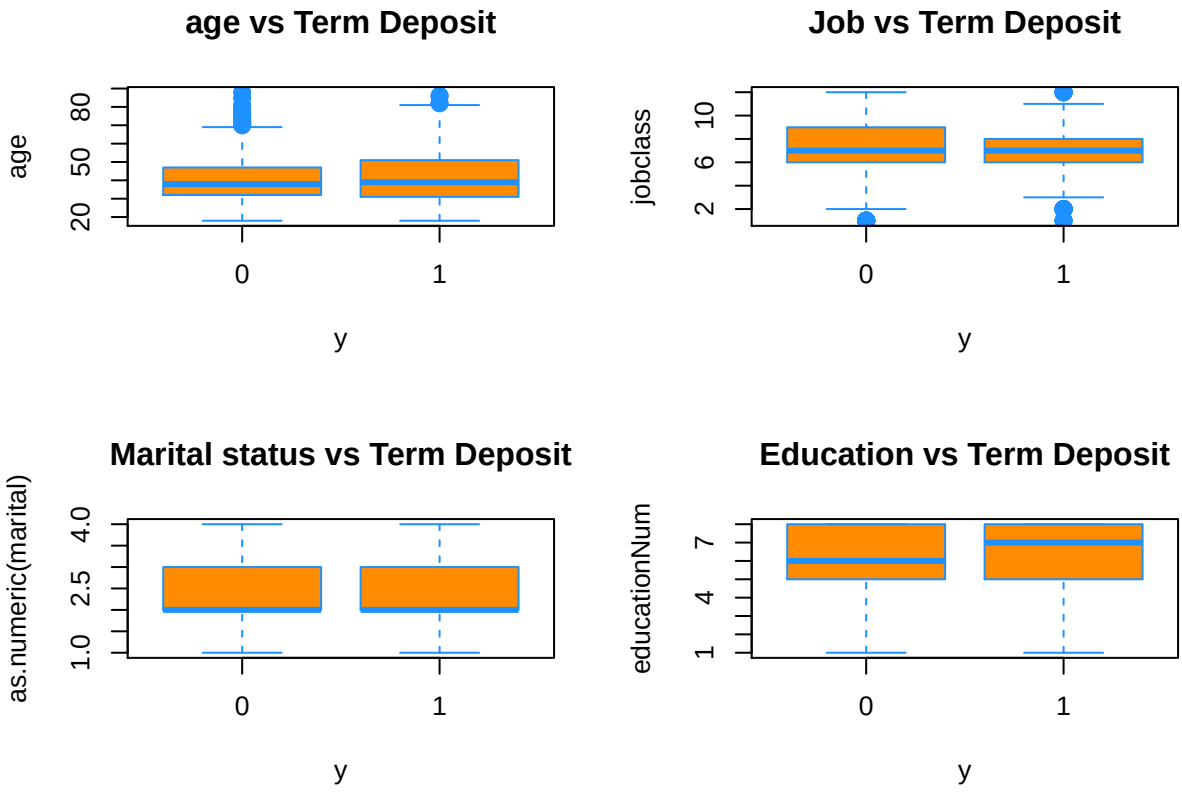
```
pairs(bank_df[c(1:4,21)], pch = 19, gap = .25, xaxt = "n", yaxt = "n"
      , col = c("#0080FF", "#F3953E")[bank_df$y]
      , label.pos = .5, oma = c(1, 1, 1, 1))
```



- pairs() function, which plots all possible scatterplots between pairs of variables in the dataset.
- There are no obvious colinearity between predictors `age`,`job`,`education` and `marital`.

```
par(mfrow=c(2,2))

#--- Factorizing the jobclass and education from lower to higher order

bank_df$jobclass = as.numeric(factor(bank_df$job
,levels=c("unknown","unemployed","housemaid", "student"
          ,"retired","technician","blue-collar","admin.","services",
"self-employed","management","entrepreneur"),ordered=TRUE))

bank_df$educationNum = as.numeric(factor(bank_df$education
,levels=c("unknown","illiterate","basic.4y","basic.6y","basic.9y"
,"high.school","professional.course", "university.degree"),ordered=TRUE))


boxplot(age~y,data=bank_df,main="age vs Term Deposit",
        pch=20,cex=2,col="darkorange",border = "dodgerblue")

boxplot(jobclass~y,data=bank_df,main="Job vs Term Deposit",
        pch=20,cex=2,col="darkorange",border = "dodgerblue")

boxplot(as.numeric(marital)~y,data=bank_df,
        main="Marital status vs Term Deposit",pch=20,cex=2,
```

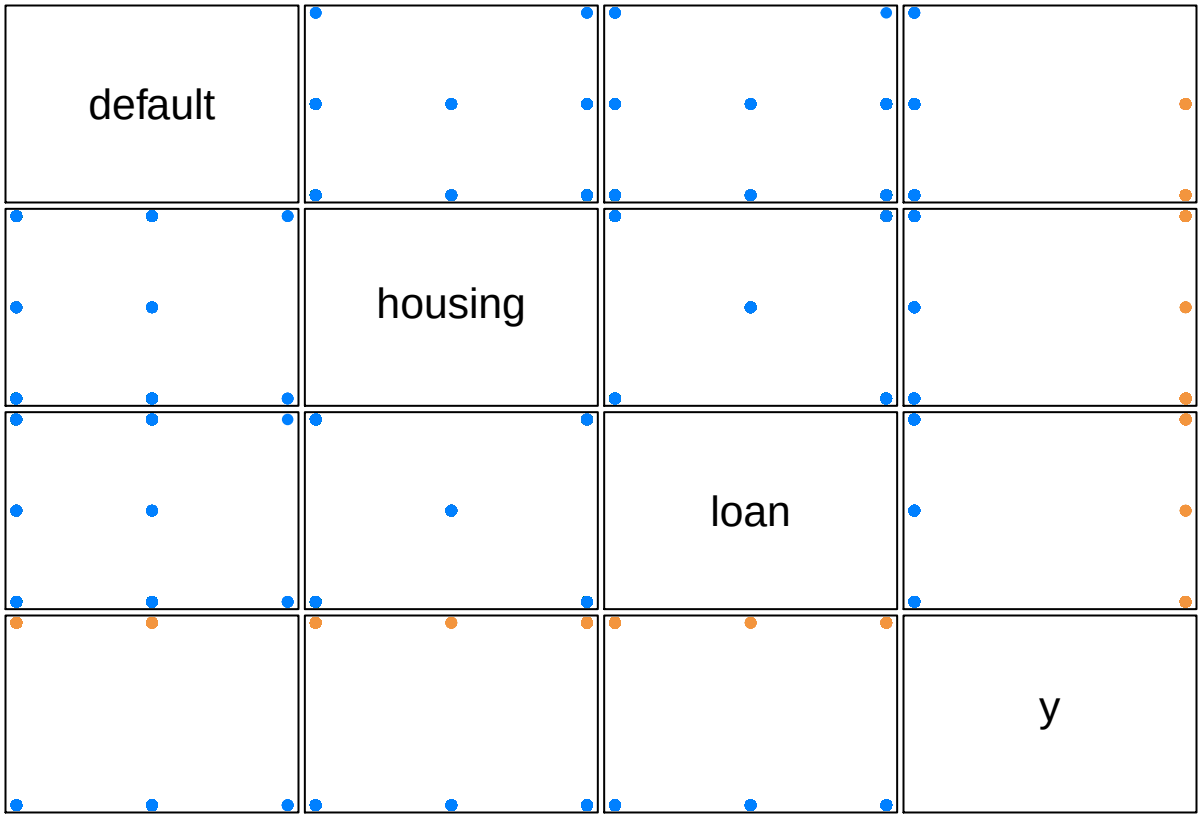
```
        col="darkorange",border = "dodgerblue")

boxplot(educationNum~y,data=bank_df,main="Education vs Term Deposit"
        ,pch=20,cex=2,col="darkorange",border = "dodgerblue")
```



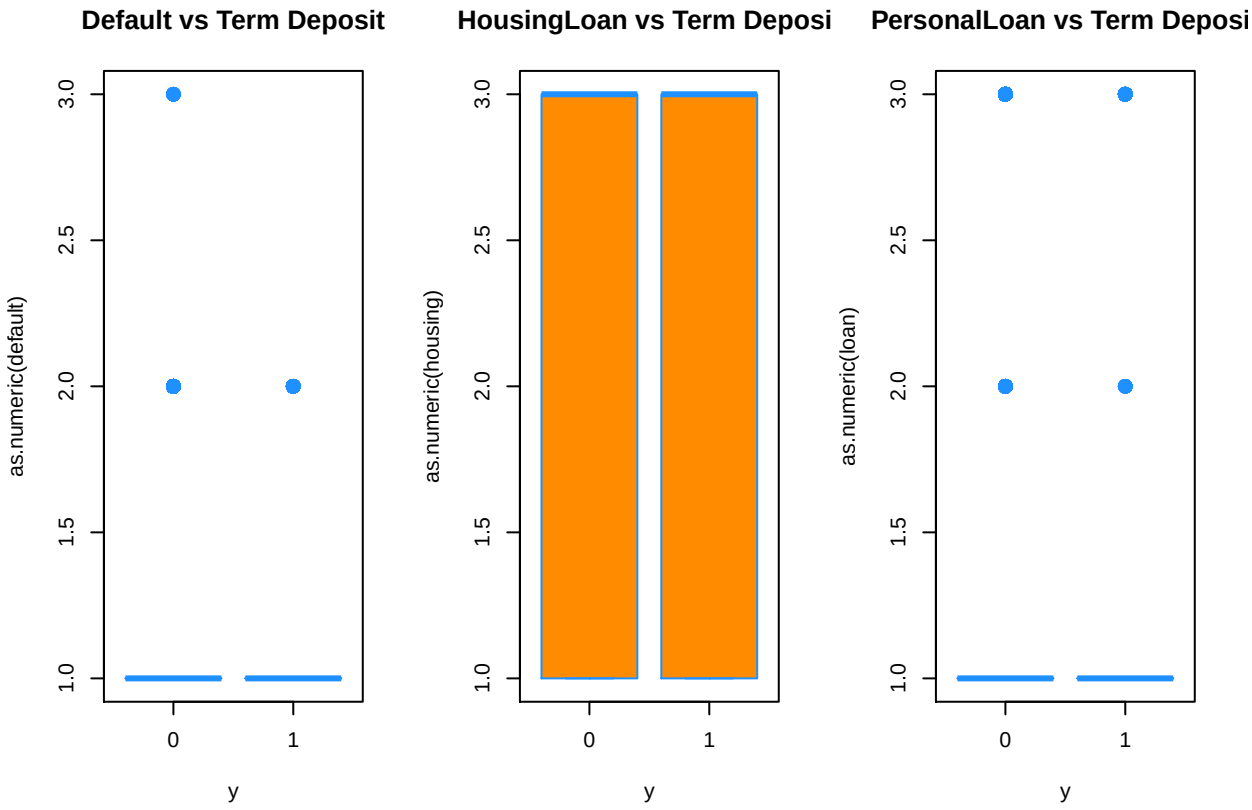
Based on the above bar plots of individual predictors against `Term Deposit`,

- It is evident that the Customers in the age group between 30 and 50 years subscribes to the term deposit than other age groups in the dataset.
- Customers with an education higher than high school takes the loan and subscribes to the term deposit vs not, but there is not a major difference between the subscription number.
- Customers who are married are considered in dataset more than others and they have similar numbers who subscribes vs not.
- Customers with job title as technician, blue-color, admin and work in services sector subscribes to the term deposit than others.
- So over all, It seems that age have some impact on term deposit subscription but need to see which exact range between 30-50 in detail. Also job class has some impact on Term deposit as well. Rest parameters does not have any significant effect based on plot.

```
pairs(bank_df[c(5:7,21)], pch = 19, gap = .25, xaxt = "n", yaxt = "n"
      , col = c("#0080FF", "#F3953E")[bank_df$y]
      , label.pos = .5, oma = c(1, 1, 1, 1))
```

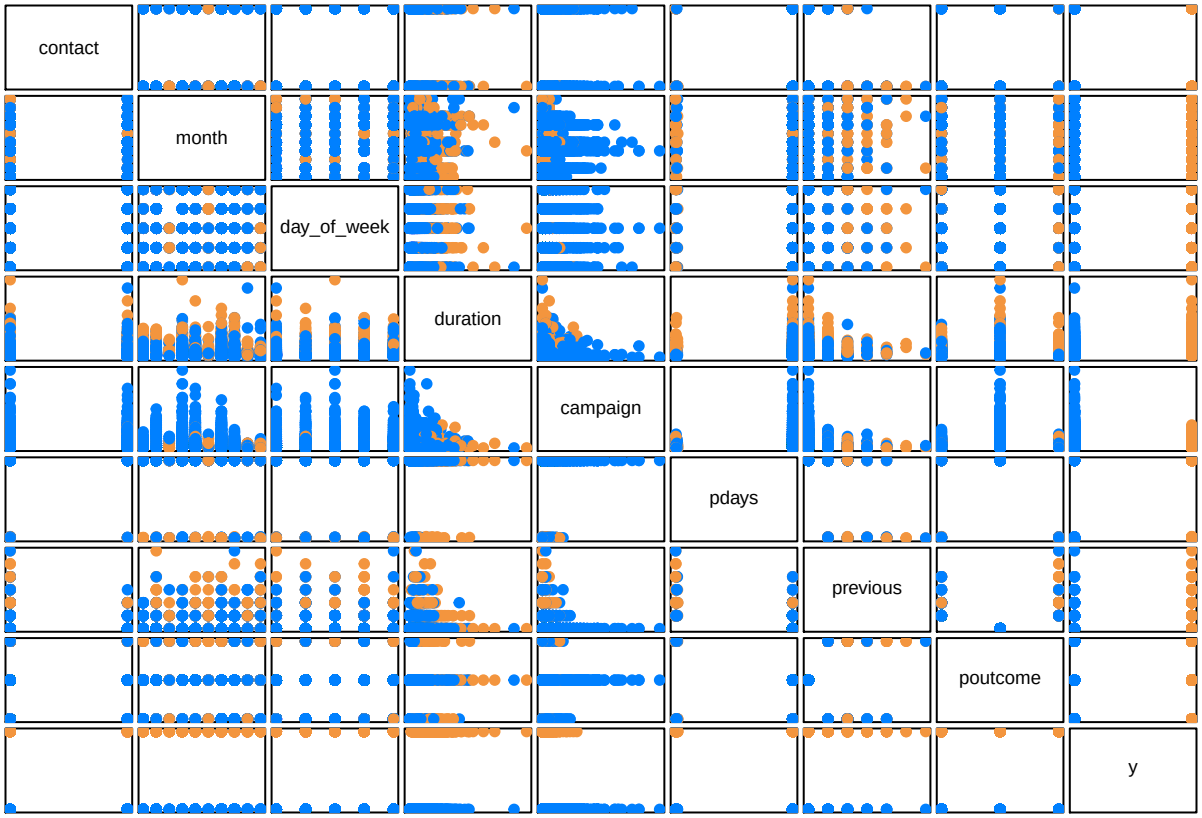



```
par(mfrow=c(1,3))
boxplot(as.numeric(default)~y,data=bank_df,main="Default vs Term Deposit",
        pch=20,cex=2,col="darkorange",border = "dodgerblue")

boxplot(as.numeric(housing)~y,data=bank_df,main="HousingLoan vs Term Deposit",
        pch=20,cex=2,col="darkorange",border = "dodgerblue")

boxplot(as.numeric(loan)~y,data=bank_df,main="PersonalLoan vs Term Deposit",
        pch=20,cex=2,col="darkorange",border = "dodgerblue")
```



- Customers with the housing loan and personal loan has no impact on term deposit subscription.
- Customers already in default certainly did not subscribed to the term deposit. So default can be a very good predictor.

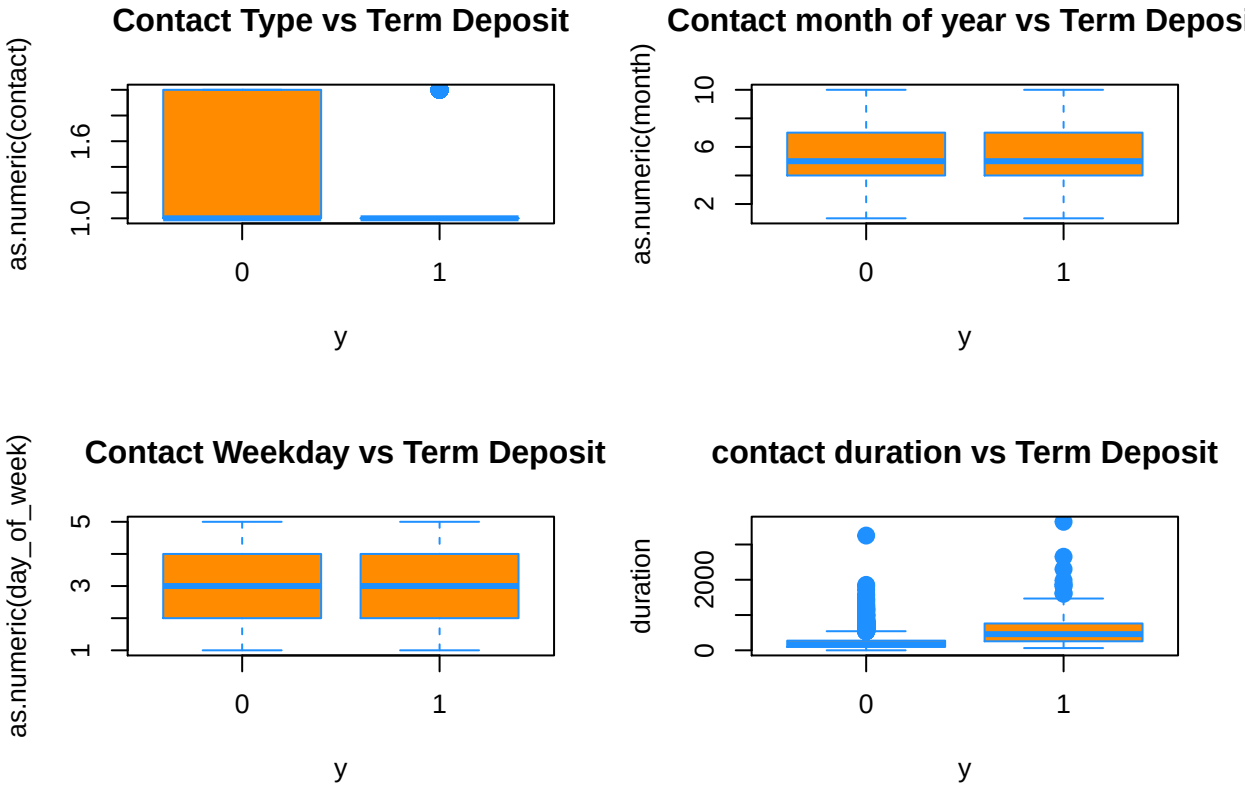
```
pairs(bank_df[c(8:15,21)], pch = 19, gap = .25, xaxt = "n", yaxt = "n"
      , col = c("#0080FF", "#F3953E")[bank_df$y]
      , label.pos = .5, oma = c(1, 1, 1, 1))
```




```
par(mfrow=c(2,2))

levels(bank_df$contact)
```

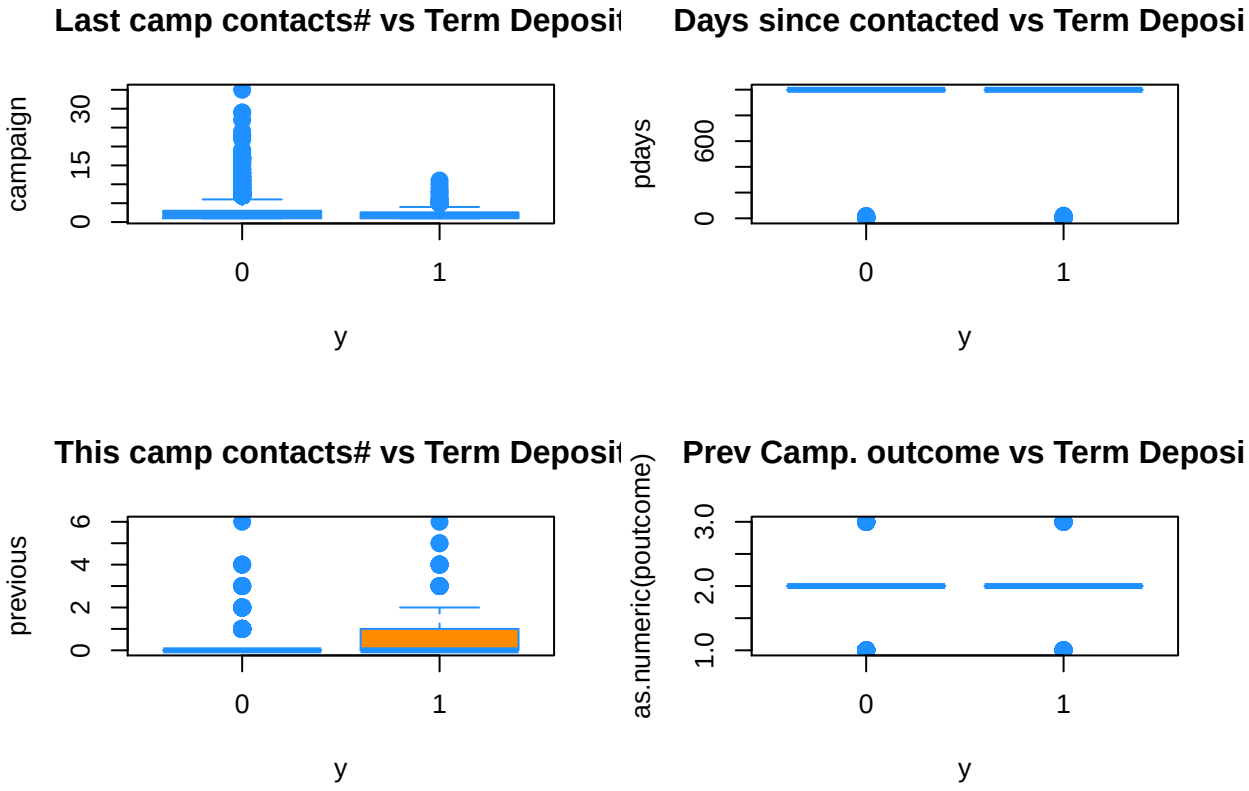
```
## [1] "cellular"  "telephone"
```

```
boxplot(as.numeric(contact)~y,data=bank_df,
        main="Contact Type vs Term Deposit",pch=20,cex=2,col="darkorange"
        ,border = "dodgerblue")

boxplot(as.numeric(month)~y,data=bank_df,
        main="Contact month of year vs Term Deposit",pch=20,cex=2,col="darkorange",
        border = "dodgerblue")

boxplot(as.numeric(day_of_week)~y,data=bank_df
        ,main=" Contact Weekday vs Term Deposit",
        pch=20,cex=2,col="darkorange",border = "dodgerblue")

boxplot(duration~y,data=bank_df,main="contact duration vs Term Deposit"
        ,pch=20,cex=2,col="darkorange",border = "dodgerblue")
```



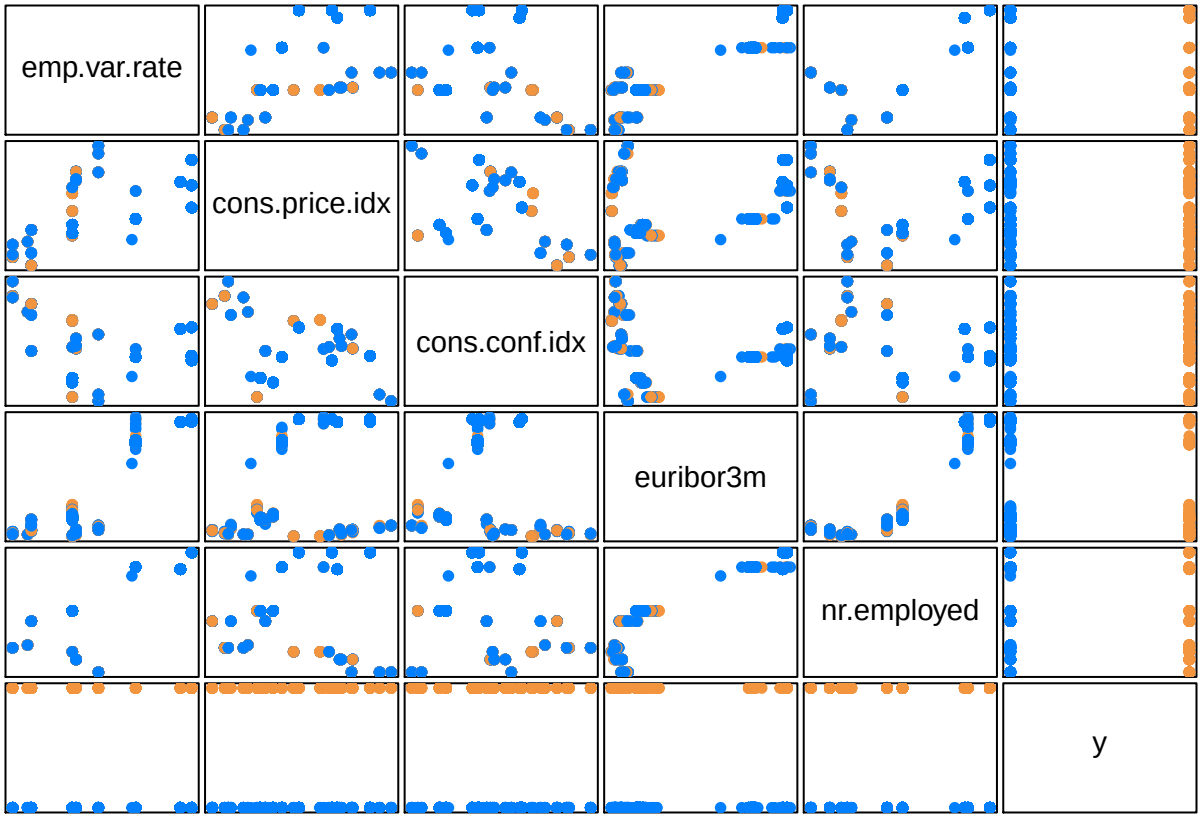
- Based on plots, Customer with contact type cellular has more probability of subscribing for term deposit than telephone. People who was recently contacted has more chance of subscribing for term deposit. Last contact day of the week does not have much impact.Contact duraction has some impact on the term deposit subscription as well.

```
par(mfrow=c(2,2))

boxplot(campaign~y,data=bank_df,main="Last camp contacts# vs Term Deposit",
        pch=20,cex=2,col="darkorange",border = "dodgerblue")

boxplot(pdays~y,data=bank_df,main="Days since contacted vs Term Deposit"
        ,pch=20,cex=2,col="darkorange",border = "dodgerblue")
boxplot(previous~y,data=bank_df,main="This camp contacts# vs Term Deposit",
        pch=20,cex=2,col="darkorange",border = "dodgerblue")
boxplot(as.numeric(poutcome)~y,data=bank_df
        ,main=" Prev Camp. outcome vs Term Deposit",
        pch=20,cex=2,col="darkorange",border = "dodgerblue")
```

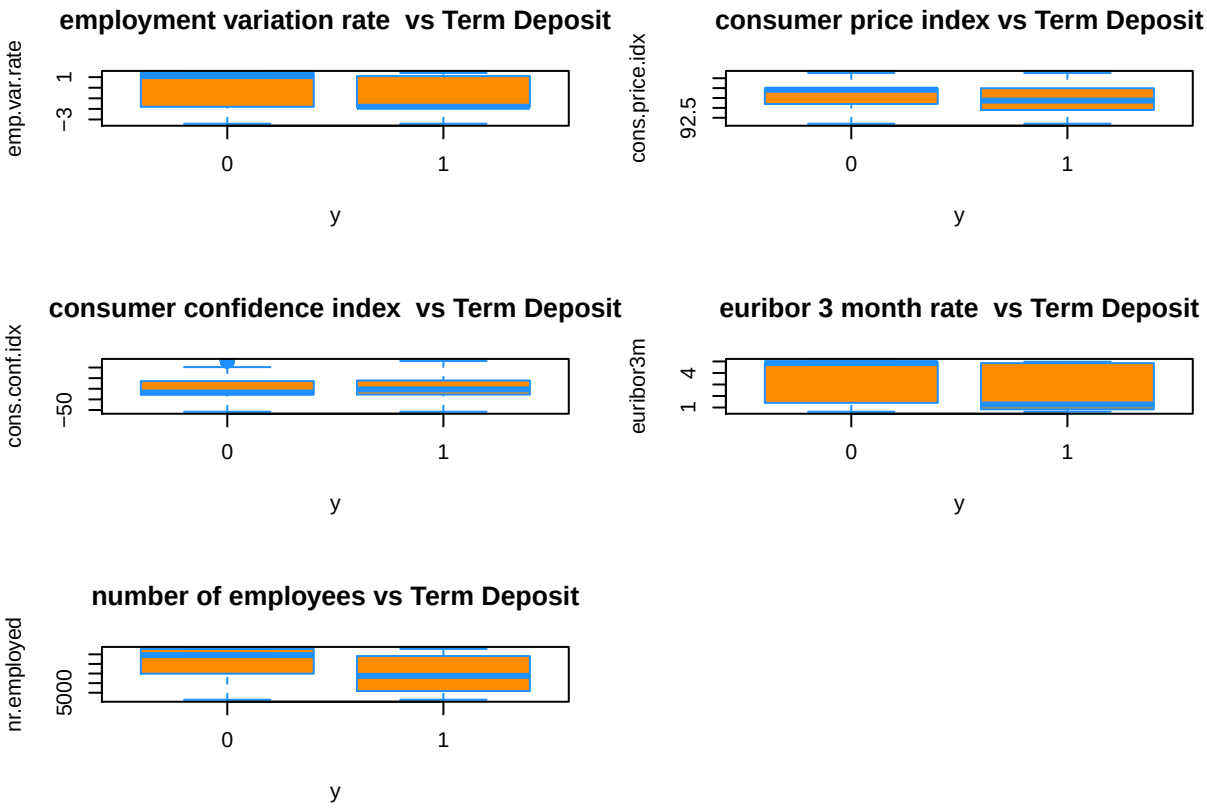


- Based on above plot, # of contact performed has some impact on Term deposit output else no other factor has significant impact.surprisingly # of contacts in this campaign decreases the probability to subscribe for term deposit.

```
pairs(bank_df[c(16:20,21)], pch = 19, gap = .25, xaxt = "n", yaxt = "n"
      , col = c("#0080FF", "#F3953E")[bank_df$y]
      , label.pos = .5, oma = c(1, 1, 1, 1))
```




```
par(mfrow=c(3,2))
boxplot(emp.var.rate~y,data=bank_df,main="employment variation rate  vs Term Deposit",
        pch=20,cex=2,col="darkorange",border = "dodgerblue")
boxplot(cons.price.idx~y,data=bank_df,main="consumer price index vs Term Deposit",
        pch=20,cex=2,col="darkorange",border = "dodgerblue")
boxplot(cons.conf.idx~y,data=bank_df,main="consumer confidence index  vs Term Deposit",
        pch=20,cex=2,col="darkorange",border = "dodgerblue")
boxplot(euribor3m~y,data=bank_df,main="euribor 3 month rate  vs Term Deposit",
        pch=20,cex=2,col="darkorange",border = "dodgerblue")
boxplot(nr.employed~y,data=bank_df,main="number of employees vs Term Deposit",
        pch=20,cex=2,col="darkorange",border = "dodgerblue")
```



- Based on plots all the market indicators has some impact on term deposit decision.

Class Imbalance Problem analysis

- We say that the dataset have the class imbalance problem, where the main class of interest is rare. That is, the data set distribution reflects a significant majority of the negative class and a minority positive class. Below table investigation shows that. we have `89` positive classes and `11%` negative classes.

```
prop.table(table(bank_df$y))
```

```
##
##         0         1
## 0.8904242 0.1095758
```

- we need to address this problem otherwise our model won't able to predict correctly. Most of the traditional classifiers assume dataset have balanced class distribution.
- There are techniques available to overcome this class imbalance problem. Lets consider few of those techniques are:
  - Oversampling: Re-sampling of data from positive class
  - Under-sampling: Randomly eliminate tuples from negative class

  - Threshold-moving: Move the decision threshold, so that the rare class tuples are easier to classify, and hence, less chance of costly false negative errors.

- We have used two library packages `DMwR` and `ROSE` which has `SMOTE` and `ROSE` functions

  - SMOTE: The general idea of this method is to artificially generate new examples of the minority class using the nearest neighbors of these cases. Furthermore, the majority class examples are also under-sampled, leading to a more balanced dataset.
  - ROSE: ROSE (Random Over-Sampling Examples) aids the task of binary classification in the presence of rare classes. It produces a synthetic, possibly balanced, sample of data simulated according to a smoothed-bootstrap approach.

Train/Test data split

- First split the dataset into train and test split using `createDataPartition` from `caret` package. The below r code splits the dataset with 70:30 ratio.

```
split<-createDataPartition(bank_df$y,p=0.7,list = FALSE)
train<-bank_df[split,]
test<-bank_df[-split,]
```

- Lets verify the class distribution in Training and Testing dataset. We see that our training and test dataset have same distribution of negative and positive classes.

```
prop.table(table(train$y))
```

```
##
##         0         1
## 0.8902735 0.1097265
```

```
prop.table(table(test$y))
```

```
##
##         0         1
## 0.8907767 0.1092233
```

- Lets apply the class imbalances improvement techniques which we discussed above and verify the results.

```
smote_train <- SMOTE(y ~ ., data = train)
rose_train <- ROSE(y ~ ., data = train)$data
```

```
prop.table(table(smote_train$y))
```

```
##
##         0         1
## 0.5714286 0.4285714
```

```r
prop.table(table(rose_train$y))
```

```
##
##         0         1
## 0.5060575 0.4939425
```

Model Builing

- In this study, we will be considering one regression model (Logistic Regression), one Ensemble model (Random Forest) and one Boosting model (Adaptive Boosting). We will be applying the techniques which we learnt in this course as well as others to better predict the chosen dataset.

Model 1: Logistic Regression

- As We have seen in boxplot comparison, age, default, contact type, last contact time, duration, # of contacts and all market indicators has some impact on terms deposit. Let's create a additive model using these parameters, full additive model and run AIC and BIC search to identify the model it comes up with.

```r
bank_lr = glm(y ~ ., data = train, family = "binomial")
summary(bank_lr)
```

Full addivite model

```
##
## Call:
## glm(formula = y ~ ., family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.8592  -0.2841  -0.1737  -0.1064   2.9274
##
## Coefficients: (3 not defined because of singularities)
##                               Estimate Std. Error z value Pr(>|z|)
## (Intercept)                 -1.815e+02  1.527e+02  -1.189  0.23460
## age                          1.122e-02  9.952e-03   1.128  0.25937
## jobblue-collar              -1.292e-01  3.353e-01  -0.385  0.69992
## jobentrepreneur             -5.279e-01  5.687e-01  -0.928  0.35330
## jobhousemaid                 2.183e-01  5.612e-01   0.389  0.69730
## jobmanagement               -6.311e-01  3.557e-01  -1.774  0.07605 .
## jobretired                   6.334e-02  4.273e-01   0.148  0.88218
## jobself-employed            -4.200e-01  4.765e-01  -0.882  0.37802
## jobservices                  5.719e-01  3.407e-01   1.679  0.09316 .
## jobstudent                   6.014e-01  4.640e-01   1.296  0.19496
## jobtechnician                3.647e-01  2.767e-01   1.318  0.18753
## jobunemployed               -2.638e-01  5.197e-01  -0.508  0.61164
## jobunknown                   1.383e-01  9.123e-01   0.152  0.87949
## maritalmarried               3.688e-02  2.823e-01   0.131  0.89604
```

```
## maritalsingle                 9.670e-02  3.240e-01   0.298  0.76533
## maritalunknown                2.123e-01  1.168e+00   0.182  0.85576
## educationbasic.6y             3.389e-01  5.038e-01   0.673  0.50117
## educationbasic.9y             2.208e-01  3.860e-01   0.572  0.56730
## educationhigh.school         -1.174e-01  3.723e-01  -0.315  0.75241
## educationilliterate          -1.191e+01  4.834e+02  -0.025  0.98033
## educationprofessional.course  3.033e-02  4.084e-01   0.074  0.94079
## educationuniversity.degree    6.809e-01  3.741e-01   1.820  0.06874 .
## educationunknown              3.339e-01  4.619e-01   0.723  0.46968
## defaultunknown                4.068e-02  2.586e-01   0.157  0.87500
## defaultyes                   -1.176e+01  5.013e+02  -0.023  0.98129
## housingunknown               -3.931e-01  6.154e-01  -0.639  0.52294
## housingyes                   -4.723e-03  1.689e-01  -0.028  0.97769
## loanunknown                         NA         NA      NA       NA
## loanyes                      -5.096e-01  2.460e-01  -2.072  0.03829 *
## contacttelephone             -1.166e+00  3.643e-01  -3.199  0.00138 **
## monthaug                      4.117e-01  5.305e-01   0.776  0.43775
## monthdec                      1.137e+00  7.689e-01   1.478  0.13934
## monthjul                     -1.282e-03  4.476e-01  -0.003  0.99772
## monthjun                      6.200e-01  5.350e-01   1.159  0.24649
## monthmar                      2.458e+00  5.991e-01   4.102 4.09e-05 ***
## monthmay                     -2.497e-01  3.608e-01  -0.692  0.48876
## monthnov                     -3.450e-01  5.091e-01  -0.678  0.49802
## monthoct                      6.756e-02  6.434e-01   0.105  0.91637
## monthsep                      2.263e-01  7.332e-01   0.309  0.75763
## day_of_weekmon                3.068e-01  2.661e-01   1.153  0.24898
## day_of_weekthu                2.255e-01  2.662e-01   0.847  0.39707
## day_of_weektue                1.785e-01  2.798e-01   0.638  0.52350
## day_of_weekwed                6.174e-01  2.791e-01   2.212  0.02698 *
## duration                      5.193e-03  3.130e-04  16.589  < 2e-16 ***
## campaign                     -9.487e-02  5.505e-02  -1.723  0.08482 .
## pdays                        -4.290e-05  8.325e-04  -0.052  0.95890
## previous                      2.107e-01  2.407e-01   0.875  0.38141
## poutcomenonexistent           5.695e-01  3.791e-01   1.502  0.13302
## poutcomesuccess               1.977e+00  8.214e-01   2.407  0.01607 *
## emp.var.rate                 -1.168e+00  5.748e-01  -2.032  0.04220 *
## cons.price.idx                1.697e+00  1.017e+00   1.668  0.09532 .
## cons.conf.idx                 4.499e-02  3.203e-02   1.405  0.16010
## euribor3m                     3.407e-02  5.019e-01   0.068  0.94587
## nr.employed                   3.668e-03  1.217e-02   0.302  0.76302
## jobclass                            NA         NA      NA       NA
## educationNum                        NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1998.9  on 2888  degrees of freedom
## Residual deviance: 1093.6  on 2836  degrees of freedom
## AIC: 1199.6
##
## Number of Fisher Scoring iterations: 14
```

```r
bank_lr_plotobs = glm(y~age+job+default+campaign+previous+contact
                      + month+duration+ pdays+emp.var.rate+cons.price.idx
                      +cons.conf.idx+euribor3m+nr.employed, data = train,
                      family = "binomial")
summary(bank_lr_plotobs)
```

Model based on data analysis from boxplot

```
##
## Call:
## glm(formula = y ~ age + job + default + campaign + previous +
##     contact + month + duration + pdays + emp.var.rate + cons.price.idx +
##     cons.conf.idx + euribor3m + nr.employed, family = "binomial",
##     data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.9225  -0.2858  -0.1796  -0.1209   2.7523
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -1.808e+02  1.445e+02  -1.251 0.210806
## age               8.309e-03  8.642e-03   0.961 0.336313
## jobblue-collar   -3.619e-01  2.628e-01  -1.377 0.168422
## jobentrepreneur  -5.743e-01  5.505e-01  -1.043 0.296869
## jobhousemaid      5.101e-02  5.252e-01   0.097 0.922632
## jobmanagement    -4.246e-01  3.396e-01  -1.250 0.211133
## jobretired       -2.257e-01  4.142e-01  -0.545 0.585778
## jobself-employed -3.592e-01  4.657e-01  -0.771 0.440553
## jobservices       1.708e-01  3.018e-01   0.566 0.571438
## jobstudent        1.715e-01  4.259e-01   0.403 0.687168
## jobtechnician     1.264e-01  2.483e-01   0.509 0.610819
## jobunemployed    -4.017e-01  4.936e-01  -0.814 0.415726
## jobunknown       -2.136e-01  8.773e-01  -0.244 0.807603
## defaultunknown   -2.845e-02  2.504e-01  -0.114 0.909543
## defaultyes       -1.197e+01  5.064e+02  -0.024 0.981139
## campaign         -9.940e-02  5.421e-02  -1.834 0.066720 .
## previous         -1.344e-01  1.477e-01  -0.910 0.363044
## contacttelephone -1.172e+00  3.561e-01  -3.292 0.000994 ***
## monthaug          5.019e-01  5.135e-01   0.977 0.328383
## monthdec          9.969e-01  7.466e-01   1.335 0.181803
## monthjul          5.227e-02  4.409e-01   0.119 0.905632
## monthjun          6.491e-01  5.104e-01   1.272 0.203429
## monthmar          2.449e+00  5.832e-01   4.198 2.69e-05 ***
## monthmay         -1.726e-01  3.527e-01  -0.489 0.624537
## monthnov         -2.881e-01  4.983e-01  -0.578 0.563086
## monthoct          1.211e-02  6.218e-01   0.019 0.984461
## monthsep          1.245e-01  7.017e-01   0.177 0.859157
## duration          5.045e-03  3.039e-04  16.600  < 2e-16 ***
## pdays            -1.723e-03  3.329e-04  -5.177 2.26e-07 ***
## emp.var.rate     -1.180e+00  5.495e-01  -2.147 0.031768 *
## cons.price.idx    1.776e+00  9.627e-01   1.845 0.065012 .
```

```
## cons.conf.idx     4.939e-02  3.046e-02   1.622 0.104906
## euribor3m         7.905e-02  4.831e-01   0.164 0.870013
## nr.employed       2.697e-03  1.155e-02   0.234 0.815316
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1998.9  on 2888  degrees of freedom
## Residual deviance: 1123.8  on 2855  degrees of freedom
## AIC: 1191.8
##
## Number of Fisher Scoring iterations: 14
```

Attribute selection We will be using step function to search forward and backward directions and use AIC and BIC as metric to find out the best model.

```
bank_lr_aic = step(bank_lr, direction = "backward", trace = 0)
summary(bank_lr_aic)
```

Model using AIC Search

```
##
## Call:
## glm(formula = y ~ loan + contact + month + duration + campaign +
##     poutcome + emp.var.rate + cons.price.idx + cons.conf.idx,
##     family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.8719  -0.2853  -0.1770  -0.1205   2.8830
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -1.351e+02  1.973e+01  -6.847 7.53e-12 ***
## loanunknown         -2.683e-01  5.747e-01  -0.467 0.640603
## loanyes             -4.610e-01  2.401e-01  -1.920 0.054838 .
## contacttelephone    -1.070e+00  3.199e-01  -3.344 0.000825 ***
## monthaug             4.948e-01  4.538e-01   1.090 0.275570
## monthdec             1.063e+00  6.851e-01   1.551 0.120828
## monthjul             1.371e-01  4.224e-01   0.325 0.745453
## monthjun             7.558e-01  3.697e-01   2.044 0.040950 *
## monthmar             2.424e+00  4.772e-01   5.081 3.75e-07 ***
## monthmay            -2.598e-01  3.243e-01  -0.801 0.423038
## monthnov            -2.709e-01  4.062e-01  -0.667 0.504793
## monthoct             8.760e-02  5.216e-01   0.168 0.866643
## monthsep             8.238e-02  5.320e-01   0.155 0.876949
## duration             5.053e-03  3.005e-04  16.814  < 2e-16 ***
## campaign            -9.180e-02  5.352e-02  -1.715 0.086331 .
## poutcomenonexistent  2.944e-01  2.367e-01   1.244 0.213452
```

```
## poutcomesuccess      1.943e+00  3.275e-01   5.933 2.97e-09 ***
## emp.var.rate        -9.246e-01  9.371e-02  -9.866  < 2e-16 ***
## cons.price.idx       1.420e+00  2.130e-01   6.664 2.66e-11 ***
## cons.conf.idx        4.743e-02  2.098e-02   2.260 0.023812 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1998.9  on 2888  degrees of freedom
## Residual deviance: 1124.1  on 2869  degrees of freedom
## AIC: 1164.1
##
## Number of Fisher Scoring iterations: 7
```

- The *best model based on AIC* include `contact`, `month`, `duration`, `campaign`, `poutcome`, `emp.var.rate`, `cons.price.idx`, `cons.conf.idx`

```
n=length(resid(bank_lr))
bank_lr_bic = step(bank_lr, direction = "backward", trace = 0,k=log(n))
bank_lr_bic
```

Model using BIC search

```
##
## Call:  glm(formula = y ~ contact + duration + pdays + emp.var.rate +
##     cons.price.idx + cons.conf.idx, family = "binomial", data = train)
##
## Coefficients:
##      (Intercept)  contacttelephone          duration             pdays
##       -1.394e+02        -1.036e+00         4.812e-03        -1.562e-03
##     emp.var.rate    cons.price.idx     cons.conf.idx
##       -9.337e-01         1.491e+00         6.152e-02
##
## Degrees of Freedom: 2888 Total (i.e. Null);  2882 Residual
## Null Deviance:       1999
## Residual Deviance: 1188  AIC: 1202
```

- The *best model based on BIC* include `contact`, `duration`, `poutcome`, `emp.var.rate`, `cons.price.idx`, `cons.conf.idx`
- The BIC model have lesser number of parameters compared to best AIC model. Let's do the Anova LRT test to find the preferred model.

```
result = data.frame(
        NullHypoThesis = c(
ifelse(anova(bank_lr_bic,bank_lr,test="LRT")[2,5]<0.05,"Rejected","Fail To Reject"),
ifelse(anova(bank_lr_aic,bank_lr,test="LRT")[2,5]<0.05,"Rejected","Fail To Reject"),
ifelse(anova(bank_lr_bic,bank_lr_aic,test="LRT")[2,5]<0.05,"Rejected","Fail To Reject"),
ifelse(anova(bank_lr_plotobs,bank_lr,test="LRT")[2,5]<0.05,"Rejected","Fail To Reject"),
```

```
ifelse(anova(bank_lr_aic,bank_lr_plotobs,test="LRT")[2,5]<0.05,"Rejected","Fail To Reject")),
          PValue = c((anova(bank_lr_bic,bank_lr,test="LRT")[2,5]),
           (anova(bank_lr_aic,bank_lr,test="LRT")[2,5]),
           (anova(bank_lr_bic,bank_lr_aic,test="LRT")[2,5]),
           anova(bank_lr_plotobs,bank_lr,test="LRT")[2,5],
           anova(bank_lr_aic,bank_lr_plotobs,test="LRT")[2,5]),
row.names = c("BIC vs Additive","AIC vs Additive ","BIC vs AIC",
              "PlotObs vs Additive","AIC vs PlotObs")
)

colnames(result) = c("Null Hypothesis","P-Value")

kable(result) %>%
  kable_styling(bootstrap_options = c("striped", "hover"),latex_options = "scale_down")
```

	Null Hypothesis	P-Value
BIC vs Additive	Rejected	0.0000342
AIC vs Additive	Fail To Reject	0.5936991
BIC vs AIC	Rejected	0.0000000
PlotObs vs Additive	Rejected	0.0489034
AIC vs PlotObs	Fail To Reject	1.0000000

- Based on Anova LRT test above, only AIC model is the preferable model among all model.
- Let's compare Null deviance, Residual Deviance, AIC and 5 fold misclassification rate to see which is the better model.

```
#train = na.omit(train)
#train = train[complete.cases(train), ]
#train = train[,-5]
set.seed(1983)
mcRate_bank_lr = cv.glm(train,bank_lr,K=5)$delta[1]
mcRate_bank_lr_aic = cv.glm(train,bank_lr_aic,K=5)$delta[1]
mcRate_bank_lr_bic = cv.glm(train,bank_lr_bic,K=5)$delta[1]
mcRate_bank_lr_plotobs = cv.glm(train,bank_lr_plotobs,K=5)$delta[1]



result = data.frame(
  Residual_Deviance=c(bank_lr$deviance,bank_lr_aic$deviance,bank_lr_bic$deviance
                      ,bank_lr_plotobs$deviance),
  Residual_DF=c(bank_lr$df.residual,bank_lr_aic$df.residual,bank_lr_bic$df.residual
                  ,bank_lr_plotobs$df.residual),
  Null_Deviance=c(bank_lr$null.deviance,bank_lr_aic$null.deviance,bank_lr_bic$null.deviance
                  ,bank_lr_plotobs$null.deviance),
  Null_DF=c(bank_lr$df.null,bank_lr_aic$df.null,bank_lr_bic$df.null,bank_lr_plotobs$df.null),
```

```
  AIC=c(AIC(bank_lr),AIC(bank_lr_aic),AIC(bank_lr_bic),AIC(bank_lr_plotobs)),
  MisclassificationRate=c(mcRate_bank_lr,mcRate_bank_lr_aic,mcRate_bank_lr_bic
                          ,mcRate_bank_lr_plotobs),
  row.names = c("Full Model","Model Selected with AIC","Model Selected with BIC"
                ,"Model built based on boxplot observation")
)

colnames(result) = c("Residual Deviance","Residual DF","Null Deviance","Null DF","AIC "
                     ,"Misclassification Rate")

kable(result) %>%
  kable_styling(bootstrap_options = c("striped", "hover"),latex_options = "scale_down")
```

	Residual Deviance	Residual DF	Null Deviance	Null DF	AIC	Misclassification Rate
Full Model	1093.590	2836	1998.86	2888	1199.590	0.0626113
Model Selected with AIC	1124.059	2869	1998.86	2888	1164.059	0.0595994
Model Selected with BIC	1188.013	2882	1998.86	2888	1202.013	0.0613051
Model built based on boxplot observation	1123.823	2855	1998.86	2888	1191.823	0.0610390

- Based on the table above, AIC model has the lowest Residual Deviance after Full model but degree of freedom is more closer to Null Deviance degree of freedom than Full Model. AIC of the AIC model and misclassification rate is also least.

Can we improve it further by applying the transformation or interaction on AIC model? Let's check.

```
bank_lr_aic_int = glm(y ~ (contact + month + duration + campaign
                           + poutcome + cons.conf.idx + nr.employed
                           +contact:month+campaign:poutcome)
                      , family = binomial(link="logit"), data = train)
```

Apply two way interaction

Apply Log Transformation on one of the predictor

- We have chosen the predictor `nr.employed` since it is numerical attribute varying from small to large numbers.

```
range(bank_df$nr.employed)
```

```
## [1] 4963.6 5228.1
```

```
bank_lr_aic_transform = glm(y ~ (contact + month + duration + campaign
    + poutcome + cons.conf.idx + log(nr.employed)),
    family = binomial(link="logit"), data = train)
```

Use Anova LRT test to compare the aic additive model vs aic interaction model

```r
anova(bank_lr_aic,bank_lr_aic_int,test="LRT")
```

```
## Analysis of Deviance Table
##
## Model 1: y ~ loan + contact + month + duration + campaign + poutcome +
##     emp.var.rate + cons.price.idx + cons.conf.idx
## Model 2: y ~ (contact + month + duration + campaign + poutcome + cons.conf.idx +
##     nr.employed + contact:month + campaign:poutcome)
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1      2869     1124.1
## 2      2861     1122.2  8   1.8124   0.9862
```

- Since p-value is very large so we fail to reject the null hypothesis and we stick with our AIC additive model

Use Anova LRT test to compare the aic additive model vs aic log model

```r
anova(bank_lr_aic,bank_lr_aic_transform,test="LRT")
```

```
## Analysis of Deviance Table
##
## Model 1: y ~ loan + contact + month + duration + campaign + poutcome +
##     emp.var.rate + cons.price.idx + cons.conf.idx
## Model 2: y ~ (contact + month + duration + campaign + poutcome + cons.conf.idx +
##     log(nr.employed))
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1      2869     1124.1
## 2      2872     1132.2 -3  -8.0965  0.04406 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- At significant level `0.01`, so we fail to reject the null hypothesis and we stick with our AIC additive model

Let's do 5 fold cross validation and calculate misclassification rate to compare and select the model

```r
set.seed(1250)

mcRate_bank_lr_aic_int = cv.glm(train,bank_lr_aic_int,K=5)$delta[1]
mcRate_bank_lr_aic_transform = cv.glm(train,bank_lr_aic_transform,K=5)$delta[1]

result = data.frame(
  Residual_Deviance=c(bank_lr$deviance,bank_lr_aic$deviance,bank_lr_bic$deviance
                      ,bank_lr_plotobs$deviance,bank_lr_aic_int$deviance
                      ,bank_lr_aic_transform$deviance),

  Residual_DF=c(bank_lr$df.residual,bank_lr_aic$df.residual,bank_lr_bic$df.residual
                ,bank_lr_plotobs$df.residual,bank_lr_aic_int$df.residual
                ,bank_lr_aic_transform$df.residual),
```

```
  Null_Deviance=c(bank_lr$null.deviance,bank_lr_aic$null.deviance,bank_lr_bic$null.deviance
                  ,bank_lr_plotobs$null.deviance,bank_lr_aic_int$null.deviance
                  ,bank_lr_aic_transform$null.deviance),

  Null_DF=c(bank_lr$df.null,bank_lr_aic$df.null,bank_lr_bic$df.null,bank_lr_plotobs$df.null
            ,bank_lr_aic_int$df.null,bank_lr_aic_transform$df.null),

  AIC=c(AIC(bank_lr),AIC(bank_lr_aic),AIC(bank_lr_bic),AIC(bank_lr_plotobs)
        ,AIC(bank_lr_aic_int),AIC(bank_lr_aic_transform)),

  MisclassificationRate=c(mcRate_bank_lr,mcRate_bank_lr_aic,mcRate_bank_lr_bic
        ,mcRate_bank_lr_plotobs,mcRate_bank_lr_aic_int,mcRate_bank_lr_aic_transform),

  row.names = c("Full Model","Model Selected with AIC","Model Selected with BIC"
                ,"Model built based on boxplot observation","Model with an interaction"
                ,"Model with transformation")
)

colnames(result) = c("Residual Deviance","Residual DF","Null Deviance","Null DF"
                     ,"AIC ","Misclassification Rate")

kable(result) %>%
  kable_styling(bootstrap_options = c("striped", "hover"),latex_options = "scale_down")
```

	Residual Deviance	Residual DF	Null Deviance	Null DF	AIC	Misclassification Rate
Full Model	1093.590	2836	1998.86	2888	1199.590	0.0626113
Model Selected with AIC	1124.059	2869	1998.86	2888	1164.059	0.0595994
Model Selected with BIC	1188.013	2882	1998.86	2888	1202.013	0.0613051
Model built based on boxplot observation	1123.823	2855	1998.86	2888	1191.823	0.0610390
Model with an interaction	1122.247	2861	1998.86	2888	1178.247	0.0620511
Model with transformation	1132.156	2872	1998.86	2888	1166.156	0.0608127

- As you see, Residual deviance goes down with integration but AIC goes up and degrees of freedom goes down.Misclassification rate is marginally changed with transformation. But we dont see any major difference which can be considered but residual deviance and AIC has gone down with gaining a degree of freedom.
- Let's consider model selected with AIC on test data to identify misclassification rate, false positives and negatives on Test data.

```
predict_bank_lr_aic = ifelse(predict(bank_lr_aic, test, type = "response")> 0.5,1,0)
misClassificationRate = mean(predict_bank_lr_aic!=test$y)
misClassificationRate
```

```
## [1] 0.08980583
```

```
make_conf_mat = function(predicted, actual) {
  table(predicted = predicted, actual = actual)
}

confusion_matrix = table(predicted = predict_bank_lr_aic, actual = test$y)
sensitivity = confusion_matrix[2,2]/confusion_matrix[,2]
specificity = confusion_matrix[1,1]/confusion_matrix[,1]

confusion_matrix
```

```
##          actual
## predicted    0    1
##         0 1072   82
##         1   29   53
```

```
sensitivity
```

```
##         0         1
## 0.6463415 1.0000000
```

```
specificity
```

```
##        0        1
##  1.00000 36.96552
```

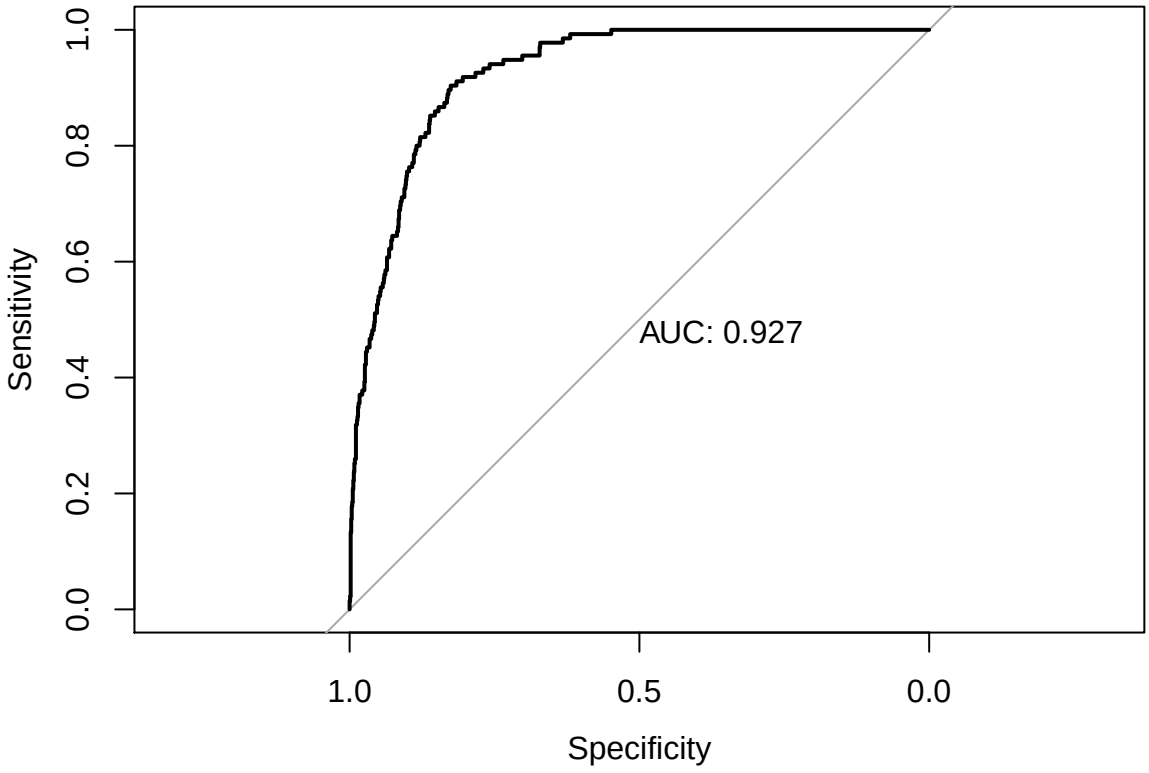
- With cutoff of 0.5 probability , we see 29 false positives and 82 false negatives among 1236 observations which is less than 10%. We can try to identify cut off where this number goes down. Let's draw ROC curve and identify AUC.

```
get_logistic_pred = function(mod, data, res = "y", pos = 1, neg = 0, cut = 0.5) {
  probs = predict(mod, newdata = data, type = "response")
  ifelse(probs > cut, pos, neg)
}

test_pred_10 = get_logistic_pred(bank_lr_aic, data = test, res = "default",
                                 pos = 1, neg = 0, cut = 0.1)
test_pred_30 = get_logistic_pred(bank_lr_aic, data = test, res = "default",
                                 pos = 1, neg = 0, cut = 0.3)
test_pred_40 = get_logistic_pred(bank_lr_aic, data = test, res = "default",
                                 pos = 1, neg = 0, cut = 0.4)
test_pred_50 = get_logistic_pred(bank_lr_aic, data = test, res = "default",
                                 pos = 1, neg = 0, cut = 0.5)
test_pred_60 = get_logistic_pred(bank_lr_aic, data = test, res = "default",
                                 pos = 1, neg = 0, cut = 0.6)
test_pred_90 = get_logistic_pred(bank_lr_aic, data = test, res = "default",
                                 pos = 1, neg = 0, cut = 0.9)

test_tab_10 = table(predicted = test_pred_10, actual = test$y)
test_tab_30 = table(predicted = test_pred_30, actual = test$y)
test_tab_40 = table(predicted = test_pred_40, actual = test$y)
test_tab_50 = table(predicted = test_pred_50, actual = test$y)
test_tab_60 = table(predicted = test_pred_60, actual = test$y)
test_tab_90 = table(predicted = test_pred_90, actual = test$y)

test_con_mat_10 = confusionMatrix(test_tab_10, positive = "1")
test_con_mat_30 = confusionMatrix(test_tab_30, positive = "1")
test_con_mat_40 = confusionMatrix(test_tab_40, positive = "1")
test_con_mat_50 = confusionMatrix(test_tab_50, positive = "1")
test_con_mat_60 = confusionMatrix(test_tab_60, positive = "1")
test_con_mat_90 = confusionMatrix(test_tab_90, positive = "1")

metrics = data.frame(
```

```
  Accuracy = c(test_con_mat_10$overall["Accuracy"], test_con_mat_30$overall["Accuracy"]
               ,test_con_mat_40$overall["Accuracy"], test_con_mat_50$overall["Accuracy"]
               ,test_con_mat_60$overall["Accuracy"], test_con_mat_90$overall["Accuracy"]),

  Sensitivity = c(test_con_mat_10$byClass["Sensitivity"],test_con_mat_30$byClass["Sensitivity"]
                ,test_con_mat_40$byClass["Sensitivity"],test_con_mat_50$byClass["Sensitivity"]
                , test_con_mat_60$byClass["Sensitivity"], test_con_mat_90$byClass["Sensitivity"]),

  Specificity = c(test_con_mat_10$byClass["Specificity"],test_con_mat_30$byClass["Specificity"]
                ,test_con_mat_40$byClass["Specificity"],test_con_mat_50$byClass["Specificity"]
                ,test_con_mat_60$byClass["Specificity"],test_con_mat_90$byClass["Specificity"])

)

rownames(metrics) = c("c = 0.10", "c = 0.30","c = 0.40","c = 0.50","c = 0.60", "c = 0.90")


colnames(metrics) = c("Accuracy","Sensitivity","Specificity")

kable(metrics) %>%
  kable_styling(bootstrap_options = c("striped", "hover"),latex_options = "scale_down")
```

	Accuracy	Sensitivity	Specificity
c = 0.10	0.8365696	0.8814815	0.8310627
c = 0.30	0.8996764	0.5629630	0.9409628
c = 0.40	0.9085761	0.4814815	0.9609446
c = 0.50	0.9101942	0.3925926	0.9736603
c = 0.60	0.9158576	0.3555556	0.9845595
c = 0.90	0.9012945	0.1111111	0.9981835

```
test_prob = predict(bank_lr_aic, newdata = test, type = "response")
test_roc = roc(test$y ~ test_prob, plot = TRUE, print.auc = TRUE)
```




```
auc(test_roc)
```

```
## Area under the curve: 0.9266
```

- With the best chosen model from logistic regression, we have AUC above 0.9 and the ratio of Sensitivity and Specificity is good as well. But as you can see sensitivity is decreasing as cut-off is increasing from 0.10 onward. Unlike sensitivity, specificity and accuracy is increasing as cut-off is increasing from 0.10 onward. But at cut-of 0.10, all 3 have the best numbers possible.

Model 2: Random Forest

- We have chosen Random Forest model (ensemble) to do binary classification to predict whether customer will subscribe to term deposit or not.

model using all predictors

```
model_rf<-randomForest(y ~ .  , data=train, mtry=3)
```

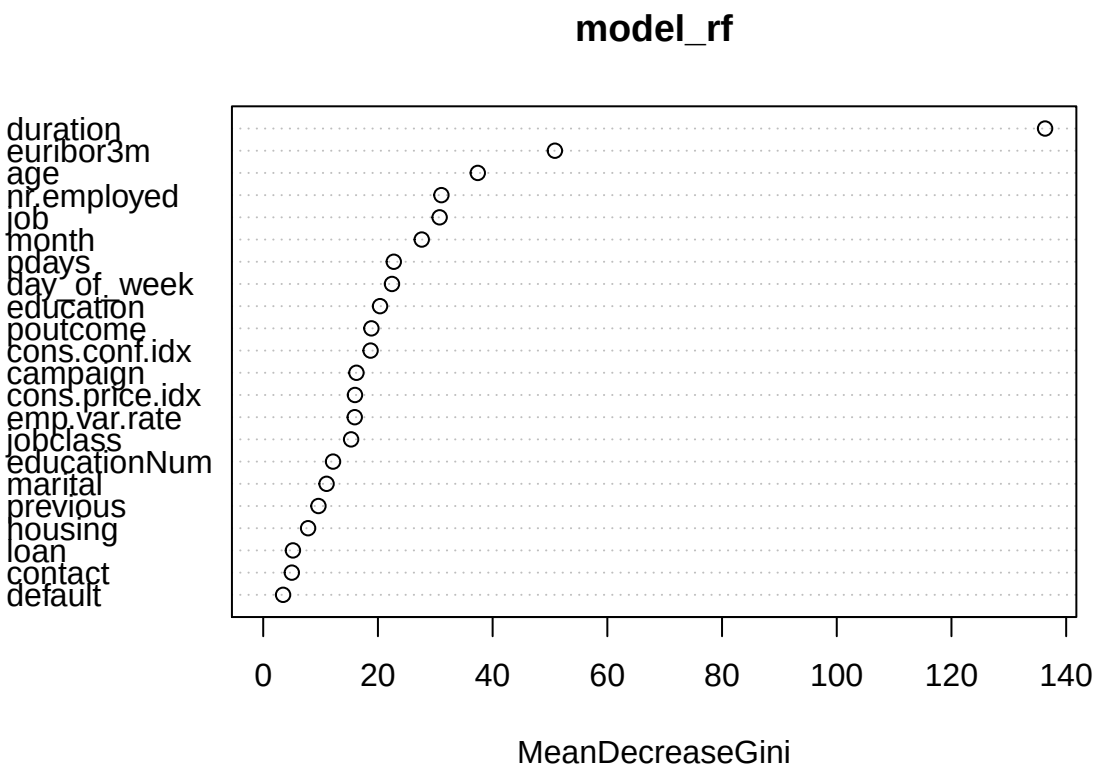
- Fitted model details

```
model_rf
```

```
##
## Call:
##  randomForest(formula = y ~ ., data = train, mtry = 3)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 8.58%
## Confusion matrix:
##      0   1 class.error
## 0 2504  68  0.02643857
## 1  180 137  0.56782334
```

- Dotchart of variable importance as measured by a Random Forest

```
varImpPlot(model_rf)
```



- Confusion Matrix for Training Dataset

```
pred_train<-predict(model_rf,train)
confusionMatrix(train$y,pred_train)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2572    0
##          1   10  307
##
##                Accuracy : 0.9965
##                  95% CI : (0.9936, 0.9983)
##     No Information Rate : 0.8937
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.982
##
##  Mcnemar's Test P-Value : 0.004427
##
##             Sensitivity : 0.9961
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.9685
##              Prevalence : 0.8937
##          Detection Rate : 0.8903
##    Detection Prevalence : 0.8903
##       Balanced Accuracy : 0.9981
##
##        'Positive' Class : 0
##
```

- Confusion Matrix for Test Dataset

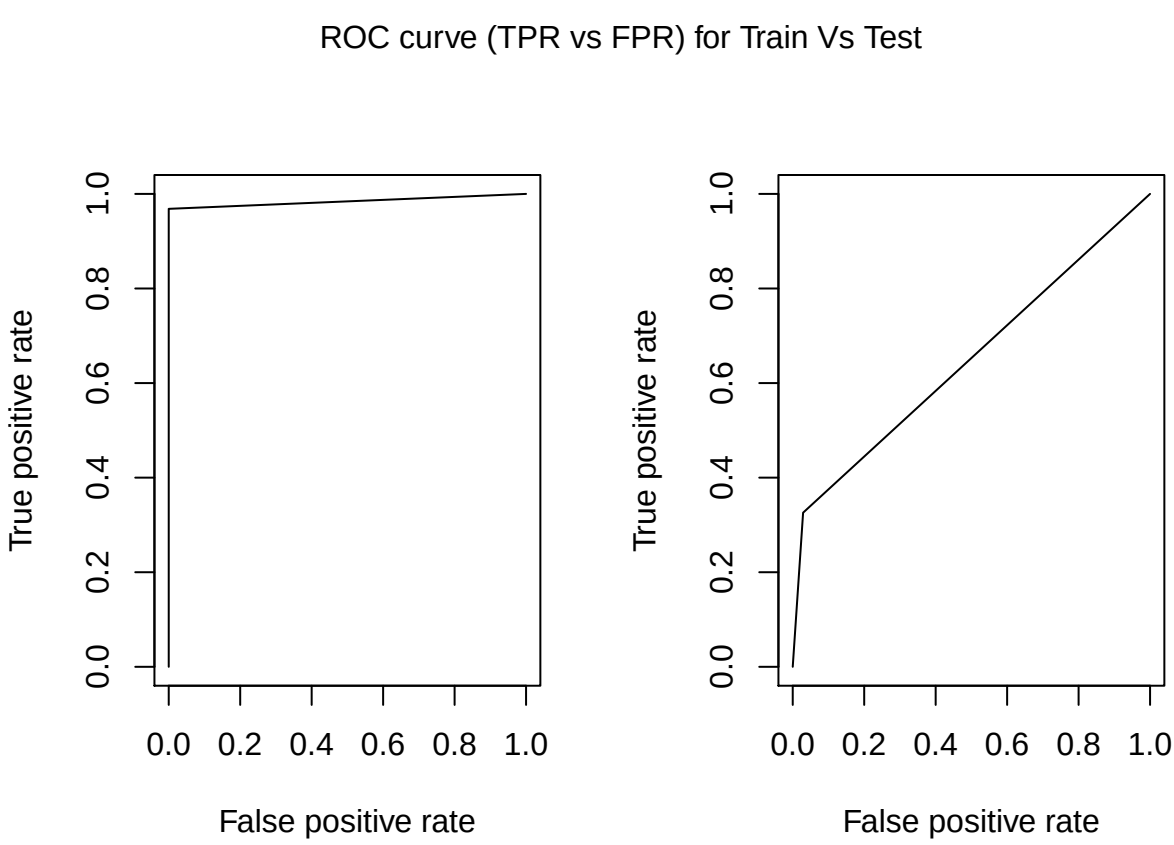
```
pred_test1<-predict(model_rf,test)
confusionMatrix(test$y,pred_test1)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1069   32
##          1   91   44
##
##                Accuracy : 0.9005
##                  95% CI : (0.8824, 0.9166)
##     No Information Rate : 0.9385
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3673
##
##  Mcnemar's Test P-Value : 1.698e-07
##
##             Sensitivity : 0.9216
##             Specificity : 0.5789
##          Pos Pred Value : 0.9709
##          Neg Pred Value : 0.3259
```

```
##              Prevalence : 0.9385
##          Detection Rate : 0.8649
##    Detection Prevalence : 0.8908
##       Balanced Accuracy : 0.7502
##
##        'Positive' Class : 0
##
```

Plot ROC curve (TPR vs FPR)

```
par(mfrow = c(1,2))
pr_train = prediction(as.numeric(pred_train),train$y)
prf_train = performance(pr_train, measure = "tpr", x.measure = "fpr")
plot(prf_train)

pr_test1 = prediction(as.numeric(pred_test1),test$y)
prf_test1 = performance(pr_test1, measure = "tpr",x.measure = "fpr")
plot(prf_test1)

mtext("ROC curve (TPR vs FPR) for Train Vs Test", side = 3, line = -1, outer = TRUE)
```




```
auc_train = performance(pr_train, measure = "auc")
auc_test1 = performance(pr_test1, measure = "auc")
```

Tuning Random Forest Model

- Need to find out optimum varible split for each decision tree by running cross validation. In general, `mtry` value is $\sqrt{(p)}$ where p is total number of predictors. We would use train function from caret library to find out the optimum split.

```
oob = trainControl(method = "oob")
cv_5 = trainControl(method = "cv", number = 5)

rf_mtry = ceiling(sqrt(20))
rf_grid =  expand.grid(mtry = 1:rf_mtry)
set.seed(1)
bank_rf_tune = train(y ~ . , data = train,
                     method = "rf",
                     trControl = oob,
                     verbose = FALSE,
                     tuneGrid = rf_grid)
bank_rf_tune
```

```
## Random Forest
##
## 2889 samples
##   22 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##   1     0.8902735  0.0000000
##   2     0.9023884  0.2331446
##   3     0.9055036  0.3006660
##   4     0.9079266  0.3607758
##   5     0.9065421  0.3900621
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 4.
```

```
model_rf_best<-randomForest(y ~ .  , data=train, mtry=bank_rf_tune$bestTune$mtry[[1]])
```

```
pred_train<-predict(model_rf_best,train)
pred_test1<-predict(model_rf_best,test)
train_cm = confusionMatrix(train$y,pred_train)
test1_cm = confusionMatrix(test$y,pred_test1)

metrics = cbind(train_cm$byClass, test1_cm$byClass)
acc = cbind(train_cm$overall["Accuracy"],test1_cm$overall["Accuracy"])
auc = cbind(auc_train@y.values,auc_test1@y.values)
rownames(auc) = c("Area Under Curve")
result = data.frame(rbind(acc,auc,metrics))
```

```
colnames(result) = c("Train","Test")
kable(result) %>%
  kable_styling(bootstrap_options = c("striped", "hover"),latex_options = "scale_down")
```

	Train	Test
Accuracy	0.999653859466944	0.899676375404531
Area Under Curve	0.984227129337539	0.648430719547886
Sensitivity	0.999611348620288	0.925893635571055
Specificity	1	0.361797752808989
Pos Pred Value	1	0.964577656675749
Neg Pred Value	0.996845425867507	0.370370370370371
Precision	1	0.964577656675749
Recall	0.999611348620288	0.925893635571055
F1	0.99980563654033	0.944839857651246
Prevalence	0.890619591554171	0.927993527508091
Detection Rate	0.890273451021115	0.859223300970874
Detection Prevalence	0.890273451021115	0.890776699029126
Balanced Accuracy	0.999805674310144	0.743845694190022

```
# to be pulled by final result session
test1_cm_rf = test1_cm
auc_test1_rf = auc_test1
```

Metrics for best Random Forest Model

Model 3: Adaptive Boosting

- AdaBoost is an ensemble learning method (also known as "meta-learning") which was initially created to increase the efficiency of binary classifiers. AdaBoost uses an iterative approach to learn from the mistakes of weak classifiers, and turn them into strong ones. We will be using the library `ada` for the same. This model have built in feature selection.
- Build the model with default parameter values

```
bank_model_ada = ada(y ~ ., data=train,loss='exponential', type='discrete', iter = 50)
```

```
bank_model_ada$confusion
```

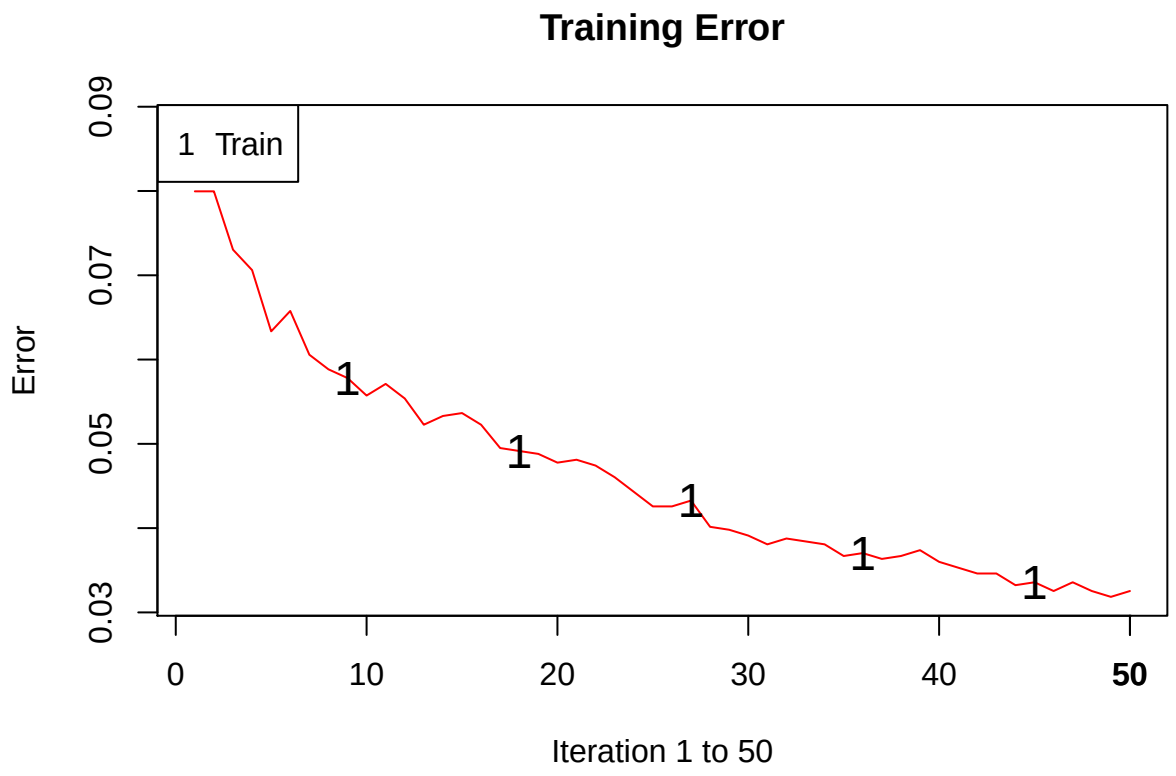
```
##            Final Prediction
## True value    0    1
##          0 2555   17
##          1   77  240
```

```
(bank_model_ada$confusion[2,2]/sum(bank_model_ada$confusion[,2]))
```

```
## [1] 0.9338521
```

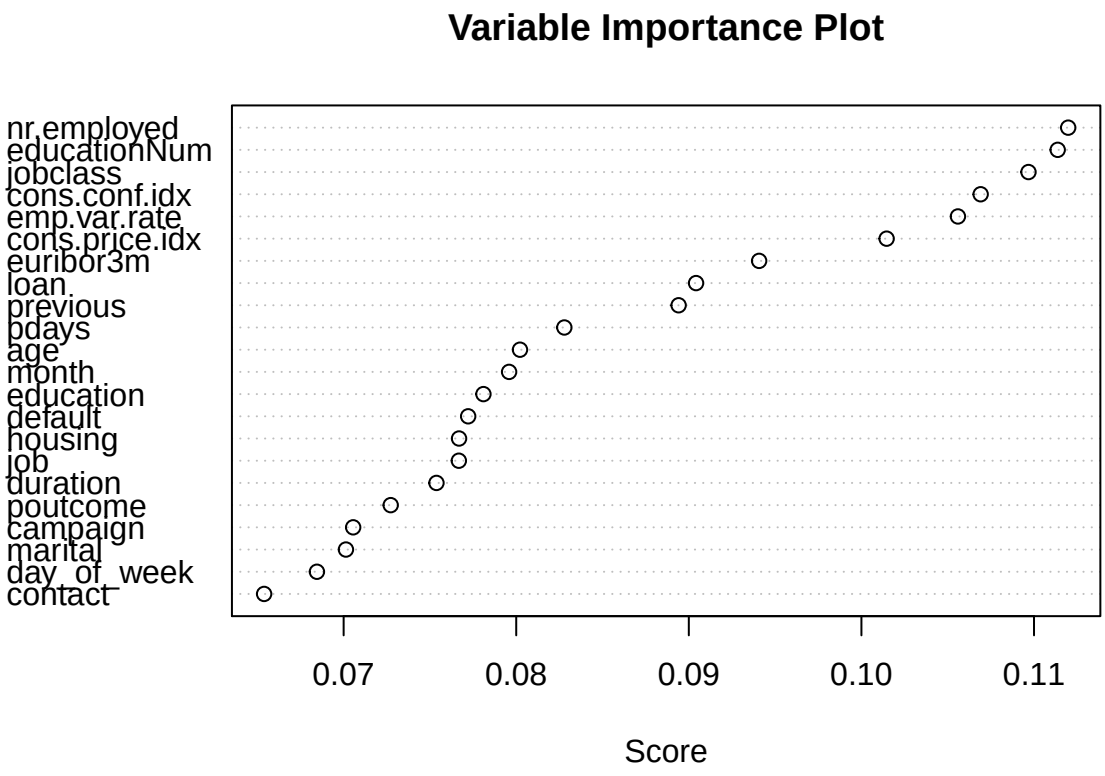
- Below plot shows training error vs iteration where training error reduces and stabilizes as iteration grows

```
plot(bank_model_ada)
```



- Plot of variables ordered by the variable importance measure

```
varplot(bank_model_ada)
```



- In the above training error plot the error rate reduced at minimal ~25 and then slightly increased. So lets try with lesser iterations and measure the metrics

```
bank_model_ada25 = ada(y ~ ., data=train,loss='exponential', type='discrete', iter = 25)
```

- Confustion Matrix for a Ada Model with 25 iterations

```
bank_model_ada25$confusion
```

```
##            Final Prediction
## True value    0    1
##          0 2528   44
##          1   93  224
```

```
(bank_model_ada25$confusion[2,2]/sum(bank_model_ada25$confusion[,2]))
```

```
## [1] 0.8358209
```

- Build model with 100 iterations

```
bank_model_ada100 = ada(y ~ ., data=train,loss='exponential', type='discrete', iter = 100)
```

- Confustion Matrix for a Ada Model with 100 iterations

```
bank_model_ada100$confusion
```

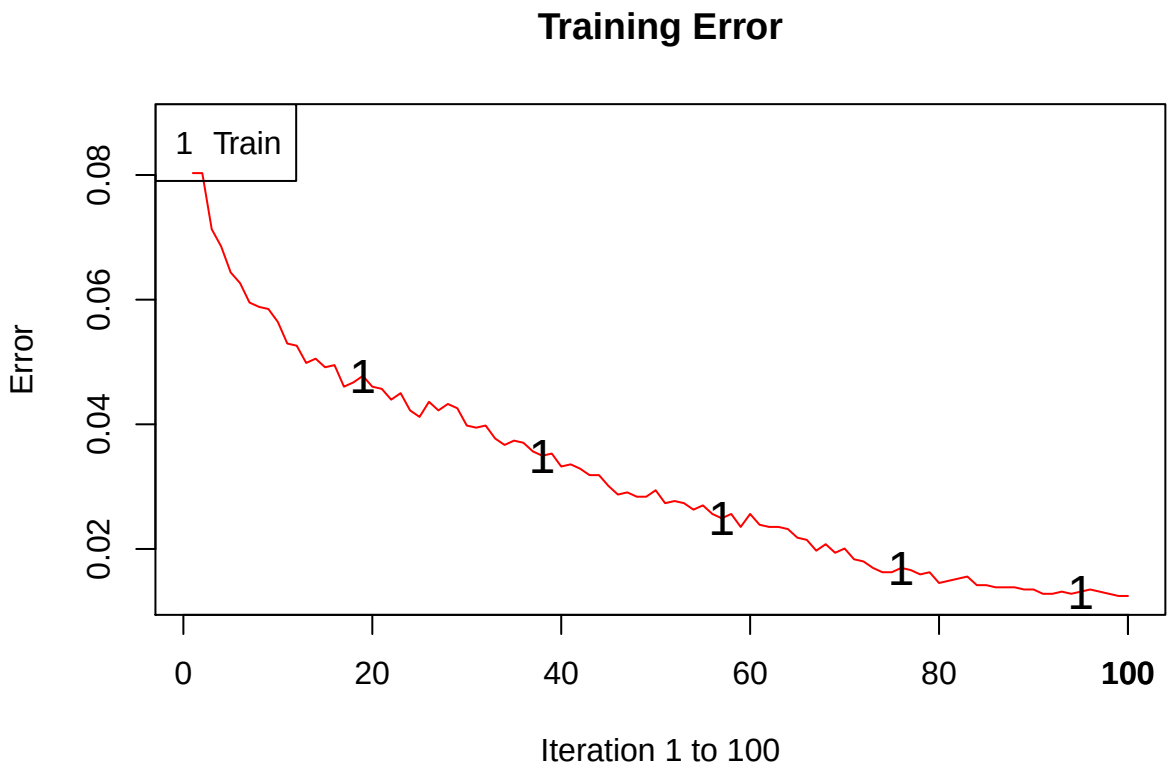
```
##            Final Prediction
## True value    0    1
##          0 2570    2
##          1   34  283
```

```
(bank_model_ada100$confusion[2,2]/sum(bank_model_ada100$confusion[,2]))
```

```
## [1] 0.9929825
```

- Below plot shows training error vs iteration where training error reduces and stabilizes as iteration grows

```
plot(bank_model_ada100)
```



- Since our dataset has class imbalance problem where less positive cases and lot of negative cases. Lets use resampled training data using `SMOTE` and `ROSE` function for training.

```
model_ada_smote = ada(y ~ ., data=smote_train,loss='exponential', type='discrete',iter=100)

model_ada_smote$confusion
```

```
##           Final Prediction
## True value    0    1
##          0 1256   12
##          1   12  939
```

```
(model_ada_smote$confusion[2,2]/sum(model_ada_smote$confusion[,2]))
```

```
## [1] 0.9873817
```

```
model_ada_rose = ada(y ~ ., data=rose_train,loss='exponential', type='discrete',iter=100)

model_ada_rose$confusion
```

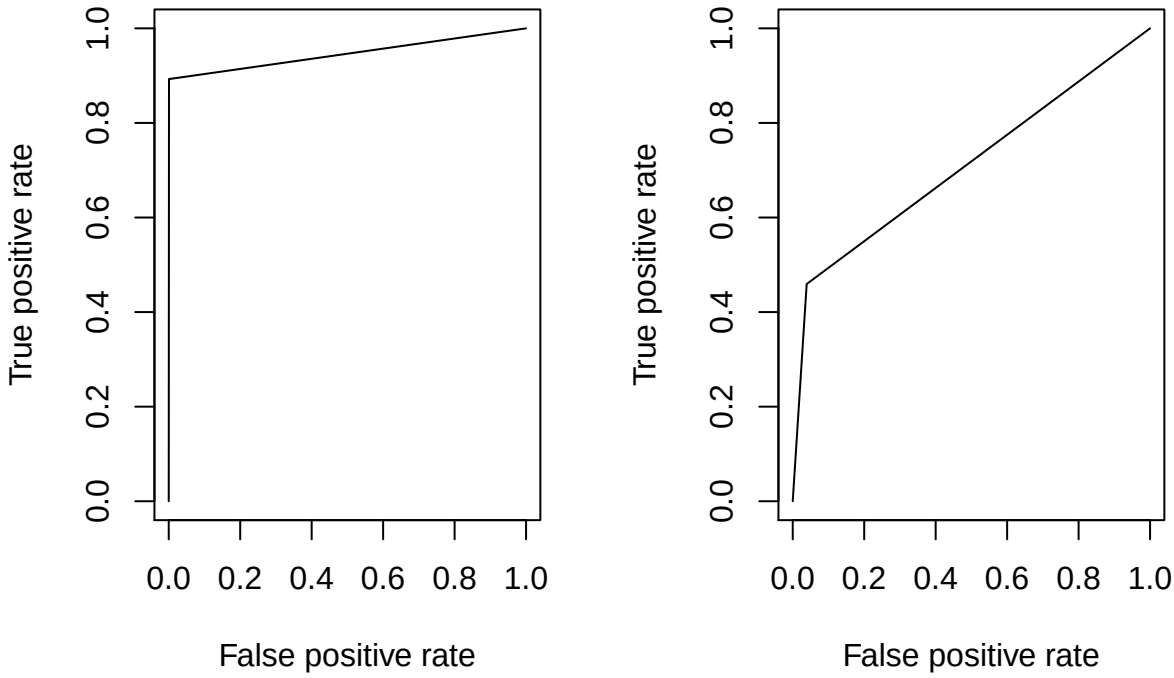
```
##           Final Prediction
## True value    0    1
##          0 1442   20
##          1   14 1413
```

```
(model_ada_rose$confusion[2,2]/sum(model_ada_rose$confusion[,2]))
```

```
## [1] 0.9860433
```

- We see that our accuracy increased after our using resampled data using smote and rose function.
- Plot TPR VS FPR for Training and Test Dataset

```
pred_train = predict(bank_model_ada100, train)
train_cm = confusionMatrix(train$y,as.factor(pred_train))
pred_test = predict(bank_model_ada100, test)
test1_cm=confusionMatrix(test$y,as.factor(pred_test))

par(mfrow = c(1,2))
pr_train = prediction(as.numeric(pred_train),train$y)
prf_train = performance(pr_train, measure = "tpr", x.measure = "fpr")
plot(prf_train)

pr_test1 = prediction(as.numeric(pred_test),test$y)
prf_test1 = performance(pr_test1, measure = "tpr",x.measure = "fpr")
plot(prf_test1)

mtext("ROC curve (TPR vs FPR) for Train Vs Test", side = 3, line = -1, outer = TRUE)
```



- Metrics for Training VS Testing Dataset

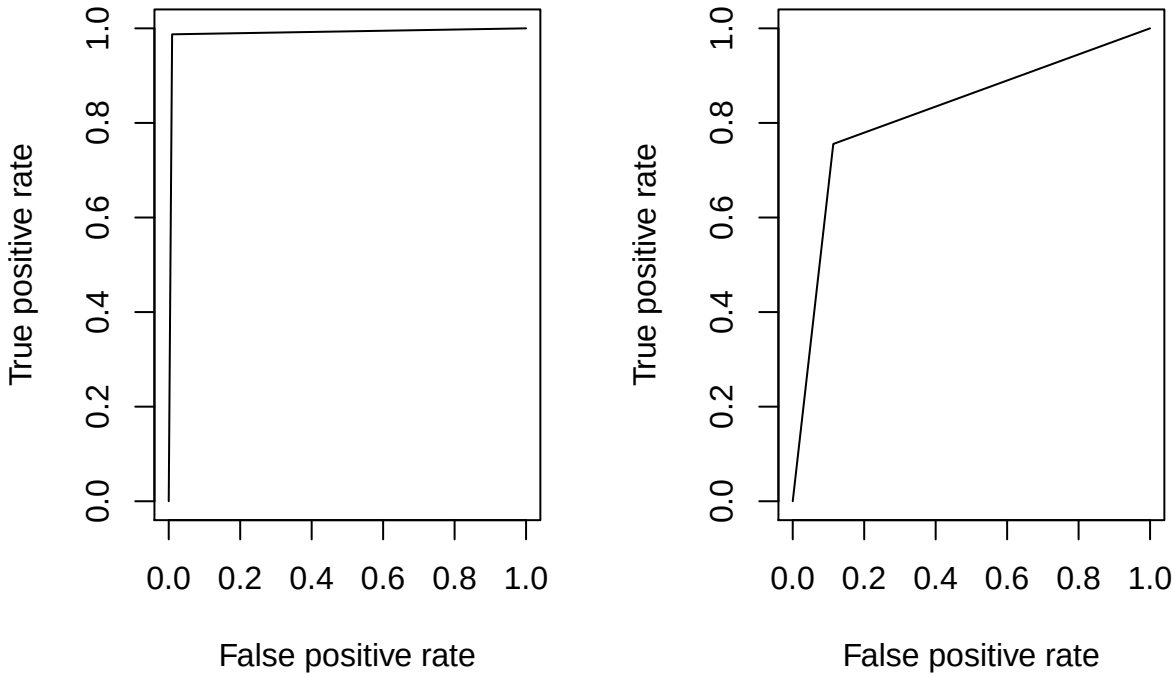
```
auc_train = performance(pr_train, measure = "auc")
auc_test1 = performance(pr_test1, measure = "auc")

metrics = cbind(train_cm$byClass, test1_cm$byClass)
acc = cbind(train_cm$overall["Accuracy"],test1_cm$overall["Accuracy"])
auc = cbind(auc_train@y.values,auc_test1@y.values)
rownames(auc) = c("Area Under Curve")
result = data.frame(rbind(acc,auc,metrics))

colnames(result) = c("Train","Test")
kable(result) %>%
  kable_styling(bootstrap_options = c("striped", "hover"),latex_options = "scale_down")
```

	Train	Test
Accuracy	0.987538940809969	0.906148867313916
Area Under Curve	0.945983437259298	0.71010192754062
Sensitivity	0.986943164362519	0.935455349248453
Specificity	0.992982456140351	0.59047619047619
Pos Pred Value	0.999222395023328	0.96094459582198
Neg Pred Value	0.892744479495268	0.459259259259259
Precision	0.999222395023328	0.96094459582198
Recall	0.986943164362519	0.935455349248453
F1	0.993044822256569	0.948028673835125
Prevalence	0.90134994807892	0.91504854368932
Detection Rate	0.889581169955002	0.855987055016181
Detection Prevalence	0.890273451021115	0.890776699029126
Balanced Accuracy	0.989962810251435	0.762965769862322

```
# to be pull by final result session
test1_cm_ab = test1_cm
auc_test1_ab = auc_test1
```

- Plot TPR vs FPR (SMOTED TRAIN vs TEST)

```
pred_train = predict(model_ada_smote, smote_train)
train_cm = confusionMatrix(smote_train$y,as.factor(pred_train))
pred_test1 = predict(model_ada_smote, test)
test1_cm=confusionMatrix(test$y,as.factor(pred_test1))

par(mfrow = c(1,2))
pr_train = prediction(as.numeric(pred_train),smote_train$y)
prf_train = performance(pr_train, measure = "tpr", x.measure = "fpr")
plot(prf_train)

pr_test1 = prediction(as.numeric(pred_test1),test$y)
prf_test1 = performance(pr_test1, measure = "tpr",x.measure = "fpr")
plot(prf_test1)

mtext("ROC curve (TPR vs FPR) for Train Vs Test", side = 3, line = -1, outer = TRUE)
```



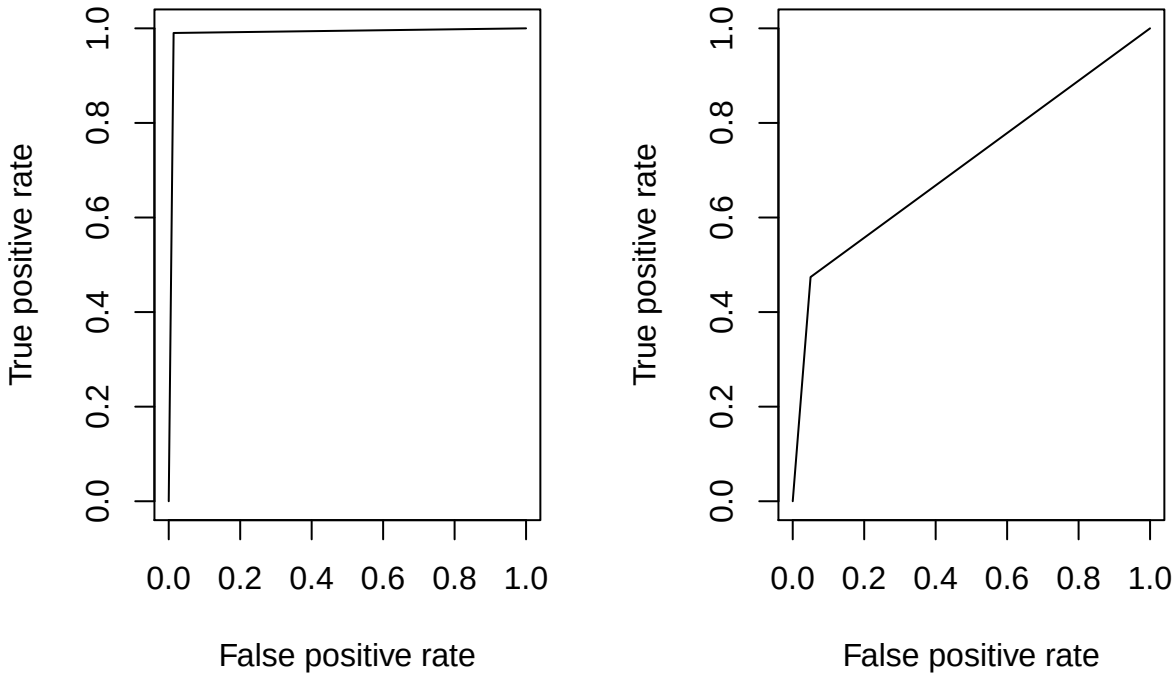
- Metrics for SMOTED TRAIN vs TEST

```
auc_train = performance(pr_train, measure = "auc")
auc_test1 = performance(pr_test1, measure = "auc")

metrics = cbind(train_cm$byClass, test1_cm$byClass)
acc = cbind(train_cm$overall["Accuracy"],test1_cm$overall["Accuracy"])
auc = cbind(auc_train@y.values,auc_test1@y.values)
rownames(auc) = c("Area Under Curve")
result = data.frame(rbind(acc,auc,metrics))

colnames(result) = c("Train","Test")
kable(result) %>%
  kable_styling(bootstrap_options = c("striped", "hover"),latex_options = "scale_down")
```

	Train	Test
Accuracy	0.989184317260027	0.872168284789644
Area Under Curve	0.988958990536278	0.821011201937632
Sensitivity	0.990536277602524	0.967294350842418
Specificity	0.987381703470032	0.449339207048458
Pos Pred Value	0.990536277602524	0.886466848319709
Neg Pred Value	0.987381703470032	0.755555555555555
Precision	0.990536277602524	0.886466848319709
Recall	0.990536277602524	0.967294350842418
F1	0.990536277602524	0.925118483412322
Prevalence	0.571428571428571	0.816343042071197
Detection Rate	0.566020730058585	0.789644012944984
Detection Prevalence	0.571428571428571	0.890776699029126
Balanced Accuracy	0.988958990536278	0.708316778945438

- Plot TPR vs FPR (ROSE TRAIN vs TEST)

```
pred_train = predict(model_ada_rose, rose_train)
train_cm = confusionMatrix(rose_train$y,as.factor(pred_train))
pred_test1 = predict(model_ada_rose, test)
test1_cm=confusionMatrix(test$y,as.factor(pred_test1))

par(mfrow = c(1,2))
pr_train = prediction(as.numeric(pred_train),rose_train$y)
prf_train = performance(pr_train, measure = "tpr", x.measure = "fpr")
plot(prf_train)

pr_test1 = prediction(as.numeric(pred_test1),test$y)
prf_test1 = performance(pr_test1, measure = "tpr",x.measure = "fpr")
plot(prf_test1)
mtext("ROC curve (TPR vs FPR) for Train Vs Test", side = 3, line = -1, outer = TRUE)
```



- Metrics for ROSE Train vs Test

```
auc_train = performance(pr_train, measure = "auc")
auc_test1 = performance(pr_test1, measure = "auc")

metrics = cbind(train_cm$byClass, test1_cm$byClass)
acc = cbind(train_cm$overall["Accuracy"],test1_cm$overall["Accuracy"])
auc = cbind(auc_train@y.values,auc_test1@y.values)
rownames(auc) = c("Area Under Curve")
result = data.frame(rbind(acc,auc,metrics))

colnames(result) = c("Train","Test")
kable(result) %>%
  kable_styling(bootstrap_options = c("striped", "hover"),latex_options = "scale_down")
```

	Train	Test
Accuracy	0.988231221876082	0.898058252427185
Area Under Curve	0.988254658784033	0.712059743667373
Sensitivity	0.990384615384615	0.936436884512086
Specificity	0.986043265875785	0.53781512605042
Pos Pred Value	0.986320109439125	0.950045413260672
Neg Pred Value	0.990189208128942	0.474074074074074
Precision	0.986320109439125	0.950045413260672
Recall	0.990384615384615	0.936436884512086
F1	0.988348183687457	0.943192064923355
Prevalence	0.503980616130149	0.903721682847896
Detection Rate	0.499134648667359	0.846278317152104
Detection Prevalence	0.506057459328487	0.890776699029126
Balanced Accuracy	0.9882139406302	0.737126005281253

Results

Compare Models (Test and Train Accuracy) The following is a table to compare the various metrics against the test data set from the three different methods: logistic regression, random forest and adaptive boosting after model tuning.

For logistic regression, we pick backward search AIC at cutoff 0.1. For random forest, we pick decision tree with 5 predictors and 500 trees. For adaptive boosting, we pick iteration of 100.

```
# compare Accuracy and ROC
compare <- data.frame(Method = c('Logistic Regression (AIC backward search cutoff = 0.1)'
                , 'Random Forest (mtry = 5, ntree = 500)', 'Adaptive boosting (iter = 100)')
                , Accuracy = NA, Sensitivity = NA, Specificity = NA, Precision = NA
                , Recall = NA, FScore = NA, 'ROC AUC' = NA)

compare$Accuracy <- c(test_con_mat_10$overall["Accuracy"], test1_cm_rf$overall['Accuracy']
                      ,test1_cm_ab$overall['Accuracy'])

compare$Sensitivity <- c(test_con_mat_10$byClass["Sensitivity"], test1_cm_rf$byClass['Sensitivity']
                         ,test1_cm_ab$byClass['Sensitivity'])

compare$Specificity <- c(test_con_mat_10$byClass["Specificity"], test1_cm_rf$byClass['Specificity']
                         ,test1_cm_ab$byClass['Specificity'])

compare$Precision <- c(test_con_mat_10$byClass["Precision"], test1_cm_rf$byClass['Precision']
                       ,test1_cm_ab$byClass['Precision'])

compare$Recall <- c(test_con_mat_10$byClass["Recall"], test1_cm_rf$byClass['Recall']
                    ,test1_cm_ab$byClass['Recall'])
```

```
compare$FScore <- c(test_con_mat_10$byClass["F1"], test1_cm_rf$byClass['F1']
                    ,test1_cm_ab$byClass['F1'])

compare$ROC.AUC <- c(auc(test_roc), auc_test1_rf@y.values, auc_test1_ab@y.values)

kable_styling(kable(compare),c("striped","bordered"), full_width = F, latex_options = "scale_down")
```

Method	Accuracy	Sensitivity	Specificity	Precision	Recall	FScore	ROC.AUC
Logistic Regression (AIC backward search cutoff = 0.1)	0.8365696	0.8814815	0.8310627	0.3901639	0.8814815	0.5409091	0.9266256265348
Random Forest (mtry = 5, ntree = 500)	0.8996764	0.9258936	0.5617978	0.9645777	0.9258936	0.9448399	0.648430719547886
Adaptive boosting (iter = 100)	0.9061489	0.9354553	0.5904762	0.9609446	0.9354553	0.9480287	0.71010192754062

Discussion

- Do we need to select model with high accuracy?
  - Accuracy may not be a good measure if the dataset is not balanced and that is the case for our dataset. F1 score becomes high only when both precision and recall are high. F1 score is the harmonic mean of precision and recall and is a better measure than accuracy.
- Do we accept False Positive or False Negative in the context of the problem statement?
  - False Positive, means the client do NOT SUBSCRIBED to term deposit, but the model thinks he/she did.
    - Type I error is more harmful because we might lose the customer and loss revenue by incorrectly considering the client was already subscribed.
  - False Negative, means the client SUBSCRIBED to term deposit, but the model thinks otherwise.
    - Type II error is less harmful because we didn't lose the customer and no loss in revenue by incorrectly considering the client was not subscribed. We may be contacting the customer for the same campaign but it won't cause to lose a customer.
- Flexibility vs Interpretability - what do we choose?
  - Smaller model(Logistic Regression) are easily interpretable with decent accuracy in prediction.
  - Complex, flexible model (Random Forest, Adaptive Boosting) are less interpretable but with high accuracy.
    - In this study, we must go with complex and flexible model eventhough we lose interpretability in the expense for higher prediction since it involves revenue to the bank.
- Speed
  - We need to choose the model which performs better and run faster. Boosting models run in parallel compared to Logistic and Random Forest.

Conclusion

- In this study, we have applied regression and advanced machine learning techniques on bank marketing data and explored the distribution of data, data curation and modeling techniques. The below are few key highlights.
  - Multiple Classification and Regression Algorithms have been evaluated.
  - Garbage in - Garbage out so we analyzed the data and curated data to further analysis.

  - Oversampling/undersampling has been implemented for class imbalanced data.
  - Cross Validation was used for parameter selection with logistic regression.
  - Random Forest used with default and best split and tree depth hyper parameters tuned to produce better result.
  - Random Forest is the best model for balanced sensitivities and interpretation of feature importance.

- Our recommendation to the manager of the bank marketing campaign would be to use random forest which has the highest sensitivity so actual positives are not overlooked. Logistic regression is useful as well to understand what are significant variables and allocate resources accordingly. For example, number of client contacts performed before this campaign is significant variable to subscribe term deposit, then contact client with the optimal number of times to maximize the chance but prevent wasting resources.

Appendix

Team members

- Bhusan Bathani (bbath2)
- Mathew Leung (wmleung2)
- Vijayakumar Sitha Mohan (VS24)

Source Repository

- Our Data Analysis Project Source

UCI machine Learning Library:

- Bank Marketing Dataset Source
- [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