Table of Contents

Bank Marketing Analysis

Designing a Telemarketing Strategy To Reduce Acquisition Costs

---

A bank sells a product (called term deposit) to prospects mainly through telemarketing. If a prospect customer buys the product, we say that he has 'responded'.

The aim of this analysis is to **reduce the marketing cost by atleast 50%** and acquire a comparable number of customers (say 80-90%).

We'll use *telemarketing data* from a past campaign of the bank. The sales team had recorded customer data like age, salary, whether he has a loan, house, the month of call etc.

The idea is to use machine learning to predict the likelihood of a person 'buying the product 'responding'. We'll identify those who are most likely to respond and telemarket only to them, thereby reducing the total cost of acquisition per customer.

The standard process followed in analytics projects is: 1. Business Understanding 2. Data Understanding
3. Modelling 4. Model Evaluation 5. Model Deployment and Recommendations

---

Business Understanding

The **overall goal** is to reduce telemarketing costs by about 50% and acquire atleast 80-90% of the customers.

The specific **objective of this analysis** is to build a **'response model'** to predict the likelihood of a prospect buying the product (or responding).

Data Understanding

The datafile is named bank-marketing.csv. You can download it here: https://archive.ics.uci.edu/ml/datasets/Bank+Marketing

```
## 'data.frame':    45211 obs. of  19 variables:
##  $ age      : int  58 44 33 47 33 35 28 42 58 43 ...
##  $ job      : Factor w/ 12 levels "admin.","blue-collar",..: 5 10 3 2 12 5
5 3 6 10 ...
##  $ salary   : int  100000 60000 120000 20000 0 100000 100000 120000 55000
60000 ...
##  $ marital  : Factor w/ 3 levels "divorced","married",..: 2 3 2 2 3 2 3 1
2 3 ...
##  $ education: Factor w/ 4 levels "primary","secondary",..: 3 2 2 4 4 3 3 3
1 2 ...
##  $ targeted : Factor w/ 2 levels "no","yes": 2 2 2 1 1 2 1 1 2 2 ...
##  $ default  : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 2 1 1 ...
##  $ balance  : int  2143 29 2 1506 1 231 447 2 121 593 ...
##  $ housing  : Factor w/ 2 levels "no","yes": 2 2 2 2 1 2 2 2 2 2 ...
##  $ loan     : Factor w/ 2 levels "no","yes": 1 1 2 1 1 1 2 1 1 1 ...
##  $ contact  : Factor w/ 3 levels "cellular","telephone",..: 3 3 3 3 3 3 3
3 3 3 ...
##  $ day      : int  5 5 5 5 5 5 5 5 5 5 ...
##  $ month    : Factor w/ 12 levels "apr","aug","dec",..: 9 9 9 9 9 9 9 9 9
9 ...
##  $ duration : int  261 151 76 92 198 139 217 380 50 55 ...
##  $ campaign : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ pdays    : int  -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 ...
##  $ previous : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ poutcome : Factor w/ 4 levels "failure","other",..: 4 4 4 4 4 4 4 4 4 4
...
##  $ response : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
```

We have 45211 observations, i.e. we have data of 45211 potential customers. There are 20 variables (or 20 columns). We have two types of variables:

1. Two type of attributes (or variables):

- **Customer attributes** like age, job, salary, marital (status), education (primary / secondary / tertiary) etc.

- **Bank related attributes** like targeted (whether he/she was targeted before), loan (yes / no), contact (whether he/she has been contacted before etc.)

The last attribute 'response' tells us whether the person had responded to the bank's marketing campaign. It thus contains only two values - Yes or No. It is called the **target attribute** since that is what we want to predict using the other attributes.

Data Cleaning

Missing Values and Outliers

We always start with cleaning the data i.e. removing the missing values, any erroneous entries etc. Let's see if this data contains **missing values**.

```
sum(is.na(bank_data))

## [1] 0
```

We have simply summed up all the missing values (denoted as 'NA'). There are none of them, so we move ahead.

Next, we should ideally look at **outliers** in the data. Outliers are extreme values for which treatment is done so that the data only represents the general trends and ignores extreme cases. For example, a person having an income of Rs 20 lacs per month will be an outlier in this dataset.

For now, we will not do outlier treatment since this data doesn't have many of them.

Exploratory Analysis

In Exploratory Data Analysis, or EDA, we use plots and summaries of data to understand the patterns in it. Let's see some examples.

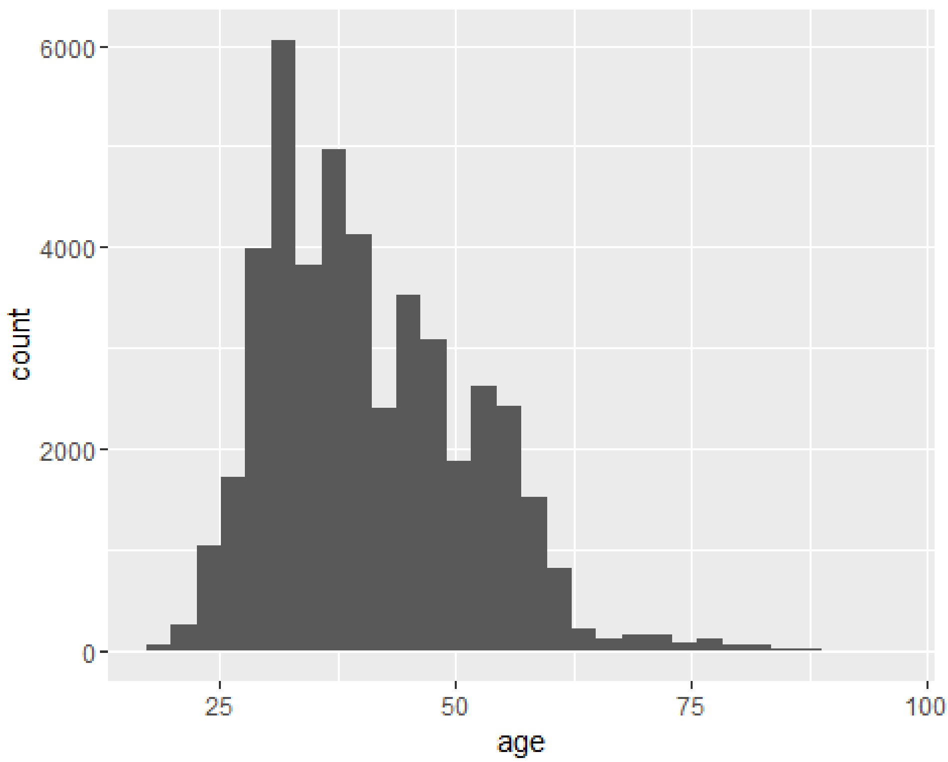
Univariate Analysis

In univariate analysis, we analyse one variable at a time.

The following plot shows the distribution of peoples' ages in the data. The ggplot library is a great data visualisation tool in R.

Since age is a numeric variable, we plot a **histogram**.

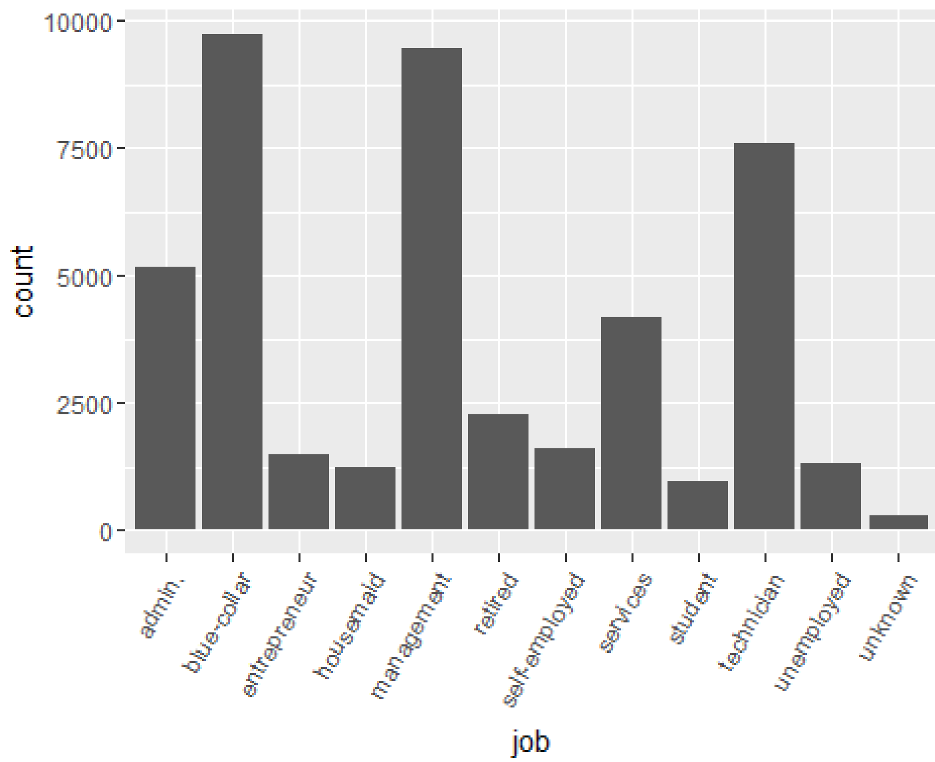
```
library(ggplot2)
ggplot(bank_data, aes(age)) + geom_histogram()

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



You can see that most people are between 25-50 years old. Very few people older than 60 years have been targeted.

Next, let's look at the types of jobs people have. Now *job* is not a numeric variable, it is a **categorical variable** and so we plot a **bar chart** for it.

```
ggplot(bank_data, aes(job)) + geom_bar() + theme(axis.text.x =
element_text(angle = 60, hjust = 1))
```



We have a large number of people with blue-collar jobs and in management, which are about 9000 each. The third highest category is technicians which are about 7500 people. Note that among 45,000 people, about 25,000 or 55% are either blue-collar workers, management employees technicians.

A very important variable for the bank would be **salary**. Let's have a look at the average and the median salary.

```
summary(bank_data$salary)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       0   20000   60000   57010   70000  120000
```

The average salary is about INR 57000 per month. Note that the maximum is only about INR 1.2 lacs per month.

Let's now look at the summary of the target variable **response**.

```
summary(bank_data$response)
```

So out of about 45,000 people, only 5289 had responded. The exact **response rate** can be calculated as:

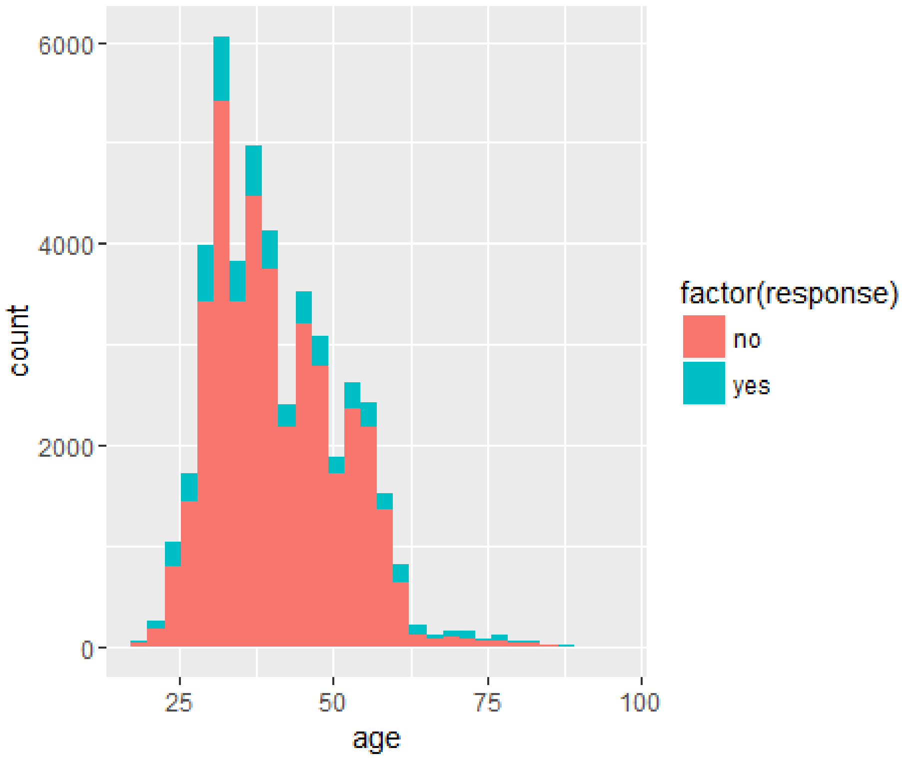
```
## [1] 0.1169848
```

An 11.7% response rate means that if the you make 100 calls to market the product (term deposit), about 11.7% will subscribe for a term-deposit.

Multivariate Analysis

Now let's analyse two variables at a time, one of which should obviously be the target variable 'response'.

```
ggplot(bank_data, aes(age, fill = factor(response))) + geom_histogram()

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



So now the plot shows information of two variables - the age on x-axis and response as a colour. This chart does not show any obvious trend of response rate with age.

The analysis will be easier if we could divide the age into **buckets**, e.g. 0-10 years, 10-20 years etc. This is called bucketing and is often done to divide **numeric variables** into smaller buckets.

```
bank_data$buckets.age <- cut(bank_data$age, breaks = c(10, 20, 30, 40, 50,
60, 70, 80, 90, 100))
str(bank_data)

## 'data.frame':    45211 obs. of  20 variables:
##  $ age          : int  58 44 33 47 33 35 28 42 58 43 ...
##  $ job          : Factor w/ 12 levels "admin.","blue-collar",..: 5 10 3 2 12
5 5 3 6 10 ...
##  $ salary       : int  100000 60000 120000 20000 0 100000 100000 120000
55000 60000 ...
##  $ marital      : Factor w/ 3 levels "divorced","married",..: 2 3 2 2 3 2 3
1 2 3 ...
```

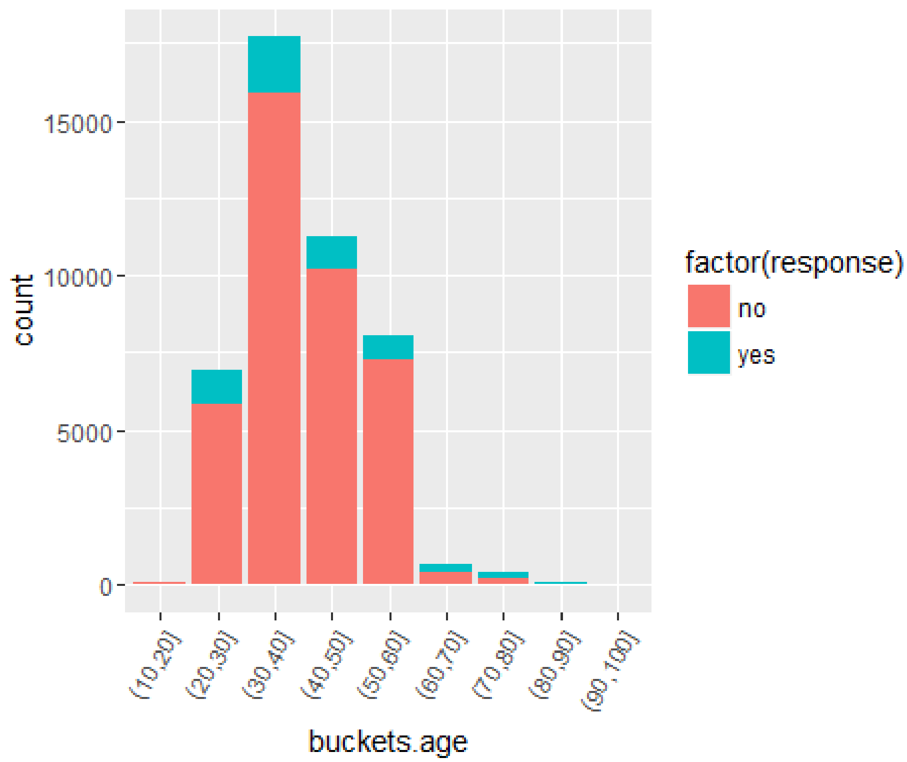
```
##  $ education  : Factor w/ 4 levels "primary","secondary",..: 3 2 2 4 4 3 3
3 1 2 ...
##  $ targeted   : Factor w/ 2 levels "no","yes": 2 2 2 1 1 2 1 1 2 2 ...
##  $ default    : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 2 1 1 ...
##  $ balance    : int  2143 29 2 1506 1 231 447 2 121 593 ...
##  $ housing    : Factor w/ 2 levels "no","yes": 2 2 2 2 1 2 2 2 2 2 ...
##  $ loan       : Factor w/ 2 levels "no","yes": 1 1 2 1 1 1 2 1 1 1 ...
##  $ contact    : Factor w/ 3 levels "cellular","telephone",..: 3 3 3 3 3 3
3 3 3 3 ...
##  $ day        : int  5 5 5 5 5 5 5 5 5 5 ...
##  $ month      : Factor w/ 12 levels "apr","aug","dec",..: 9 9 9 9 9 9 9 9
9 9 ...
##  $ duration   : int  261 151 76 92 198 139 217 380 50 55 ...
##  $ campaign   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ pdays      : int  -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 ...
##  $ previous   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ poutcome   : Factor w/ 4 levels "failure","other",..: 4 4 4 4 4 4 4 4 4
4 ...
##  $ response   : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ buckets.age: Factor w/ 9 levels "(10,20]","(20,30]",..: 5 4 3 4 3 3 2 4
5 4 ...
```

```
sum(is.na(bank_data))

## [1] 0
```

Note that a new variable named *buckets.age* is now added to bank_data. Ages are now bucketed into this variable.

Let's use the buckets to see if age affects the response rate.

```
ggplot(bank_data, aes(buckets.age, fill = factor(response))) + geom_bar() +
theme(axis.text.x = element_text(angle = 60, hjust = 1))
```



It will be easier if we could see the response rate in numbers as well. We can convert the 'yes-no' values to '1-0' respectively and then calculate the response rate by summing up the 1s.

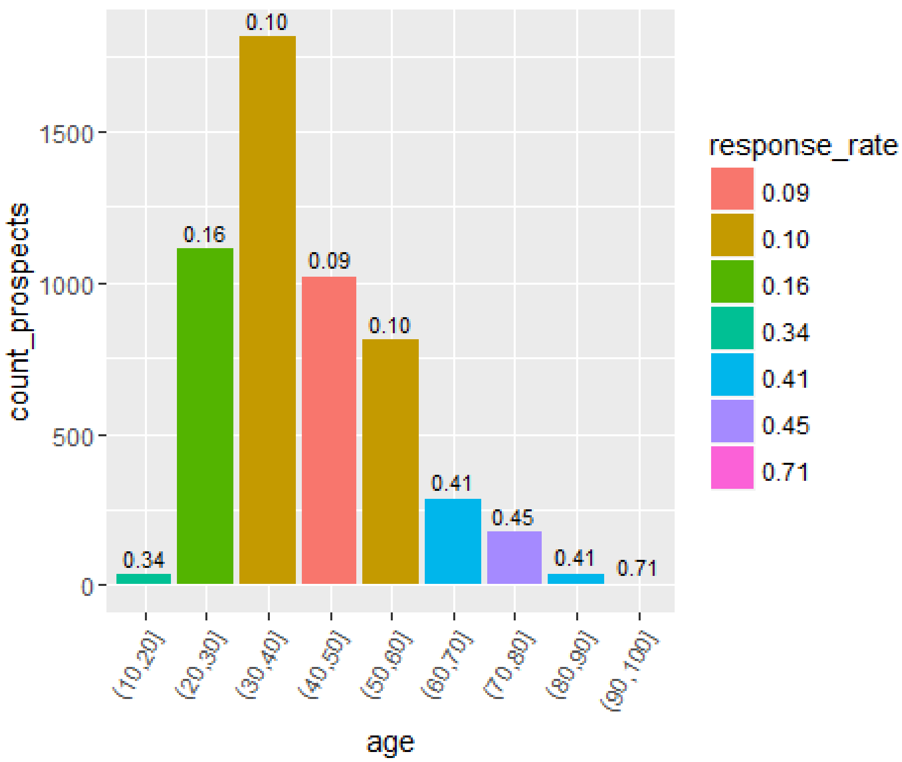
```
bank_data$response.numeric <- ifelse(bank_data$response == "yes", 1, 0)
str(bank_data)
```

```
## 'data.frame':    45211 obs. of  21 variables:
##  $ age              : int  58 44 33 47 33 35 28 42 58 43 ...
##  $ job              : Factor w/ 12 levels "admin.","blue-collar",..: 5 10 3
2 12 5 5 3 6 10 ...
##  $ salary           : int  100000 60000 120000 20000 0 100000 100000 120000
55000 60000 ...
##  $ marital          : Factor w/ 3 levels "divorced","married",..: 2 3 2 2 3
2 3 1 2 3 ...
##  $ education        : Factor w/ 4 levels "primary","secondary",..: 3 2 2 4
4 3 3 3 1 2 ...
##  $ targeted         : Factor w/ 2 levels "no","yes": 2 2 2 1 1 2 1 1 2 2
...
##  $ default          : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 2 1 1
...
##  $ balance          : int  2143 29 2 1506 1 231 447 2 121 593 ...
##  $ housing          : Factor w/ 2 levels "no","yes": 2 2 2 2 1 2 2 2 2 2
...
##  $ loan             : Factor w/ 2 levels "no","yes": 1 1 2 1 1 1 2 1 1 1
...
##  $ contact          : Factor w/ 3 levels "cellular","telephone",..: 3 3 3 3
```

```
3 3 3 3 3 3 ...
##  $ day            : int  5 5 5 5 5 5 5 5 5 5 ...
##  $ month          : Factor w/ 12 levels "apr","aug","dec",..: 9 9 9 9 9 9
9 9 9 9 ...
##  $ duration       : int  261 151 76 92 198 139 217 380 50 55 ...
##  $ campaign       : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ pdays          : int  -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 ...
##  $ previous       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ poutcome       : Factor w/ 4 levels "failure","other",..: 4 4 4 4 4 4
4 4 4 4 ...
##  $ response       : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1
...
##  $ buckets.age    : Factor w/ 9 levels "(10,20]","(20,30]",..: 5 4 3 4 3
3 2 4 5 4 ...
##  $ response.numeric: num  0 0 0 0 0 0 0 0 0 0 ...
```

We can now aggregate the 1s by each age bucket and see the response rates for each bucket.

```
agg_age <- merge(aggregate(response.numeric ~ buckets.age, bank_data, mean),
                 aggregate(response.numeric~buckets.age, bank_data, sum),
                 by = "buckets.age")

colnames(agg_age) <- c("age", "response_rate", "count_prospects")
agg_age$response_rate <- format(round(agg_age$response_rate, 2))
agg_age

##        age response_rate count_prospects
## 1  (10,20]          0.34              33
## 2  (20,30]          0.16            1112
## 3  (30,40]          0.10            1812
## 4  (40,50]          0.09            1019
## 5  (50,60]          0.10             811
## 6  (60,70]          0.41             284
## 7  (70,80]          0.45             175
## 8  (80,90]          0.41              38
## 9 (90,100]          0.71               5

ggplot(agg_age, aes(age, count_prospects, fill = response_rate, label =
response_rate)) + geom_bar(stat = 'identity') + theme(axis.text.x =
element_text(angle = 60, hjust = 1)) + geom_text(size = 3, vjust = -0.5)
```



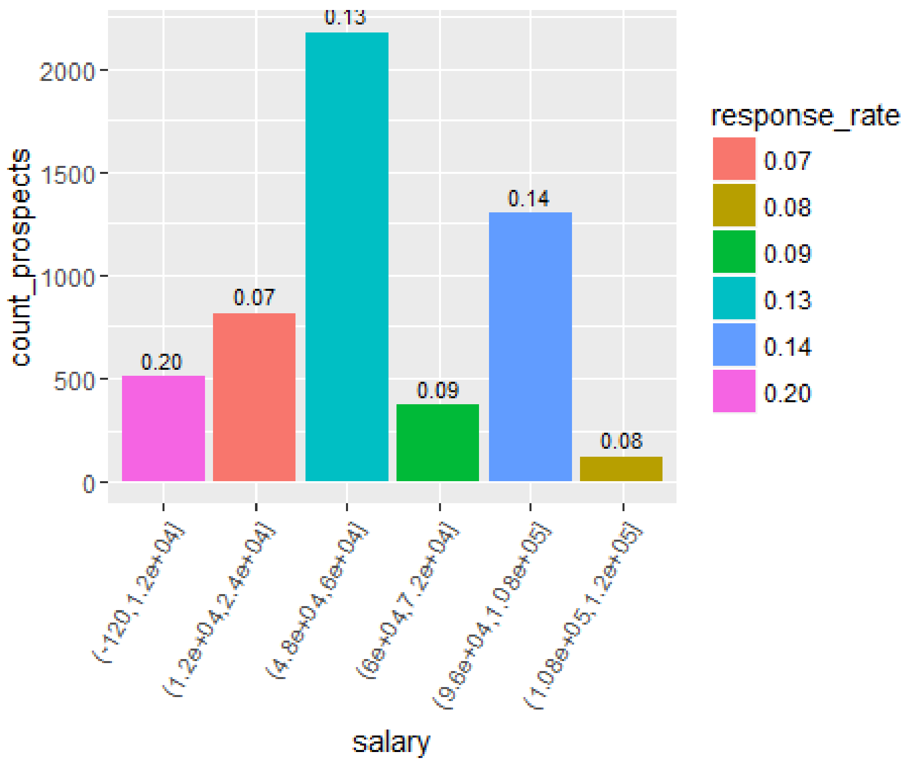
Note that although the bucket 10-20 has about 34% response rate, there are only a few prospects in that bucket (only 33, see code below) and thus we should ignore the bucket.

The buckets 20-30 has 16% response rate while 40-50 and 50-60 have low response ratesat around 10%.

Similarly, we can measure the response rate with salary and jobs.

```
bank_data$buckets.salary <- cut(bank_data$salary, breaks = 10)
agg_salary <- merge(aggregate(response.numeric ~ buckets.salary, data =
bank_data, mean),
                    aggregate(response.numeric ~ buckets.salary, bank_data,
sum),
                              by = "buckets.salary")

colnames(agg_salary) <- c("salary", "response_rate", "count_prospects")
agg_salary$response_rate <- format(round(agg_salary$response_rate, 2))
agg_salary

##               salary response_rate count_prospects
## 1     (-120,1.2e+04]          0.20             505
## 2 (1.08e+05,1.2e+05]          0.08             123
## 3  (1.2e+04,2.4e+04]          0.07             817
## 4    (4.8e+04,6e+04]          0.13            2174
## 5    (6e+04,7.2e+04]          0.09             369
## 6 (9.6e+04,1.08e+05]          0.14            1301
```

```
ggplot(agg_salary, aes(salary, count_prospects, fill = response_rate, label =
response_rate)) + geom_bar(stat = 'identity') + theme(axis.text.x =
element_text(angle = 60, hjust = 1)) + geom_text(size = 3, vjust = -0.5)
```



You can see that the response rate is highest for the lowest salary band (19.96%) while it contains 505 prospects. This might tell you something about the banking products you should be selling (which are used by people in this salary band.)

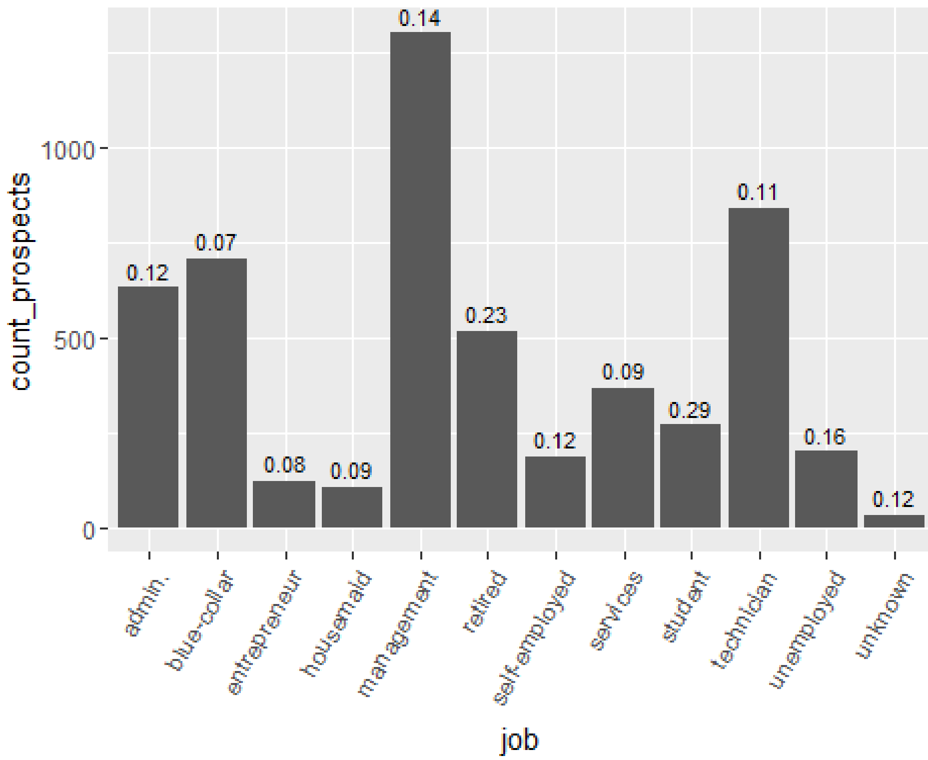
The number of prospects is highest in the fourth salary bucket at 2174 where the response rate is about 13.08% (higher than the average of 11.7%).

Let's also compare response rates across various jobs.

```
agg_job <- merge(aggregate(response.numeric~job, data = bank_data, mean),
                 aggregate(response.numeric~job, bank_data, sum),
                 by = "job")
colnames(agg_job) <- c("job", "response_rate", "count_prospects")
agg_job$response_rate <- format(round(agg_job$response_rate, 2))
agg_job

##               job response_rate count_prospects
## 1          admin.          0.12             631
## 2     blue-collar          0.07             708
## 3    entrepreneur          0.08             123
## 4       housemaid          0.09             109
## 5      management          0.14            1301
## 6         retired          0.23             516
```

```
## 7  self-employed          0.12              187
## 8       services          0.09              369
## 9        student          0.29              269
## 10    technician          0.11              840
## 11    unemployed          0.16              202
## 12       unknown          0.12               34
```

```
ggplot(agg_job, aes(job, count_prospects, label = response_rate)) +
geom_bar(stat = 'identity') + theme(axis.text.x = element_text(angle = 60,
hjust = 1)) + geom_text(size = 3, vjust = -0.5)
```



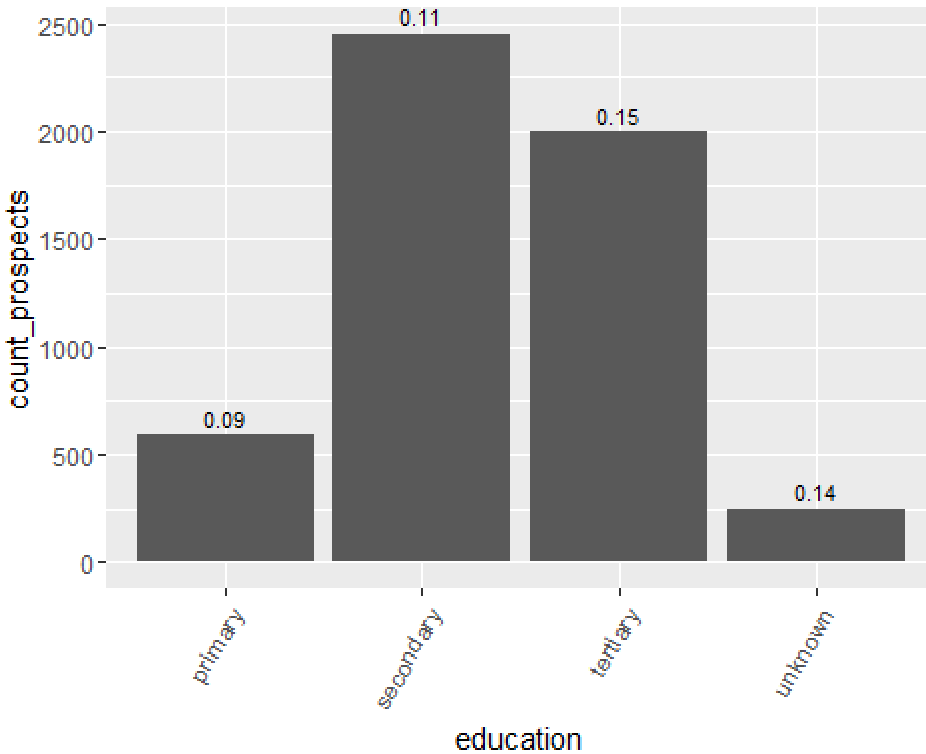
Interestingly, response rate is highest for students and second highest for retired people. It is quite low for blue-collar workers, housemaids and entrepreneurs.

Similarly, you can analyse response rates with other variables like education, marital status, loan etc. The plots below show how response rate varies with education, marital status, contact method and monh of contact.

```
agg_ed <- merge(aggregate(response.numeric~education, bank_data, mean),
                 aggregate(response.numeric~education, bank_data, sum),
                 by = "education")
colnames(agg_ed) <- c("education", "response_rate", "count_prospects")
agg_ed$response_rate <- format(round(agg_ed$response_rate, 2))
agg_ed

##   education response_rate count_prospects
## 1   primary          0.09             591
```

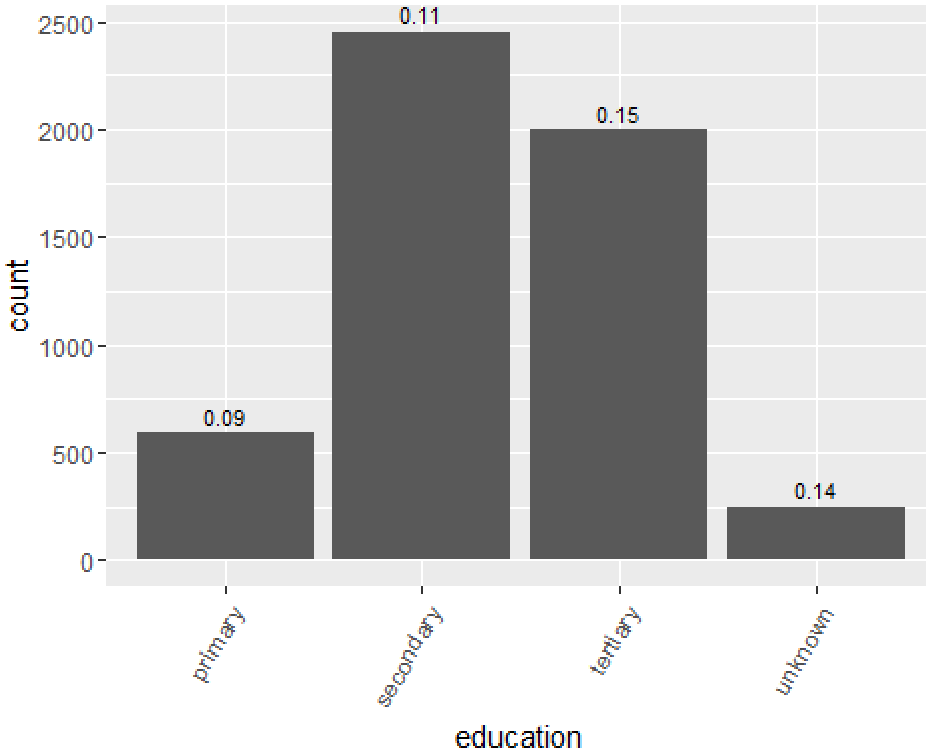
```
## 2 secondary          0.11            2450
## 3  tertiary          0.15            1996
## 4   unknown          0.14             252
```

```
ggplot(agg_ed, aes(education, count_prospects, label = response_rate)) +
geom_bar(stat = 'identity') + theme(axis.text.x = element_text(angle = 60,
hjust = 1)) + geom_text(size = 3, vjust = -0.5)
```

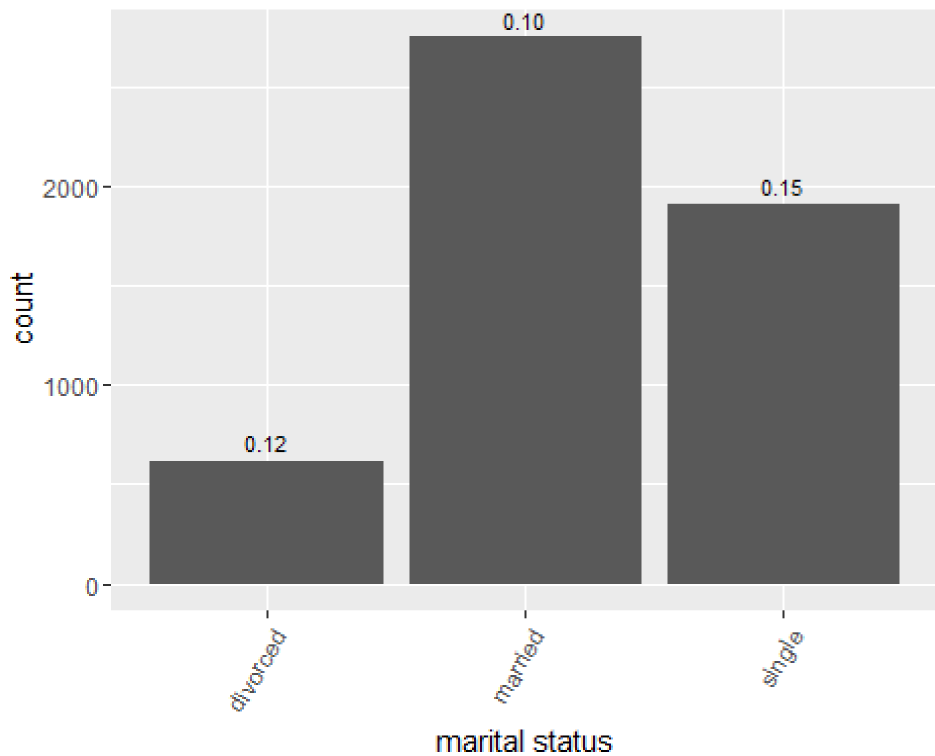



```
plot_response <- function(cat_var, var_name){
  a <- aggregate(response.numeric~cat_var, bank_data, mean)
  b <- aggregate(response.numeric~cat_var, bank_data, sum)

  colnames(a)[1] <- var_name
  colnames(b)[1] <- var_name
  agg_response <- merge(a, b, by = var_name)


  colnames(agg_response) <- c(var_name, "response_rate","count")
  agg_response[, 2] <- format(round(agg_response[, 2], 2))

  ggplot(agg_response, aes(agg_response[, 1], count, label = response_rate))
+ geom_bar(stat = 'identity') + theme(axis.text.x = element_text(angle = 60,
hjust = 1)) + geom_text(size = 3, vjust = -0.5) + xlab(var_name)
  #
  # return(agg_response)
  }
```

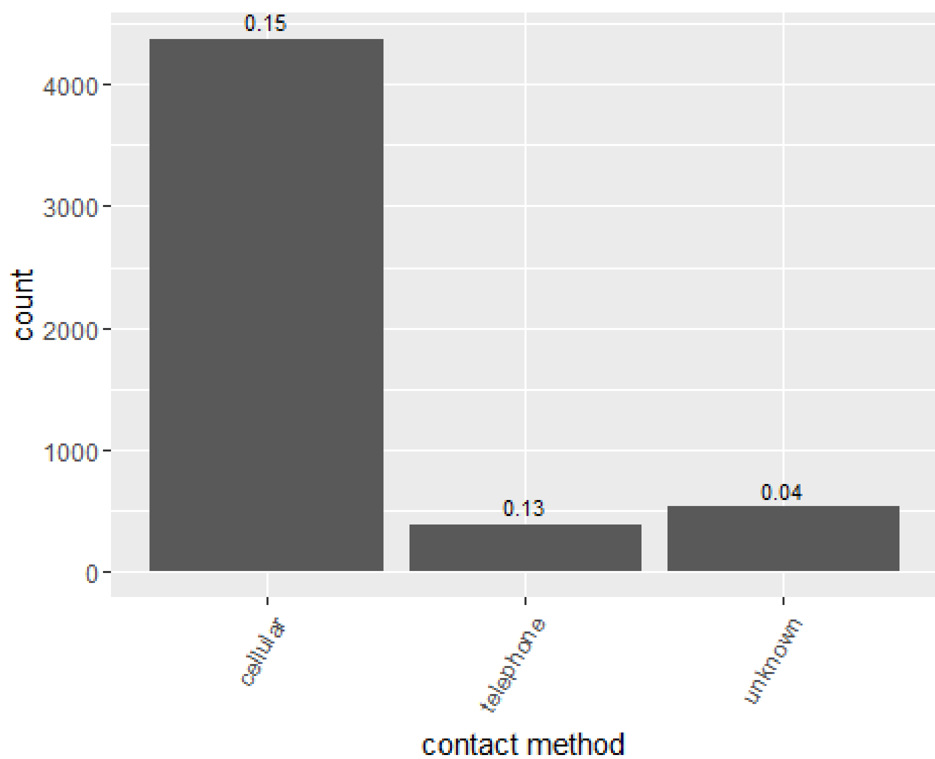
```r
plot_response(bank_data$education, "education")
```



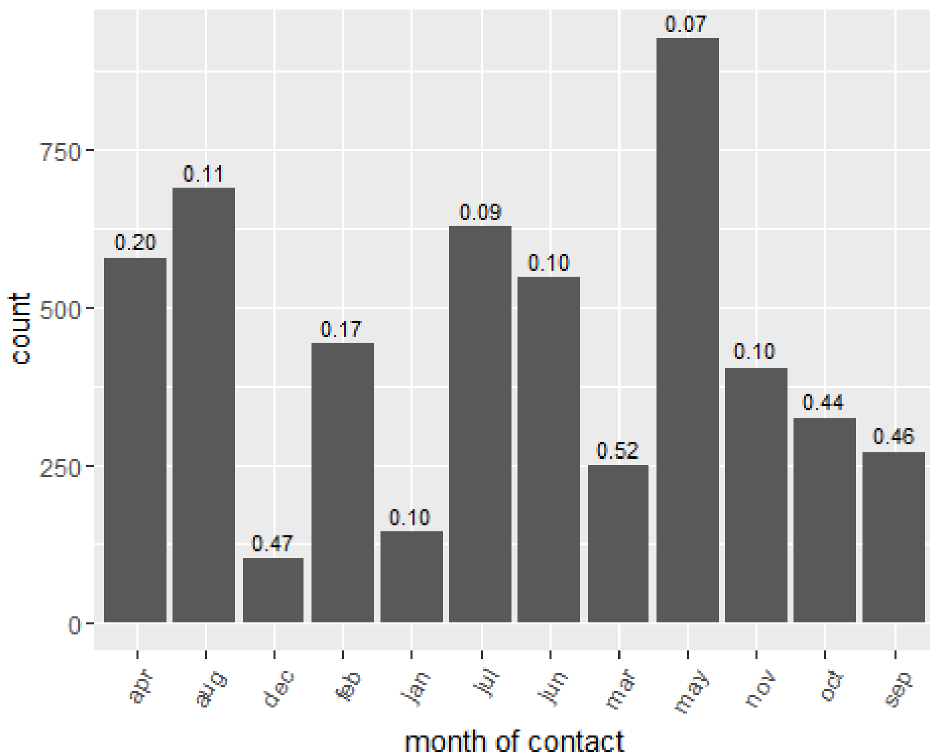

```r
plot_response(bank_data$marital, "marital status")
```




```r
plot_response(bank_data$contact, "contact method")
```




```r
plot_response(bank_data$month, "month of contact")
```



- From the plots shown above, we can draw the following observations:
- **Education**: Response rate is only 9% for primary education, 11% for secondary and 15% for tertiary; implying that education clearly plays a crucial role in predicting response
- __ Marital Status__: Response rate is 15% for single prospects and 10% for married
- **Contact method**: Response rate is 15% and 13% for cellular and telephonic contact methods, respectively
- **Month of contact**: Interestingly, response rate varies drastically with month of contact; it is 52% in March, 47% in December and only 7% in May

We now have a decent understanding of the variable which are important in predicting response.

But this way, we can only only analyse the effect of each variable separately. We saw that multiple attributes like month of contact, marital status etc. affect the reponse rate. How do we analyse the *combined effect* of the variables? Also, how can we know which variables affect response rate more than others?

This is why we need machine learning **models.** We'll see that in the next section.

Modelling

Let's now build some machine learning models to predict the type of potential customers who are more likely to respond.

To build machine learning models, we use only a part of the data to train the model. This is called **training data**.

Rest of the data is used to test or evaluate the model, which is called **test data**.

We'll use 70% data to train the model and the rest 30% to test it.

Data Preparation

```
library(caret)

## Loading required package: lattice

library(caTools)
library(dummies)

## dummies-1.5.6 provided by Decision Patterns

bank_data <- bank_data[, -c(20, 21, 22)]


#creating dummy variables
bank_data$response <- as.integer(bank_data$response)
bank_data <- dummy.data.frame(bank_data)
bank_data$response <- as.factor(ifelse(bank_data$response == 1, "no", "yes"))


# splitting into train and test data
set.seed(1)
split_indices <- sample.split(bank_data$response, SplitRatio = 0.70)
train <- bank_data[split_indices, ]
test <- bank_data[!split_indices, ]
nrow(train)/nrow(bank_data)

## [1] 0.6999845

nrow(test)/nrow(bank_data)

## [1] 0.3000155
```

Model 1: Logistic Regression

Let's build the first model - **logistic regression**.

```
library(MASS)
library(car)
logistic_1 <- glm(response ~ ., family = "binomial", data = train)
summary(logistic_1)

## 
## Call:
## glm(formula = response ~ ., family = "binomial", data = train)
## 
## Deviance Residuals: 
##     Min       1Q   Median       3Q      Max  
## -4.7547  -0.3728  -0.2496  -0.1458   3.4916  
## 
## Coefficients: (11 not defined because of singularities)
##                         Estimate Std. Error z value Pr(>|z|)    
## (Intercept)           -4.457e+00  3.990e-01 -11.170  < 2e-16 ***
## age                   -2.450e-03  2.641e-03  -0.928  0.35354    
## jobadmin.              3.545e-01  2.679e-01   1.323  0.18579    
## `jobblue-collar`       2.078e-02  2.673e-01   0.078  0.93802    
## jobentrepreneur       -6.857e-02  2.951e-01  -0.232  0.81627    
## jobhousemaid          -2.042e-01  2.988e-01  -0.684  0.49427    
## jobmanagement          2.123e-01  2.659e-01   0.799  0.42448    
## jobretired             5.813e-01  2.727e-01   2.132  0.03304 *  
## `jobself-employed`     2.009e-01  2.833e-01   0.709  0.47819    
## jobservices            6.568e-02  2.735e-01   0.240  0.81021    
## jobstudent             7.528e-01  2.825e-01   2.665  0.00769 ** 
## jobtechnician          1.799e-01  2.659e-01   0.677  0.49866    
## jobunemployed          1.390e-01  2.852e-01   0.487  0.62599    
## jobunknown                    NA         NA      NA       NA    
## salary                        NA         NA      NA       NA    
## maritaldivorced       -4.294e-02  8.107e-02  -0.530  0.59636    
## maritalmarried        -2.074e-01  6.652e-02  -3.118  0.00182 ** 
## maritalsingle                 NA         NA      NA       NA    
## educationprimary      -2.118e-01  1.511e-01  -1.402  0.16080    
## educationsecondary -9.074e-02  1.418e-01  -0.640  0.52218    
## educationtertiary      8.320e-02  1.224e-01   0.680  0.49654    
## educationunknown              NA         NA      NA       NA    
## targetedno             8.358e-03  9.161e-02   0.091  0.92731    
## targetedyes                   NA         NA      NA       NA    
## defaultno             -8.588e-03  1.928e-01  -0.045  0.96447    
## defaultyes                    NA         NA      NA       NA    
## balance                1.529e-05  6.019e-06   2.539  0.01111 *  
## housingno              7.693e-01  5.306e-02  14.500  < 2e-16 ***
## housingyes                    NA         NA      NA       NA    
## loanno                 3.636e-01  7.175e-02   5.067 4.04e-07 ***
## loanyes                       NA         NA      NA       NA    
## contactcellular        1.643e+00  8.798e-02  18.670  < 2e-16 ***
## contacttelephone       1.509e+00  1.210e-01  12.475  < 2e-16 ***
## contactunknown                NA         NA      NA       NA    
## day                    7.950e-03  2.977e-03   2.671  0.00757 ** 
```

```
## monthapr           -8.189e-01  1.425e-01  -5.745 9.21e-09 ***
## monthaug           -1.530e+00  1.372e-01 -11.149  < 2e-16 ***
## monthdec           -1.471e-01  2.332e-01  -0.631  0.52825    
## monthfeb           -9.456e-01  1.441e-01  -6.562 5.31e-11 ***
## monthjan           -2.124e+00  1.817e-01 -11.686  < 2e-16 ***
## monthjul           -1.659e+00  1.406e-01 -11.798  < 2e-16 ***
## monthjun           -3.797e-01  1.473e-01  -2.577  0.00996 ** 
## monthmar            7.240e-01  1.733e-01   4.177 2.95e-05 ***
## monthmay           -1.191e+00  1.366e-01  -8.725  < 2e-16 ***
## monthnov           -1.678e+00  1.456e-01 -11.524  < 2e-16 ***
## monthoct            1.125e-01  1.627e-01   0.692  0.48908    
## monthsep                   NA         NA      NA       NA    
## duration            4.272e-03  7.820e-05  54.631  < 2e-16 ***
## campaign           -9.677e-02  1.230e-02  -7.866 3.65e-15 ***
## pdays               5.737e-04  3.556e-04   1.613  0.10667    
## previous            1.169e-02  7.290e-03   1.604  0.10872    
## poutcomefailure    -6.733e-02  1.113e-01  -0.605  0.54529    
## poutcomeother       4.195e-02  1.279e-01   0.328  0.74290    
## poutcomesuccess     2.269e+00  1.007e-01  22.535  < 2e-16 ***
## poutcomeunknown            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: 22840  on 31646  degrees of freedom
## Residual deviance: 15016  on 31603  degrees of freedom
## AIC: 15104
## 
## Number of Fisher Scoring iterations: 6
```

```
#stepAIC(logistic_1, direction = "both")
# stepAIC has removed some variables and only the following ones remain
logistic_2 <- glm(formula = response ~ jobadmin. + jobhousemaid + 
jobmanagement + 
    jobretired + jobstudent + jobtechnician + maritalmarried + 
    educationprimary + educationsecondary + balance + housingno + 
    loanno + contactcellular + contacttelephone + day + monthapr + 
    monthaug + monthfeb + monthjan + monthjul + monthjun + monthmar + 
    monthmay + monthnov + duration + campaign + pdays + previous + 
    poutcomesuccess, family = "binomial", data = train)
```

```
##          jobadmin.       jobhousemaid      jobmanagement 
##           1.279574           1.075275           1.848829 
##         jobretired         jobstudent      jobtechnician 
##           1.269010           1.186903           1.356935 
##     maritalmarried   educationprimary educationsecondary 
##           1.094159           1.481352           1.639156 
##            balance          housingno             loanno 
##           1.033559           1.410136           1.057855 
```

```
##    contactcellular   contacttelephone                day
##           2.472082           1.951957           1.315402
##           monthapr           monthaug           monthfeb
##           2.146210           2.555384           1.980877
##           monthjan           monthjul           monthjun
##           1.398465           2.495161           2.511823
##           monthmar           monthmay           monthnov
##           1.337560           3.180751           1.937072
##           duration           campaign              pdays
##           1.131015           1.102743           1.357429
##           previous    poutcomesuccess
##           1.161571           1.133248
```

```
##
## Call:
## glm(formula = response ~ jobadmin. + jobhousemaid + jobmanagement +
##     jobretired + jobstudent + jobtechnician + maritalmarried +
##     educationprimary + educationsecondary + balance + housingno +
##     loanno + contactcellular + contacttelephone + day + monthapr +
##     monthaug + monthfeb + monthjan + monthjul + monthjun + monthmar +
##     monthmay + monthnov + duration + campaign + pdays + previous +
##     poutcomesuccess, family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.7557  -0.3727  -0.2504  -0.1459   3.4618
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -4.406e+00  1.614e-01 -27.295  < 2e-16 ***
## jobadmin.           2.964e-01  7.371e-02   4.021 5.79e-05 ***
## jobhousemaid       -2.749e-01  1.541e-01  -1.784 0.074388 .
## jobmanagement       1.560e-01  6.858e-02   2.274 0.022951 *
## jobretired          4.643e-01  8.931e-02   5.198 2.01e-07 ***
## jobstudent          7.148e-01  1.171e-01   6.101 1.05e-09 ***
## jobtechnician       1.234e-01  6.775e-02   1.821 0.068580 .
## maritalmarried     -2.172e-01  4.532e-02  -4.792 1.65e-06 ***
## educationprimary   -3.120e-01  8.208e-02  -3.800 0.000144 ***
## educationsecondary -1.738e-01  5.534e-02  -3.141 0.001685 **
## balance             1.519e-05  5.983e-06   2.539 0.011128 *
## housingno           7.677e-01  5.242e-02  14.646  < 2e-16 ***
## loanno              3.658e-01  7.136e-02   5.126 2.95e-07 ***
## contactcellular     1.651e+00  8.739e-02  18.894  < 2e-16 ***
## contacttelephone    1.503e+00  1.199e-01  12.534  < 2e-16 ***
## day                 8.415e-03  2.956e-03   2.847 0.004420 **
## monthapr           -8.538e-01  1.037e-01  -8.232  < 2e-16 ***
## monthaug           -1.560e+00  9.726e-02 -16.039  < 2e-16 ***
## monthfeb           -9.689e-01  1.087e-01  -8.910  < 2e-16 ***
## monthjan           -2.153e+00  1.516e-01 -14.204  < 2e-16 ***
## monthjul           -1.692e+00  1.003e-01 -16.878  < 2e-16 ***
```

```
## monthjun          -4.069e-01  1.128e-01  -3.608 0.000309 ***
## monthmar           6.990e-01  1.440e-01   4.854 1.21e-06 ***
## monthmay          -1.222e+00  9.672e-02 -12.638  < 2e-16 ***
## monthnov          -1.717e+00  1.077e-01 -15.931  < 2e-16 ***
## duration           4.272e-03  7.812e-05  54.690  < 2e-16 ***
## campaign          -9.724e-02  1.229e-02  -7.915 2.48e-15 ***
## pdays              4.559e-04  2.182e-04   2.089 0.036703 *
## previous           1.183e-02  7.001e-03   1.690 0.091013 .
## poutcomesuccess    2.290e+00  8.089e-02  28.309  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 22840  on 31646  degrees of freedom
## Residual deviance: 15025  on 31617  degrees of freedom
## AIC: 15085
##
## Number of Fisher Scoring iterations: 6
```

```
##
## Call:
## glm(formula = response ~ jobadmin. + jobhousemaid + jobmanagement +
##     jobretired + jobstudent + jobtechnician + maritalmarried +
##     educationprimary + educationsecondary + balance + housingno +
##     loanno + contactcellular + contacttelephone + day + monthapr +
##     monthaug + monthfeb + monthjan + monthjul + monthjun + monthmar +
##     +monthnov + duration + campaign + pdays + previous + poutcomesuccess,
##     family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.6947  -0.3816  -0.2524  -0.1457   3.4782
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)       -5.564e+00  1.343e-01 -41.441  < 2e-16 ***
## jobadmin.          3.234e-01  7.321e-02   4.418 9.98e-06 ***
## jobhousemaid      -2.133e-01  1.530e-01  -1.394 0.163357
## jobmanagement      1.885e-01  6.815e-02   2.766 0.005680 **
## jobretired         5.705e-01  8.814e-02   6.473 9.61e-11 ***
## jobstudent         7.570e-01  1.161e-01   6.523 6.88e-11 ***
## jobtechnician      1.505e-01  6.744e-02   2.232 0.025612 *
## maritalmarried    -1.959e-01  4.503e-02  -4.351 1.35e-05 ***
## educationprimary  -3.293e-01  8.158e-02  -4.037 5.41e-05 ***
## educationsecondary -1.842e-01  5.502e-02  -3.347 0.000817 ***
## balance            1.712e-05  5.917e-06   2.894 0.003800 **
## housingno          9.394e-01  5.098e-02  18.425  < 2e-16 ***
## loanno             3.815e-01  7.118e-02   5.359 8.36e-08 ***
## contactcellular    1.834e+00  8.457e-02  21.681  < 2e-16 ***
```

```
## contacttelephone    1.716e+00  1.173e-01  14.633  < 2e-16 ***
## day                 9.709e-03  2.951e-03   3.289 0.001004 **
## monthapr           -4.432e-03  8.090e-02  -0.055 0.956316
## monthaug           -7.991e-01  7.815e-02 -10.225  < 2e-16 ***
## monthfeb           -1.644e-01  9.005e-02  -1.826 0.067853 .
## monthjan           -1.392e+00  1.404e-01  -9.915  < 2e-16 ***
## monthjul           -8.762e-01  7.845e-02 -11.168  < 2e-16 ***
## monthjun            4.922e-01  8.870e-02   5.549 2.87e-08 ***
## monthmar            1.467e+00  1.328e-01  11.045  < 2e-16 ***
## monthnov           -9.104e-01  8.835e-02 -10.305  < 2e-16 ***
## duration            4.246e-03  7.769e-05  54.648  < 2e-16 ***
## campaign           -1.018e-01  1.235e-02  -8.240  < 2e-16 ***
## pdays               4.146e-04  2.194e-04   1.890 0.058793 .
## previous            1.414e-02  7.667e-03   1.844 0.065154 .
## poutcomesuccess     2.354e+00  8.025e-02  29.327  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 22840  on 31646  degrees of freedom
## Residual deviance: 15183  on 31618  degrees of freedom
## AIC: 15241
##
## Number of Fisher Scoring iterations: 6
```

```
##          jobadmin.       jobhousemaid      jobmanagement
##           1.278345           1.073900           1.852500
##         jobretired         jobstudent      jobtechnician
##           1.255725           1.186584           1.355827
##     maritalmarried   educationprimary educationsecondary
##           1.093686           1.478612           1.640305
##            balance          housingno             loanno
##           1.034041           1.347969           1.057721
##    contactcellular   contacttelephone                day
##           2.331771           1.885092           1.314812
##           monthapr           monthaug           monthfeb
##           1.295233           1.649712           1.344307
##           monthjan           monthjul           monthjun
##           1.191206           1.524433           1.517659
##           monthmar           monthnov           duration
##           1.119970           1.296945           1.127920
##           campaign              pdays           previous
##           1.100228           1.390133           1.197572
##    poutcomesuccess
##           1.136761
```

```
##
## Call:
## glm(formula = response ~ jobadmin. + jobmanagement + jobretired +
```

```
##     jobstudent + jobtechnician + maritalmarried + educationprimary +
##     educationsecondary + balance + housingno + loanno + contactcellular +
##     contacttelephone + day + monthapr + monthaug + monthfeb +
##     monthjan + monthjul + monthjun + monthmar + +monthnov + duration +
##     campaign + pdays + previous + poutcomesuccess, family = "binomial",
##     data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.6895  -0.3818  -0.2529  -0.1457   3.4820
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -5.569e+00  1.342e-01 -41.485  < 2e-16 ***
## jobadmin.           3.346e-01  7.282e-02   4.594 4.34e-06 ***
## jobmanagement       2.019e-01  6.753e-02   2.990 0.002790 **
## jobretired          5.886e-01  8.724e-02   6.747 1.51e-11 ***
## jobstudent          7.701e-01  1.157e-01   6.656 2.82e-11 ***
## jobtechnician       1.628e-01  6.691e-02   2.433 0.014984 *
## maritalmarried     -1.967e-01  4.503e-02  -4.367 1.26e-05 ***
## educationprimary   -3.408e-01  8.122e-02  -4.196 2.71e-05 ***
## educationsecondary -1.823e-01  5.500e-02  -3.314 0.000919 ***
## balance             1.714e-05  5.919e-06   2.896 0.003784 **
## housingno           9.350e-01  5.088e-02  18.376  < 2e-16 ***
## loanno              3.804e-01  7.119e-02   5.342 9.17e-08 ***
## contactcellular     1.833e+00  8.459e-02  21.671  < 2e-16 ***
## contacttelephone    1.713e+00  1.173e-01  14.606  < 2e-16 ***
## day                 9.628e-03  2.951e-03   3.263 0.001104 **
## monthapr           -4.825e-03  8.091e-02  -0.060 0.952441
## monthaug           -8.037e-01  7.807e-02 -10.296  < 2e-16 ***
## monthfeb           -1.650e-01  9.003e-02  -1.833 0.066842 .
## monthjan           -1.388e+00  1.403e-01  -9.896  < 2e-16 ***
## monthjul           -8.788e-01  7.845e-02 -11.202  < 2e-16 ***
## monthjun            4.877e-01  8.865e-02   5.501 3.78e-08 ***
## monthmar            1.464e+00  1.328e-01  11.025  < 2e-16 ***
## monthnov           -9.111e-01  8.834e-02 -10.314  < 2e-16 ***
## duration            4.243e-03  7.766e-05  54.641  < 2e-16 ***
## campaign           -1.015e-01  1.234e-02  -8.222  < 2e-16 ***
## pdays               4.166e-04  2.193e-04   1.900 0.057494 .
## previous            1.408e-02  7.651e-03   1.840 0.065727 .
## poutcomesuccess     2.354e+00  8.023e-02  29.345  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 22840  on 31646  degrees of freedom
## Residual deviance: 15185  on 31619  degrees of freedom
## AIC: 15241
```

```
##
## Number of Fisher Scoring iterations: 6
```

```
##          jobadmin.      jobmanagement         jobretired
##           1.265142           1.818987           1.230380
##         jobstudent      jobtechnician     maritalmarried
##           1.179704           1.335063           1.093653
##   educationprimary educationsecondary            balance
##           1.464271           1.639060           1.034000
##          housingno             loanno    contactcellular
##           1.342635           1.057621           2.333056
##   contacttelephone                day           monthapr
##           1.884936           1.314397           1.295365
##           monthaug           monthfeb           monthjan
##           1.647107           1.344399           1.190980
##           monthjul           monthjun           monthmar
##           1.523644           1.516960           1.119701
##           monthnov           duration           campaign
##           1.297082           1.126930           1.099934
##              pdays           previous    poutcomesuccess
##           1.389528           1.196844           1.136652
```

```
##
## Call:
## glm(formula = response ~ jobadmin. + jobmanagement + jobretired +
##     jobstudent + jobtechnician + maritalmarried + educationprimary +
##     educationsecondary + balance + housingno + loanno + contactcellular +
##     contacttelephone + day + monthaug + monthjan + monthjul +
##     monthjun + monthmar + +monthnov + duration + campaign +
poutcomesuccess,
##     family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.6847  -0.3811  -0.2531  -0.1468   3.4937
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -5.587e+00  1.326e-01 -42.146  < 2e-16 ***
## jobadmin.           3.410e-01  7.276e-02   4.686 2.78e-06 ***
## jobmanagement       2.061e-01  6.746e-02   3.055  0.00225 **
## jobretired          5.915e-01  8.716e-02   6.787 1.15e-11 ***
## jobstudent          7.819e-01  1.156e-01   6.763 1.35e-11 ***
## jobtechnician       1.630e-01  6.690e-02   2.437  0.01482 *
## maritalmarried     -1.928e-01  4.497e-02  -4.288 1.80e-05 ***
## educationprimary   -3.370e-01  8.117e-02  -4.152 3.29e-05 ***
## educationsecondary -1.787e-01  5.494e-02  -3.253  0.00114 **
## balance             1.724e-05  5.909e-06   2.917  0.00353 **
## housingno           8.965e-01  4.937e-02  18.159  < 2e-16 ***
## loanno              3.806e-01  7.115e-02   5.349 8.84e-08 ***
```

```
## contactcellular     1.867e+00  8.052e-02  23.183  < 2e-16 ***
## contacttelephone    1.745e+00  1.147e-01  15.213  < 2e-16 ***
## day                 1.090e-02  2.785e-03   3.915 9.04e-05 ***
## monthaug           -7.961e-01  7.053e-02 -11.287  < 2e-16 ***
## monthjan           -1.382e+00  1.372e-01 -10.078  < 2e-16 ***
## monthjul           -8.852e-01  7.136e-02 -12.405  < 2e-16 ***
## monthjun            5.168e-01  8.586e-02   6.019 1.75e-09 ***
## monthmar            1.493e+00  1.292e-01  11.561  < 2e-16 ***
## monthnov           -9.040e-01  8.293e-02 -10.900  < 2e-16 ***
## duration            4.241e-03  7.748e-05  54.728  < 2e-16 ***
## campaign           -1.030e-01  1.229e-02  -8.383  < 2e-16 ***
## poutcomesuccess     2.433e+00  7.682e-02  31.668  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 22840  on 31646  degrees of freedom
## Residual deviance: 15200  on 31623  degrees of freedom
## AIC: 15248
##
## Number of Fisher Scoring iterations: 6
```

```
##          jobadmin.      jobmanagement         jobretired
##           1.264792           1.817724           1.230185
##         jobstudent      jobtechnician     maritalmarried
##           1.177893           1.334885           1.091911
##   educationprimary educationsecondary            balance
##           1.463544           1.636925           1.033673
##          housingno             loanno    contactcellular
##           1.265137           1.056964           2.115809
##   contacttelephone                day           monthaug
##           1.806276           1.178723           1.346639
##           monthjan           monthjul           monthjun
##           1.137174           1.264032           1.423663
##           monthmar           monthnov           duration
##           1.062195           1.144795           1.123197
##           campaign    poutcomesuccess
##           1.092087           1.041965
```

```
##
## Call:
## glm(formula = response ~ jobadmin. + jobmanagement + jobretired +
##     jobstudent + maritalmarried + educationprimary + educationsecondary +
##     balance + housingno + loanno + contactcellular + contacttelephone +
##     day + monthaug + monthjan + monthjul + monthjun + monthmar +
##     +monthnov + duration + campaign + poutcomesuccess, family =
"binomial",
##     data = train)
##
```

```
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.6982  -0.3810  -0.2534  -0.1468   3.4930
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -5.531e+00  1.304e-01 -42.419  < 2e-16 ***
## jobadmin.           2.856e-01  6.893e-02   4.144 3.42e-05 ***
## jobmanagement       1.448e-01  6.241e-02   2.321 0.020292 *
## jobretired          5.463e-01  8.499e-02   6.427 1.30e-10 ***
## jobstudent          7.216e-01  1.128e-01   6.396 1.60e-10 ***
## maritalmarried     -1.989e-01  4.491e-02  -4.429 9.48e-06 ***
## educationprimary   -3.765e-01  7.942e-02  -4.741 2.13e-06 ***
## educationsecondary -1.838e-01  5.492e-02  -3.346 0.000818 ***
## balance             1.727e-05  5.895e-06   2.929 0.003395 **
## housingno           8.976e-01  4.938e-02  18.178  < 2e-16 ***
## loanno              3.794e-01  7.113e-02   5.333 9.64e-08 ***
## contactcellular     1.871e+00  8.055e-02  23.234  < 2e-16 ***
## contacttelephone    1.746e+00  1.147e-01  15.226  < 2e-16 ***
## day                 1.111e-02  2.784e-03   3.990 6.59e-05 ***
## monthaug           -7.763e-01  7.008e-02 -11.078  < 2e-16 ***
## monthjan           -1.384e+00  1.371e-01 -10.095  < 2e-16 ***
## monthjul           -8.868e-01  7.136e-02 -12.426  < 2e-16 ***
## monthjun            5.181e-01  8.590e-02   6.031 1.63e-09 ***
## monthmar            1.499e+00  1.292e-01  11.609  < 2e-16 ***
## monthnov           -9.047e-01  8.290e-02 -10.912  < 2e-16 ***
## duration            4.237e-03  7.744e-05  54.717  < 2e-16 ***
## campaign           -1.033e-01  1.228e-02  -8.405  < 2e-16 ***
## poutcomesuccess     2.436e+00  7.680e-02  31.719  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 22840  on 31646  degrees of freedom
## Residual deviance: 15206  on 31624  degrees of freedom
## AIC: 15252
##
## Number of Fisher Scoring iterations: 6

##
## Call:
## glm(formula = response ~ jobadmin. + jobretired + jobstudent +
##     maritalmarried + educationprimary + educationsecondary +
##     balance + housingno + loanno + contactcellular + contacttelephone +
##     day + monthaug + monthjan + monthjul + monthjun + monthmar +
##     +monthnov + duration + campaign + poutcomesuccess, family =
"binomial",
##     data = train)
##
```

```
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.6989  -0.3814  -0.2541  -0.1467   3.4880
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -5.461e+00  1.266e-01 -43.123  < 2e-16 ***
## jobadmin.           2.570e-01  6.773e-02   3.794 0.000148 ***
## jobretired          5.138e-01  8.376e-02   6.134 8.57e-10 ***
## jobstudent          6.756e-01  1.110e-01   6.085 1.17e-09 ***
## maritalmarried     -1.970e-01  4.489e-02  -4.388 1.15e-05 ***
## educationprimary   -4.411e-01  7.419e-02  -5.945 2.76e-09 ***
## educationsecondary -2.457e-01  4.777e-02  -5.143 2.70e-07 ***
## balance             1.757e-05  5.890e-06   2.983 0.002854 **
## housingno           8.985e-01  4.938e-02  18.196  < 2e-16 ***
## loanno              3.806e-01  7.113e-02   5.351 8.72e-08 ***
## contactcellular     1.878e+00  8.052e-02  23.328  < 2e-16 ***
## contacttelephone    1.750e+00  1.147e-01  15.258  < 2e-16 ***
## day                 1.113e-02  2.784e-03   3.997 6.42e-05 ***
## monthaug           -7.735e-01  7.006e-02 -11.040  < 2e-16 ***
## monthjan           -1.388e+00  1.371e-01 -10.122  < 2e-16 ***
## monthjul           -8.876e-01  7.135e-02 -12.440  < 2e-16 ***
## monthjun            5.188e-01  8.592e-02   6.038 1.56e-09 ***
## monthmar            1.506e+00  1.291e-01  11.664  < 2e-16 ***
## monthnov           -9.020e-01  8.292e-02 -10.878  < 2e-16 ***
## duration            4.234e-03  7.739e-05  54.718  < 2e-16 ***
## campaign           -1.029e-01  1.227e-02  -8.389  < 2e-16 ***
## poutcomesuccess     2.436e+00  7.676e-02  31.736  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 22840  on 31646  degrees of freedom
## Residual deviance: 15211  on 31625  degrees of freedom
## AIC: 15255
##
## Number of Fisher Scoring iterations: 6
```

```
##          jobadmin.         jobretired         jobstudent
##           1.095196           1.136371           1.085759
##     maritalmarried   educationprimary educationsecondary
##           1.088615           1.226031           1.238695
##            balance          housingno             loanno
##           1.032898           1.266552           1.057005
##    contactcellular   contacttelephone                day
##           2.116738           1.808209           1.177530
##           monthaug           monthjan           monthjul
##           1.327510           1.137262           1.264405
##           monthjun           monthmar           monthnov
```

```
##          1.426528          1.061397          1.144361
##          duration          campaign    poutcomesuccess
##          1.121823          1.091999          1.041863
```

We now have a logistic model named logistic_final. The variables in the final model can be seen above, like job, education, balance, housing, contact method, month of contact etc.

Next, we'll use the model to predict the response in the test data.

```
predictions_logit <- predict(logistic_final, newdata = test[, -55], type =
"response")
summary(predictions_logit)

##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 0.0000564 0.0190000 0.0446700 0.1161000 0.1090000 1.0000000
```

So now we have predicted the 'probabilities of responding' (for the test data). Note that the average probability (as shown above in summary(predictions_logit) is 11.6%, which is the average response rate.

Next comes the interesting part. We need to convert the probabilities to an actual prediction, i.e. **yes or no**. Can we just say that anything *above 50% probability of response is yes and no otherwise*? Yes, we could, but we can do better.

We can rather experiment with other **probability thresholds** like 30%, 40% etc. We will go with whatever gives us the highest (loosely speaking) **accuracy**. In fact, apart from accuracy, there are other metrics to **evaluate the model** like sensitivity, specificity etc.

Model Evaluation

In model evaluation, we use the test data to evaluate how good the model is (note that it was trained on 'train data' and hasn't seen the test data, so we are not cheating).

Let us first look at how **accurate** the predictions are. For now, let's use a probability cutoff of 50% and then we'll iterate.

```
predicted_response <- factor(ifelse(predictions_logit >= 0.50, "yes", "no"))
conf <- confusionMatrix(predicted_response, test$response, positive = "yes")
conf

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    no   yes
##        no  11684  1046
##        yes   293   541
##
##                Accuracy : 0.9013
##                  95% CI : (0.8961, 0.9063)
##     No Information Rate : 0.883
##     P-Value [Acc > NIR] : 6.515e-12
```

```
##
##                  Kappa : 0.3984
##  Mcnemar's Test P-Value : < 2.2e-16
##
##            Sensitivity : 0.34089
##            Specificity : 0.97554
##         Pos Pred Value : 0.64868
##         Neg Pred Value : 0.91783
##             Prevalence : 0.11700
##         Detection Rate : 0.03988
##   Detection Prevalence : 0.06149
##      Balanced Accuracy : 0.65822
##
##       'Positive' Class : yes
##
```

Firstly, note that the **accuracy** is approx. 90% which means that the model has made about 90% predictions correct (whether yes or no).

There are two other important metrics - **sensitivity** and **specificity**.

Sensitivity is the fraction of correctly identified responses, i.e. out of those who will actually respond, how many has the model identified.

Specificity is the fraction of **incorrectly identified responses**, i.e. out of those who will actually NOT respond, how many has the model identified.

These two metrics can be calculated using the table of predictions as follows:

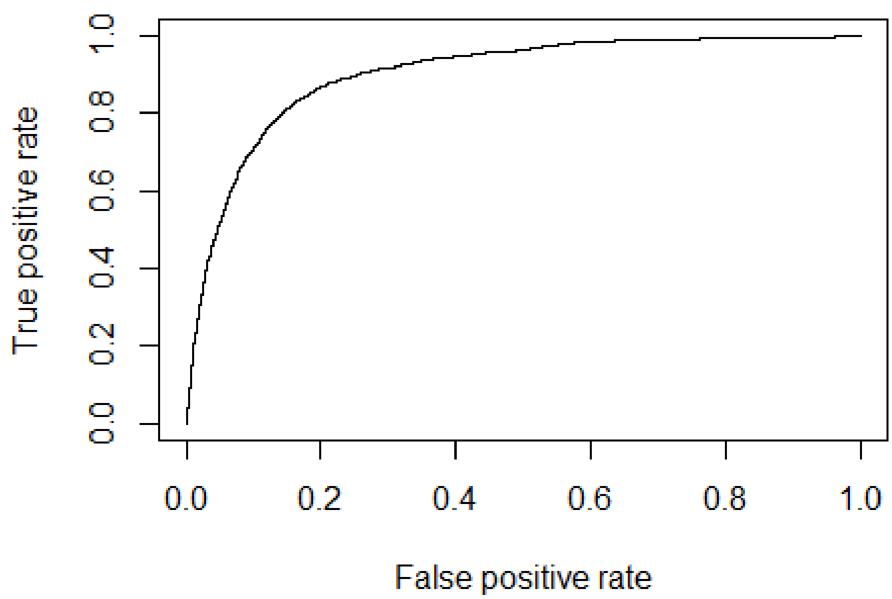
```
sensitivity <- conf$byClass[1]
specificity <- conf$byClass[2]
sensitivity

## Sensitivity
##   0.3408948

specificity

## Specificity
##   0.9755364
```

The values of sensitivity and specificity are about 31.75% and 97.72% respectively. This means that the model predicts 97.72% of those who will NOT buy correctly while only 31.75% of those who'll buy.

Since the number of "yes" responders are few (only 11% respond), it is hard to predict them. So if you market the product to about 10,000 people, you know that about 1100 will respond. The model will identify about 31% or 350 of them correctly.

But these predictions are based on an arbitrary cut-off of 0.50 probability. Now that we know what accuracy, specificity and sensitivity mean, we can find a cutoff which optimises the most important metric. In our case, it is sensitivity.

```
library(ROCR)

## Loading required package: gplots

##
## Attaching package: 'gplots'

## The following object is masked from 'package:stats':
##
##     lowess

predictions_object <- prediction(predictions_logit, test$response)
perf_object <- performance(predictions_object, "tpr", "fpr")
plot(perf_object)
```

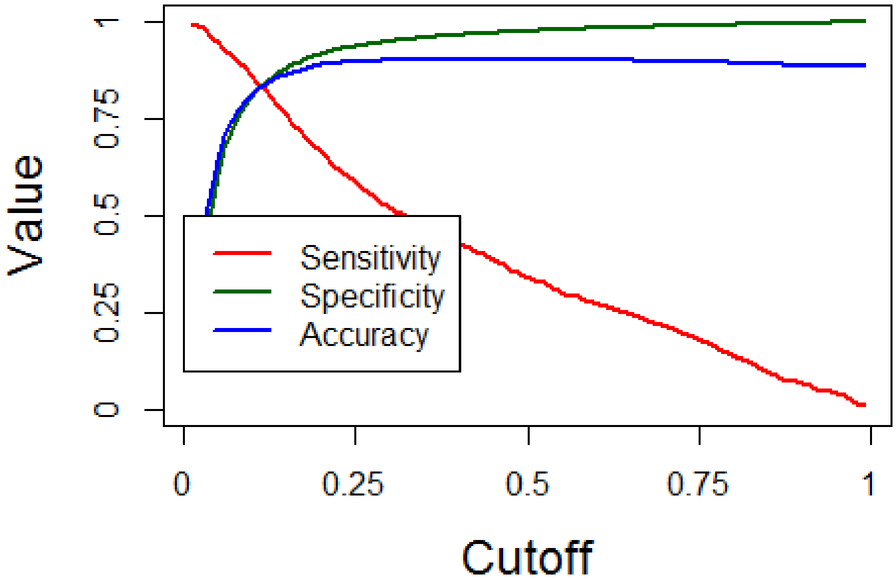


The plot shown above is called the **ROC** curve. It has False Positive Rate (FPR) and True Positive Rate (TPR, or sensitivity) on the x and y axes respectively.

The objective is to **maximise the TPR** and **minimise the FPR** which means that we want the curve to be aligned towards the **top-left**.

Now, let's find oput the optimal probabilty cutoff, i.e. the value abpve which we'll predict "yes" and "no" otherwise. We can plot the three metrics against cutoff values ranging from 0% to 100% and choose the one which gives high accuracy, sensitivity and specificity.

```
perform_fn <- function(cutoff)
  {
  predicted_response <- factor(ifelse(predictions_logit >= cutoff, "yes",
"no"))
  conf <- confusionMatrix(predicted_response, test$response, positive =
"yes")
  acc <- conf$overall[1]
  sens <- conf$byClass[1]
  spec <- conf$byClass[2]
  out <- t(as.matrix(c(sens, spec, acc)))
  colnames(out) <- c("sensitivity", "specificity", "accuracy")
  return(out)
}

# creating cutoff values from 0.01 to 0.99 for plotting and initialising a
matrix of size 1000x4
s = seq(.01,.99,length=100)
OUT = matrix(0,100,3)

# calculate the sens, spec and acc for different cutoff values
for(i in 1:100)
  {
  OUT[i,] = perform_fn(s[i])
}


# plotting cutoffs
plot(s,
OUT[,1],xlab="Cutoff",ylab="Value",cex.lab=1.5,cex.axis=1.5,ylim=c(0,1),type=
"l",lwd=2,axes=FALSE,col=2)
axis(1,seq(0,1,length=5),seq(0,1,length=5),cex.lab=1.5)
axis(2,seq(0,1,length=5),seq(0,1,length=5),cex.lab=1.5)
lines(s,OUT[,2],col="darkgreen",lwd=2)
lines(s,OUT[,3],col=4,lwd=2)
box()
legend(0,.50,col=c(2,"darkgreen",4,"darkred"),lwd=c(2,2,2,2),c("Sensitivity",
"Specificity","Accuracy"))
```



The plot above shows the sensitivity, specificity and accuracy for cutoff probabilities ranging from 0 to 100. It is clear that a cutoff around 12-13% will optimise the three metrics.

```
cutoffs <- s[which(abs(OUT[, 1] - OUT[, 2]) < 0.01)]
```

Let's choose a cutoff value of 12% for the final model.

```
##  Accuracy
## 0.8409024

## Sensitivity
##   0.8204159

## Specificity
##   0.8436169
```

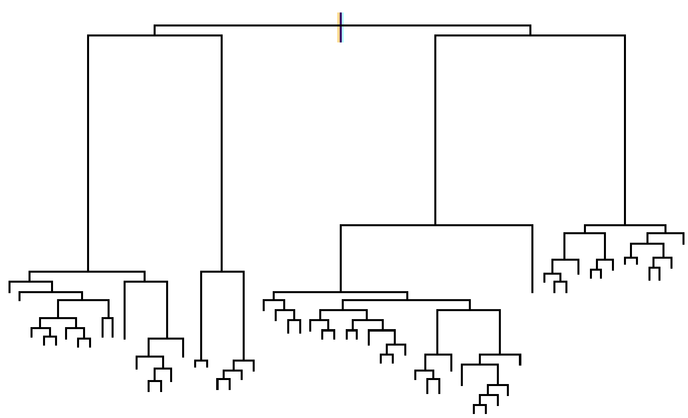
We have accuracy = 82.62%, sensitivity = 75.29% and specificity = 83.59%. This is a remarkable improvement over cutoff = 0.50, where the sensitivity was around 31% only.

Now, if you market the product to 10,000 people (out of which around 1100 usually respond), the model will be able to identify 75% of 1100 or approx. 825 people correctly.

Model 2: Decision Tree

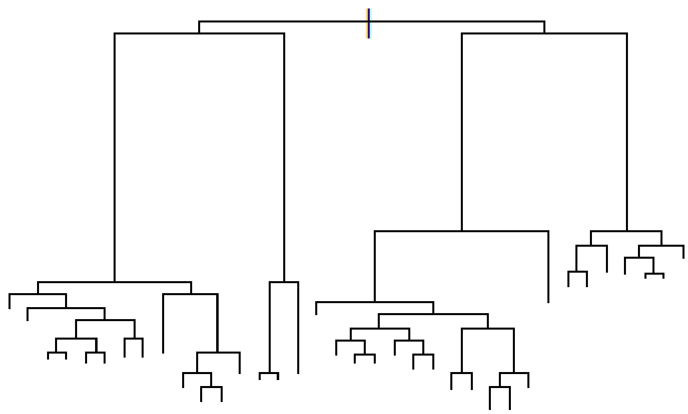
Let's build a decision tree and compare that with the logistic regression model.

```r
library(rpart)
library(rattle)

## Rattle: A free graphical interface for data mining with R.
## Version 4.1.0 Copyright (c) 2006-2015 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.

bank <- read.csv("bank-marketing.csv")
split_indices <- sample.split(bank$response, SplitRatio = 0.70)
train_dt <- bank[split_indices, ]
test_dt <- bank[!split_indices, ]
nrow(train_dt)/nrow(bank)

## [1] 0.6999845

nrow(test_dt)/nrow(bank)

## [1] 0.3000155

# building a tree with arbitrary minsplit and cp
banktree_1 <-  rpart(response ~ ., data=train_dt, method= "class",
                     control=rpart.control(minsplit=65, cp=0.001))
plot(banktree_1)
```

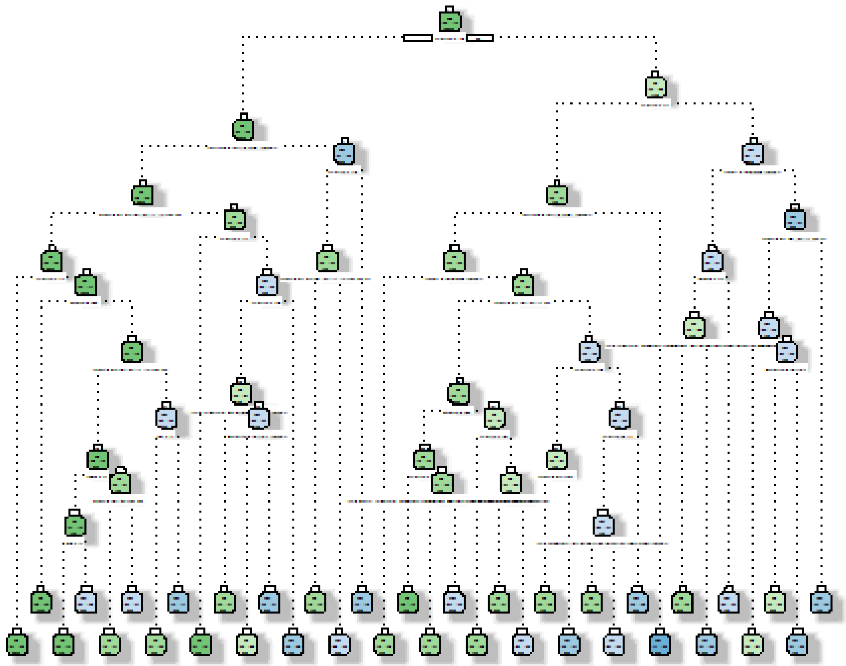


```r
# This is clearly an overfitted tree

# Increasing the minsplit two fold to 130
banktree_2 <-  rpart(response ~ ., data=train_dt, method= "class",
```

```
                                 control=rpart.control(minsplit=130, cp=0.001))
plot(banktree_2)
```

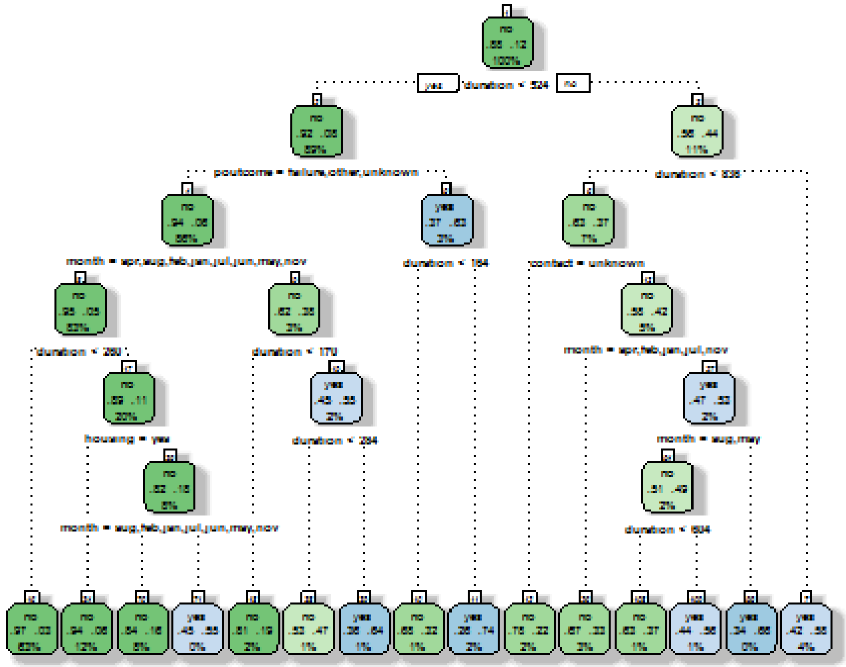


```
# This one is better, but still looks a little too complex
fancyRpartPlot(banktree_2)

## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```



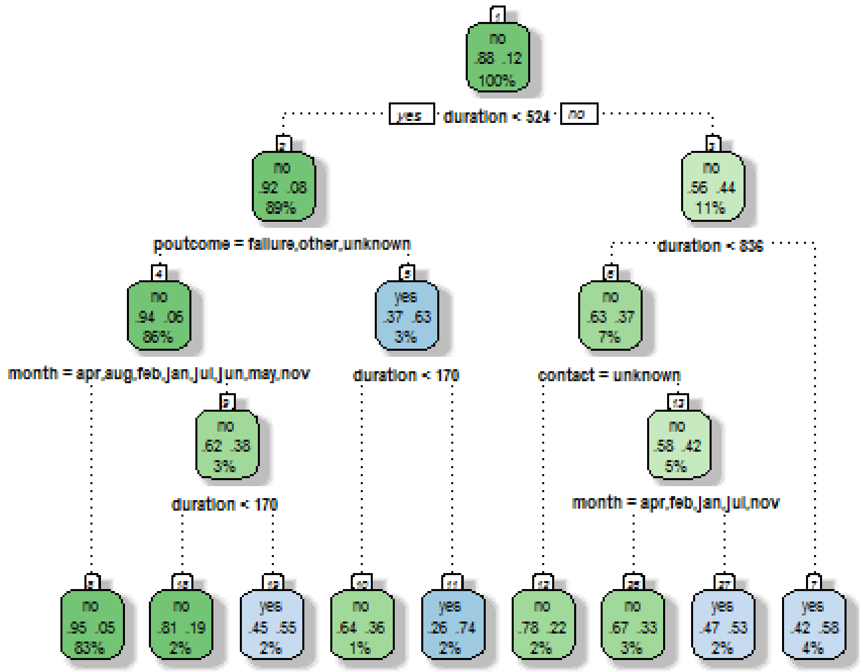
Rattle 2016-Dec-21 18:28:25 Pratika

```
# Listing the variables by importance: Duration, poutcome, month are the top
3
banktree_2$variable.importance

##      duration      poutcome         month       housing           age
## 1100.0806054   629.1219282   355.8465846    58.5726337    47.2998892
##           job       contact         pdays      previous           day
##    41.7918574    39.2661168    31.0693077    24.2778212    22.1378873
##        salary     education       balance      targeted       marital
##    11.4936286    10.8707432     9.5013333     4.8280236     2.1555044
##      campaign          loan       default
##     1.9067268     1.6462427     0.2932873

# We can further simplify the tree by increasing minsplit
banktree_3 <-  rpart(response ~ ., data=train_dt, method= "class",
                     control=rpart.control(minsplit=400, cp=0.001))

banktree_3$variable.importance

##      duration      poutcome         month       housing       contact
## 1086.3435731   563.7976032   336.4775041    48.0529388    41.8905974
##           day           age           job         pdays       balance
##    10.2809741     7.9403807     6.5981194     5.1803734     3.5911400
##        salary      previous          loan      campaign     education
##     3.3759324     2.7766546     1.6462427     1.1720273     0.3502029

fancyRpartPlot(banktree_3)
```



Rattle 2016-Dec-21 18:28:33 Pratika

```r
# banktree_3 looks like an acceptable model; contains
banktree_4 <- rpart(response ~ ., data=train_dt, method= "class",
                    control=rpart.control(minsplit=800, cp=0.001))
fancyRpartPlot(banktree_4)
```

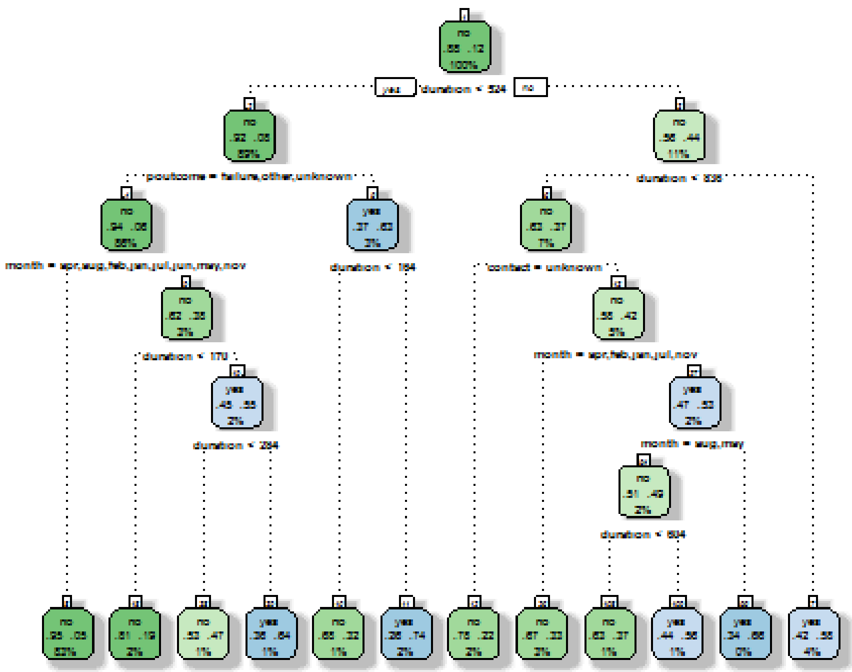


Rattle 2016-Dec-21 18:28:37 Pratika

```
banktree_4$variable.importance

##     duration     poutcome        month      contact          day        pdays
## 994.0661890 563.7976032 267.4717306  40.2783307   8.8805044   4.4344924
##      balance          age      housing     previous         loan     campaign
##    3.4058204    3.2271714    2.7437378    2.1176411    1.6462427    0.9971059

# We'll evaluate banktree _3 and banktree_4 using cross validation and choose
the optimal value of cp parameter
# The xerror is the average error measured during cross validation. The cp
value for which the xerror is minimum is our ideal choice for the cp
parameter.
# cp = 0.0024 minimises the xerror
printcp(banktree_3)

##
## Classification tree:
## rpart(formula = response ~ ., data = train_dt, method = "class",
##     control = rpart.control(minsplit = 400, cp = 0.001))
##
## Variables actually used in tree construction:
## [1] contact  duration housing  month    poutcome
##
## Root node error: 3702/31647 = 0.11698
##
## n= 31647
##
```

```
##          CP nsplit rel error  xerror     xstd
## 1 0.0374572      0   1.00000 1.00000 0.015444
## 2 0.0232307      3   0.88763 0.89762 0.014731
## 3 0.0074284      4   0.86440 0.87061 0.014533
## 4 0.0051324      6   0.84954 0.85521 0.014419
## 5 0.0015307     11   0.81929 0.83657 0.014278
## 6 0.0010000     14   0.81469 0.83522 0.014268
```

```
banktree_3 <- rpart(response ~ ., data=train_dt, method= "class",
                     control=rpart.control(minsplit=400, cp=0.0024))

fancyRpartPlot(banktree_3)
```

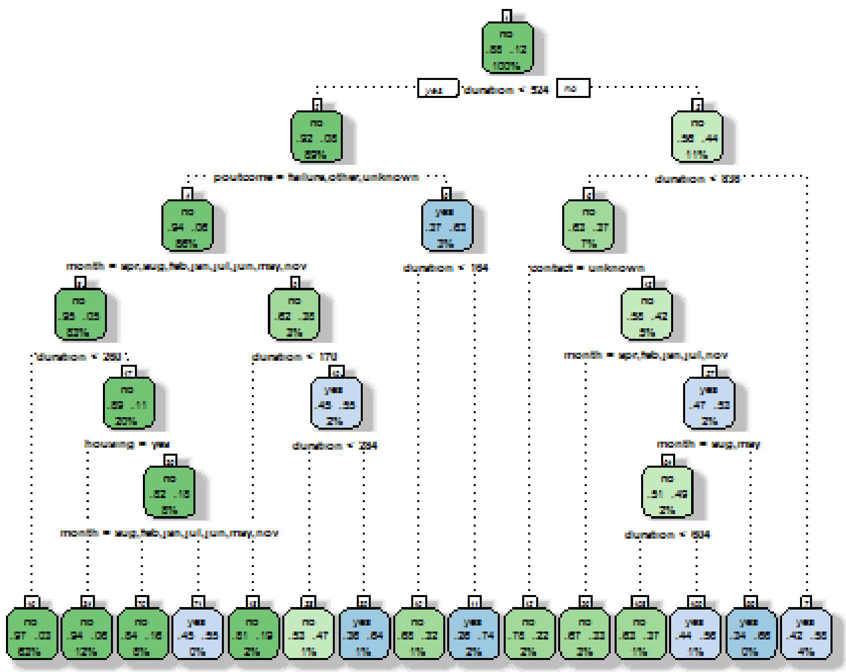


Rattle 2016-Dec-21 18:28:41 Pratika

```
# find optimal cp for banktree 4
printcp(banktree_4)

##
## Classification tree:
## rpart(formula = response ~ ., data = train_dt, method = "class",
##     control = rpart.control(minsplit = 800, cp = 0.001))
##
## Variables actually used in tree construction:
## [1] contact  duration month    poutcome
##
## Root node error: 3702/31647 = 0.11698
##
## n= 31647
```

```
##
##          CP nsplit rel error  xerror     xstd
## 1 0.0374572      0   1.00000 1.00000 0.015444
## 2 0.0197191      3   0.88763 0.89870 0.014739
## 3 0.0074284      4   0.86791 0.89141 0.014686
## 4 0.0051324      6   0.85305 0.89492 0.014712
## 5 0.0010000      8   0.84279 0.88682 0.014653
```

```
banktree_4 <- rpart(response ~ ., data=train_dt, method= "class",
                    control=rpart.control(minsplit=400, cp=0.001))
fancyRpartPlot(banktree_4)
```



Rattle 2016-Dec-21 18:28:46 Pratika

```
banktree_4$variable.importance
```

```
##     duration     poutcome        month      housing      contact
## 1086.3435731  563.7976032  336.4775041   48.0529388   41.8905974
##          day          age          job        pdays      balance
##   10.2809741    7.9403807    6.5981194    5.1803734    3.5911400
##       salary     previous         loan     campaign    education
##    3.3759324    2.7766546    1.6462427    1.1720273    0.3502029
```

```
## Model Evaluation for banktree_3 and banktree_4
# using test data from now on
# banktree_3
banktree_3_pred <- predict(banktree_3, test_dt[, -19], type = "class")
confusionMatrix(banktree_3_pred, test_dt[, 19], positive = "yes")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    no   yes
##        no  11577   898
##        yes   400   689
##
##                Accuracy : 0.9043
##                  95% CI : (0.8992, 0.9092)
##     No Information Rate : 0.883
##     P-Value [Acc > NIR] : 1.152e-15
##
##                   Kappa : 0.4639
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.43415
##             Specificity : 0.96660
##          Pos Pred Value : 0.63269
##          Neg Pred Value : 0.92802
##              Prevalence : 0.11700
##          Detection Rate : 0.05080
##    Detection Prevalence : 0.08029
##       Balanced Accuracy : 0.70038
##
##        'Positive' Class : yes
##
```

```
# Accuracy is 90.14%, sensitivity is only 42.91%

# banktree_4
banktree_4_pred <- predict(banktree_4, test_dt[, -19], type = "class")
confusionMatrix(banktree_4_pred, test_dt[, 19], positive = "yes")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    no   yes
##        no  11547   867
##        yes   430   720
##
##                Accuracy : 0.9044
##                  95% CI : (0.8993, 0.9093)
##     No Information Rate : 0.883
##     P-Value [Acc > NIR] : 9.166e-16
##
##                   Kappa : 0.4745
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.45369
##             Specificity : 0.96410
```

```
##          Pos Pred Value : 0.62609
##          Neg Pred Value : 0.93016
##              Prevalence : 0.11700
##          Detection Rate : 0.05308
##    Detection Prevalence : 0.08478
##       Balanced Accuracy : 0.70889
##
##        'Positive' Class : yes
##

# Sensitivity is only about 42%; we can improve the model quite a bit since
logistic model has sensitivtiy around 75%


# Model 3: Random forest
library(randomForest)

## randomForest 4.6-12

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin

split_indices <- sample.split(bank$response, SplitRatio = 0.70)
train_rf <- bank[split_indices, ]
test_rf <- bank[!split_indices, ]
nrow(train_rf)/nrow(bank)

## [1] 0.6999845

nrow(test_rf)/nrow(bank)

## [1] 0.3000155

bank_rf <- randomForest(response ~., data = train_rf, proximity = F, do.trace
= T, mtry = 5)

## ntree      OOB      1      2
##     1:  12.27%  7.04% 53.87%
##     2:  12.53%  7.17% 53.72%
##     3:  12.46%  7.12% 52.98%
##     4:  12.30%  7.13% 51.77%
##     5:  11.97%  6.71% 51.98%
##     6:  12.00%  6.69% 52.32%
##     7:  11.67%  6.30% 52.49%
##     8:  11.41%  6.03% 52.11%
##     9:  11.21%  5.87% 51.64%
```

```
##     10:  11.11%  5.71% 51.97%
##     11:  10.98%  5.58% 51.85%
##     12:  10.87%  5.38% 52.32%
##     13:  10.67%  5.26% 51.50%
##     14:  10.69%  5.21% 52.07%
##     15:  10.39%  5.03% 50.87%
##     16:  10.31%  4.93% 50.89%
##     17:  10.28%  4.80% 51.65%
##     18:  10.22%  4.81% 51.04%
##     19:  10.14%  4.71% 51.13%
##     20:  10.14%  4.71% 51.13%
##     21:  10.03%  4.54% 51.49%
##     22:  10.01%  4.57% 51.11%
##     23:   9.95%  4.50% 51.13%
##     24:   9.87%  4.48% 50.62%
##     25:   9.86%  4.41% 50.97%
##     26:   9.82%  4.38% 50.92%
##     27:   9.78%  4.35% 50.78%
##     28:   9.81%  4.35% 51.00%
##     29:   9.75%  4.25% 51.30%
##     30:   9.72%  4.30% 50.62%
##     31:   9.60%  4.14% 50.81%
##     32:   9.69%  4.22% 50.97%
##     33:   9.66%  4.19% 50.92%
##     34:   9.63%  4.16% 50.95%
##     35:   9.68%  4.20% 51.00%
##     36:   9.63%  4.19% 50.70%
##     37:   9.50%  4.10% 50.24%
##     38:   9.53%  4.07% 50.81%
##     39:   9.57%  4.09% 50.95%
##     40:   9.55%  4.08% 50.86%
##     41:   9.52%  4.04% 50.89%
##     42:   9.50%  4.04% 50.65%
##     43:   9.56%  4.10% 50.76%
##     44:   9.47%  4.03% 50.57%
##     45:   9.52%  4.06% 50.73%
##     46:   9.55%  4.06% 51.00%
##     47:   9.56%  4.12% 50.68%
##     48:   9.53%  4.06% 50.84%
##     49:   9.50%  4.02% 50.86%
##     50:   9.54%  4.06% 50.92%
##     51:   9.53%  4.05% 50.84%
##     52:   9.56%  4.08% 50.95%
##     53:   9.49%  4.06% 50.51%
##     54:   9.46%  3.99% 50.73%
##     55:   9.46%  4.03% 50.46%
##     56:   9.45%  4.03% 50.41%
##     57:   9.49%  4.02% 50.78%
##     58:   9.47%  4.04% 50.51%
##     59:   9.48%  4.07% 50.32%
```

```
##    60:   9.48%  4.07% 50.38%
##    61:   9.45%  4.00% 50.54%
##    62:   9.47%  4.03% 50.57%
##    63:   9.48%  4.04% 50.51%
##    64:   9.42%  3.98% 50.49%
##    65:   9.48%  4.02% 50.70%
##    66:   9.43%  3.99% 50.49%
##    67:   9.44%  3.95% 50.86%
##    68:   9.45%  3.95% 50.95%
##    69:   9.40%  3.94% 50.65%
##    70:   9.46%  3.97% 50.89%
##    71:   9.47%  3.98% 50.95%
##    72:   9.47%  3.96% 51.05%
##    73:   9.46%  3.97% 50.95%
##    74:   9.45%  3.96% 50.86%
##    75:   9.43%  3.91% 51.08%
##    76:   9.45%  3.94% 51.03%
##    77:   9.44%  3.93% 51.05%
##    78:   9.45%  3.90% 51.38%
##    79:   9.45%  3.93% 51.13%
##    80:   9.43%  3.88% 51.30%
##    81:   9.46%  3.91% 51.35%
##    82:   9.47%  3.89% 51.59%
##    83:   9.50%  3.91% 51.70%
##    84:   9.51%  3.92% 51.67%
##    85:   9.44%  3.89% 51.32%
##    86:   9.44%  3.90% 51.24%
##    87:   9.37%  3.87% 50.86%
##    88:   9.38%  3.85% 51.19%
##    89:   9.38%  3.86% 51.08%
##    90:   9.41%  3.85% 51.35%
##    91:   9.38%  3.86% 51.11%
##    92:   9.40%  3.86% 51.27%
##    93:   9.40%  3.83% 51.49%
##    94:   9.41%  3.85% 51.32%
##    95:   9.35%  3.77% 51.49%
##    96:   9.40%  3.81% 51.57%
##    97:   9.37%  3.78% 51.59%
##    98:   9.33%  3.76% 51.35%
##    99:   9.34%  3.78% 51.27%
##   100:   9.31%  3.77% 51.16%
##   101:   9.32%  3.77% 51.22%
##   102:   9.33%  3.79% 51.16%
##   103:   9.34%  3.79% 51.27%
##   104:   9.30%  3.75% 51.16%
##   105:   9.32%  3.78% 51.11%
##   106:   9.32%  3.77% 51.22%
##   107:   9.33%  3.74% 51.51%
##   108:   9.33%  3.76% 51.32%
##   109:   9.31%  3.77% 51.08%
```

```
##   110:   9.30%  3.77% 51.05%
##   111:   9.27%  3.74% 51.08%
##   112:   9.34%  3.74% 51.59%
##   113:   9.30%  3.73% 51.38%
##   114:   9.33%  3.75% 51.40%
##   115:   9.26%  3.76% 50.81%
##   116:   9.26%  3.73% 50.97%
##   117:   9.30%  3.75% 51.19%
##   118:   9.26%  3.71% 51.08%
##   119:   9.27%  3.73% 51.13%
##   120:   9.27%  3.72% 51.19%
##   121:   9.27%  3.70% 51.32%
##   122:   9.26%  3.70% 51.19%
##   123:   9.31%  3.73% 51.38%
##   124:   9.25%  3.70% 51.13%
##   125:   9.26%  3.71% 51.13%
##   126:   9.23%  3.70% 51.00%
##   127:   9.25%  3.69% 51.19%
##   128:   9.25%  3.72% 51.00%
##   129:   9.27%  3.71% 51.22%
##   130:   9.26%  3.72% 51.05%
##   131:   9.26%  3.70% 51.19%
##   132:   9.25%  3.70% 51.11%
##   133:   9.20%  3.69% 50.76%
##   134:   9.20%  3.71% 50.65%
##   135:   9.20%  3.70% 50.65%
##   136:   9.23%  3.71% 50.84%
##   137:   9.24%  3.71% 51.03%
##   138:   9.24%  3.74% 50.73%
##   139:   9.23%  3.70% 50.92%
##   140:   9.23%  3.72% 50.78%
##   141:   9.23%  3.71% 50.92%
##   142:   9.29%  3.76% 51.05%
##   143:   9.26%  3.74% 50.97%
##   144:   9.28%  3.72% 51.22%
##   145:   9.30%  3.75% 51.24%
##   146:   9.29%  3.76% 51.03%
##   147:   9.24%  3.74% 50.76%
##   148:   9.25%  3.74% 50.84%
##   149:   9.22%  3.74% 50.62%
##   150:   9.17%  3.68% 50.57%
##   151:   9.19%  3.68% 50.76%
##   152:   9.19%  3.70% 50.65%
##   153:   9.17%  3.68% 50.59%
##   154:   9.21%  3.71% 50.70%
##   155:   9.21%  3.70% 50.84%
##   156:   9.18%  3.67% 50.78%
##   157:   9.15%  3.66% 50.59%
##   158:   9.17%  3.66% 50.73%
##   159:   9.20%  3.68% 50.89%
```

```
##   160:   9.17%  3.65% 50.78%
##   161:   9.17%  3.65% 50.84%
##   162:   9.16%  3.65% 50.70%
##   163:   9.20%  3.65% 51.03%
##   164:   9.19%  3.67% 50.84%
##   165:   9.18%  3.67% 50.73%
##   166:   9.15%  3.63% 50.86%
##   167:   9.18%  3.64% 50.97%
##   168:   9.12%  3.60% 50.76%
##   169:   9.17%  3.64% 50.92%
##   170:   9.17%  3.64% 50.89%
##   171:   9.14%  3.63% 50.78%
##   172:   9.17%  3.65% 50.84%
##   173:   9.17%  3.63% 50.95%
##   174:   9.17%  3.62% 51.05%
##   175:   9.19%  3.65% 50.95%
##   176:   9.18%  3.64% 51.00%
##   177:   9.17%  3.64% 50.92%
##   178:   9.19%  3.65% 50.97%
##   179:   9.19%  3.66% 50.92%
##   180:   9.15%  3.64% 50.70%
##   181:   9.16%  3.65% 50.81%
##   182:   9.20%  3.68% 50.89%
##   183:   9.21%  3.68% 50.97%
##   184:   9.19%  3.65% 51.03%
##   185:   9.19%  3.64% 51.05%
##   186:   9.17%  3.65% 50.86%
##   187:   9.17%  3.64% 50.92%
##   188:   9.19%  3.66% 50.89%
##   189:   9.15%  3.64% 50.81%
##   190:   9.21%  3.65% 51.22%
##   191:   9.17%  3.64% 50.95%
##   192:   9.19%  3.66% 50.95%
##   193:   9.19%  3.65% 51.03%
##   194:   9.17%  3.62% 51.11%
##   195:   9.21%  3.65% 51.19%
##   196:   9.21%  3.65% 51.22%
##   197:   9.21%  3.65% 51.19%
##   198:   9.24%  3.66% 51.35%
##   199:   9.21%  3.66% 51.13%
##   200:   9.20%  3.65% 51.08%
##   201:   9.21%  3.67% 51.05%
##   202:   9.20%  3.65% 51.13%
##   203:   9.19%  3.66% 50.92%
##   204:   9.21%  3.67% 51.08%
##   205:   9.20%  3.64% 51.16%
##   206:   9.21%  3.67% 51.03%
##   207:   9.22%  3.67% 51.08%
##   208:   9.16%  3.64% 50.84%
##   209:   9.17%  3.63% 51.03%
```

```
##   210:   9.18%  3.64% 51.00%
##   211:   9.14%  3.60% 50.92%
##   212:   9.18%  3.62% 51.11%
##   213:   9.19%  3.61% 51.27%
##   214:   9.15%  3.61% 51.00%
##   215:   9.16%  3.62% 51.03%
##   216:   9.19%  3.63% 51.13%
##   217:   9.16%  3.60% 51.11%
##   218:   9.15%  3.59% 51.16%
##   219:   9.15%  3.60% 51.08%
##   220:   9.14%  3.58% 51.16%
##   221:   9.15%  3.58% 51.19%
##   222:   9.14%  3.58% 51.05%
##   223:   9.14%  3.59% 50.97%
##   224:   9.13%  3.60% 50.92%
##   225:   9.09%  3.55% 50.95%
##   226:   9.10%  3.56% 50.95%
##   227:   9.12%  3.57% 50.97%
##   228:   9.15%  3.58% 51.24%
##   229:   9.11%  3.55% 51.05%
##   230:   9.11%  3.55% 51.05%
##   231:   9.11%  3.54% 51.11%
##   232:   9.12%  3.56% 51.08%
##   233:   9.12%  3.56% 51.13%
##   234:   9.11%  3.56% 51.05%
##   235:   9.11%  3.56% 50.95%
##   236:   9.12%  3.55% 51.11%
##   237:   9.11%  3.58% 50.86%
##   238:   9.10%  3.55% 51.00%
##   239:   9.15%  3.59% 51.11%
##   240:   9.12%  3.57% 51.00%
##   241:   9.14%  3.57% 51.13%
##   242:   9.14%  3.61% 50.95%
##   243:   9.15%  3.59% 51.08%
##   244:   9.15%  3.60% 51.08%
##   245:   9.16%  3.61% 51.05%
##   246:   9.17%  3.61% 51.11%
##   247:   9.18%  3.61% 51.24%
##   248:   9.16%  3.60% 51.11%
##   249:   9.17%  3.61% 51.13%
##   250:   9.17%  3.61% 51.16%
##   251:   9.17%  3.60% 51.19%
##   252:   9.18%  3.60% 51.27%
##   253:   9.20%  3.62% 51.30%
##   254:   9.17%  3.61% 51.13%
##   255:   9.15%  3.60% 51.08%
##   256:   9.19%  3.62% 51.22%
##   257:   9.17%  3.60% 51.22%
##   258:   9.16%  3.59% 51.22%
##   259:   9.17%  3.59% 51.27%
```

```
##    260:    9.17%  3.57% 51.43%
##    261:    9.17%  3.56% 51.49%
##    262:    9.14%  3.56% 51.27%
##    263:    9.13%  3.57% 51.13%
##    264:    9.15%  3.56% 51.38%
##    265:    9.13%  3.56% 51.13%
##    266:    9.10%  3.56% 50.89%
##    267:    9.11%  3.56% 50.97%
##    268:    9.12%  3.56% 51.11%
##    269:    9.10%  3.56% 50.95%
##    270:    9.09%  3.57% 50.78%
##    271:    9.09%  3.57% 50.78%
##    272:    9.09%  3.57% 50.76%
##    273:    9.10%  3.59% 50.70%
##    274:    9.12%  3.59% 50.81%
##    275:    9.10%  3.57% 50.81%
##    276:    9.12%  3.57% 50.97%
##    277:    9.11%  3.57% 50.92%
##    278:    9.12%  3.56% 51.05%
##    279:    9.11%  3.56% 50.97%
##    280:    9.08%  3.55% 50.84%
##    281:    9.12%  3.57% 51.00%
##    282:    9.13%  3.57% 51.11%
##    283:    9.10%  3.57% 50.86%
##    284:    9.11%  3.56% 51.05%
##    285:    9.11%  3.56% 51.03%
##    286:    9.12%  3.58% 50.95%
##    287:    9.10%  3.58% 50.81%
##    288:    9.12%  3.60% 50.81%
##    289:    9.13%  3.60% 50.89%
##    290:    9.12%  3.58% 50.89%
##    291:    9.11%  3.57% 50.89%
##    292:    9.10%  3.57% 50.89%
##    293:    9.08%  3.56% 50.76%
##    294:    9.08%  3.56% 50.76%
##    295:    9.08%  3.55% 50.81%
##    296:    9.10%  3.56% 50.89%
##    297:    9.09%  3.57% 50.76%
##    298:    9.10%  3.57% 50.84%
##    299:    9.10%  3.56% 50.92%
##    300:    9.06%  3.54% 50.76%
##    301:    9.08%  3.56% 50.76%
##    302:    9.07%  3.55% 50.73%
##    303:    9.08%  3.56% 50.70%
##    304:    9.06%  3.55% 50.65%
##    305:    9.07%  3.56% 50.70%
##    306:    9.06%  3.54% 50.78%
##    307:    9.02%  3.54% 50.46%
##    308:    9.04%  3.55% 50.43%
##    309:    9.04%  3.55% 50.51%
```

```
##   310:   9.04%  3.55% 50.49%
##   311:   9.05%  3.55% 50.54%
##   312:   9.03%  3.55% 50.46%
##   313:   9.02%  3.54% 50.38%
##   314:   9.00%  3.52% 50.38%
##   315:   9.05%  3.54% 50.65%
##   316:   9.06%  3.54% 50.70%
##   317:   9.06%  3.55% 50.62%
##   318:   9.03%  3.52% 50.62%
##   319:   9.04%  3.52% 50.68%
##   320:   9.07%  3.54% 50.78%
##   321:   9.03%  3.52% 50.62%
##   322:   9.03%  3.52% 50.59%
##   323:   9.04%  3.52% 50.70%
##   324:   9.04%  3.55% 50.46%
##   325:   9.02%  3.55% 50.38%
##   326:   9.02%  3.54% 50.38%
##   327:   9.03%  3.54% 50.51%
##   328:   9.05%  3.55% 50.59%
##   329:   9.02%  3.54% 50.35%
##   330:   9.02%  3.53% 50.43%
##   331:   9.04%  3.55% 50.46%
##   332:   9.02%  3.55% 50.32%
##   333:   9.01%  3.53% 50.35%
##   334:   9.01%  3.52% 50.41%
##   335:   9.04%  3.56% 50.41%
##   336:   9.03%  3.55% 50.41%
##   337:   9.02%  3.55% 50.30%
##   338:   9.03%  3.55% 50.46%
##   339:   9.04%  3.55% 50.51%
##   340:   9.04%  3.56% 50.41%
##   341:   9.03%  3.55% 50.41%
##   342:   9.02%  3.54% 50.32%
##   343:   9.03%  3.54% 50.49%
##   344:   9.01%  3.52% 50.38%
##   345:   9.00%  3.52% 50.35%
##   346:   9.00%  3.52% 50.35%
##   347:   9.01%  3.52% 50.43%
##   348:   9.02%  3.54% 50.43%
##   349:   9.01%  3.52% 50.43%
##   350:   9.04%  3.54% 50.51%
##   351:   9.02%  3.55% 50.27%
##   352:   9.02%  3.51% 50.57%
##   353:   9.00%  3.51% 50.43%
##   354:   9.00%  3.50% 50.49%
##   355:   9.03%  3.51% 50.73%
##   356:   9.02%  3.52% 50.51%
##   357:   9.03%  3.53% 50.57%
##   358:   9.03%  3.54% 50.49%
##   359:   9.03%  3.53% 50.54%
```

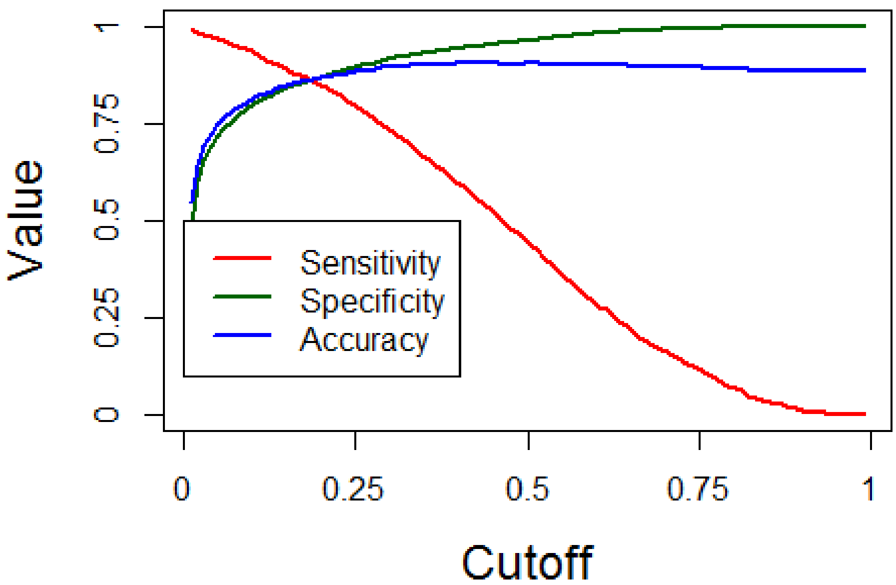
```
##    360:   9.02%  3.52% 50.54%
##    361:   9.05%  3.52% 50.84%
##    362:   9.03%  3.51% 50.73%
##    363:   9.05%  3.53% 50.76%
##    364:   9.07%  3.53% 50.86%
##    365:   9.05%  3.52% 50.76%
##    366:   9.02%  3.51% 50.65%
##    367:   9.03%  3.51% 50.70%
##    368:   9.02%  3.51% 50.68%
##    369:   9.05%  3.53% 50.76%
##    370:   9.05%  3.52% 50.81%
##    371:   9.04%  3.51% 50.73%
##    372:   9.01%  3.51% 50.49%
##    373:   9.02%  3.51% 50.62%
##    374:   9.01%  3.49% 50.62%
##    375:   9.05%  3.52% 50.73%
##    376:   9.03%  3.50% 50.78%
##    377:   9.03%  3.51% 50.70%
##    378:   9.05%  3.51% 50.86%
##    379:   9.01%  3.50% 50.62%
##    380:   9.04%  3.51% 50.78%
##    381:   9.00%  3.49% 50.59%
##    382:   9.03%  3.51% 50.68%
##    383:   9.01%  3.48% 50.76%
##    384:   9.02%  3.51% 50.65%
##    385:   9.04%  3.51% 50.73%
##    386:   9.04%  3.52% 50.73%
##    387:   9.03%  3.50% 50.73%
##    388:   9.04%  3.51% 50.84%
##    389:   9.04%  3.51% 50.78%
##    390:   9.02%  3.49% 50.73%
##    391:   9.04%  3.51% 50.81%
##    392:   9.03%  3.52% 50.62%
##    393:   9.02%  3.51% 50.62%
##    394:   9.03%  3.53% 50.57%
##    395:   9.02%  3.52% 50.57%
##    396:   9.02%  3.53% 50.46%
##    397:   9.01%  3.51% 50.57%
##    398:   9.03%  3.53% 50.51%
##    399:   9.03%  3.54% 50.54%
##    400:   9.01%  3.51% 50.57%
##    401:   9.02%  3.52% 50.51%
##    402:   9.00%  3.50% 50.49%
##    403:   9.02%  3.52% 50.51%
##    404:   9.02%  3.54% 50.43%
##    405:   9.01%  3.51% 50.49%
##    406:   9.01%  3.52% 50.49%
##    407:   9.00%  3.52% 50.30%
##    408:   9.03%  3.53% 50.51%
##    409:   9.03%  3.54% 50.51%
```

```
##   410:   9.04%  3.54% 50.59%
##   411:   9.04%  3.54% 50.62%
##   412:   9.02%  3.52% 50.54%
##   413:   9.02%  3.52% 50.49%
##   414:   9.05%  3.54% 50.59%
##   415:   9.02%  3.54% 50.41%
##   416:   9.05%  3.55% 50.54%
##   417:   9.05%  3.56% 50.49%
##   418:   9.02%  3.54% 50.35%
##   419:   9.03%  3.54% 50.46%
##   420:   9.03%  3.55% 50.38%
##   421:   9.03%  3.54% 50.43%
##   422:   8.99%  3.52% 50.30%
##   423:   9.03%  3.55% 50.43%
##   424:   9.01%  3.55% 50.27%
##   425:   9.02%  3.54% 50.38%
##   426:   8.99%  3.52% 50.27%
##   427:   9.00%  3.52% 50.38%
##   428:   8.99%  3.52% 50.30%
##   429:   8.97%  3.51% 50.19%
##   430:   9.00%  3.52% 50.32%
##   431:   8.99%  3.50% 50.43%
##   432:   8.97%  3.50% 50.27%
##   433:   8.98%  3.50% 50.30%
##   434:   8.99%  3.51% 50.30%
##   435:   9.00%  3.54% 50.22%
##   436:   9.02%  3.54% 50.38%
##   437:   8.99%  3.51% 50.38%
##   438:   9.01%  3.54% 50.30%
##   439:   9.00%  3.52% 50.35%
##   440:   9.00%  3.52% 50.35%
##   441:   9.03%  3.54% 50.51%
##   442:   9.03%  3.55% 50.41%
##   443:   9.01%  3.54% 50.24%
##   444:   9.00%  3.52% 50.32%
##   445:   9.02%  3.54% 50.41%
##   446:   9.03%  3.53% 50.54%
##   447:   9.02%  3.53% 50.49%
##   448:   9.02%  3.53% 50.43%
##   449:   9.04%  3.54% 50.57%
##   450:   9.05%  3.53% 50.70%
##   451:   9.06%  3.55% 50.68%
##   452:   9.05%  3.54% 50.62%
##   453:   9.01%  3.52% 50.41%
##   454:   9.02%  3.51% 50.65%
##   455:   9.04%  3.52% 50.73%
##   456:   9.04%  3.52% 50.73%
##   457:   9.02%  3.52% 50.54%
##   458:   9.02%  3.52% 50.57%
##   459:   9.02%  3.52% 50.59%
```

```
##   460:   9.02%  3.51% 50.54%
##   461:   9.03%  3.52% 50.62%
##   462:   9.04%  3.54% 50.54%
##   463:   9.05%  3.54% 50.65%
##   464:   9.03%  3.53% 50.57%
##   465:   9.05%  3.54% 50.62%
##   466:   9.05%  3.53% 50.70%
##   467:   9.05%  3.54% 50.65%
##   468:   9.04%  3.52% 50.68%
##   469:   9.06%  3.54% 50.76%
##   470:   9.06%  3.54% 50.73%
##   471:   9.04%  3.53% 50.62%
##   472:   9.05%  3.54% 50.65%
##   473:   9.06%  3.55% 50.65%
##   474:   9.04%  3.53% 50.62%
##   475:   9.05%  3.54% 50.65%
##   476:   9.06%  3.54% 50.68%
##   477:   9.04%  3.53% 50.65%
##   478:   9.02%  3.51% 50.59%
##   479:   9.05%  3.53% 50.70%
##   480:   9.03%  3.52% 50.62%
##   481:   9.08%  3.56% 50.73%
##   482:   9.02%  3.52% 50.59%
##   483:   9.04%  3.53% 50.65%
##   484:   9.05%  3.54% 50.68%
##   485:   9.03%  3.55% 50.46%
##   486:   9.05%  3.55% 50.51%
##   487:   9.06%  3.55% 50.65%
##   488:   9.03%  3.53% 50.54%
##   489:   9.04%  3.54% 50.54%
##   490:   9.03%  3.54% 50.54%
##   491:   9.04%  3.55% 50.49%
##   492:   9.03%  3.55% 50.41%
##   493:   9.03%  3.54% 50.51%
##   494:   9.03%  3.54% 50.51%
##   495:   9.04%  3.54% 50.57%
##   496:   9.04%  3.54% 50.57%
##   497:   9.07%  3.56% 50.62%
##   498:   9.06%  3.56% 50.57%
##   499:   9.06%  3.56% 50.54%
##   500:   9.06%  3.56% 50.62%
```

```r
rf_pred <- predict(bank_rf, test_rf[, -19], type = "prob")


perform_fn_rf <- function(cutoff)
  {
  predicted_response <- factor(ifelse(rf_pred[, 2] >= cutoff, "yes", "no"))
  conf <- confusionMatrix(predicted_response, test_rf$response, positive =
"yes")
```

```
  acc <- conf$overall[1]
  sens <- conf$byClass[1]
  spec <- conf$byClass[2]
  out <- t(as.matrix(c(sens, spec, acc)))
  colnames(out) <- c("sensitivity", "specificity", "accuracy")
  return(out)
}

# creating cutoff values from 0.01 to 0.99 for plotting and initialising a
matrix of size 1000x4
s = seq(.01,.99,length=100)
OUT = matrix(0,100,3)

# calculate the sens, spec and acc for different cutoff values
for(i in 1:100)
  {
  OUT[i,] = perform_fn_rf(s[i])
}

## Warning in confusionMatrix.default(predicted_response, test_rf$response, :
## Levels are not in the same order for reference and data. Refactoring data
## to match.

## Warning in confusionMatrix.default(predicted_response, test_rf$response, :
## Levels are not in the same order for reference and data. Refactoring data
## to match.

## Warning in confusionMatrix.default(predicted_response, test_rf$response, :
## Levels are not in the same order for reference and data. Refactoring data
## to match.

## Warning in confusionMatrix.default(predicted_response, test_rf$response, :
## Levels are not in the same order for reference and data. Refactoring data
## to match.

## Warning in confusionMatrix.default(predicted_response, test_rf$response, :
## Levels are not in the same order for reference and data. Refactoring data
## to match.

# plotting cutoffs
plot(s,
OUT[,1],xlab="Cutoff",ylab="Value",cex.lab=1.5,cex.axis=1.5,ylim=c(0,1),type=
"l",lwd=2,axes=FALSE,col=2)
axis(1,seq(0,1,length=5),seq(0,1,length=5),cex.lab=1.5)
axis(2,seq(0,1,length=5),seq(0,1,length=5),cex.lab=1.5)
lines(s,OUT[,2],col="darkgreen",lwd=2)
lines(s,OUT[,3],col=4,lwd=2)
box()
legend(0,.50,col=c(2,"darkgreen",4,"darkred"),lwd=c(2,2,2,2),c("Sensitivity",
"Specificity","Accuracy"))
```




```r
# the plot shows that cutoff value of around 22% optimises sensitivity and accuracy
predicted_response_22 <- factor(ifelse(rf_pred[, 2] >= 0.22, "yes", "no"))
confusionMatrix(predicted_response_22, test_rf[, 19], positive = "yes")

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    no   yes
##        no  10548   271
##        yes  1429  1316
##
##                Accuracy : 0.8747
##                  95% CI : (0.869, 0.8802)
##     No Information Rate : 0.883
##     P-Value [Acc > NIR] : 0.9987
##
##                   Kappa : 0.5393
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.82924
##             Specificity : 0.88069
##          Pos Pred Value : 0.47942
##          Neg Pred Value : 0.97495
##              Prevalence : 0.11700
##          Detection Rate : 0.09702
##    Detection Prevalence : 0.20237
```

```
##       Balanced Accuracy : 0.85496
##
##        'Positive' Class : yes
##

# Final RF important variables
importance <- bank_rf$importance
```

We'll choose the **Random Forest** as the final model. The top 5 important variables are duration, month, balance, age and day.

Model Deployment and Recommendations

Now that we have a model which predicts the probability of response, we can arrive at some interesting recommendations.

Our objective is to reduce the marketing cost and get almost the same number of customers as before.

The usual response rate is 11%, which means that if we telemarket to 10,000 people, 1100 will buy the product.

We can rather telemarket to only thoso whose **probability of purchase is high**. Let's look at the probabilities of purchase. Note that we will use only test data for this analysis.

```
test_rf$predicted_probs <- rf_pred[, 2]
test_rf$predicted_response <- predicted_response_22

test_predictions_rf <- test_rf[, c("response", "predicted_probs",
"predicted_response")]
head(test_predictions_rf)

##    response predicted_probs predicted_response
## 2        no           0.000                 no
## 3        no           0.004                 no
## 5        no           0.018                 no
## 6        no           0.002                 no
## 8        no           0.040                 no
## 12       no           0.000                 no

write.csv(test_predictions_rf, file = "response_predictions_rf.csv")
```

We have 13,564 observations in test data. Since we now have the probabilities of response, we can sort them and market only to those with high probabilities.

Reducing Customer Acquision Cost

Let's assume that telemarketing to each person costs INR 1. In the test data, we have 13,564 observations, so the total cost is INR 13564.
Among these, about 11.7% respond, so we get 1587 customers for INR 13564, or Rs 8.54 per customer.

```
summary(test_predictions_rf$response)

##    no   yes
## 11977  1587
```

Let's sort the observations in decreasing order of probability.

```
test_predictions_rf <-
test_predictions_rf[order(test_predictions_rf$predicted_probs, decreasing =
T), ]
head(test_predictions_rf)

##       response predicted_probs predicted_response
## 44159      yes           0.944                yes
## 44864      yes           0.942                yes
## 44745      yes           0.934                yes
## 45007      yes           0.932                yes
## 41473      yes           0.930                yes
## 40149       no           0.924                yes
```

Now if we market to, say, only 50% population (approx. 6800 people), then about 1576 will respond (see below). The response rate is improved to 23.1%, almost double of what you'll get by randomly marketing. The acquision cost comes down to Rs 4.31 per customer.

```
summary(test_predictions_rf$response[1:6800])

##   no  yes
## 5232 1568

1576/6800

## [1] 0.2317647

6800/1576

## [1] 4.314721
```

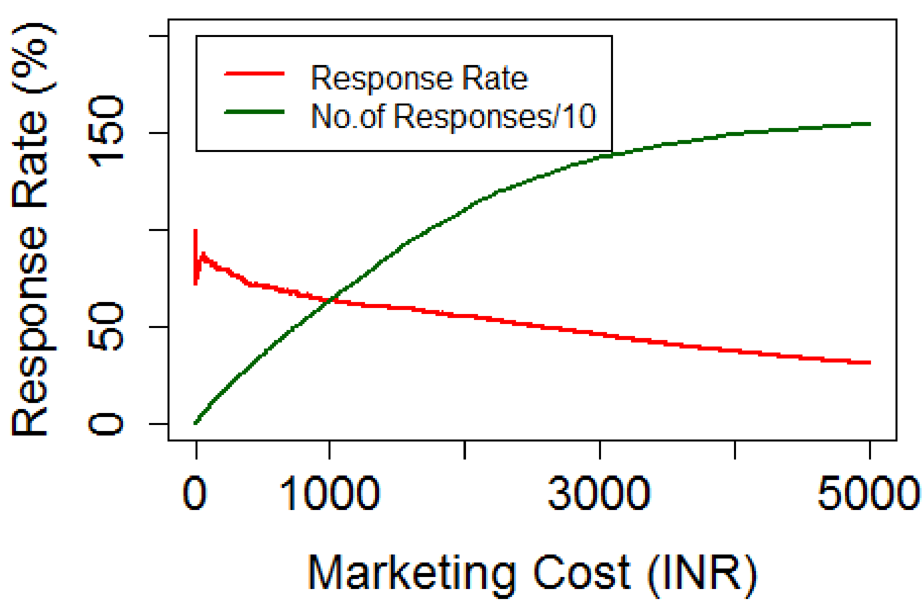
We can also visualise how the response rate varies with the marketing cost.

```
seq_prospects <- seq(1, 5000, by = 1)
cost_matrix <- matrix(0, length(seq_prospects), 3)
for (i in seq_prospects)
{
  cost_matrix[i, 1] = i
  response <- length(which(test_predictions_rf$response[1:i] == "yes"))
  cost_matrix[i, 2] = response/i
  cost_matrix[i, 3] = response
}
colnames(cost_matrix) <- c("number of prospects targeted (marketing cost)",
"response rate", "number of responses")
head(cost_matrix)
```

```
##       number of prospects targeted (marketing cost) response rate
## [1,]                                              1     1.0000000
## [2,]                                              2     1.0000000
## [3,]                                              3     1.0000000
## [4,]                                              4     1.0000000
## [5,]                                              5     1.0000000
## [6,]                                              6     0.8333333
##       number of responses
## [1,]                    1
## [2,]                    2
## [3,]                    3
## [4,]                    4
## [5,]                    5
## [6,]                    5
```

The cost_matrix stores the number of prospects targeted, the response rates and the number of responses. The marketing cost is same as number of people targted since we've assumed Re 1 per call.

```
plot(cost_matrix[, 1], cost_matrix[,2]*100,xlab="Marketing Cost
(INR)",ylab="Response Rate (%)",cex.lab=1.5,cex.axis=1.5,
ylim=c(0,200),type="l",lwd=2,axes=TRUE,col=2)

lines(seq_prospects, cost_matrix[, 3]/10, col="darkgreen",lwd=2)
box()
legend(0, 200,col=c(2,"darkgreen"),lwd=c(2,2),c("Response Rate","No.of
Responses/10"))
```



The plot shows how the number of responses and the response rate varies with marketing cost (no. of prospects targeted).

You can see that for INR 3000, almost 1379 prospects are expected to respond. Earlier, about 1587 would respond at a cost of Rs 13500.

```
cost_matrix[3000:3010, ]

##       number of prospects targeted (marketing cost) response rate
##  [1,]                                          3000     0.4566667
##  [2,]                                          3001     0.4565145
##  [3,]                                          3002     0.4563624
##  [4,]                                          3003     0.4562105
##  [5,]                                          3004     0.4560586
##  [6,]                                          3005     0.4559068
##  [7,]                                          3006     0.4557552
##  [8,]                                          3007     0.4556036
##  [9,]                                          3008     0.4554521
## [10,]                                          3009     0.4553008
## [11,]                                          3010     0.4551495
##       number of responses
##  [1,]                1370
##  [2,]                1370
##  [3,]                1370
##  [4,]                1370
##  [5,]                1370
##  [6,]                1370
##  [7,]                1370
##  [8,]                1370
##  [9,]                1370
## [10,]                1370
## [11,]                1370

1379/1587

## [1] 0.8689351

3000/13500

## [1] 0.2222222
```

Thus, we can acquire **about 86% of the customers for only about 22% of the marketing cost.**