Predicting Bank Marketing Campaign Outcome

Vivek Krishnan

16 March 2020

Introduction

Organizations such as banks spend considerable time, effort and money on marketing campaigns to increase their customer base or to increase the sign-up for their products. There are two approaches that financial institutions take to launch their campaigns - one is to launch a mass campaign targeting the general public, and the second is to launch a directed campaign targeting specific customers.

While mass campaigns may have been effective in the past, they are increasingly losing appeal due to the low rate of success. Direct marketing, **if targeted at the right audience**, can be very effective and efficient.

However, that's the big IF - are banks targeting the right set of contacts and customers?

That's the question that this study seeks to answer. Will a bank be able to predict if a campaign directed at a given set of customers will have a successful outcome or not?

This study is based on a dataset related to directed marketing campaigns (phone calls) for term deposits provided by a Portuguese banking institution.

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

Dataset available at the UCI Machine Learning Repository https://archive.ics.uci.edu/ml/datasets/Bank+Marketing.

Dataset Description

The dataset is a csv file containing data used for a directed marketing campaign for term deposits. The dataset consists of the following fields:

- Bank Client Data:
  - age (numeric)
  - job : type of job (categorical: 'admin.','blue-collar', 'entrepreneur', 'housemaid', 'management', 'retired', 'self-employed', 'services', 'student', 'technician', 'unemployed', 'unknown')
  - marital : marital status (categorical: 'divorced','married','single','unknown'; note: 'divorced' means divorced or widowed)
  - education (categorical: 'basic.4y', 'basic.6y', 'basic.9y', 'high.school', 'illiterate', 'professional.course', 'university.degree', 'unknown')
  - default: has credit in default? (categorical: 'no','yes','unknown')
  - housing: has housing loan? (categorical: 'no','yes','unknown')
  - loan: has personal loan? (categorical: 'no','yes','unknown')

- Related with the last contact of the current campaign:
  - contact: previous contact communication type (categorical: 'cellular','telephone')
  - month: previous contact month of year (categorical: 'jan', 'feb', 'mar', ..., 'nov', 'dec')
  - day_of_week: previous contact day of the week (categorical: 'mon','tue','wed','thu','fri')
  - duration: contact duration, in seconds (numeric)
- Other attributes:
  - campaign: number of contacts performed during this campaign and for this client (numeric, includes last contact)
  - pdays: number of days that passed by after the client was last contacted from a previous campaign (numeric; 999 means client was not previously contacted)
  - previous: number of contacts performed before this campaign and for this client (numeric)
  - poutcome: outcome of the previous marketing campaign (categorical: 'failure','nonexistent','success')
- Social and economic context attributes:
  - emp.var.rate: employment variation rate - quarterly indicator (numeric)
  - cons.price.idx: consumer price index - monthly indicator (numeric)
  - cons.conf.idx: consumer confidence index - monthly indicator (numeric)
  - euribor3m: euribor 3 month rate - daily indicator (numeric)
  - nr.employed: number of employees - quarterly indicator (numeric)
- Output/Target:
  - y - has the client subscribed to a term deposit? (binary: 'yes','no')

Prediction Objectives:

1. Predict if a potential client would subscribe or not even before contacting him/her - this will help save the bank's time, effort and money so that they only contact the customers who are likely to subscribe. For this objective, the duration field will not be considered in model training and prediction
2. Predict if a potential client who has been contacted will subscribe or not - this will also be useful since quite often, there is a big timegap between contacting a client and the client taking a decision. Being able to predict the client's decision will help the bank to improve their planning and decision-making activities. For this objective, the call duration will be included in the model training and prediction

The analysis will be structured as follows:

1. Method and Analysis for the study
2. Data Wrangling, data exploration and analysis of the dataset
3. Identifying influencers for building a prediction model
4. Building different prediction models for the first objective
5. Building different prediction models for the second objective (with duration)
6. Model aggregation for both objectives
7. Final model and evaluation against the validation set
8. Results
9. Conclusion and future work that can be done

Method and Analysis

The total dataset of around *41100* records was split into a dataset called *data* consisting of *37000* records and another dataset called *validation* consisting of *4100* records.

All of the data exploration, analysis, visualisation and building and testing of prediction models have been done using only the *37000* record *data* dataset.

The *validation* set was used only for final evaluation of the predictions made using the best model.

The *data* dataset itself was split into a training set called *train* and a test set called *test* with a **training:test split ratio of 80% : 20% (29600 training : 7400 test)**.

This *train* set consisting of around *29600* records was used for building prediction models. Each model was tested against the *7400* record *test_set* and iteratively tuned.

The best model based on the evaluation against the *test* set was used as the final model.

This final model was run against the *validation* set at the end to generate the final result.

The evaluation metric used for the prediction models was **balanced accuracy** as this dataset had a high class imbalance and *accuracy* would not have been a useful metric. The *balanced accuracy*, on the other hand balances the sensitivity and the specificity and ensures that a prediction model does not just predict the majority class all the time (which would boost *accuracy*, but not be a useful prediction model).

The *balanced accuracy* is calculated as the average of the *sensitivity* (the ability of a model to predict a positive outcome when the actual outcome is positive) and the *specificity* (the ability of a model to predict a negative outcome when the actual outcome is negative)

	Actually +	Actually -
Predicted +	True + (TP)	False + (FP)
Predicted -	False - (FN)	True - (TN)

$Sensitivity = TP/(TP + FN)$

$Specificity = TN/(TN + FP)$

$BalancedAccuracy = (Sensitivity + Specificity)/2$

The model chosen was the one that maximised the balanced accuracy over the *test* dataset consisting of 7400 records.

Different rule-based models as well as models based on different machine learning algorithms were tested. The best model from these different models was then chosen and the hyperparameters tuned to get the optimum model.

The top two models were then aggregated to get the final model to be used.

This final model was evaluated by running the model to generate predictions against the validation set of 4100 records and the prediction metrics documented.

The above was done for both the case where the duration field was not included in the prediction model and the case where it was included.

Data Wrangling, Data Exploration and Analysis of the dataset

Before attempting to build any prediction model, it is imperative to first explore the data and understand the underlying patterns. This section will focus on the data exploration that was done on the *data* dataset and analysis of the patterns in the data.

Let's download the dataset and split it into a data set for analysis and a validation set. All data exploration, training and testing and tuning will happen using the analysis set. The validation set will be used for the final evaluation of the model chosen.

```
# Bank Marketing dataset:
# https://archive.ics.uci.edu/ml/datasets/Bank+Marketing
# https://archive.ics.uci.edu/ml/machine-learning-databases/00222/bank-additional.zip
```

```
dl <- tempfile()
url<-"https://archive.ics.uci.edu/ml/machine-learning-databases/00222/bank-additional.zip"
download.file(url, dl)

data <- fread(text = gsub(";", "\t",
                          readLines(unzip(dl, "bank-additional/bank-additional-full.csv"))),
              stringsAsFactors = TRUE)
#All character fields are read as factors since they are all categorical.

#Let's first change the target y to 0/1 instead of no/yes
#This is so that we get the right sensitivity and specificity calculations later on.
data$y <- ifelse(data$y=='yes',1,0)

set.seed(1, sample.kind="Rounding")
idx <- createDataPartition(y=data$y, times=1, p = 0.1, list = FALSE)
validation <- data[idx,]
data <- data[-idx,]
```

There are **37069** rows in the analysis data set and **4119** in the validation set.

Let's have a quick look at the structure of the dataset.

```
## Classes 'data.table' and 'data.frame':   37069 obs. of  21 variables:
##  $ age           : int  56 57 37 40 56 45 59 41 24 25 ...
##  $ job           : Factor w/ 12 levels "admin.","blue-collar",..: 4 8 8 1 8 8 1 2 10 8 ...
##  $ marital       : Factor w/ 4 levels "divorced","married",..: 2 2 2 2 2 2 2 2 3 3 ...
##  $ education     : Factor w/ 8 levels "basic.4y","basic.6y",..: 1 4 4 2 4 3 6 8 6 4 ...
##  $ default       : Factor w/ 3 levels "no","unknown",..: 1 2 1 1 1 2 1 2 1 1 ...
##  $ housing       : Factor w/ 3 levels "no","unknown",..: 1 1 3 1 1 1 1 1 3 3 ...
##  $ loan          : Factor w/ 3 levels "no","unknown",..: 1 1 1 1 3 1 1 1 1 1 ...
##  $ contact       : Factor w/ 2 levels "cellular","telephone": 2 2 2 2 2 2 2 2 2 2 ...
##  $ month         : Factor w/ 10 levels "apr","aug","dec",..: 7 7 7 7 7 7 7 7 7 7 ...
##  $ day_of_week   : Factor w/ 5 levels "fri","mon","thu",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ duration      : int  261 149 226 151 307 198 139 217 380 50 ...
##  $ campaign      : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ pdays         : int  999 999 999 999 999 999 999 999 999 999 ...
##  $ previous      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ poutcome      : Factor w/ 3 levels "failure","nonexistent",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ emp.var.rate  : num  1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 ...
##  $ cons.price.idx: num  94 94 94 94 94 ...
##  $ cons.conf.idx : num  -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 ...
##  $ euribor3m     : num  4.86 4.86 4.86 4.86 4.86 ...
##  $ nr.employed   : num  5191 5191 5191 5191 5191 ...
##  $ y             : num  0 0 0 0 0 0 0 0 0 0 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

Let's look at the distribution of positive and negative cases.

```
data %>% group_by(y) %>% summarize(accept_count=n())
```

```
## # A tibble: 2 x 2
##       y accept_count
```

```
##   <dbl>         <int>
## 1     0         32873
## 2     1          4196
```

```
yes_proportion <- mean(data$y)
yes_proportion
```

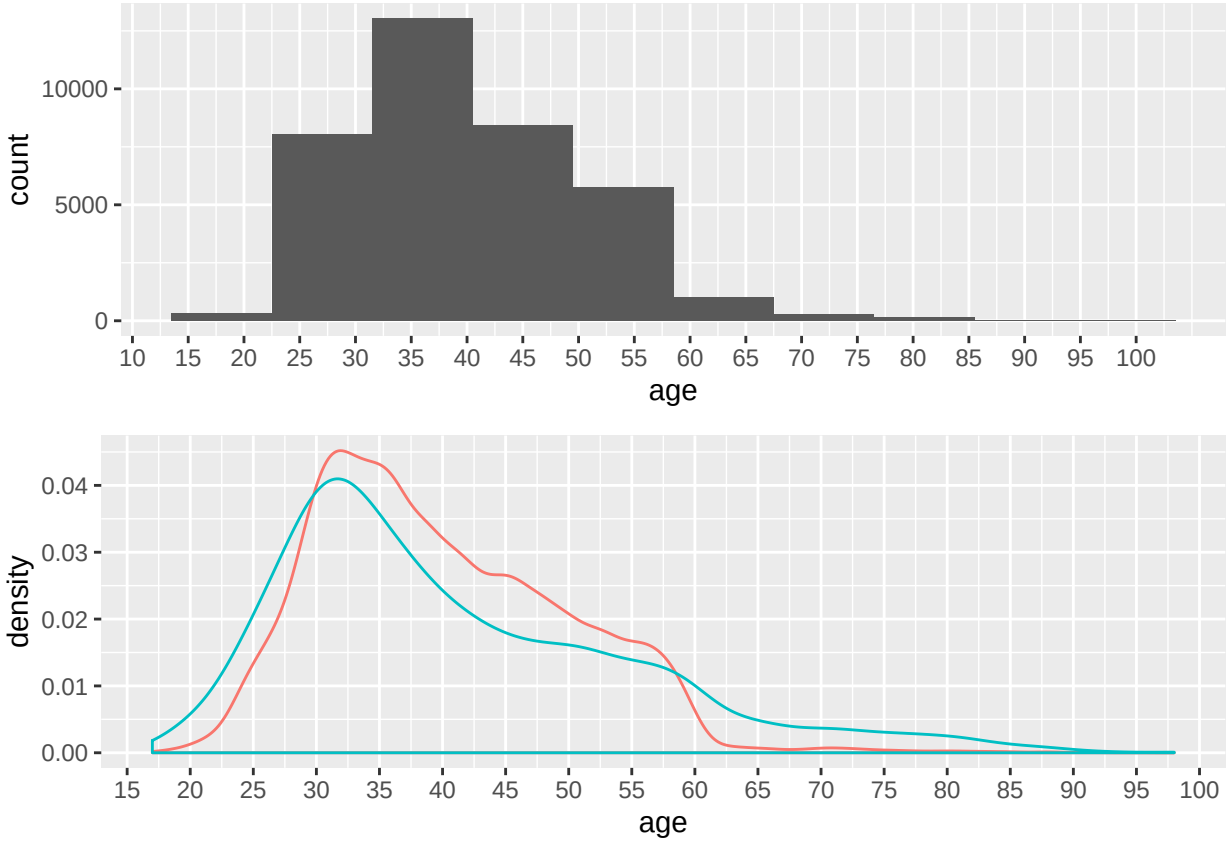
```
## [1] 0.1131943
```

We can see this is a very imbalanced dataset with only about 11% of positive records.

Let's start looking at the distribution of values and trends for each field:

Let's start with the distribution of the age field and how the target varies by age.

```
g1<-data %>% ggplot(aes(age)) + geom_histogram(bins=10) +
  scale_x_continuous(breaks=seq(10, 100, 5))
g2<-data %>% mutate(y=factor(y)) %>%
  ggplot(aes(x=age, color=y)) + geom_density(show.legend=FALSE) +
  scale_x_continuous(breaks=seq(10, 100, 5))
grid.arrange(g1, g2, nrow=2)
```

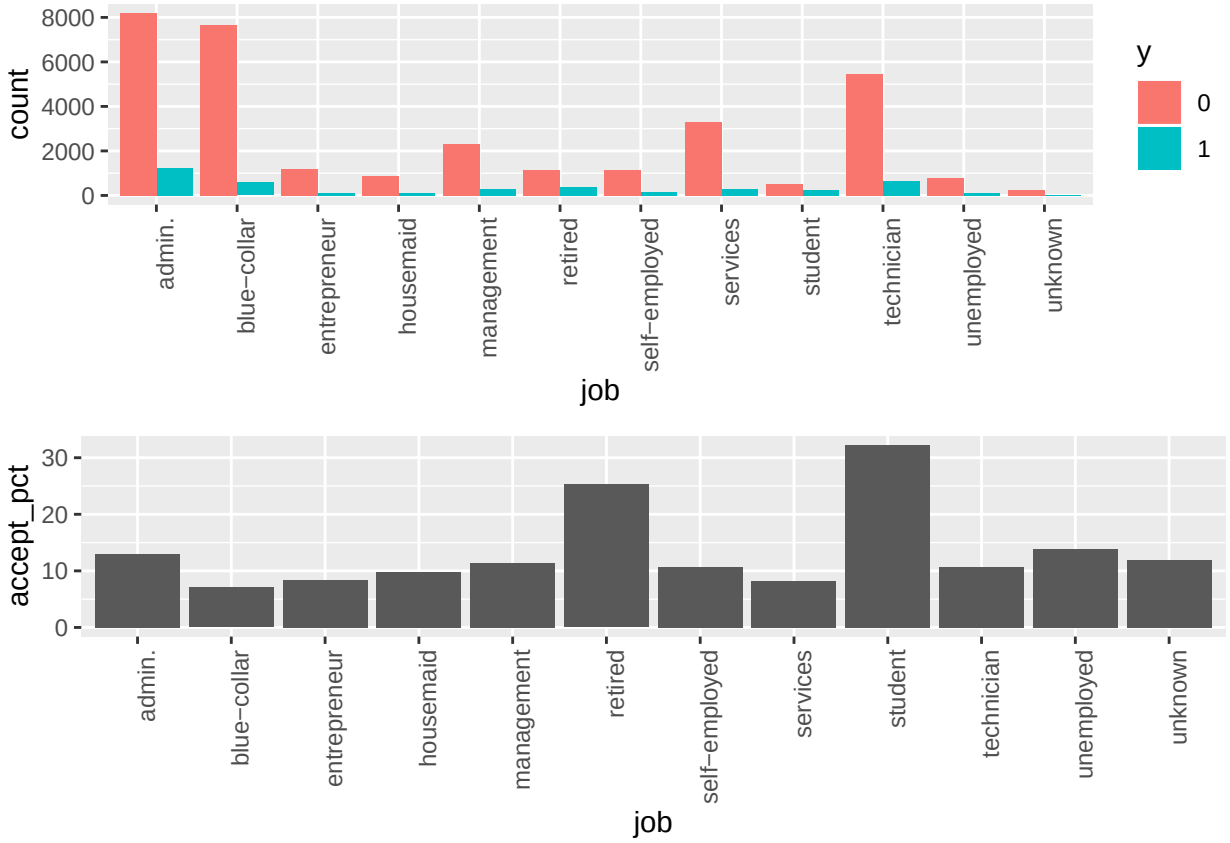


We see majority of the dataset between the age of 23 and 58 - this is expected as banks tend to approach customers who are in the working age range.

We don't see much of an influence on the target, except that people over 58 are more likely to subscribe and this is intuitive as older/retired people tend to prefer relatively safer investment avenues such as term deposits.

Let's look at job now.

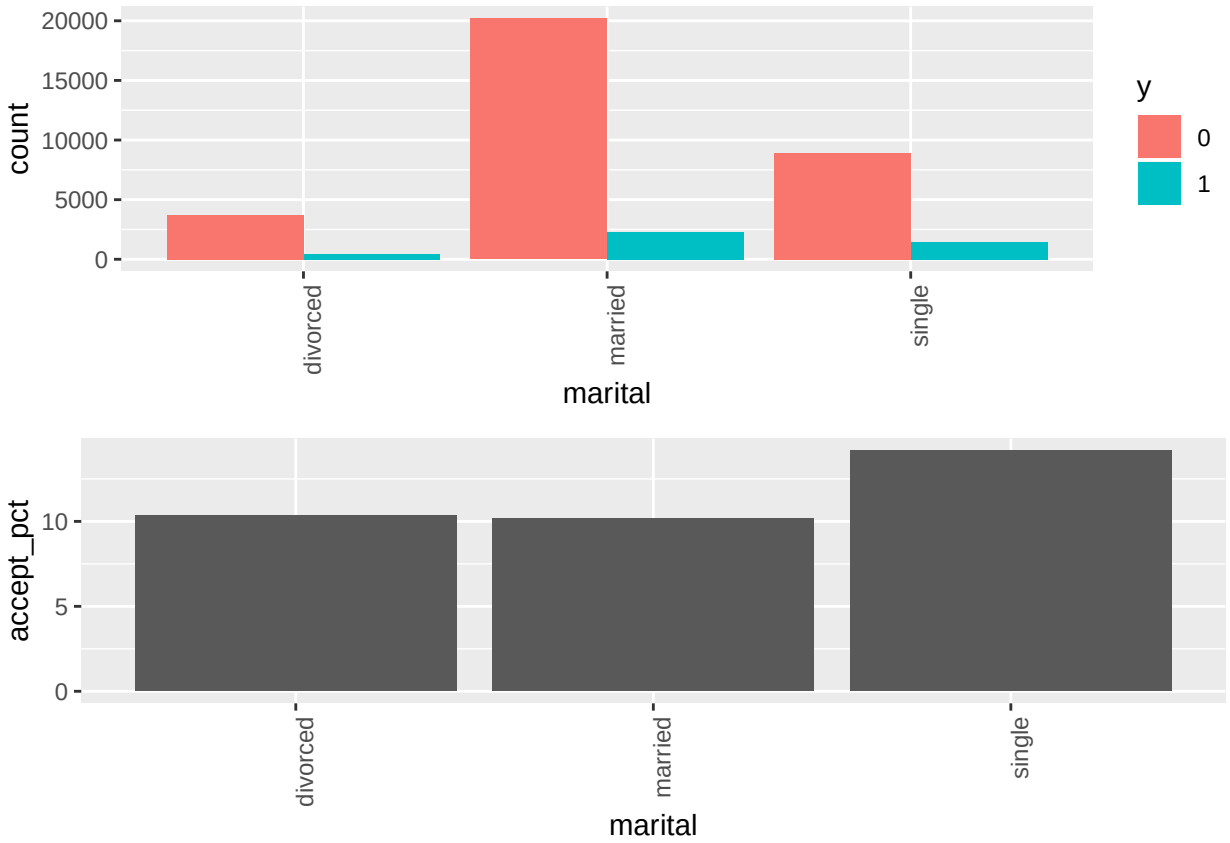
```
g1<-data %>% mutate(y=factor(y)) %>% ggplot(aes(x=job, fill=y)) +
  geom_bar(position=position_dodge()) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
g2<-data %>% group_by(job) %>% summarize(count=n(), accept_pct=100*mean(y==1)) %>%
  ggplot(aes(x=job, y=accept_pct)) + geom_bar(stat='identity') +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
grid.arrange(g1, g2, nrow=2)
```



We can see bulk of the people are admin or blue-collared or technician. We can see that the proportion of acceptances varies by job. Students and retired people are more likely to accept than others, however, since the number of retired/student records is relatively far lower, this field may not be a very good influencer.

Let's look at marital status now. There are very few records with unknown marital status, so those are filtered out.

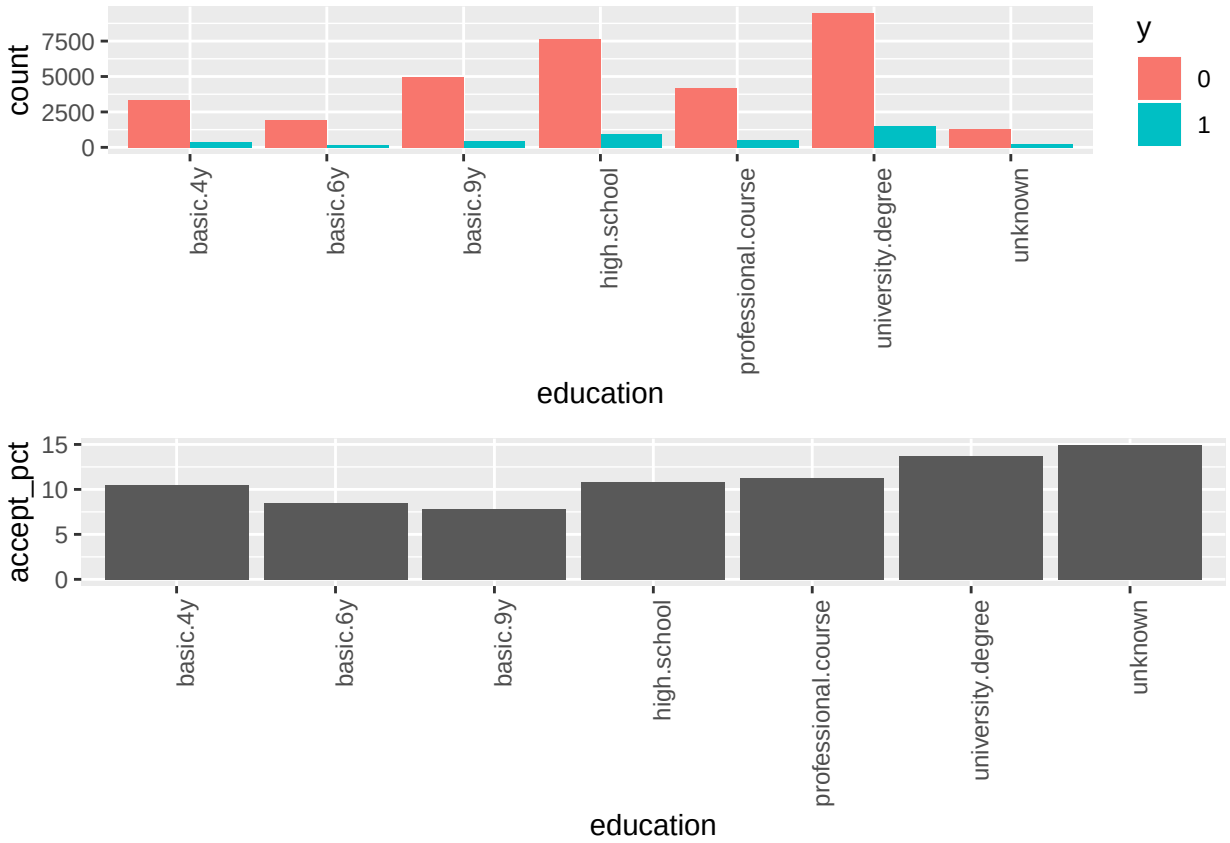
```
g1<-data %>% filter(marital!='unknown') %>%
  mutate(y=factor(y)) %>% ggplot(aes(x=marital, fill=y)) +
  geom_bar(position=position_dodge()) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
g2<-data %>% filter(marital!='unknown') %>% group_by(marital) %>%
  summarize(count=n(), accept_pct=100*mean(y==1)) %>%
  ggplot(aes(x=marital, y=accept_pct)) + geom_bar(stat='identity') +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
grid.arrange(g1, g2, nrow=2)
```



We don't see much variation here. Single people have a slightly higher percentage of acceptances. But this is minor and this field is probably not a good influencer or use in prediction.

Let's look at education now. There are very few illiterate cases. Let's filter them out and check.

```
g1<-data %>% mutate(y=factor(y)) %>% filter(education!='illiterate') %>%
  ggplot(aes(x=education, fill=y)) + geom_bar(position=position_dodge()) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
g2<-data %>% filter(education!='illiterate') %>% group_by(education) %>%
  summarize(count=n(), accept_pct=100*mean(y==1)) %>%
  ggplot(aes(x=education, y=accept_pct)) + geom_bar(stat='identity') +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
grid.arrange(g1, g2, nrow=2)
```



We see very slight variation with customers with university degree or unknown having a few percentage points more.

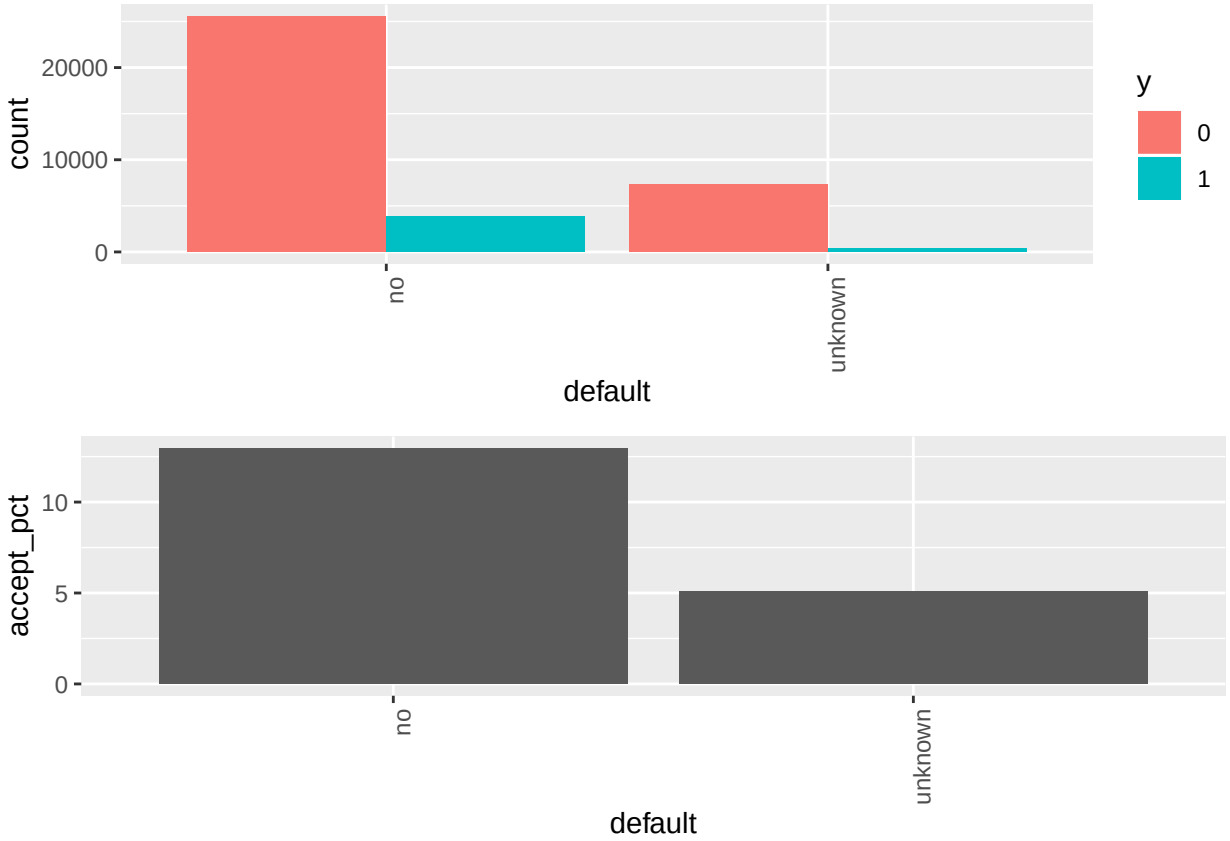
Let's look at default status now. There are hardly any default=yes cases. How many actually are there?

```
data %>% group_by(default) %>% summarize(count=n())
```

```
## # A tibble: 3 x 2
##   default count
##   <fct>   <int>
## 1 no      29355
## 2 unknown  7711
## 3 yes         3
```

Since there are only 3 yes cases, let's filter them out and compare only the "no" and the "unknown" cases.

```
g1<-data %>% mutate(y=factor(y)) %>% filter(default!='yes') %>%
  ggplot(aes(x=default, fill=y)) + geom_bar(position=position_dodge()) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
g2<-data %>% filter(default!='yes') %>% group_by(default) %>%
  summarize(count=n(), accept_pct=100*mean(y==1)) %>%
  ggplot(aes(x=default, y=accept_pct)) + geom_bar(stat='identity') +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
grid.arrange(g1, g2, nrow=2)
```



We see some variation here. People whose default status is unknown have a lower acceptance rate.

Let's look at housing loan status now.

```
data %>% group_by(housing) %>%
  summarize(count=n(), accept_pct=100*mean(y==1))
```

```
## # A tibble: 3 x 3
##   housing count accept_pct
##   <fct>   <int>      <dbl>
## 1 no      16733       10.9
## 2 unknown   884       10.4
## 3 yes     19452       11.7
```

We see almost no variation here. So perhaps, the acceptance does not depend much on housing loan status.

Let's look at personal loan status now.

```
data %>% group_by(loan) %>%
  summarize(count=n(), accept_pct=100*mean(y==1))
```

```
## # A tibble: 3 x 3
##   loan    count accept_pct
##   <fct>   <int>      <dbl>
## 1 no      30559       11.4
## 2 unknown   884       10.4
## 3 yes      5626       11.0
```

We see almost no variation here too. So perhaps, the subscription does not depend much on personal loan status.

Let's look at contact now.

```
data %>% group_by(contact) %>%
  summarize(count=n(), accept_pct=100*mean(y==1))
```

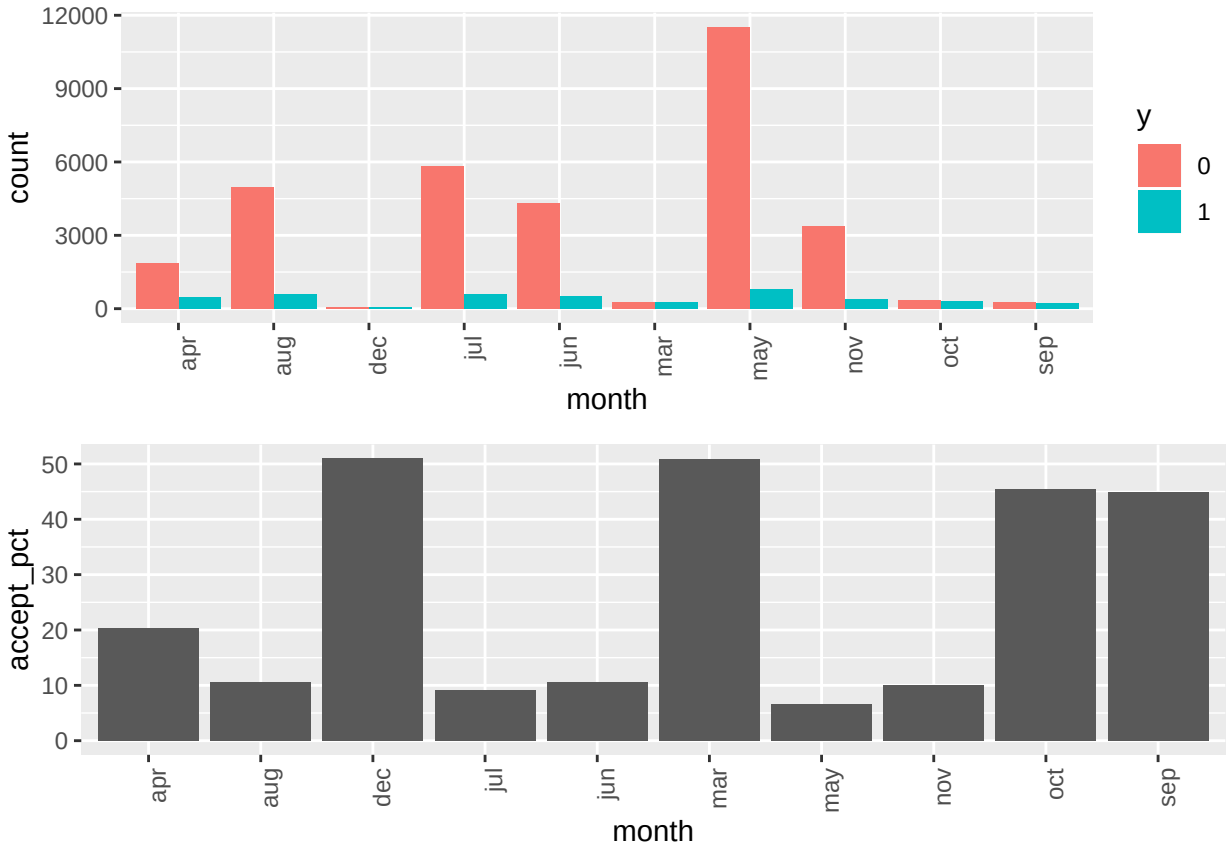
```
## # A tibble: 2 x 3
##   contact   count accept_pct
##   <fct>     <int>      <dbl>
## 1 cellular  23558      14.8
## 2 telephone 13511       5.31
```

We see a significant variation here. People contacted on their mobiles are more likely to subscribe than the ones contacted by telephone.

Let's look at last month contact now. The month column is changed to an ordered factor so that the x axis gets ordered by month correctly in the plots

We see most of the contacts in the past happened in May and the summer months. Very few or none in winter (Dec-Mar).

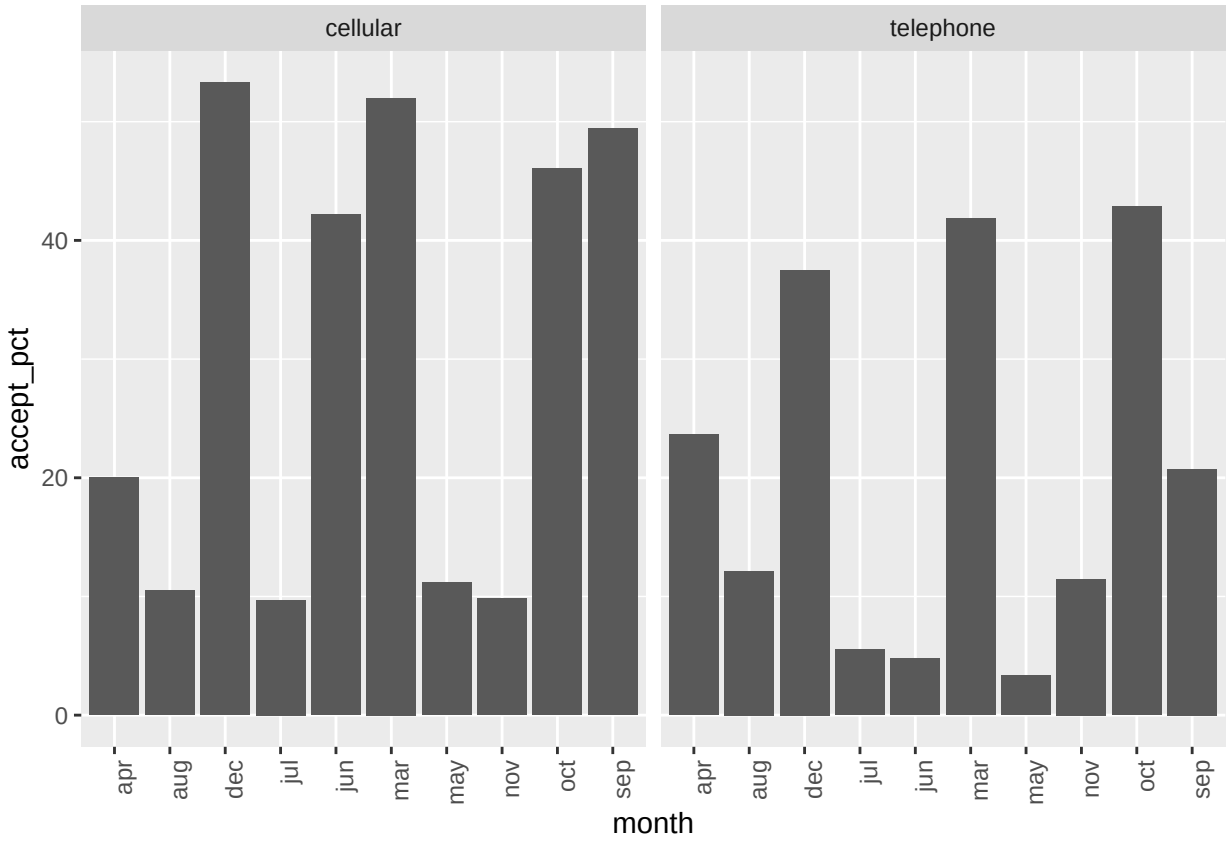
```
g1<-data %>% mutate(y=factor(y)) %>% ggplot(aes(x=month, fill=y)) +
  geom_bar(position=position_dodge()) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
g2<-data %>% group_by(month) %>%
  summarize(count=n(), accept_pct=100*mean(y==1)) %>%
  ggplot(aes(x=month, y=accept_pct)) + geom_bar(stat='identity') +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
grid.arrange(g1, g2, nrow=2)
```



We also see an interesting trend above. Customers last contacted in March/Sep/Oct/Dec seem to have a much higher subscription rate. It may not be a coincidence that March, Sep and Dec are all quarter ending months.

How about trend of acceptance by both mode and month of last contact?

```
data %>% group_by(contact, month) %>%
  summarize(count=n(), accept_pct=100*mean(y==1)) %>%
  ggplot(aes(x=month, y=accept_pct)) + geom_bar(stat='identity') +
  facet_wrap(~contact) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



That gives a pretty interesting result - for example, though the overall month-wise plot earlier showed a low % of around 10% in June, customers contacted in June by cellular have a much higher subscription % at 44%.

This then means that all the quarter ending months are good times to attempt the contact - March, June (cellular), Sep and Dec are all quarter ending months with a high subscription rate than other months.

Let's check the trend of subscription by day of week.

```
days <- c('sun', 'mon', 'tue', 'wed', 'thu', 'fri', 'sat')
data$day_of_week <- factor(data$day_of_week, levels=days)
data %>% group_by(day_of_week) %>%
  summarize(count=n(), accept_pct=100*mean(y==1))
```

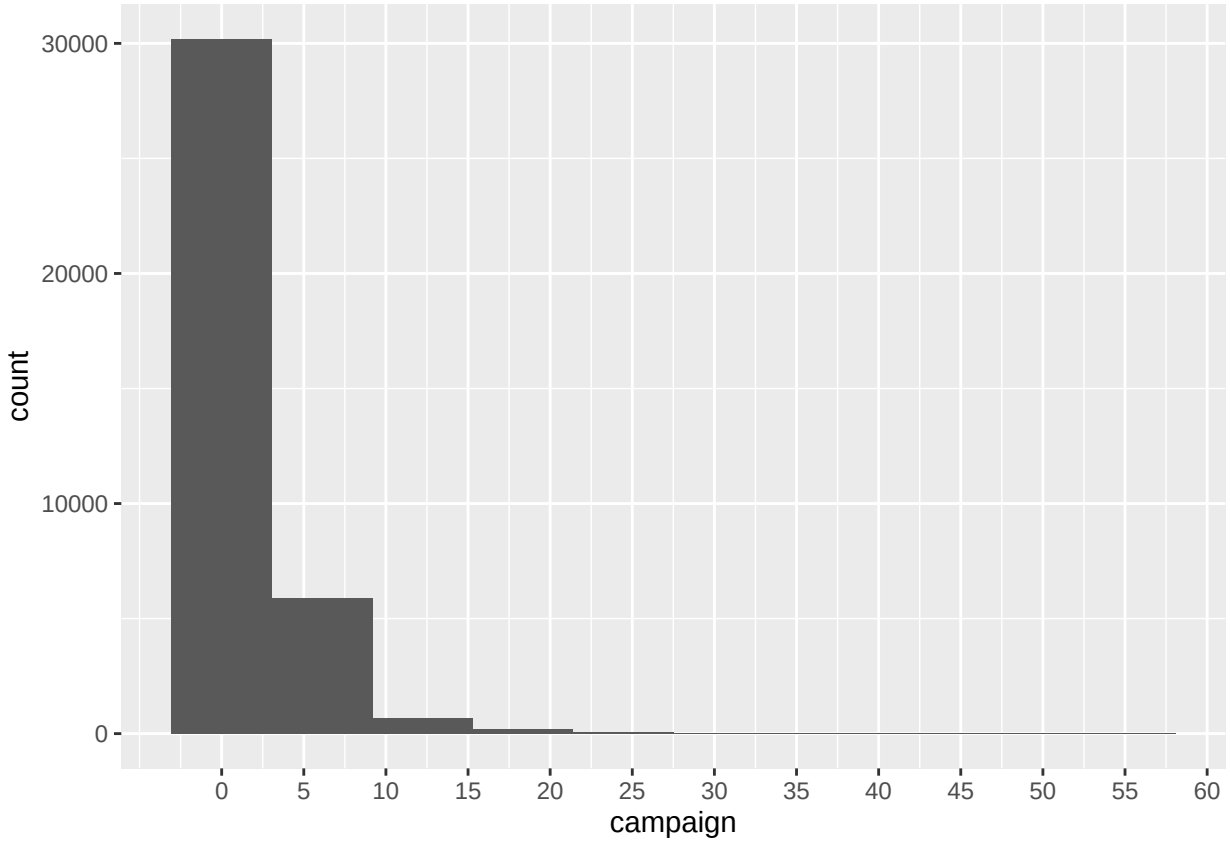
```
## # A tibble: 5 x 3
##   day_of_week count accept_pct
##   <fct>       <int>      <dbl>
## 1 mon          7690       10
## 2 tue          7271       12.0
## 3 wed          7304       11.6
## 4 thu          7740       12.2
## 5 fri          7064       10.8
```

There is not much of variation by day_of_week. The data seems to be evenly spread.

Let's look the trend by campaign now.

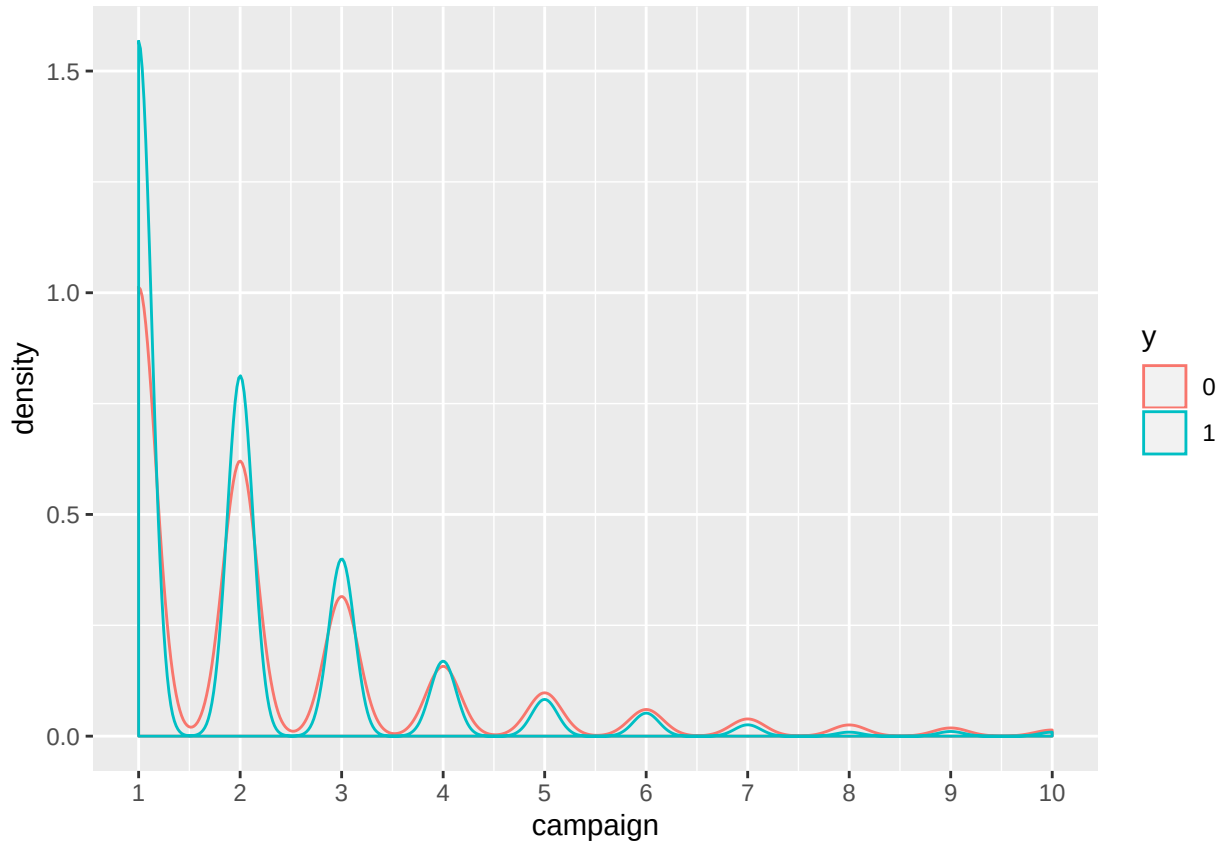
We see that the number of contacts performed in campaign ranges from 1 to 56.

```
data %>% ggplot(aes(campaign)) + geom_histogram(bins=10) +
  scale_x_continuous(breaks=seq(0, 60, 5))
```



We see majority of the dataset have campaign contact count falling in the range 1 to 10. So, we will focus on that range and check how the target vary?

```
data %>% mutate(y=factor(y)) %>% filter(campaign<=10) %>%
  ggplot(aes(x=campaign, color=y)) + geom_density() +
  scale_x_continuous(breaks=seq(0, 10, 1))
```



We see that the subscription acceptance is likely to happen for campaign contact count = 1. Again, this is intuitive since if a customer is looking to subscribe, the first contact itself is likely to result in a subscription.

Let's try pdays now. Looks like most of the data has pdays=999 which means they were never contacted before:

```
data %>% summarize(mean(pdays!=999))
```

```
##   mean(pdays != 999)
## 1         0.03663438
```

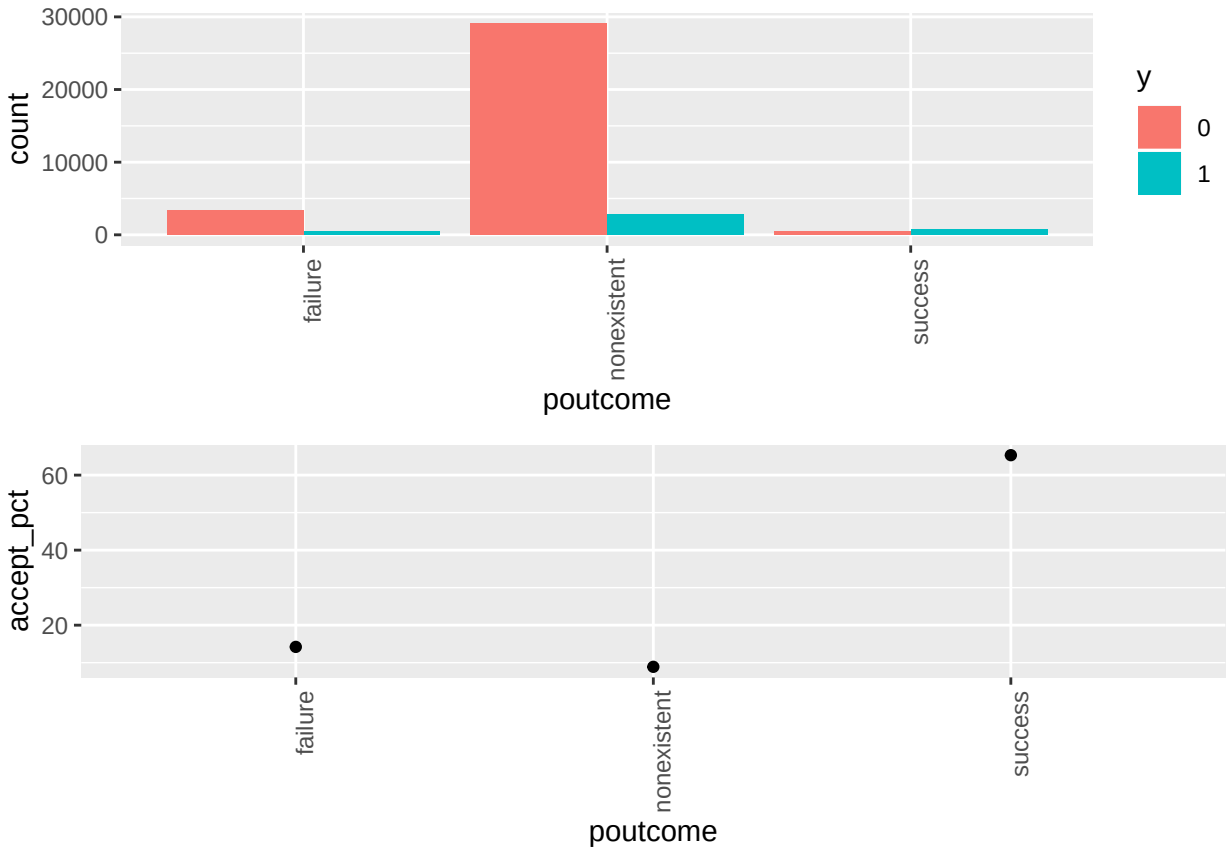
Only 3.6% of the data has pdays!=999. We can ignore this field as it will not give any meaningful analysis.

The same is likely for "previous" too since it should be non-zero only when pdays!=999 based on the dataset description.

So, we can ignore this field too.

Let's look at poutcome now.

```
g1<-data %>% mutate(y=factor(y)) %>% ggplot(aes(x=poutcome, fill=y)) +
  geom_bar(position=position_dodge()) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
g2<-data %>% group_by(poutcome) %>%
  summarize(count=n(), accept_pct=100*mean(y==1)) %>%
  ggplot(aes(x=poutcome, y=accept_pct)) + geom_point() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
grid.arrange(g1, g2, nrow=2)
```



We can clearly see that people who had previously accepted are likely to accept this time too. So, this is probably a good influencer/predictor.

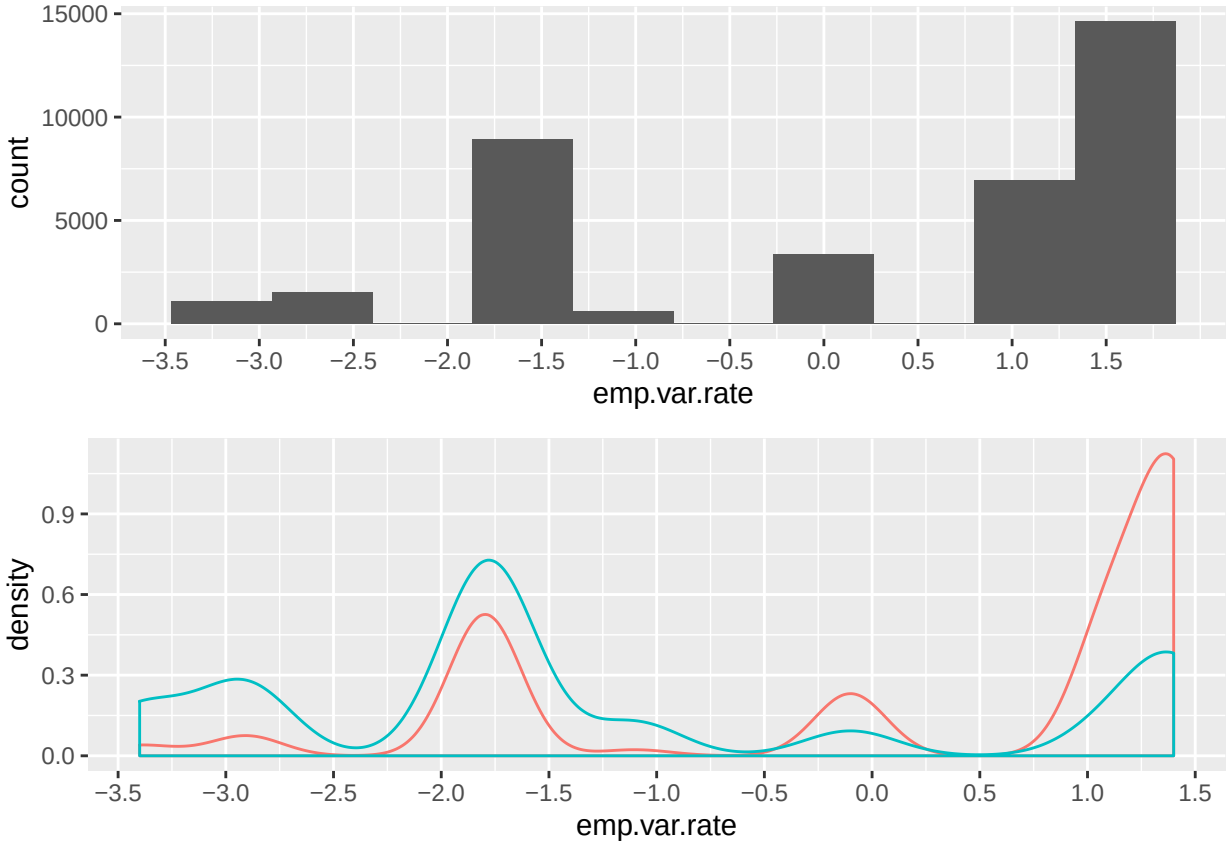
Let's now look at the social and economic context fields.

Let's look at the employment variation rate and how the target varies.

```
unique(data$emp.var.rate)
```

```
##  [1]  1.1  1.4 -0.1 -0.2 -1.8 -2.9 -3.4 -3.0 -1.7 -1.1
```

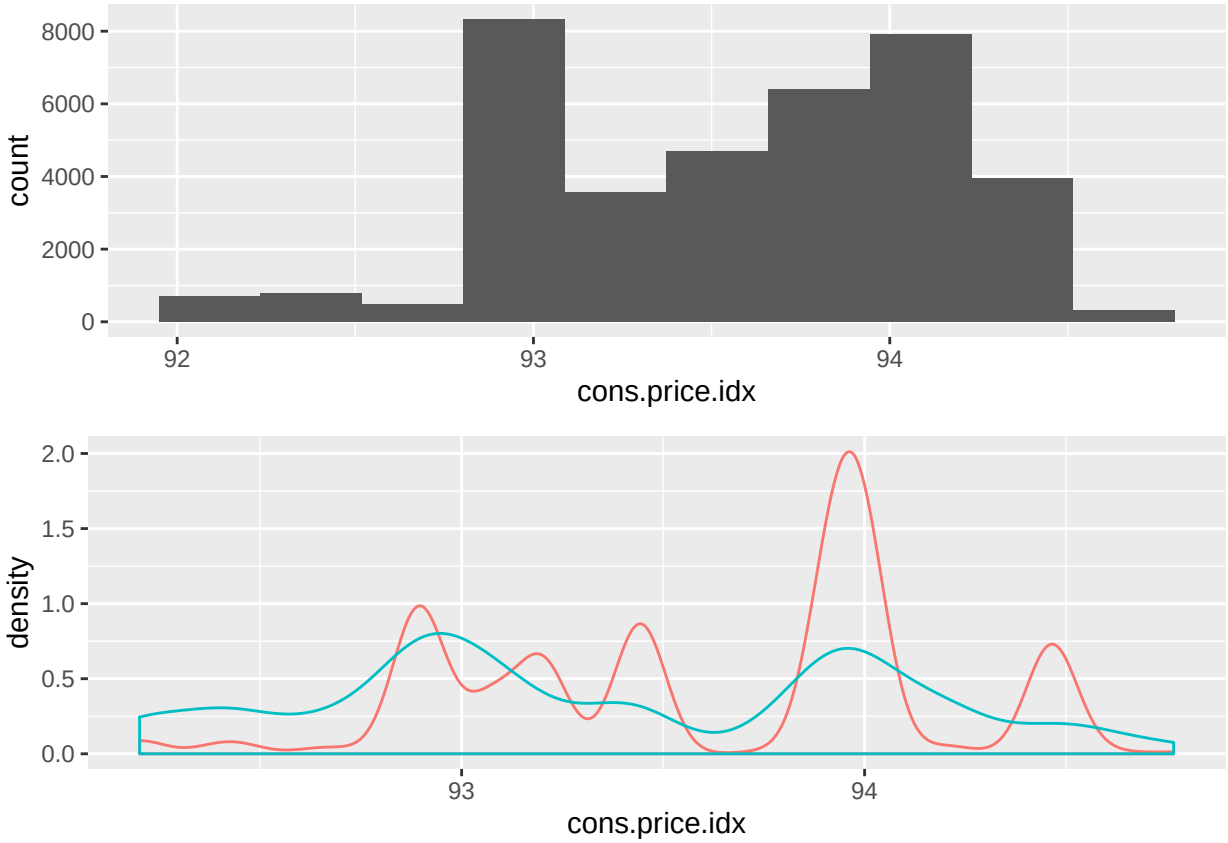
```
g1 <- data %>% ggplot(aes(emp.var.rate)) + geom_histogram(bins=10) +
  scale_x_continuous(breaks=seq(-4, 1.5, 0.5))
g2 <- data %>% mutate(y=factor(y)) %>%
  ggplot(aes(x=emp.var.rate, color=y)) + geom_density(show.legend = FALSE) +
  scale_x_continuous(breaks=seq(-4, 1.5, 0.5))
grid.arrange(g1, g2, nrow=2)
```



We can see a distinguishing trend here with negative values of employment variation rate having a higher trend of acceptance compared to positive values.

Let's look at the consumer price index. We see majority of the data has consumer price index between 93 and 94.5 with very few records outside of this range.

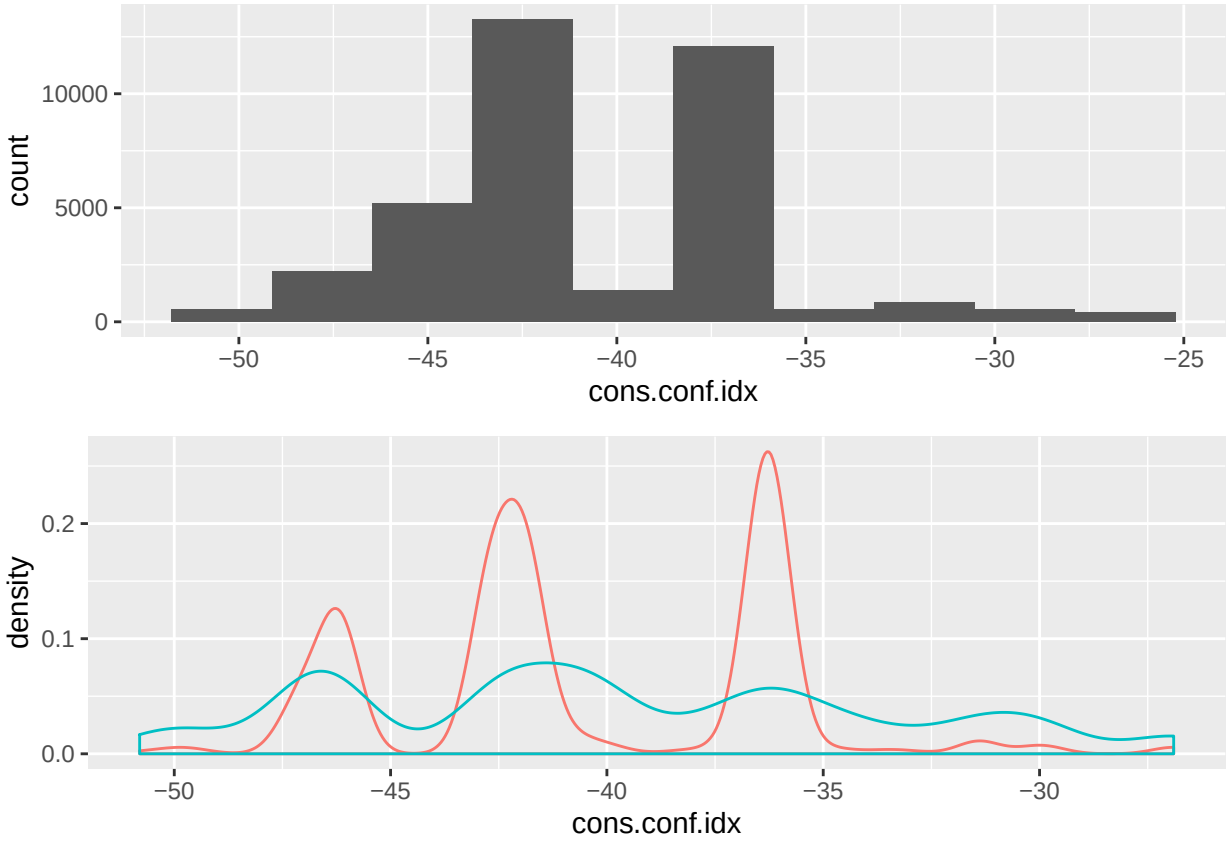
```
g1 <- data %>% ggplot(aes(cons.price.idx)) + geom_histogram(bins=10)
g2 <- data %>% mutate(y=factor(y)) %>%
  ggplot(aes(x=cons.price.idx, color=y)) +
  geom_density(show.legend=FALSE)
grid.arrange(g1, g2, nrow=2)
```



There are specific small ranges where we see a slightly higher proportion of acceptances.

Let's look at the consumer confidence index.

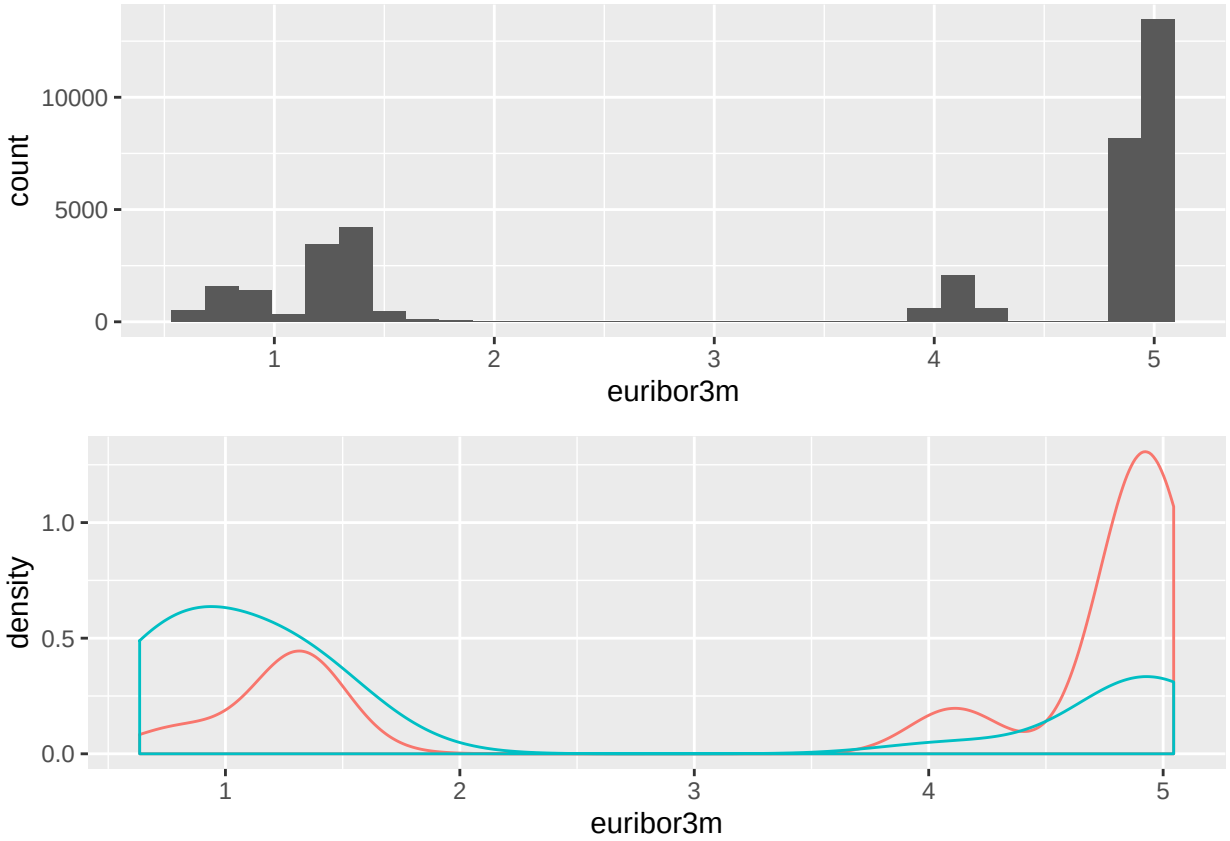
```
g1 <- data %>% ggplot(aes(cons.conf.idx)) + geom_histogram(bins=10)
g2 <- data %>% mutate(y=factor(y)) %>%
  ggplot(aes(x=cons.conf.idx, color=y)) +
  geom_density(show.legend = FALSE)
grid.arrange(g1, g2, nrow=2)
```



We can see a distinguishing trend here with consumer conf index in the range of -40 to -37.5 and >-35.

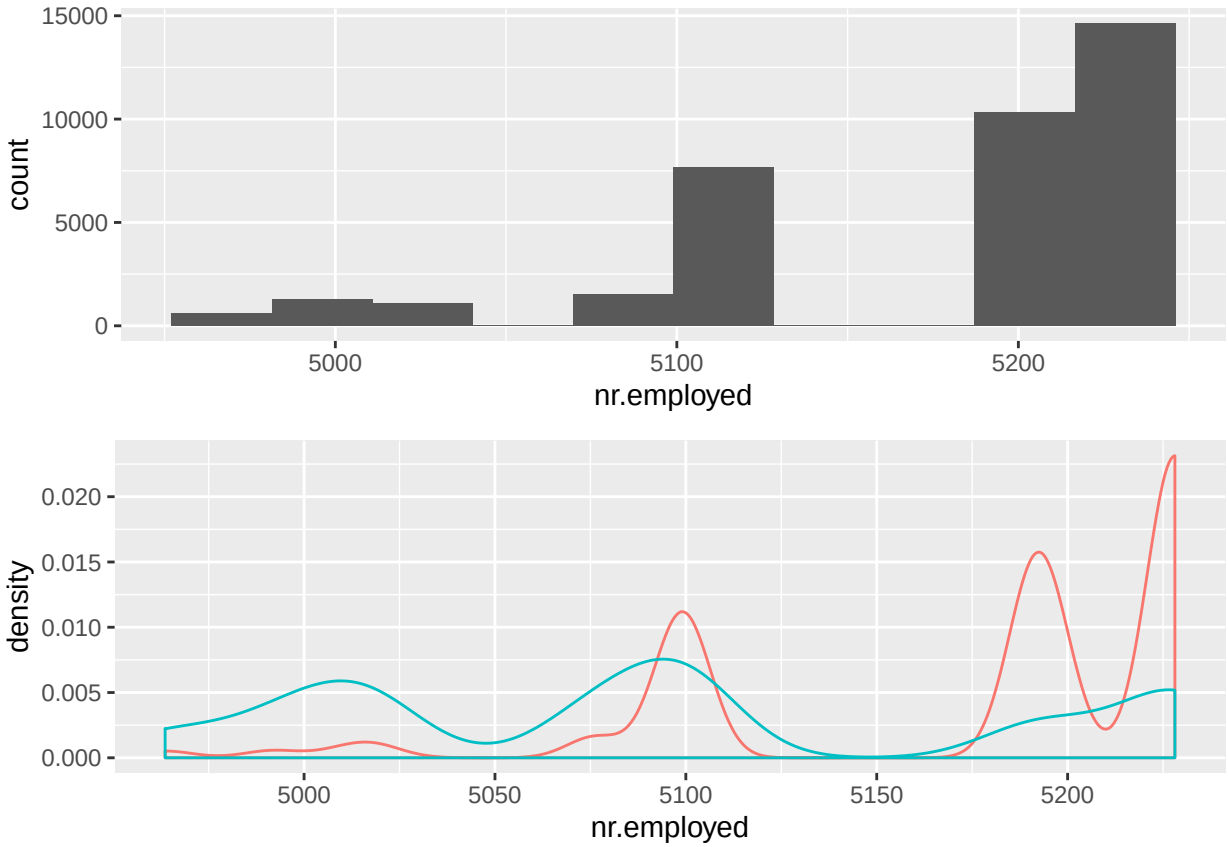
Let's look at the euribor3m rate.

```
g1 <- data %>% ggplot(aes(euribor3m)) + geom_histogram()
g2 <- data %>% mutate(y=factor(y)) %>%
  ggplot(aes(x=euribor3m, color=y)) +
  geom_density(show.legend = FALSE)
grid.arrange(g1, g2, nrow=2)
```



We can see a distinguishing trend here with low values of euribor3m having more of the acceptance cases. Let's look at the number of employees now.

```
g1 <- data %>% ggplot(aes(nr.employed)) + geom_histogram(bins=10)
g2 <- data %>% mutate(y=factor(y)) %>%
  ggplot(aes(x=nr.employed, color=y)) +
  geom_density(show.legend = FALSE)
grid.arrange(g1, g2, nrow=2)
```



We can see a distinguishing trend here as well with <5100 employees having a higher acceptance ratio.

Is there a correlation between these social and economic context fields and the acceptance?

```
temp <- data %>%
  select(emp.var.rate, cons.price.idx, cons.conf.idx, euribor3m, nr.employed, y)
cor(temp)[1:5,6]
```

```
##   emp.var.rate cons.price.idx  cons.conf.idx      euribor3m    nr.employed
##    -0.29881771    -0.13384325     0.05742473    -0.30876602    -0.35687347
```

Based on the correlation coefficients, we can see that emp.var.rate, euribor3m and nr.employed have a moderately high correlation and that is what was seen in the plots above as well.

Identifying influencers for building a prediction model

So, let's summarise what we have observed so far:

- Age - target shows slight variation by age. People above 58 are more likely to subscribe
- Job - target shows variation especially for retired and student
- Marital status - not a good influencer
- Education - Target shows slight variation here with customers with university degree or unknown having a very few percentage points more

- Previous Defaulters - target shows some variation here. People whose default status is unknown have a slightly lower subscription rate
- Housing loan status - no influence
- Personal loan status - no influence
- Contact mode - good influencer. Target shows significant variation here. People contacted on their mobiles are more likely to subscribe than the ones contacted by telephone
- Last contact month - good influencer. Customers last contacted in quarter ending months seem to have a much higher subscription rate
- day_of_week - not much of an influencer by itself, but in conjuction with month, seems to play a role though that could just be the month influence
- Number of times contacted - people contacted more than once are less likely to susbcribe
- pdays and previous - no influence. can be ignored as they don't show much selectivity
- poutcome - Good influencer. People who had previously subscribed are likely to subscribe this time too.
- emp.var.rate, euribor3m and nr.employed - all have an influence on the target outcome
- The consumer conf index shows a minor distinguishing trend in specific ranges of -40 to -37.5 and >-35.

Building different prediction models for the first objective

We can now try a few prediction models. Let's first split the data set into training and test sets.

Let's also take the duration column out as we want to build two different models, one for the bank to be able to predict which customers they should contact now (for which the duration value will not be available), and one for predicting the outcome of a contact that has been done (in which case the duration value will be available).

```
duration <- data[,'duration']
data <- data[,-'duration']

idx <- createDataPartition(y=data$y, times=1, p = 0.2, list = FALSE)
test <- data[idx,]
train <- data[-idx,]
duration_test <- duration[idx]
duration_train <- duration[-idx]
```

Let's set a naive benchmark with a naive model that just predicts no/0 for the target (since it is a heavily imbalanced dataset).

```
pred <- rep(0, length(test$y))
model_results <- data.frame(model='Naive benchmark',
                            category='rule',
                            accuracy=round(mean(pred==test$y),3),
                            sensitivity=round(sensitivity(test$y, pred),3),
                            specificity=round(specificity(test$y, pred),3)) %>%
  mutate(balanced_accuracy=(sensitivity+specificity)/2)
model_results %>% select(-category) %>% knitr::kable()
```

model	accuracy	sensitivity	specificity	balanced_accuracy
Naive benchmark	0.879	0	1	0.5

As can be seen, though the accuracy is high at 88% and though we have specificity=1, the sensitivity=0

since we are always predicting 0 or "no". The balanced accuracy is 50%.

While it will be near impossible to get a balanced accuracy close to 90 or 100% since this is a heavily imbalanced dataset with very few positive samples, any prediction model built should at least improve on this 0 sensitivity and 50% balanced accuracy.

We saw earlier people with age>58 may subscribe. Let's try that rule without using any machine learning.

```
pred <- ifelse(test$age>58, 1, 0)
res <- data.frame(model='Age Rule', category='rule',
                  accuracy=round(mean(pred==test$y),3),
                  sensitivity=round(sensitivity(test$y, pred),3),
                  specificity=round(specificity(test$y, pred),3)) %>%
  mutate(balanced_accuracy=(sensitivity+specificity)/2)
model_results <- bind_rows(model_results, res)
res %>% select(-category) %>% knitr::kable()
```

model	accuracy	sensitivity	specificity	balanced_accuracy
Age Rule	0.867	0.106	0.971	0.5385

This overall accuracy is not as good as the naive benchmark but we see an improvement in sensitivity now.

Let's try by job since we saw earlier that retired people and students are more likely to subscribe:

```
pred <- ifelse(test$job %in% c('student', 'retired'), 1, 0)
res <- data.frame(model='Job Rule', category='rule',
                  accuracy=round(mean(pred==test$y),3),
                  sensitivity=round(sensitivity(test$y, pred),3),
                  specificity=round(specificity(test$y, pred),3)) %>%
  mutate(balanced_accuracy=(sensitivity+specificity)/2)
model_results <- bind_rows(model_results, res)
res %>% select(-category) %>% knitr::kable()
```

model	accuracy	sensitivity	specificity	balanced_accuracy
Job Rule	0.856	0.14	0.954	0.547

Though the overall accuracy is still lower than the naive benchmark, the sensitivity shows further improvement.

Let's try one final rule model based on contact mode and month of contact:

```
pred <- ifelse(test$contact == 'cellular' & test$month %in% c('mar','jun','sep','dec'),1,0)
res <- data.frame(model='Month+Contact Rule', category='rule',
                  accuracy=round(mean(pred==test$y),3),
                  sensitivity=round(sensitivity(test$y, pred),3),
                  specificity=round(specificity(test$y, pred),3)) %>%
  mutate(balanced_accuracy=(sensitivity+specificity)/2)
model_results <- bind_rows(model_results, res)
res %>% select(-category) %>% knitr::kable()
```

model	accuracy	sensitivity	specificity	balanced_accuracy
Month+Contact Rule	0.876	0.193	0.969	0.581

We see a marked improvement in the sensitivity now at 0.19 and balanced accuracy at 58%.

Let's see if a machine learning model can improve on this.

First define a metric function to be used by the caret train function:

```
specSummary <- function(data, lev=NULL, model=NULL) {
  tp <- sum(data$obs == data$pred & data$obs == 1)
  fp <- sum(data$obs != data$pred & data$pred == 1)
  fn <- sum(data$obs != data$pred & data$pred == 0)
  tn <- sum(data$obs == data$pred & data$obs == 0)
  out <- (tp/(tp+fn) + tn/(tn+fp))/2
  names(out) <- 'bal_acc'
  out
}
metricControl <- trainControl(summaryFunction = specSummary)
```

Let's try a logistic regression model based on month and contact mode since the month and contact model rule model worked best among all the rule based models.

```
month_contact_lr <- train %>% mutate(y=factor(y, levels=c(0,1))) %>%
  train(y~month+contact, data=., method='glm', preProcess=c('center','scale'),
        metric='bal_acc', trControl=metricControl)
pred <- ifelse(predict(month_contact_lr, newdata=test)==1,1,0)
res <- data.frame(model='Month+Contact LR model',
                  category='ml1',
                  accuracy=round(mean(pred==test$y),3),
                  sensitivity=round(sensitivity(test$y, pred),3),
                  specificity=round(specificity(test$y, pred),3)) %>%
  mutate(balanced_accuracy=(sensitivity+specificity)/2)
model_results <- bind_rows(model_results, res)
res %>% select(-category) %>% knitr::kable()
```

model	accuracy	sensitivity	specificity	balanced_accuracy
Month+Contact LR model	0.878	0.122	0.982	0.552

This is better than the naive benchmark both in terms of accuracy and sensitivity, but the sensitivity is still low. It is not as good as the contact+month rule.

Let's try based on month and day_of_week and contact mode:

```
month_day_contact_lr <- train %>% mutate(y=factor(y, levels=c(0,1))) %>%
  train(y~month+day_of_week+contact, data=., method='glm',
        preProcess=c('center','scale'), metric='bal_acc',
        trControl=metricControl)
pred <- ifelse(predict(month_day_contact_lr, newdata=test)==1,1,0)
res <- data.frame(model='Month+Contact+Day LR model',
                  category='ml1',
```

```
                  accuracy=round(mean(pred==test$y),3),
                  sensitivity=round(sensitivity(test$y, pred),3),
                  specificity=round(specificity(test$y, pred),3)) %>%
  mutate(balanced_accuracy=(sensitivity+specificity)/2)
model_results <- bind_rows(model_results, res)
res %>% select(-category) %>% knitr::kable()
```

model	accuracy	sensitivity	specificity	balanced_accuracy
Month+Contact+Day LR model	0.882	0.145	0.983	0.564

While the overall accuracy is better than all of the previous models including the rule-based models and while the sensitivity has also improved over the previous LR model, it is still not as good as the simple month + contact rule model.

Let's try a logistic regression model with all the relevant features that we found may be good influencers.

```
lr <- train %>% mutate(y=factor(y, levels=c(0,1))) %>%
  train(y~age+job+education+contact+month+day_of_week+campaign+poutcome+emp.var.rate+euribor3m+nr.emplo
        data=., method='glm', preProcess=c('center','scale'), metric='bal_acc',
        trControl=metricControl)
pred <- ifelse(predict(lr, newdata=test)==1,1,0)
res <- data.frame(model='Selected Features LR', category='ml1',
                  accuracy=round(mean(pred==test$y),3),
                  sensitivity=round(sensitivity(test$y, pred),3),
                  specificity=round(specificity(test$y, pred),3)) %>%
  mutate(balanced_accuracy=(sensitivity+specificity)/2)
model_results <- bind_rows(model_results, res)
res %>% select(-category) %>% knitr::kable()
```

model	accuracy	sensitivity	specificity	balanced_accuracy
Selected Features LR	0.894	0.234	0.985	0.6095

We see a good increase in overall accuracy, sensitivity and balanced accuracy. This is the best result so far. Sensitivity has gone up to 0.23 and balanced accuracy to 60%.

Let's try a logistic regression model with all the features in the data set:

```
lr <- train %>% mutate(y=factor(y, levels=c(0,1))) %>%
  train(y~., data=., method='glm', preProcess=c('center','scale'), metric='bal_acc',
        trControl=metricControl)
pred <- ifelse(predict(lr, newdata=test)==1,1,0)
res<-data.frame(model='All features LR', category='ml1',
                accuracy=round(mean(pred==test$y),3),
                sensitivity=round(sensitivity(test$y, pred),3),
                specificity=round(specificity(test$y, pred),3)) %>%
  mutate(balanced_accuracy=(sensitivity+specificity)/2)
model_results <- bind_rows(model_results, res)
res %>% select(-category) %>% knitr::kable()
```

model	accuracy	sensitivity	specificity	balanced_accuracy
All features LR	0.895	0.24	0.985	0.6125

This shows a marginal improvement over the previous model, but this is probably a result of overfitting against the test set since we know some of the extra features are not good influencers.

Let's move on to other types of machine learning models now that we have done a detailed testing of logistic regression.

Let's try a gamLoess model now.

```
gammodel <- train %>% mutate(y=factor(y, levels=c(0,1))) %>%
  train(y~age+contact+month+day_of_week+campaign+poutcome+emp.var.rate+euribor3m+nr.employed+cons.price
        data=., method='gamLoess', preProcess=c('center','scale'), metric='bal_acc',
        trControl=metricControl)
pred <- ifelse(predict(gammodel, newdata=test)==1,1,0)
res <- data.frame(model='GAM Loess model', category='ml1',
                  accuracy=round(mean(pred==test$y),3),
                  sensitivity=round(sensitivity(test$y, pred),3),
                  specificity=round(specificity(test$y, pred),3)) %>%
  mutate(balanced_accuracy=(sensitivity+specificity)/2)
model_results <- bind_rows(model_results, res)
res %>% select(-category) %>% knitr::kable()
```

model	accuracy	sensitivity	specificity	balanced_accuracy
GAM Loess model	0.888	0.217	0.981	0.599

This model does not perform as well as the best lr model. The sensitivity is lower at 0.22 and balanced accuracy a little lower than 0.6.

Let's try a naive Bayes model now.

```
nbmodel <- train %>% mutate(y=factor(y, levels=c(0,1))) %>%
  train(y~age+contact+month+day_of_week+campaign+poutcome+emp.var.rate+euribor3m+nr.employed+cons.price
        data=., method='naive_bayes', preProcess=c('center','scale'), metric='bal_acc',
        trControl=metricControl)
pred <- ifelse(predict(nbmodel, newdata=test)==1,1,0)
res <- data.frame(model='Naive Bayes', category='ml1',
                  accuracy=round(mean(pred==test$y),3),
                  sensitivity=round(sensitivity(test$y, pred),3),
                  specificity=round(specificity(test$y, pred),3)) %>%
  mutate(balanced_accuracy=(sensitivity+specificity)/2)
model_results <- bind_rows(model_results, res)
res %>% select(-category) %>% knitr::kable()
```

model	accuracy	sensitivity	specificity	balanced_accuracy
Naive Bayes	0.88	0.417	0.943	0.68

This model is the best so far and performs much better than the other models. It gives a sensitivity of 0.42

and a balanced accuracy of 68%.

Compare this with the naive benchmark of always predicting "no" which has 0 sensitivity and 50% balanced accuracy.

Let's see if we can further improve on this by using a decision tree model. Recall that there were a few key features which showed distinct selectivity for the target - for example, months that are quarter ending months, age > 58, contact = cellular and a combination of such rules.

A decision tree based model should ideally perform better than all the other models that have been tested so far.

Let's see if the results bear out that supposition.

Let's try a decision tree model now with the key influencers (with default parameters for the algorithm for now. The model will be tuned next).

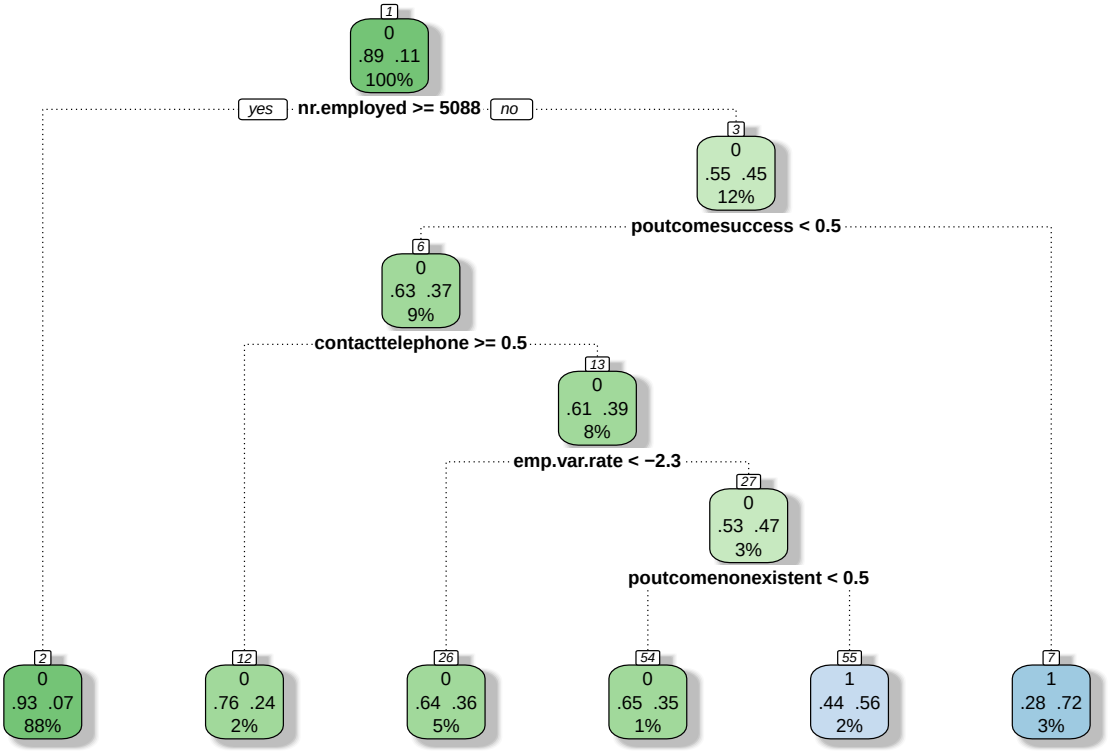
```
rpart_model <- train %>% mutate(y=factor(y, levels=c(0,1))) %>%
  train(y~contact+month+day_of_week+campaign+poutcome+emp.var.rate+euribor3m+nr.employed+cons.price.idx
        method='rpart', metric='bal_acc', trControl=metricControl)
pred <- ifelse(predict(rpart_model, newdata=test)==1,1,0)
res <- data.frame(model='Decision Tree (def params)', category='ml1',
                  accuracy=round(mean(pred==test$y),3),
                  sensitivity=round(sensitivity(test$y, pred),3),
                  specificity=round(specificity(test$y, pred),3)) %>%
  mutate(balanced_accuracy=(sensitivity+specificity)/2)
res %>% select(-category) %>% knitr::kable()
```

model	accuracy	sensitivity	specificity	balanced_accuracy
Decision Tree (def params)	0.895	0.248	0.983	0.6155

The sensitivity is now at 0.25 and balanced accuracy at 0.62 which is better than the previous LR models but not as good as the naive_bayes model. But note that we have not yet tuned the hyperparameters for the decision tree.

Let's look at how the decision tree is generating predictions:

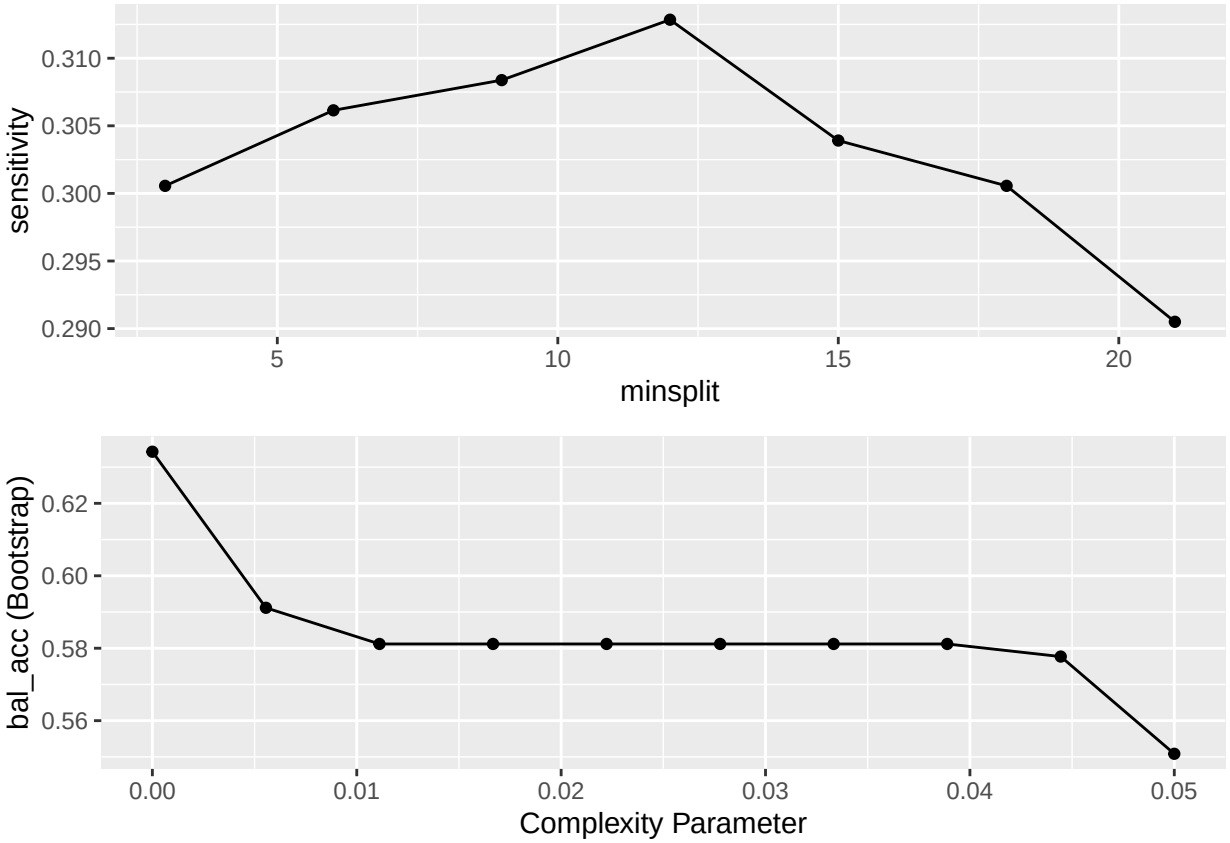
```
fancyRpartPlot(rpart_model$finalModel, caption='Decision Tree')
```



Decision Tree

Let's try to tune the decision tree model now.

Since rpart does not allow tuning of multiple parameters in tuneGrid, we will have to manually iterate for minsplit.



After tuning, the best minsplit parameter is 12 and the best sensitivity achieved is 0.313

Let's check its performance against the test set:

```
pred <- ifelse(predict(best_model, newdata=test)==1,1,0)
res <- data.frame(model='Tuned Decision Tree', category='ml1',
                  accuracy=round(mean(pred==test$y),3),
                  sensitivity=round(sensitivity(test$y, pred),3),
                  specificity=round(specificity(test$y, pred),3)) %>%
  mutate(balanced_accuracy=(sensitivity+specificity)/2)
model_results <- bind_rows(model_results, res)
res %>% select(-category) %>% knitr::kable()
```

model	accuracy	sensitivity	specificity	balanced_accuracy
Tuned Decision Tree	0.894	0.313	0.974	0.6435

This tuned decision tree model gives a sensitivity of 0.31 and a balanced accuracy of 0.64. While this is much better than all the LR models and rule-based models, it still is lower than the naive bayes model which gave a better sensitivity and balanced accuracy.

Training a random forest model would potentially give a better performance, however, due to computing resource limitations, this is not covered in this study.

Store the two best models for the first objective. This is for model aggregation later on.

```
obj1_top_1_model <- nbmodel#the naive bayes model
obj1_top_2_model <- best_model#the tuned decision tree model
```

Building different prediction models for the second objective (with duration)

Now that the first prediction model is selected as the naive bayes model, the second model will now be selected for the second objective of predicting if a contact made results in a successul subscription or not.

Let's build and compare an LR model, a naive bayes model and a tuned decision tree model after including the duration field.

Let's add the duration field back to the train and test sets:

```
train$duration <- duration_train
test$duration <- duration_test
```

Let's start with an LR model:

```
duration_lr <- train %>% mutate(y=factor(y, levels=c(0,1))) %>%
  train(y~., data=., method='glm', preProcess=c('center','scale'),
        metric='bal_acc', trControl=metricControl)
pred <- ifelse(predict(duration_lr, newdata=test)==1,1,0)
res <- data.frame(model='All features with duration LR',
                  category='ml2',
                  accuracy=round(mean(pred==test$y),3),
                  sensitivity=round(sensitivity(test$y, pred),3),
                  specificity=round(specificity(test$y, pred),3)) %>%
  mutate(balanced_accuracy=(sensitivity+specificity)/2)
model_results <- bind_rows(model_results, res)
res %>% select(-category) %>% knitr::kable()
```

model	accuracy	sensitivity	specificity	balanced_accuracy
All features with duration LR	0.903	0.397	0.973	0.685

We can see this LR model gives us a sensitivity of 0.4 and a balanced accuracy of 68%, with an overall accuracy of 90%.

Let's build a naive bayes model now:

```
nbmodel <- train %>% mutate(y=factor(y, levels=c(0,1))) %>%
  train(y~duration+age+contact+month+day_of_week+campaign+poutcome+emp.var.rate+euribor3m+nr.employed+c
        data=., method='naive_bayes', preProcess=c('center','scale'),
        metric='bal_acc', trControl=metricControl)
pred <- ifelse(predict(nbmodel, newdata=test)==1,1,0)
res <- data.frame(model='Naive Bayes with duration', category='ml2',
                  accuracy=round(mean(pred==test$y),3),
                  sensitivity=round(sensitivity(test$y, pred),3),
                  specificity=round(specificity(test$y, pred),3)) %>%
  mutate(balanced_accuracy=(sensitivity+specificity)/2)
model_results <- bind_rows(model_results, res)
res %>% select(-category) %>% knitr::kable()
```

model	accuracy	sensitivity	specificity	balanced_accuracy
Naive Bayes with duration	0.883	0.456	0.941	0.6985

This naive bayes model that includes the duration gives a sensitivity of 0.46 and a balanced accuracy of 70%.

Let's build a decision tree model now (default parameters chosen for the decision tree for now).

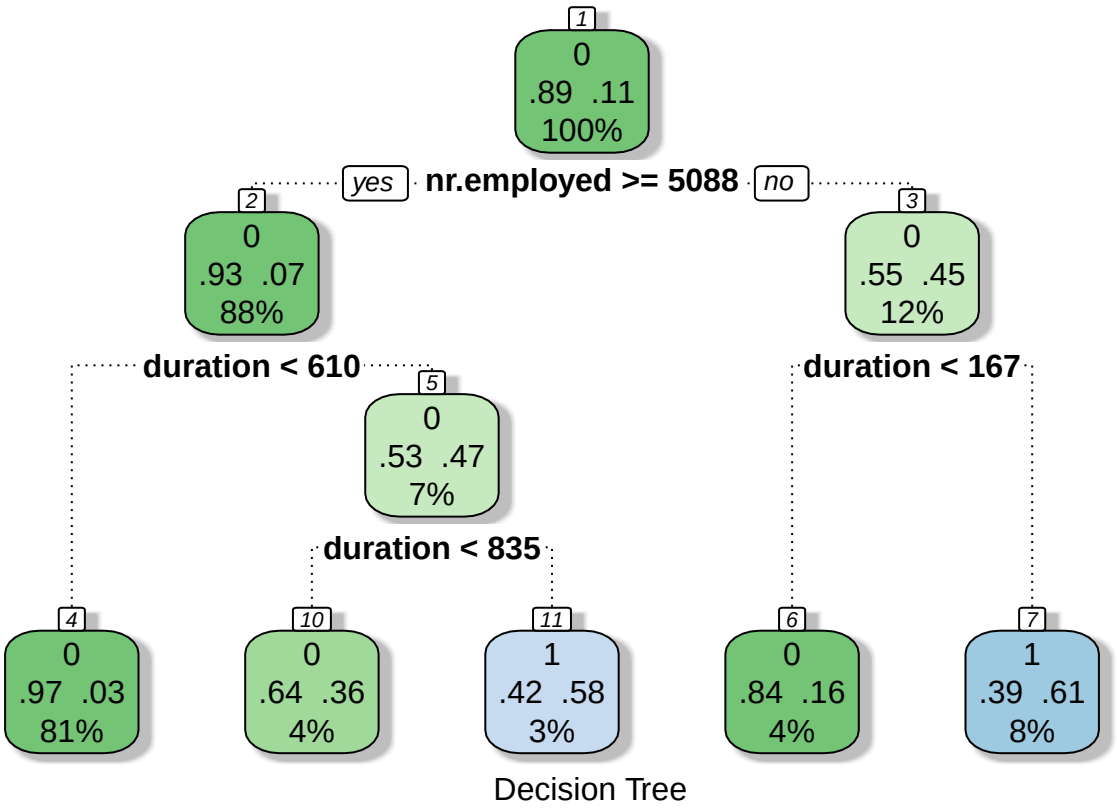
```
rpart_model <- train %>% mutate(y=factor(y, levels=c(0,1))) %>%
  train(y~duration+contact+month+day_of_week+campaign+poutcome+emp.var.rate+euribor3m+nr.employed+cons.p
pred <- ifelse(predict(rpart_model, newdata=test)==1,1,0)
res <- data.frame(model='Dec Tree with duration',
                  category='ml2',
                  accuracy=round(mean(pred==test$y),3),
                  sensitivity=round(sensitivity(test$y, pred),3),
                  specificity=round(specificity(test$y, pred),3)) %>%
  mutate(balanced_accuracy=(sensitivity+specificity)/2)
model_results <- bind_rows(model_results, res)
res %>% select(-category) %>% knitr::kable()
```

model	accuracy	sensitivity	specificity	balanced_accuracy
Dec Tree with duration	0.902	0.558	0.949	0.7535

This model gives a much better result than the LR model and the naive bayes model. The sensitivity is 0.56 and the balanced accuracy goes up to 75% with an overall accuracy of 90%.

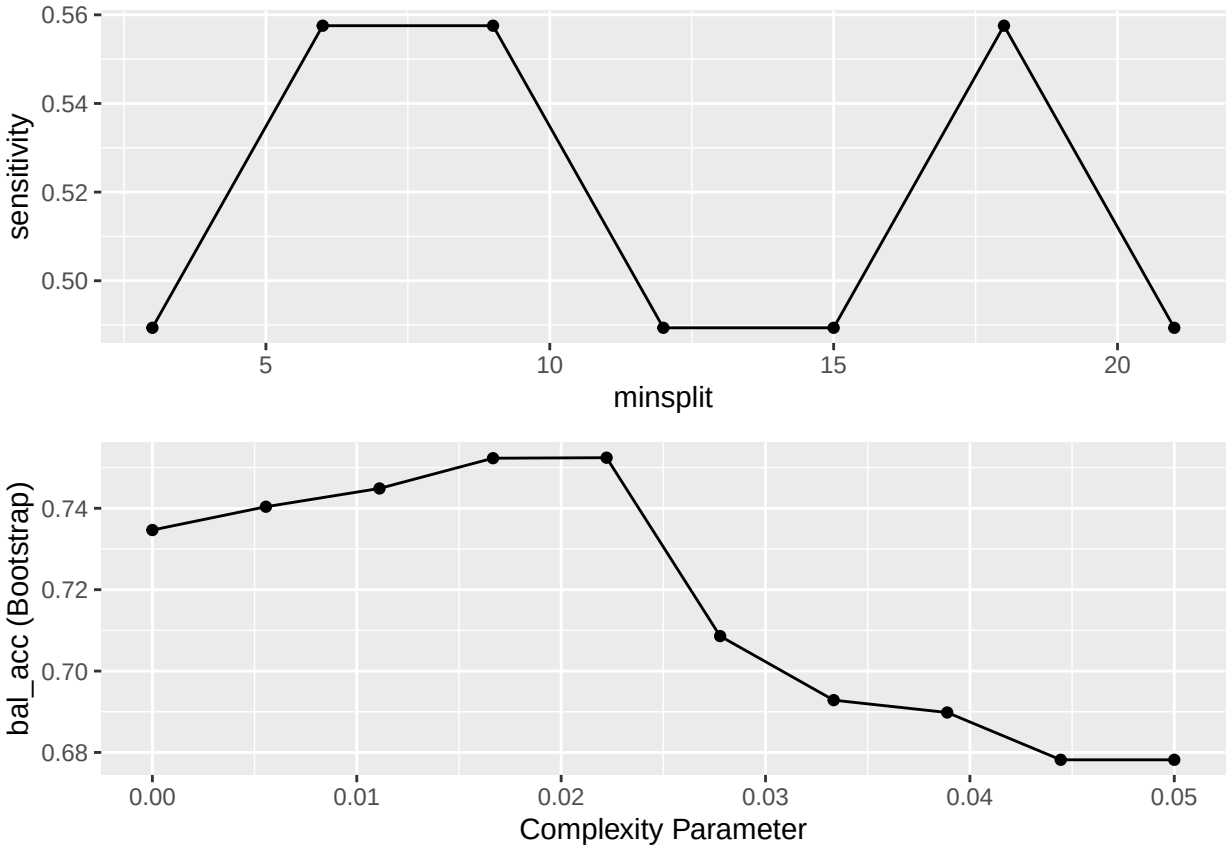
We can see how the duration of the call plays a key role in deciding the subscription outcome:

```
fancyRpartPlot(rpart_model$finalModel, caption='Decision Tree')
```



Decision Tree

Let's tune the decision tree model now.



After tuning, the best minsplit parameter is 6 and the best sensitivity achieved is 0.558

Let's check its performance against the test set:

```
pred <- ifelse(predict(best_model_dur, newdata=test)==1,1,0)
res <- data.frame(model='Tuned Dec Tree with duration',
                  category='ml2',
                  accuracy=round(mean(pred==test$y),3),
                  sensitivity=round(sensitivity(test$y, pred),3),
                  specificity=round(specificity(test$y, pred),3)) %>%
  mutate(balanced_accuracy=(sensitivity+specificity)/2)
model_results <- bind_rows(model_results, res)
res %>% select(-category) %>% knitr::kable()
```

model	accuracy	sensitivity	specificity	balanced_accuracy
Tuned Dec Tree with duration	0.902	0.558	0.949	0.7535

After tuning, we get the same performance from the decision tree model with a sensitivity of 0.6 and a balanced accuracy of 75%. This is now the best model for the duration-based prediction.

Training a random forest model would potentially give a better performance, however, due to computing resource limitations, this is not covered in this study.

Store the two best models for the second objective. This is for model aggregation.

```
obj2_top_1_model <- best_model_dur#the tuned decision tree model
obj2_top_2_model <- nbmodel#the naive bayes model
```

Model aggregation for both objectives

We already know that this dataset is a highly imbalanced one with very few positive samples in the dataset. Any machine learning model is very likely to underestimate the number of positive cases in the test set due to this class imbalance.

One way we can improve the performance of prediction is to aggregate multiple prediction models and instead of averaging as is typically done in model aggregation, we can predict a postive outcome if either of the prediction models predicts a positive outcome.

That way, we compensate for the underestimation of positive outcomes by each individual prediction model and this is likely to give a better performance.

So, let's try this now for both the objectives one by one.

Let's start with the two best models for the first objective. Predict 1 if either of the two models predicts 1.

```
pred1 <- ifelse(predict(obj1_top_1_model, newdata=test)==1,1,0)
pred2 <- ifelse(predict(obj1_top_2_model, newdata=test)==1,1,0)
pred <- ifelse(pred1==1 | pred2==1, 1, 0)

res <- data.frame(model='Final Aggregated Model',
                  category='ml1',
                  accuracy=round(mean(pred==test$y),3),
                  sensitivity=round(sensitivity(test$y, pred),3),
                  specificity=round(specificity(test$y, pred),3)) %>%
  mutate(balanced_accuracy=(sensitivity+specificity)/2)
model_results <- bind_rows(model_results, res)
res %>% select(-category) %>% knitr::kable()
```

model	accuracy	sensitivity	specificity	balanced_accuracy
Final Aggregated Model	0.88	0.475	0.935	0.705

We can see that these results are way better than each of the individual models. We get a sensitivity of 0.48 and a balanced accuracy of 70.5%.

Let's do the same for the two best models for the second objective. Predict 1 if either of them predicts 1.

```
pred1 <- ifelse(predict(obj2_top_1_model, newdata=test)==1,1,0)
pred2 <- ifelse(predict(obj2_top_2_model, newdata=test)==1,1,0)
pred <- ifelse(pred1==1 | pred2==1, 1, 0)

res <- data.frame(model='Final Aggregated Model',
                  category='ml2',
                  accuracy=round(mean(pred==test$y),3),
                  sensitivity=round(sensitivity(test$y, pred),3),
                  specificity=round(specificity(test$y, pred),3)) %>%
  mutate(balanced_accuracy=(sensitivity+specificity)/2)
model_results <- bind_rows(model_results, res)
res %>% select(-category) %>% knitr::kable()
```

model	accuracy	sensitivity	specificity	balanced_accuracy
Final Aggregated Model	0.886	0.668	0.916	0.792

We can see that these results are way better than each of the individual models. We get a sensitivity of 0.67 and a balanced accuracy of 79.2%.

Final model and evaluation against the validation set

Now that we have our final models chosen, let's run the final evaluation against the *validation* dataset.

```
pred1 <- ifelse(predict(obj1_top_1_model, newdata=validation[,-'duration'])==1,1,0)
pred2 <- ifelse(predict(obj1_top_2_model, newdata=validation[,-'duration'])==1,1,0)
pred <- ifelse(pred1==1 | pred2==1, 1, 0)

res_valid <- data.frame(model='Final Validation Evaluation - objective 1',
                  category='final',
                  accuracy=round(mean(pred==validation$y), 3),
                  sensitivity=round(sensitivity(validation$y, pred), 3),
                  specificity=round(specificity(validation$y, pred), 3)) %>%
  mutate(balanced_accuracy=(sensitivity+specificity)/2)
model_results <- bind_rows(model_results, res_valid)

pred1 <- ifelse(predict(obj2_top_1_model, newdata=validation)==1,1,0)
pred2 <- ifelse(predict(obj2_top_2_model, newdata=validation)==1,1,0)
pred <- ifelse(pred1==1 | pred2==1, 1, 0)

res <- data.frame(model='Final Validation Evaluation - objective 2',
                  category='final',
                  accuracy=mean(pred==validation$y),
                  sensitivity=sensitivity(validation$y, pred),
                  specificity=specificity(validation$y, pred)) %>%
  mutate(balanced_accuracy=(sensitivity+specificity)/2)
model_results <- bind_rows(model_results, res)
res_valid <- bind_rows(res_valid, res)
res_valid %>% select(-category) %>% knitr::kable()
```

model	accuracy	sensitivity	specificity	balanced_accuracy
Final Validation Evaluation - objective 1	0.8760000	0.4100000	0.9330000	0.6715000
Final Validation Evaluation - objective 2	0.8807963	0.6554054	0.9080272	0.7817163

Results

A lot of ground has been covered in this analysis. The data was analysed and explored to identify trends in the subscription outcome, and the key influencers that impact the subscription outcome.

Various models were tested starting from simple rule-based models, logistic regression models based on different combinations of features, multinomial models, naive bayes models, decision tree models with tuning

of hyperparameters and finally aggregating the best models.

The above was done for both the objectives (predicting if a customer if contacted would subscribe and predicting if a customer once contacted would subscribe).

The results of all the iterations are summarised below in ascending order of balanced accuracy:

Rule-based models:

model	category	accuracy	sensitivity	specificity	balanced_accuracy
Naive benchmark	rule	0.879	0.000	1.000	0.500
Age Rule	rule	0.867	0.106	0.971	0.538
Job Rule	rule	0.856	0.140	0.954	0.547
Month+Contact Rule	rule	0.876	0.193	0.969	0.581

ML models for objective 1:

model	category	accuracy	sensitivity	specificity	balanced_accuracy
Month+Contact LR model	ml1	0.878	0.122	0.982	0.552
Month+Contact+Day LR model	ml1	0.882	0.145	0.983	0.564
GAM Loess model	ml1	0.888	0.217	0.981	0.599
Selected Features LR	ml1	0.894	0.234	0.985	0.610
All features LR	ml1	0.895	0.240	0.985	0.612
Tuned Decision Tree	ml1	0.894	0.313	0.974	0.644
Naive Bayes	ml1	0.880	0.417	0.943	0.680
Final Aggregated Model	ml1	0.880	0.475	0.935	0.705

ML models for objective 2:

model	category	accuracy	sensitivity	specificity	balanced_accuracy
All features with duration LR	ml2	0.903	0.397	0.973	0.685
Naive Bayes with duration	ml2	0.883	0.456	0.941	0.698
Dec Tree with duration	ml2	0.902	0.558	0.949	0.754
Tuned Dec Tree with duration	ml2	0.902	0.558	0.949	0.754
Final Aggregated Model	ml2	0.886	0.668	0.916	0.792

The final evaluation result on the validation set is listed below for both the objectives:

model	accuracy	sensitivity	specificity	balanced_accuracy
Final Validation Evaluation - objective 1	0.8760000	0.4100000	0.9330000	0.6715000
Final Validation Evaluation - objective 2	0.8807963	0.6554054	0.9080272	0.7817163

Conclusion and future work that can be done

This dataset though consisting of over 41000 records, is an extremely class imbalanced dataset. Evaluating prediction models based on just accuracy would not help as a naive model which always predicts “no” would achieve a high accuracy of around 88%.

The balanced accuracy was chosen as the performance metric where balanced accuracy is calculated as the average of sensitivity and specificity.

Various models were tested ranging from simple rule-based models, logistic regression models, naive bayes model, decision tree models and finally, aggregated models.

The best result was seen through model aggregation where a naive bayes model and a tuned decision tree model where aggregated to generate predictions.

The model aggregation was done by predicting a positive outcome if any of the underlying prediction models predicts a positive outcome. This helps to compensate for the underestimation of the positive outcomes and improves prediction performance on a highly class imbalanced dataset.

What else could be done?

- Since it was seen that decision tree models give good performance, intuitively, random forest models would give even better performance. However, due to computing resource constraints, random forest models could not be trained on this large dataset
- SVM-based models could also provide good performance, however, this could not be tested either due to computing resource constraints
- Further hyperparameter tuning of the decision tree models, random forest models and SVM kernels could improve prediction performance
- Model aggregation across random forest models and SVM models could further boost performance
- Finally, given the class imbalance in the dataset, more positive samples could be simulated and generated before prediction models are trained. This would help overcome the class imbalance problem and get a much better accuracy