Bank marketing

Introduction

This is the Capstone project of the HarvardX Data science program.

Harvard's data science Professional certificate program covers several data science methods, like visualization data wrangling or machine learning. This capstone project will use each of these concepts in a real world example.

My project deals with a dataset from Portuguese bank direct marketing campaigns. The goal is to develop a model which predicts if the client will subscribe a term deposit. If it is possible to know which client will subscribe a term deposit, than this could help save time and resources for the bank to concentrate on the important clients.

Dataset Description

The dataset comes from direct marketing campaigns for term deposits from Portuguese banking institutions. The dataset contains several input variables, that describe the calls:

1.age (numeric)

2.job (categorical: 'admin.','blue-collar','entrepreneur','housemaid','management','retired','self-employed','services','student','

3.martial (categorical: 'divorced','married','single','unknown'; note: 'divorced' means divorced or widowed)

4.education (categorical: 'basic.4y','basic.6y','basic.9y','high.school','illiterate','professional.course','university.degree','unknow

5.default (categorical: 'no','yes','unknown')

6.housing (categorical: 'no','yes','unknown')

7.loan (categorical: 'no','yes','unknown')

8.contact (categorical: 'cellular','telephone')

9.month (categorical: 'jan', 'feb', 'mar', . . . , 'nov', 'dec')

10.day_of_week (categorical: 'mon','tue','wed','thu','fri')

11.duration in seconds (numeric) The duration variable is not useful for building a realistic classification model, because the duration is not known before the call was made and when the call is over the output variable is already known. Therefore we will only explore the duration variable in the visualization part of the project, but it won't be incorporated in the feature selection or model building sections.

12.campaign: number of contacts performed during this campaign and for this client

13.pdays: number of days that passed by after the client was last contacted from a previous campaign

14.previous: number of contacts performed before this campaign and for this client (numeric)

15.poutcome: outcome of the previous marketing campaign (categorical: 'failure','nonexistent','success')

16.emp.var.rate: employment variation rate - quarterly indicator (numeric)

17.cons.price.idx: consumer price index - monthly indicator (numeric)

18.cons.conf.idx: consumer confidence index - monthly indicator (numeric)

19.euribor3m: euribor 3 month rate - daily indicator (numeric)

20.nr.employed: number of employees - quarterly indicator (numeric)

And the dataset contains one Output variable that is either "yes" or "no" depending on whether the client subscribed a term deposit or not:

1.y: has the client subscribed a term deposit? (binary: 'yes','no')

Key steps

We start by preparing the data for the analysis, we will get a quick overview, clean the data and finally split the dataset into a training set and a testing set, in the Data preparation section.

Then we will analyze and explore the data by making several plots for the various features of the dataset in the data exploration part.

Furthermore we will explain in the modeling section, what algorithms we will use for the prediction and why.

After this is complete we will train those algorithms on the training set and test them on the test set.

Than we continue by explaining the model evaluation approach and choose the best model accordingly.

Finally we validate our best performing model by applying it to the Validation set.

We will draw conclusions from what we have done in this project and explain some limitations of our approach in the conclusion section.

Data preparation

In this section we import the library we will use and the data set from the UCI Machine learning repository. We will also get an overview over the data and prepare the data for further analysis. At the end of this section we will split the data into training and test set.

In this Project we will use the updated data set bank-additional-full.csv for analysis training, testing and evaluating which model to choose. The bank-additional.csv data is only used for validating the algorithms result.

First we load the library needed for the project.

```r
library(tidyverse)
library(readxl)
library(ggpubr)
library(ggthemes)
library(caret)
library(rpart.plot)
library(corrgram)
```

Then we load the data from the UCI Repository

```r
knitr::opts_chunk$set(echo = TRUE)
data <- read_csv2("bank-additional-full.csv")
Validation <- read_csv2("bank-additional.csv")
head(data)
```

```
## # A tibble: 6 x 21
##     age job    marital education default housing loan  contact month day_of_week
##   <dbl> <chr>  <chr>   <chr>     <chr>   <chr>   <chr> <chr>   <chr> <chr>
## 1    56 house~ married basic.4y  no      no      no    teleph~ may   mon
## 2    57 servi~ married high.sch~ unknown no      no    teleph~ may   mon
## 3    37 servi~ married high.sch~ no      yes     no    teleph~ may   mon
## 4    40 admin. married basic.6y  no      no      no    teleph~ may   mon
## 5    56 servi~ married high.sch~ no      no      yes   teleph~ may   mon
## 6    45 servi~ married basic.9y  unknown no      no    teleph~ may   mon
## # ... with 11 more variables: duration <dbl>, campaign <dbl>, pdays <dbl>,
## #   previous <dbl>, poutcome <chr>, emp.var.rate <dbl>, cons.price.idx <dbl>,
## #   cons.conf.idx <dbl>, euribor3m <chr>, nr.employed <dbl>, y <chr>
```

To get a quick overview over the data for this project, we use the summary() function.

```
summary(data)
```

```
##       age            job              marital           education        
##  Min.   :17.00   Length:41188       Length:41188       Length:41188      
##  1st Qu.:32.00   Class :character   Class :character   Class :character  
##  Median :38.00   Mode  :character   Mode  :character   Mode  :character  
##  Mean   :40.02                                                           
##  3rd Qu.:47.00                                                           
##  Max.   :98.00                                                           
##    default            housing             loan             contact         
##  Length:41188       Length:41188       Length:41188       Length:41188      
##  Class :character   Class :character   Class :character   Class :character  
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character  
##                                                                             
##                                                                             
##                                                                             
##     month           day_of_week           duration         campaign     
##  Length:41188       Length:41188       Min.   :   0.0   Min.   : 1.000  
##  Class :character   Class :character   1st Qu.: 102.0   1st Qu.: 1.000  
##  Mode  :character   Mode  :character   Median : 180.0   Median : 2.000  
##                                        Mean   : 258.3   Mean   : 2.568  
##                                        3rd Qu.: 319.0   3rd Qu.: 3.000  
##                                        Max.   :4918.0   Max.   :56.000  
##      pdays          previous       poutcome          emp.var.rate     
##  Min.   :  0.0   Min.   :0.000   Length:41188       Min.   :-34.0000  
##  1st Qu.:999.0   1st Qu.:0.000   Class :character   1st Qu.:-18.0000  
##  Median :999.0   Median :0.000   Mode  :character   Median : 11.0000  
##  Mean   :962.5   Mean   :0.173                      Mean   :  0.9316  
##  3rd Qu.:999.0   3rd Qu.:0.000                      3rd Qu.: 14.0000  
##  Max.   :999.0   Max.   :7.000                      Max.   : 14.0000  
##  cons.price.idx  cons.conf.idx     euribor3m          nr.employed   
##  Min.   :  932   Min.   :-508.0   Length:41188       Min.   : 5191  
##  1st Qu.:92893   1st Qu.:-427.0   Class :character   1st Qu.:50175  
##  Median :93749   Median :-403.0   Mode  :character   Median :50991  
##  Mean   :85475   Mean   :-365.7                      Mean   :42865  
##  3rd Qu.:93994   3rd Qu.:-361.0                      3rd Qu.:52281  
##  Max.   :94767   Max.   : -33.0                      Max.   :52281  
##       y            
##  Length:41188
##  Class :character
##  Mode  :character
##
##
##
```

Then we want to know where the data set contains Na values. Na values could lead to calculation problems later. We can see that in neither the data dataset nor in the Validation dataset are any Na values

```
colSums(is.na(data))
```

```
##            age            job        marital      education        default
##              0              0              0              0              0
##        housing           loan        contact          month    day_of_week
##              0              0              0              0              0
##       duration       campaign          pdays       previous       poutcome
##              0              0              0              0              0
##   emp.var.rate cons.price.idx  cons.conf.idx      euribor3m    nr.employed
##              0              0              0              0              0
##              y
##              0
```

```
colSums(is.na(Validation))
```

```
##            age            job        marital      education        default
##              0              0              0              0              0
##        housing           loan        contact          month    day_of_week
##              0              0              0              0              0
##       duration       campaign          pdays       previous       poutcome
##              0              0              0              0              0
##   emp.var.rate cons.price.idx  cons.conf.idx      euribor3m    nr.employed
##              0              0              0              0              0
##              y
##              0
```

Then we turn the character features into factors for the machine learning methods later. And we create a vector of all potential predictor names called x_cols.

```
xcols <- c("age","job","marital","education","default","housing","loan","contact","month", "day_of_week
data$job = as.factor(data$job)
data$marital = as.factor(data$marital)
data$education = as.factor(data$education)
data$default = as.factor(data$default)
data$housing = as.factor(data$housing)
data$contact = as.factor(data$contact)
data$month = as.factor(data$month)
data$day_of_week = as.factor(data$day_of_week)
data$poutcome = as.factor(data$poutcome)

data$y <- as.factor(data$y)

data$euribor3m <- as.numeric(data$euribor3m)
```

After cleaning the data we divide the data set into a training set and a test set. 70% will be used for training and 30% will be used for testing.

```r
set.seed(2,sample.kind = "Rounding")
test_index <- createDataPartition(y = data$y, times = 1,
                                  p = 0.3, list = FALSE)
train_set <- data[-test_index,]
test_set <- data[test_index,]
```

Data exploration

In this section we will explore the data to get insights we can use later for the modeling section and that are helpful for banking institutions to better target clients.

Variation

Variables, that do not vary across the dataset contain little or no useful information for the algorithm. The function nearZeroVar() shows the features that do not vary threw out the data set. We remove the only variable pdays that varies almost not at all from observation to observation.

```r
nearZeroVar(train_set[,xcols],saveMetrics = TRUE) %>% select(nzv)
```

```
##                  nzv
## age            FALSE
## job            FALSE
## marital        FALSE
## education      FALSE
## default        FALSE
## housing        FALSE
## loan           FALSE
## contact        FALSE
## month          FALSE
## day_of_week    FALSE
## campaign       FALSE
## pdays           TRUE
## previous       FALSE
## poutcome       FALSE
## emp.var.rate   FALSE
## cons.price.idx FALSE
## cons.conf.idx  FALSE
## euribor3m      FALSE
## nr.employed    FALSE
```

```r
train_set <- train_set %>% select(-pdays)
xcols <- c("age","job","marital","education","default","housing","loan","contact","month", "day_of_week
```

Variable importance

There are 18 features in the data set that could be used for prediction. But we don't know if those features really help us predict the output variable. In order to to find which features are important for the classification model, we use the filterVarImp function from the caret package. Since our output variable is binary the function uses a series of cutoffs to predict the output variable. The importance is then computed by calculating the area under the ROC curve.

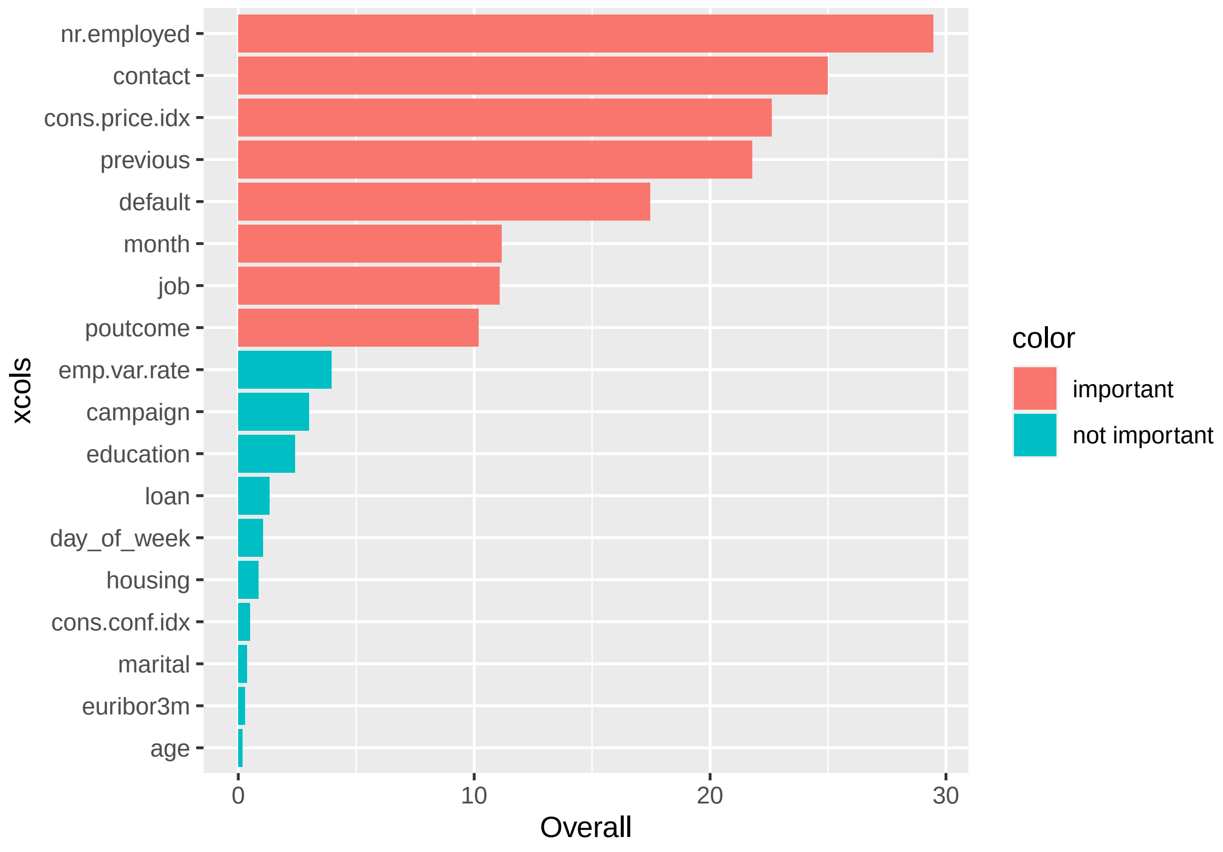
ROC

The receiver operating characteristic curve is being constructed by plotting True positive rate(Sensitivity) against the False positive rate(1-specificity). The ROC plot gives information about how good a predictor separates positive from negative values. An ideal predictor has already a high True positive rate when the False positive rate is still low. Therefore the area under the curve is meaningful when deciding how important a predictor is of the classification problem.

Variable importance plot

The plot shows the importance calculated by the filterVarImp function for each predictor. It is clear to see that features like marital, housing, euribor3m, loan and age are significantly less important for predicting y than features like nr.employed, contact, cons.price.idx, previous, default, month and job. For the model we will use only predictors with an overall importance of over 5 that are colored in red.

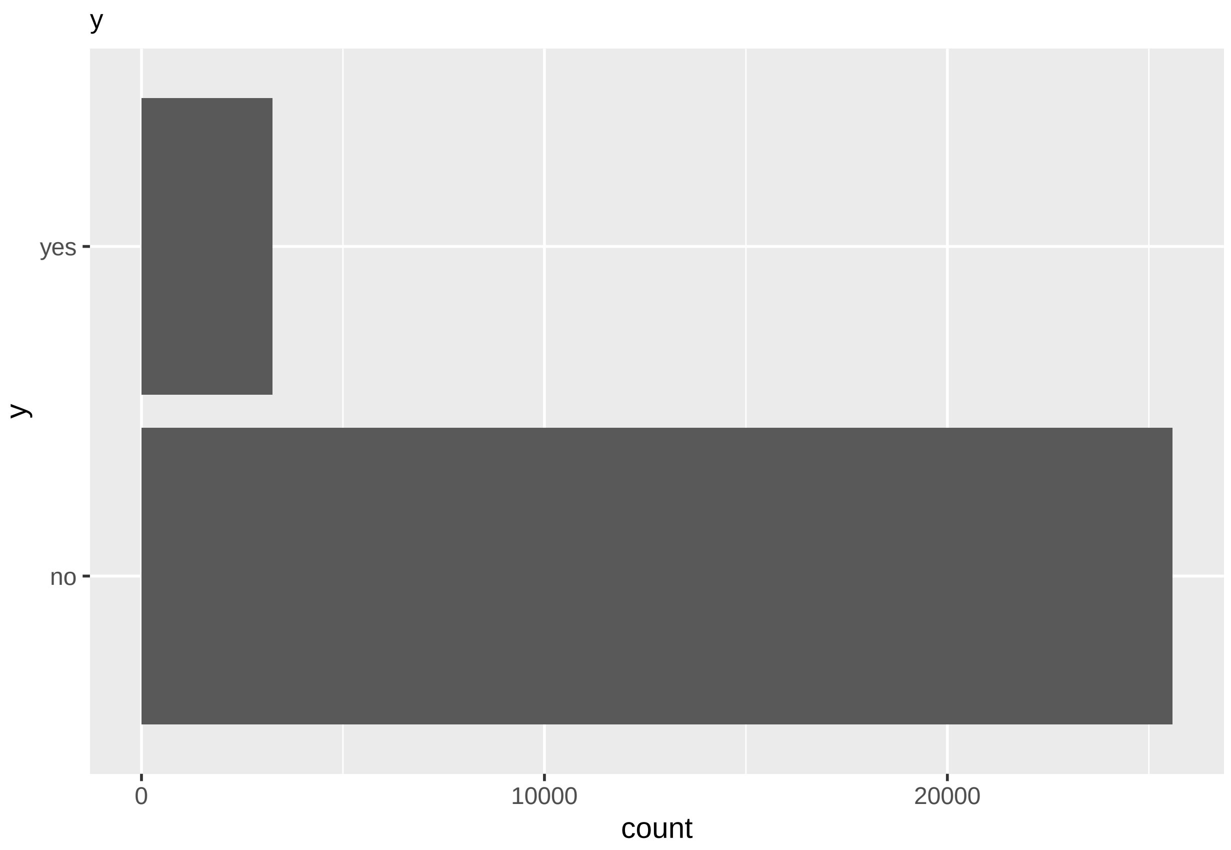
```
varimp <- data.frame( Overall = as.data.frame(filterVarImp(train_set[,xcols],as.double(train_set$y)))$Ov

varimp %>% mutate(xcols=reorder(xcols,(Overall)),color = ifelse(Overall>5,"important","not important"))%
```



Prevalence

Prevalence describes the proportion of positive cases in the population. This plot shows the prevalence of the “yes” cases in the y variable of the dataset. As we can see the prevalence is quite low, because most people do not subscribe a term deposit. This means that simple overall accuracy is not good enough for evaluating the algorithm, since simply predicting the call to be answered with a “no” would result in a very high overall accuracy. Therefore we will use balanced accuracy when evaluating the algorithms later, because it combines both sensitivity and specificity in a single number.

```r
y_plot<- train_set %>%
  group_by(y) %>%
    summarise(count = n()) %>%
  mutate(y=reorder(y,(-count)))%>%
    ggplot(aes(x =count , y =y)) +
    geom_bar(stat = 'identity')+ggtitle("y")+
  theme(plot.title = element_text(size=10))
y_plot
```



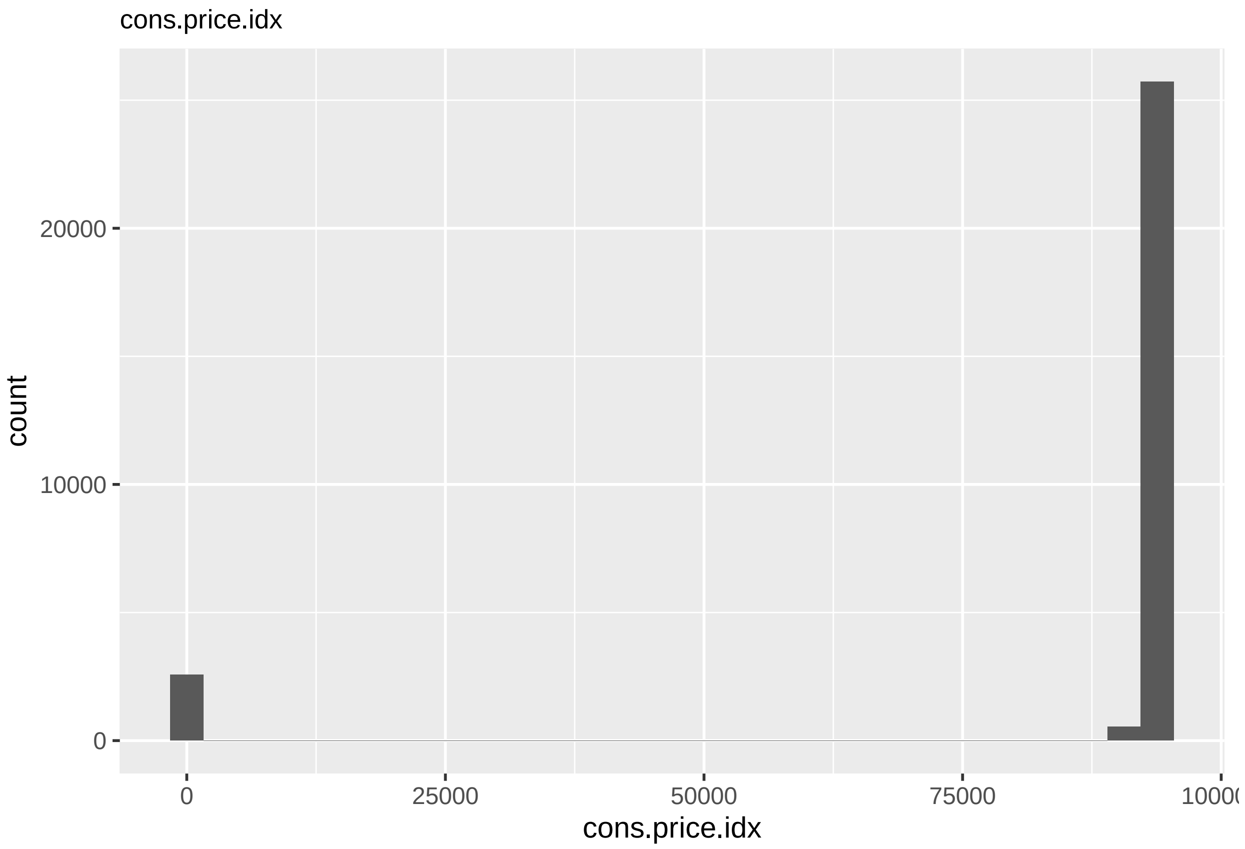
Vizualisation of features

Next we will try to explore if the features have an effect on the outcome, which we will try to predict in the modeling phase.

Conumer price index plot

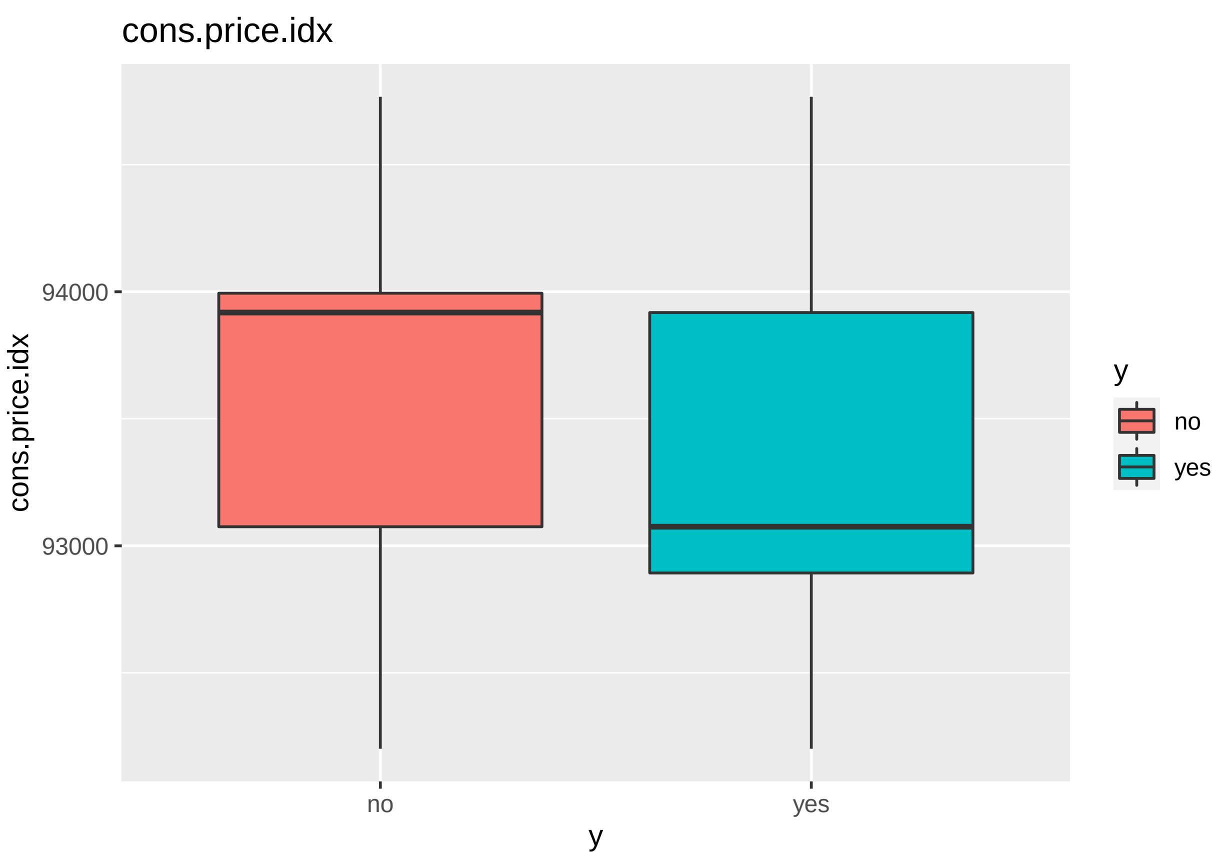
The Histogram bellow shows the number of appearances of the cons.price.idx feature in the data set. We can clearly see that almost all of the calls were made when the consumer price index was above 85000.

```
cons.price.idx_plot<- train_set %>% ggplot(aes(x=cons.price.idx))+geom_histogram()+ggtitle("cons.price.i
cons.price.idx_plot
```



The plot bellow is a Boxplot of the consumer price index that shows the difference between the consumer price index of successful calls and unsuccessful calls. It looks like there is no big difference between calls that were answered with no or with yes. But calls answered with “yes” still tend to have a lower con.price.idx.

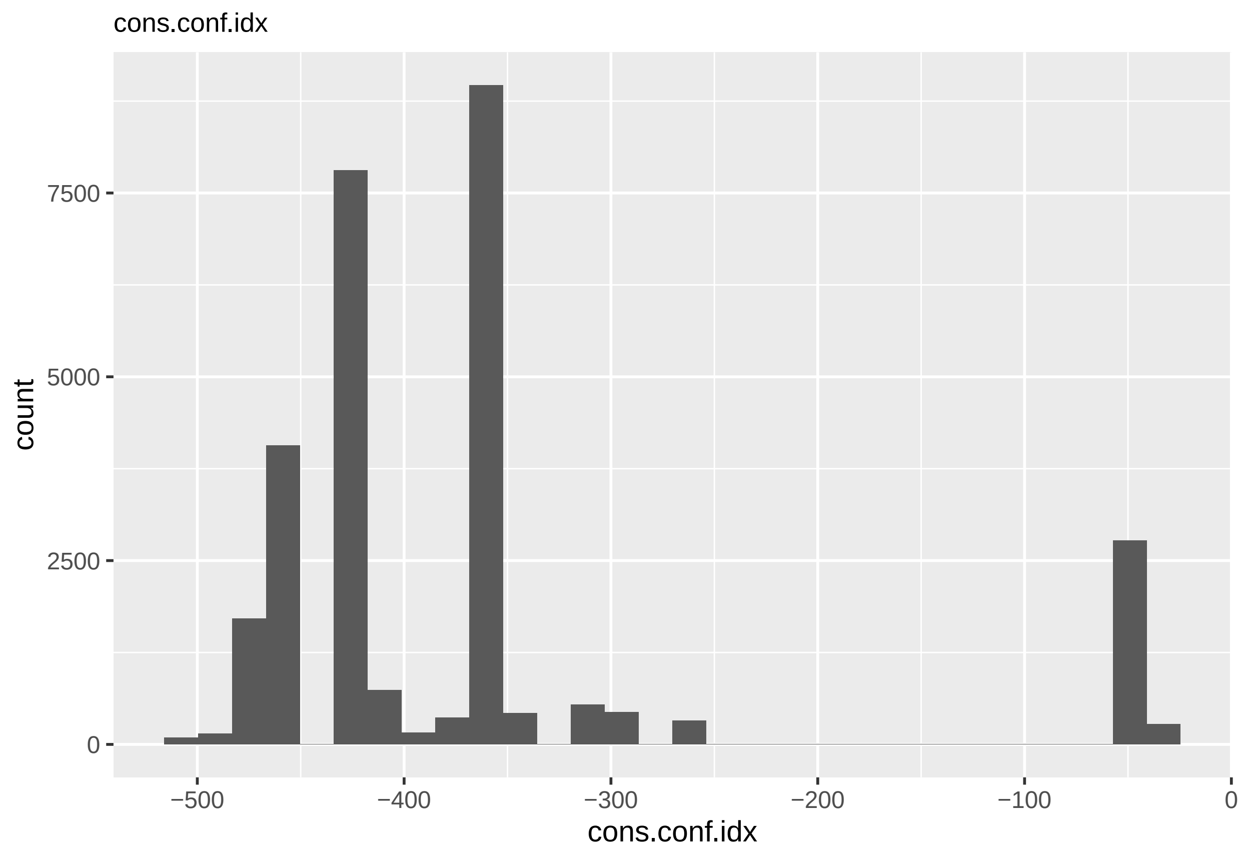
```
cons.price.idx_rate_plot<- train_set %>% filter(cons.price.idx>1000) %>% ggplot(aes(x=cons.price.idx,y=y
  geom_boxplot()+coord_flip()+ggtitle("cons.price.idx")
cons.price.idx_rate_plot
```



Consumer Confidence Index Plot

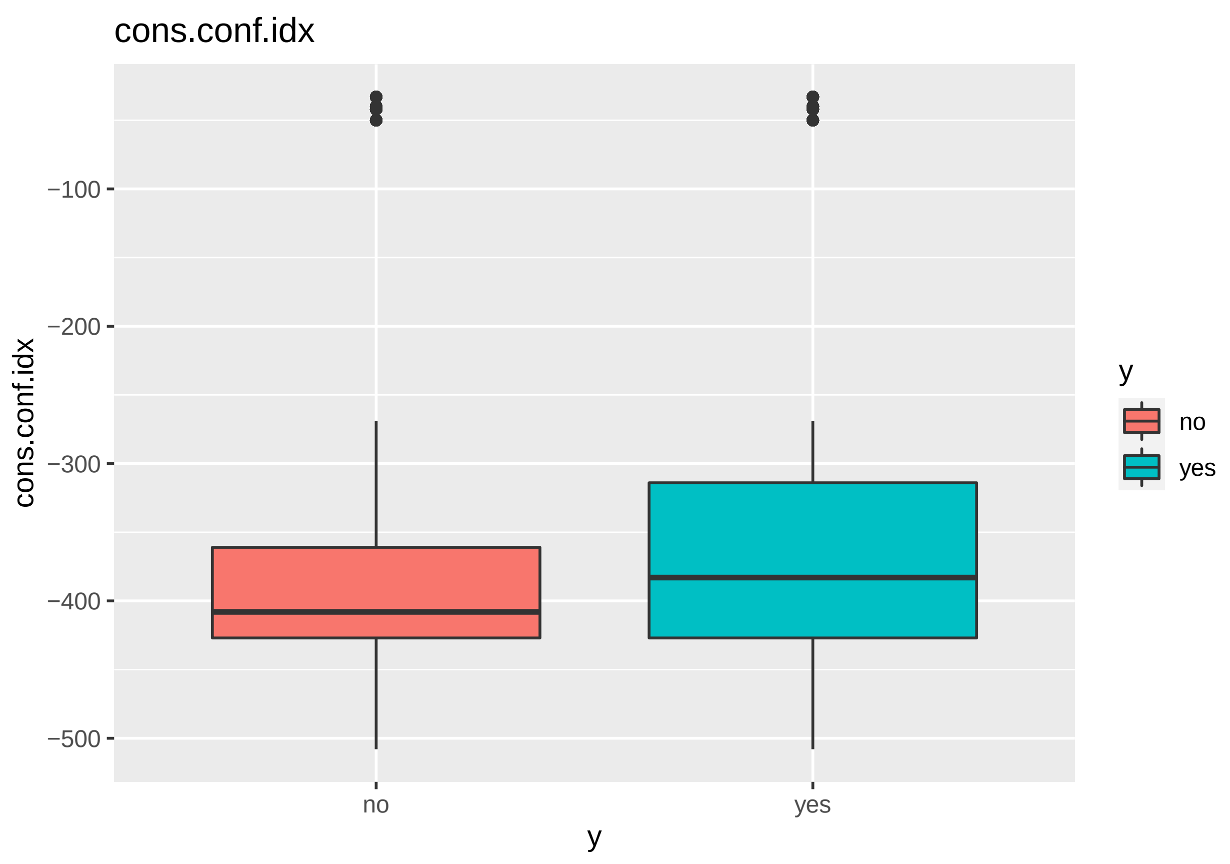
The next histogram shows the appearance distribution of the cons.cong.idx variable. We can see that almost all of the cons.conf.idx data is between -500 and - 300.

```
cons.conf.idx_plot<- train_set %>% ggplot(aes(x=cons.conf.idx))+geom_histogram()+ggtitle("cons.conf.idx
cons.conf.idx_plot
```



The Next plot shows the cons.conf.idx variable and displays the difference of the consumer confidence index when a call was successful to the consumer confidence index when the call was unsuccessful. As we can see, there isn't a a big difference, but calls that were successful tended to have a higher consumer confidence index.

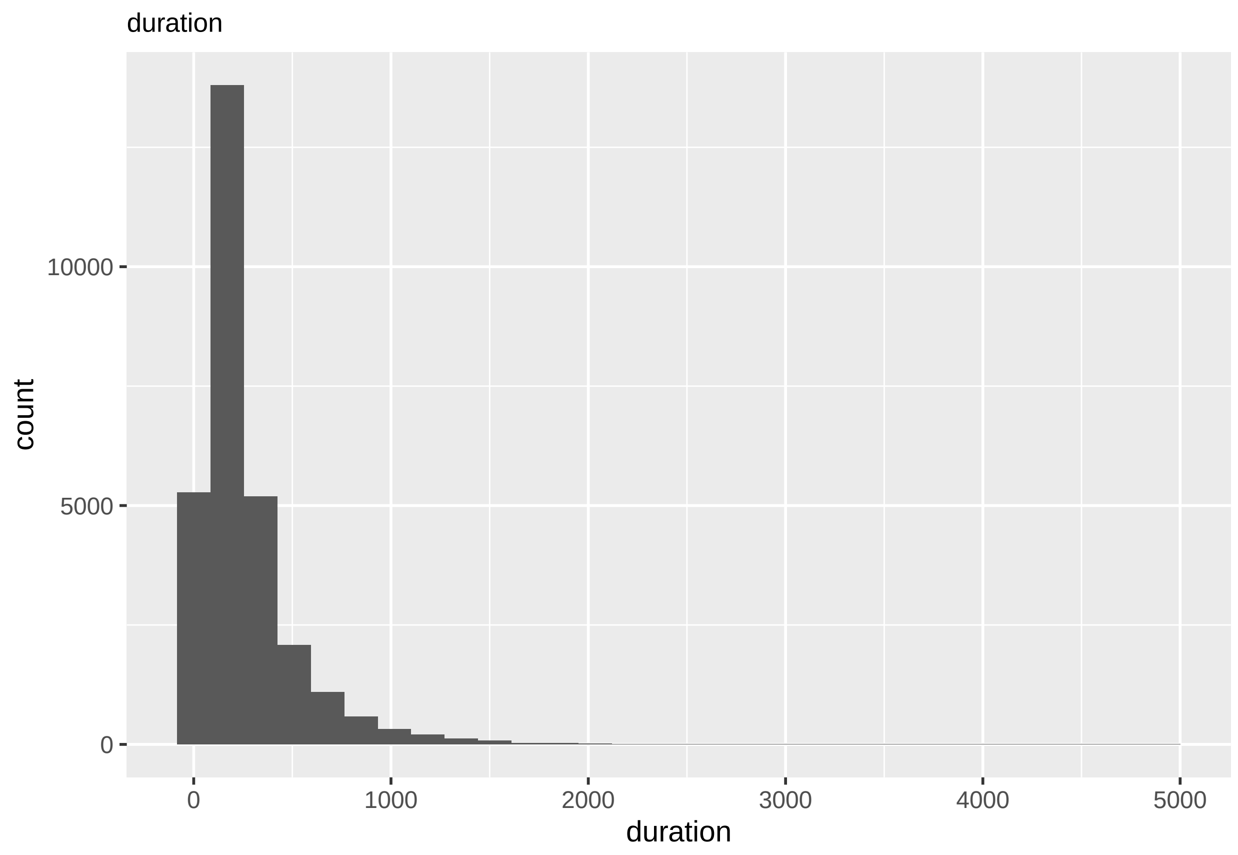
```
cons.conf.idx_rate_plot<- data %>% ggplot(aes(x=cons.conf.idx,y=y,fill=y)) +
  geom_boxplot()+ggtitle("cons.conf.idx")+coord_flip()
cons.conf.idx_rate_plot
```



Duration plot

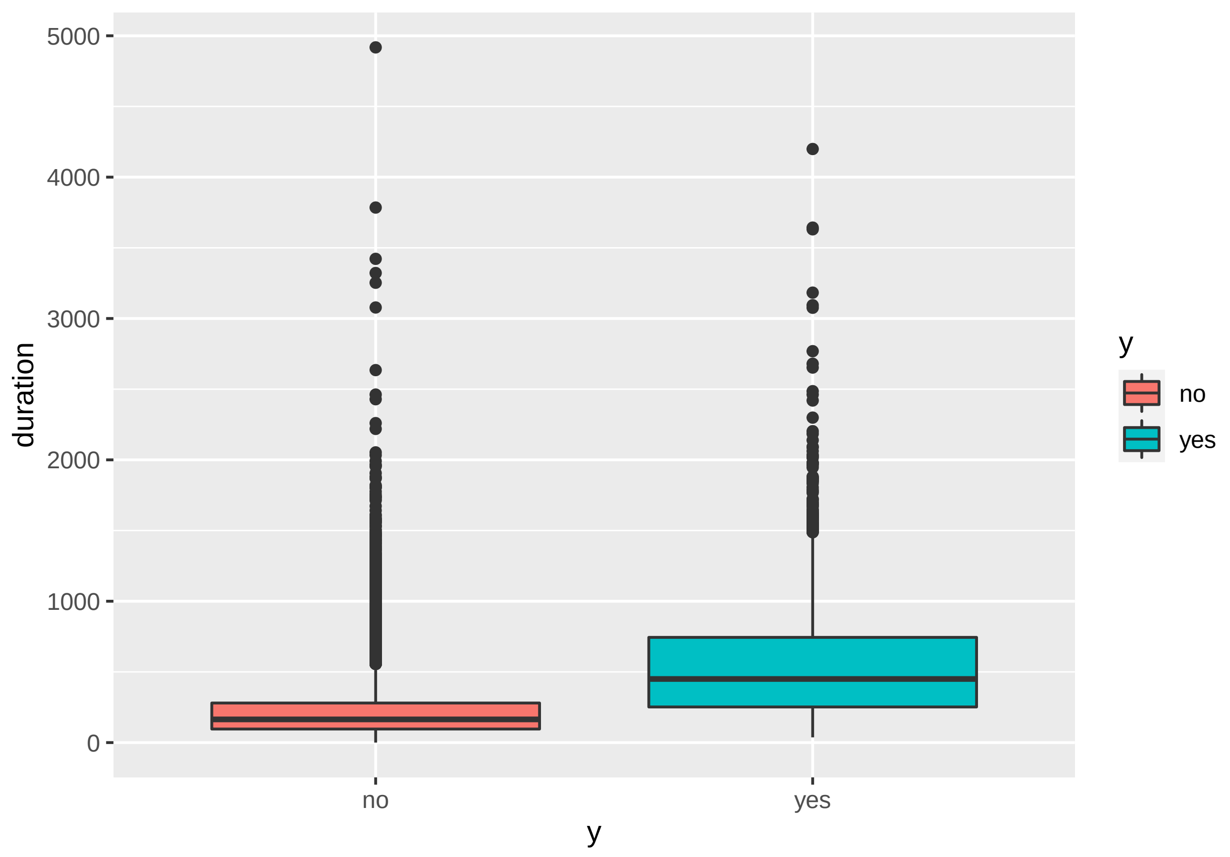
The plot displays the distribution of appearances of the different duration values in the data set. As we can see in the plot most calls are shorter than 1000 seconds.

```
duration_plot <- train_set %>% ggplot(aes(x=duration))+geom_histogram()+ggtitle("duration")+theme(plot.
duration_plot
```



The following plot looks at the duration of the calls made. This variable can't be used for predictions, because before a call we don't know the how long a call will be, before the call is made. We see that calls that were answered with "yes" tend to have a longer duration.

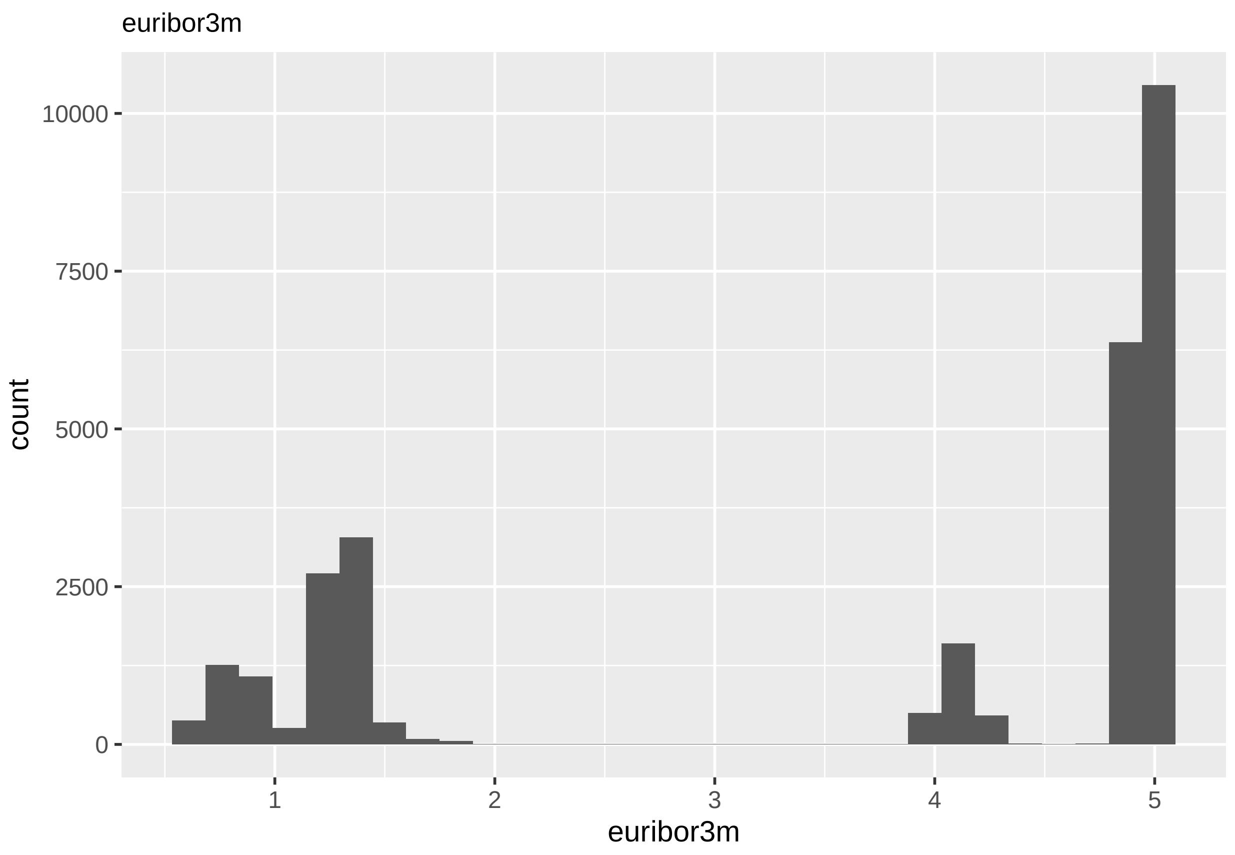
```
duration_rate_plot<-train_set  %>% ggplot(aes(x=duration,y=y,fill=y)) +
  geom_boxplot()+coord_flip()
duration_rate_plot
```



Euribor3m plot

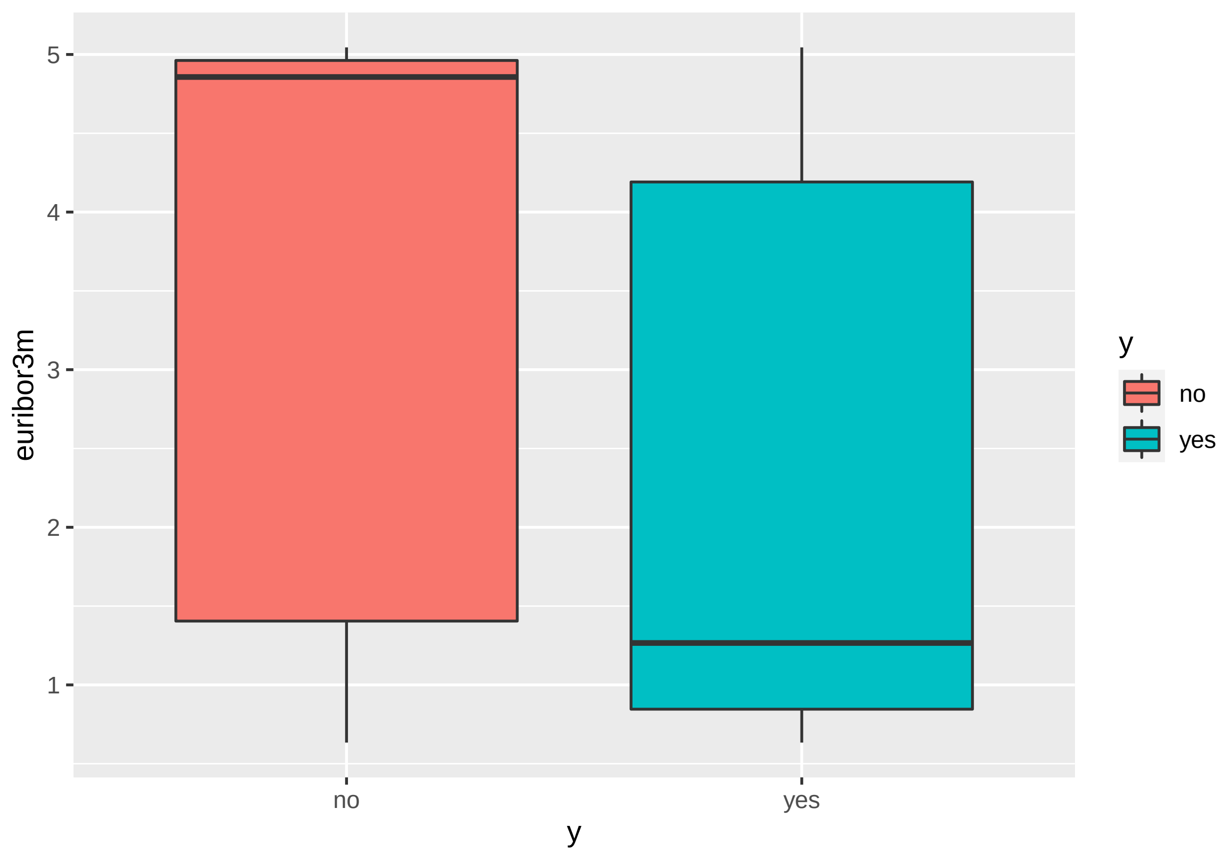
The next histogram shows the appearance distribution of the euribor3m variable. Here we can see that the mods calls were made when the euribor3m was at 5.

```
euribor3m_plot <- train_set %>%  ggplot(aes(x=euribor3m))+geom_histogram()+ggtitle("euribor3m") + theme
euribor3m_plot
```



The Euribor3m-effect plot shows the difference of the euribor3m values of successful calls and unsuccessful calls. We see that the output variable was more often "yes" when the euribor3m was low.

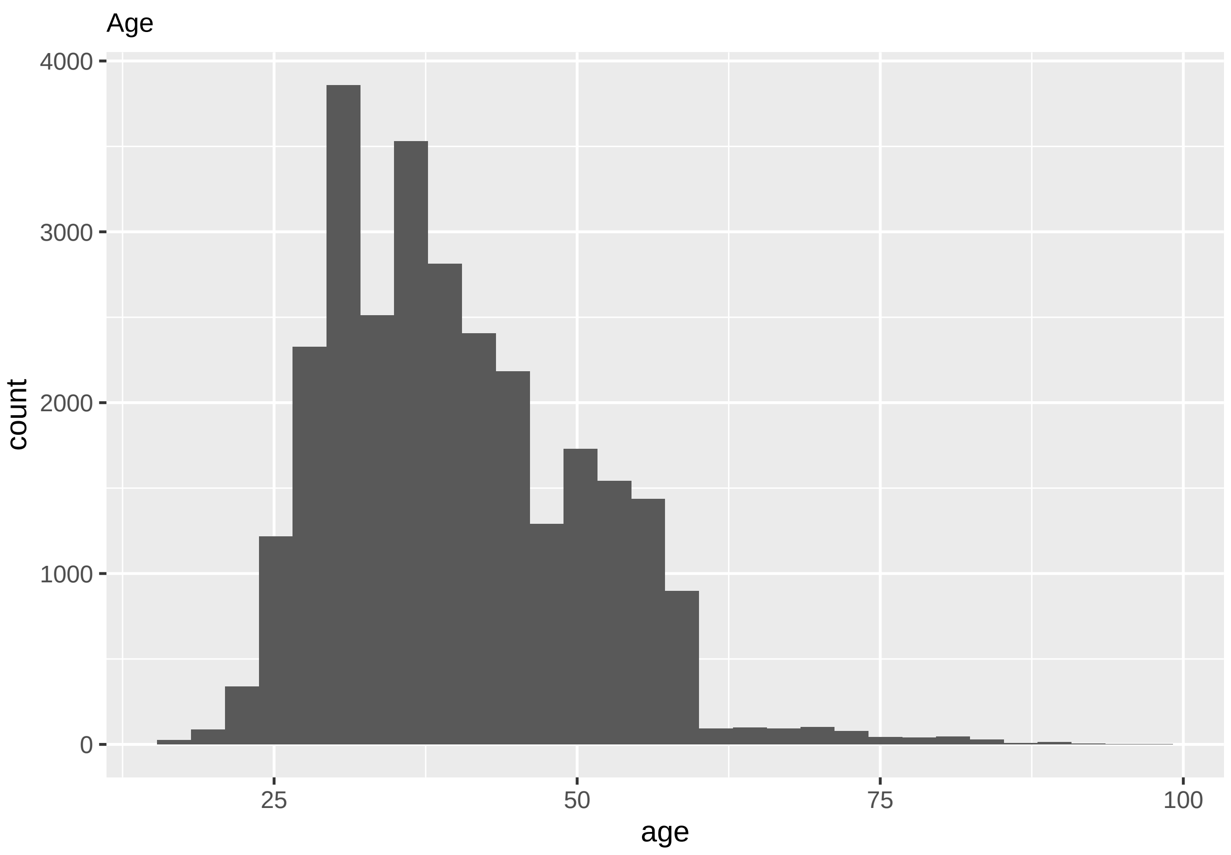
```
train_set  %>% ggplot(aes(y=euribor3m,x=y,fill=y)) +
  geom_boxplot()
```



Age Plot

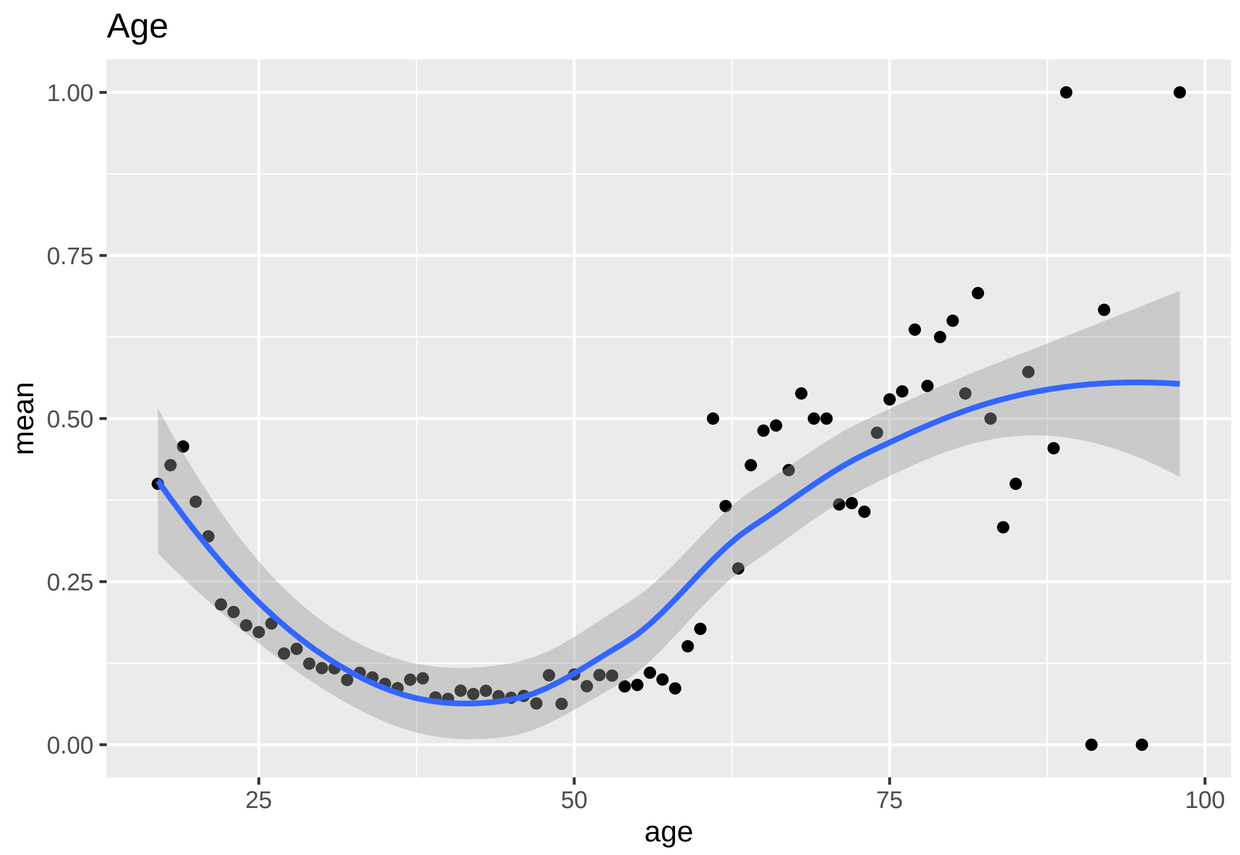
The plot shows the distribution of appearances of the different ages in the data set. Most of the clients in the data set is apparently between the age of 25 and 60.

```
age_plot <- train_set %>% ggplot(aes(x=age))+geom_histogram()+ggtitle("Age") + theme(plot.title = elemen
age_plot
```



The following plot compares the success rate of all ages. The x-axis displays the age and the y-axis the average success rate for the given age.We can see that very young clients and very old clients are more likely to subscribe to the term deposits. At the age most people have a job the success rate of the calls is lower. When looking at the plot, it seems like age could be a good predictor, but we saw that these age groups, that have a very high success rate, donot appear as often in the data.

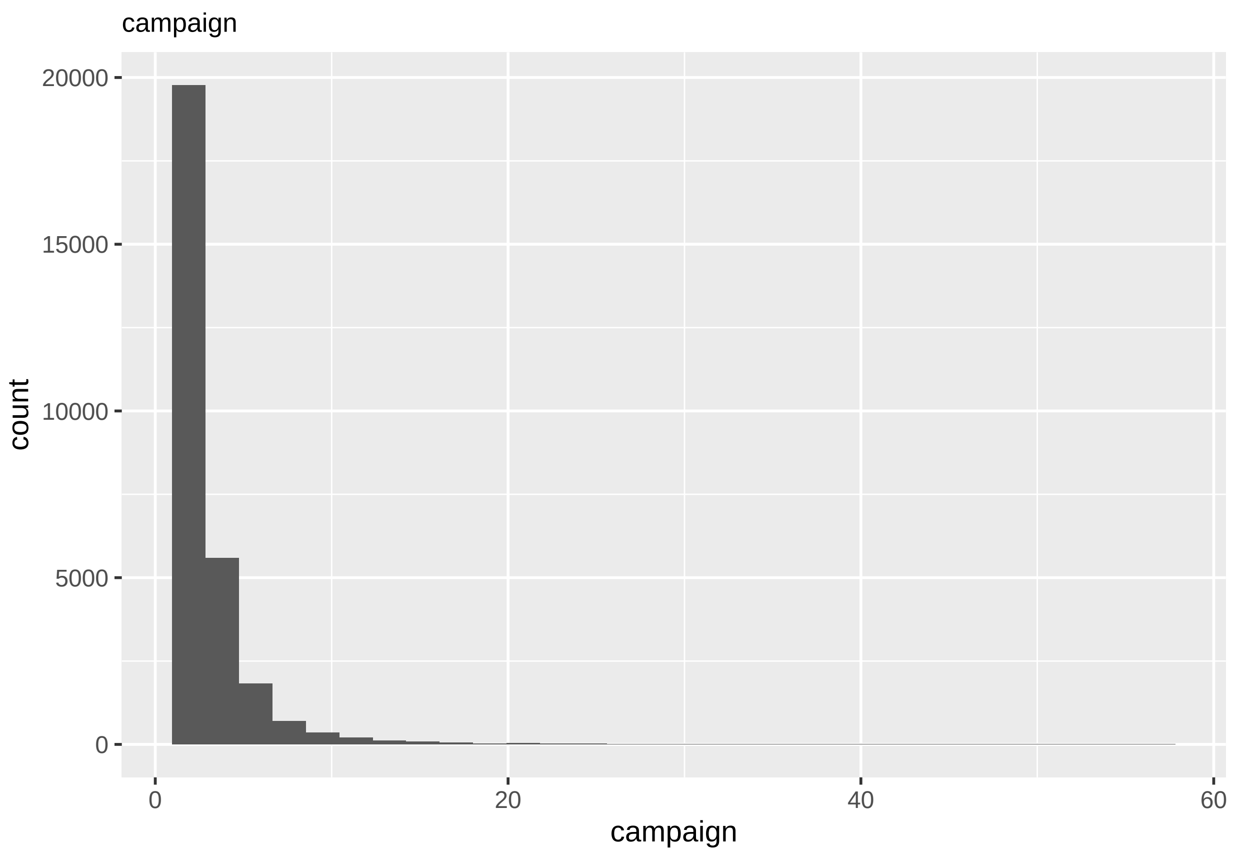
```
train_set$y <- ifelse(train_set$y=="yes",1,0)
test_set$y <- ifelse(test_set$y=="yes",1,0)
age_rate_plot<- train_set %>%   group_by(age) %>% summarise(mean=mean(y))  %>% ggplot(aes(x=age,y=mean)
age_rate_plot
```



Campaign plot

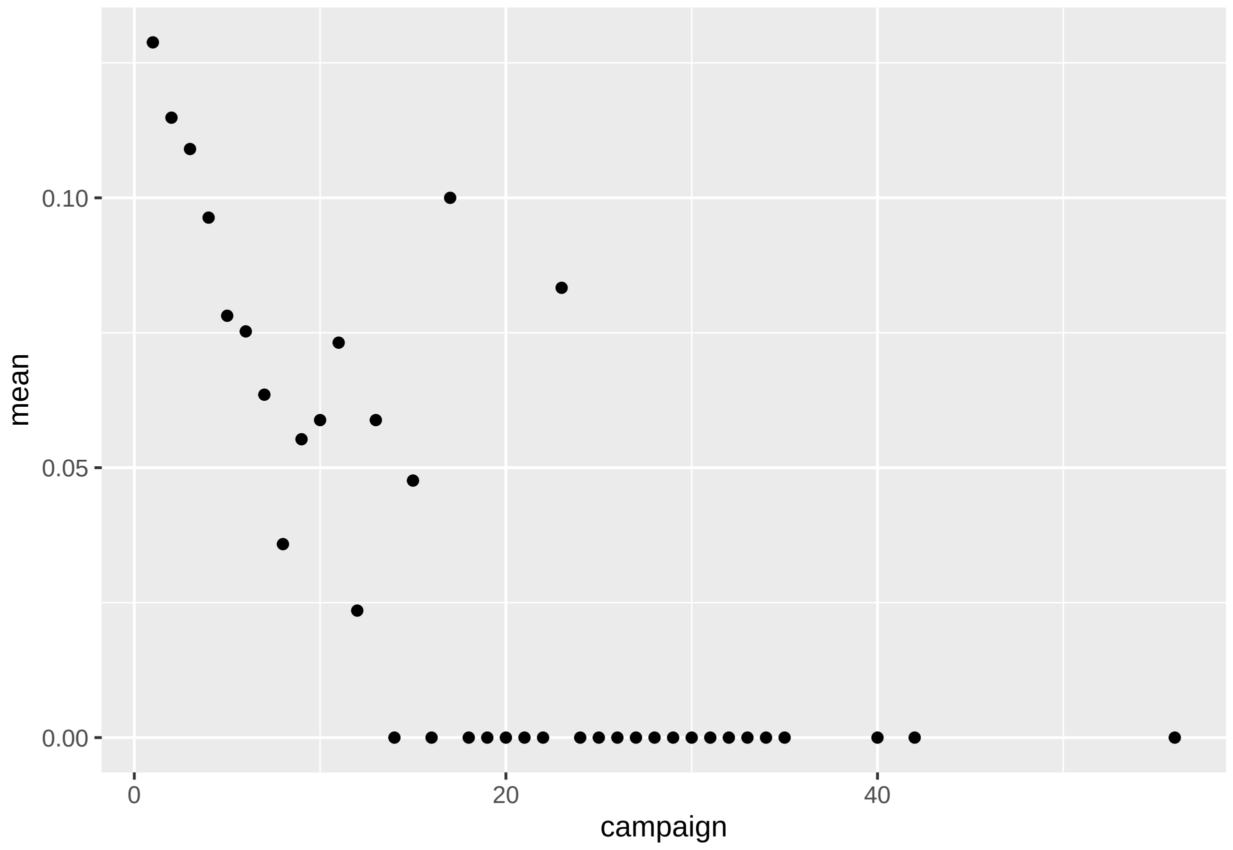
The Histogram bellow shows the number of appearances of the campaign feature in the dataset.

```
campaign_plot<- train_set %>% ggplot(aes(x=campaign))+geom_histogram()+ggtitle("campaign")+theme(plot.t
campaign_plot
```



This graph shows the effect the number of contact that were made during the campaign has on the success rate. The x-axis shows the number of contacts that were made during the campaign from one client and the y-axis shows the average of y being "yes". The graph shows that the more contacts were made the less probable it is that the variable y is yes and the graph also reveals that if the number of contacts is above 16 the chance is nearly zero.

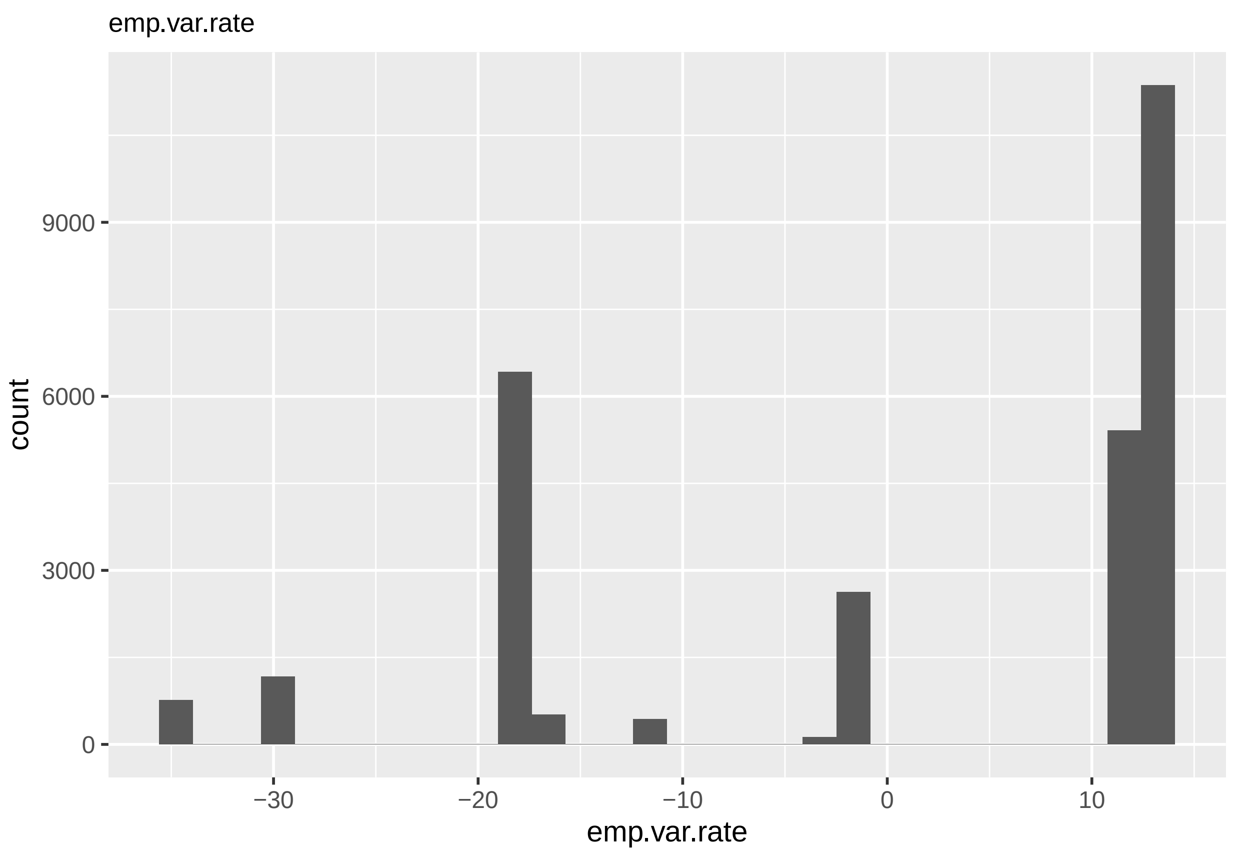
```
campaign_rate_plot<- train_set %>%group_by(campaign) %>% summarise(mean=mean(y))  %>% ggplot(aes(x=campa
campaign_rate_plot
```



emp.var.rate plot

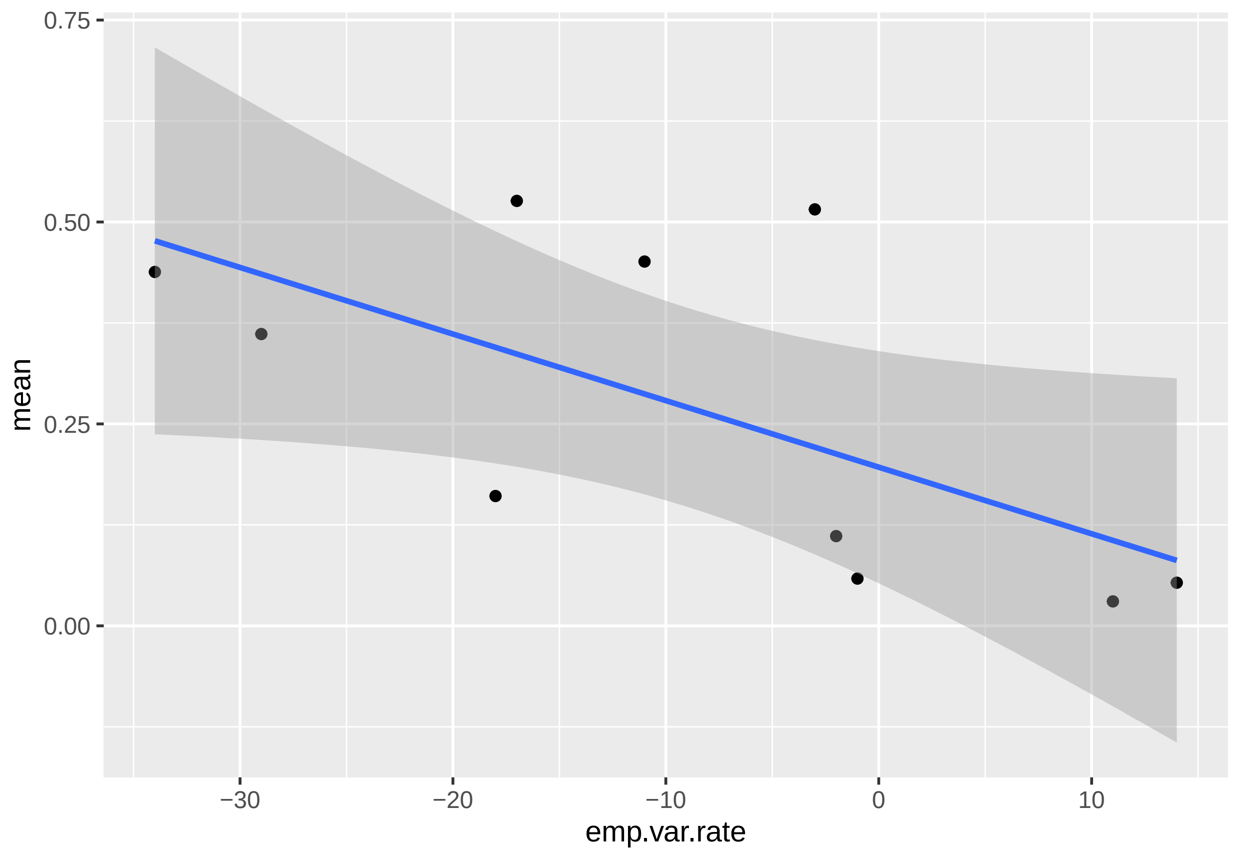
This Histogram shows the number of appearances of the emp.var.rate feature in the dataset.

```
emp.var.rate_plot<- train_set %>% ggplot(aes(x=emp.var.rate))+geom_histogram()+ggtitle("emp.var.rate")+
emp.var.rate_plot
```



The next plot compares the success rate of the emp.var.rate with the emp.var.rate. The x-axis shows the emp.var.rate and the y-axis shows the average of the output variable being equal to "yes" for each of the emp.var.rate values. By interpreting the plot, we can see that there is a weak negative linear correlation between the two variables.

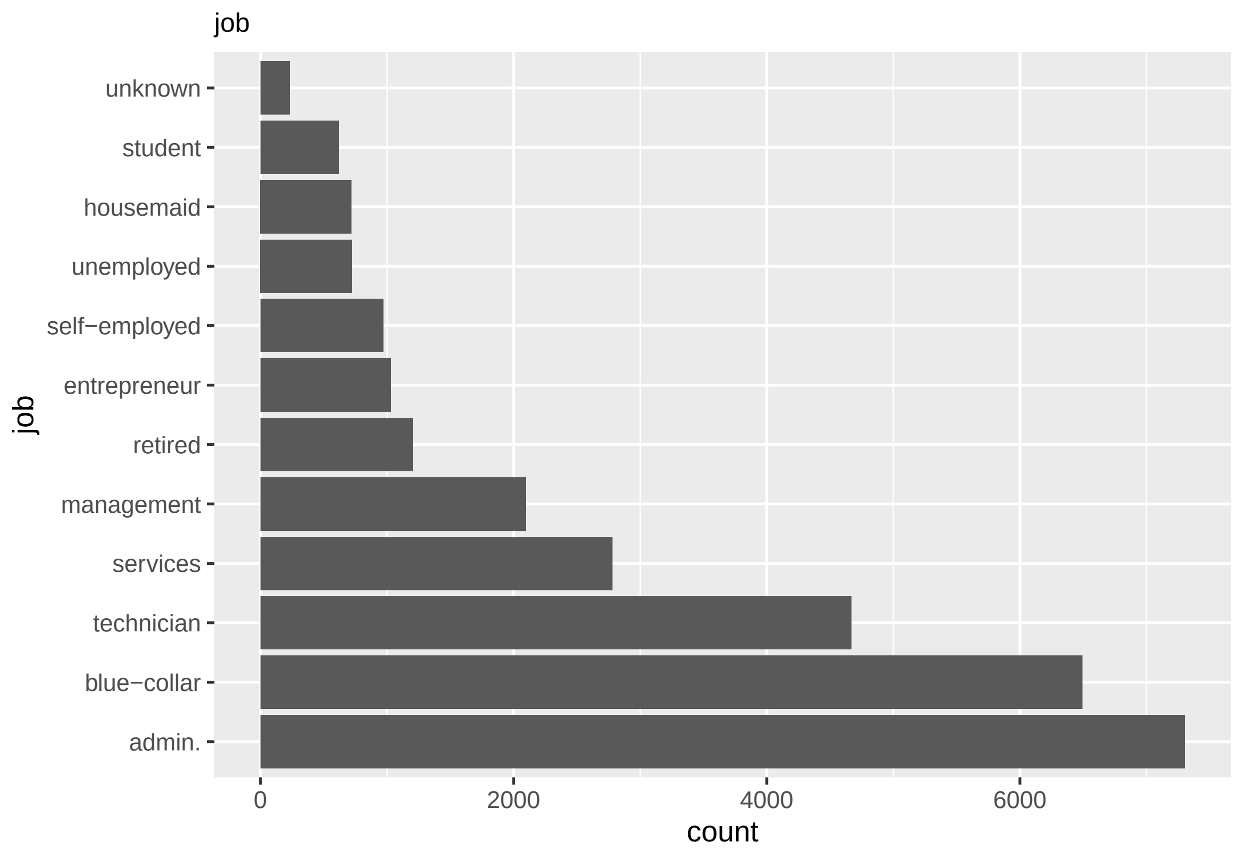
```
emp.var.rate_rate_plot<- train_set %>%
  group_by(emp.var.rate) %>%
    summarise(mean=mean(y)) %>%
  ggplot(aes(x=emp.var.rate,y=mean))+
  geom_point()+geom_smooth(method = "lm")
emp.var.rate_rate_plot
```



Job Plot

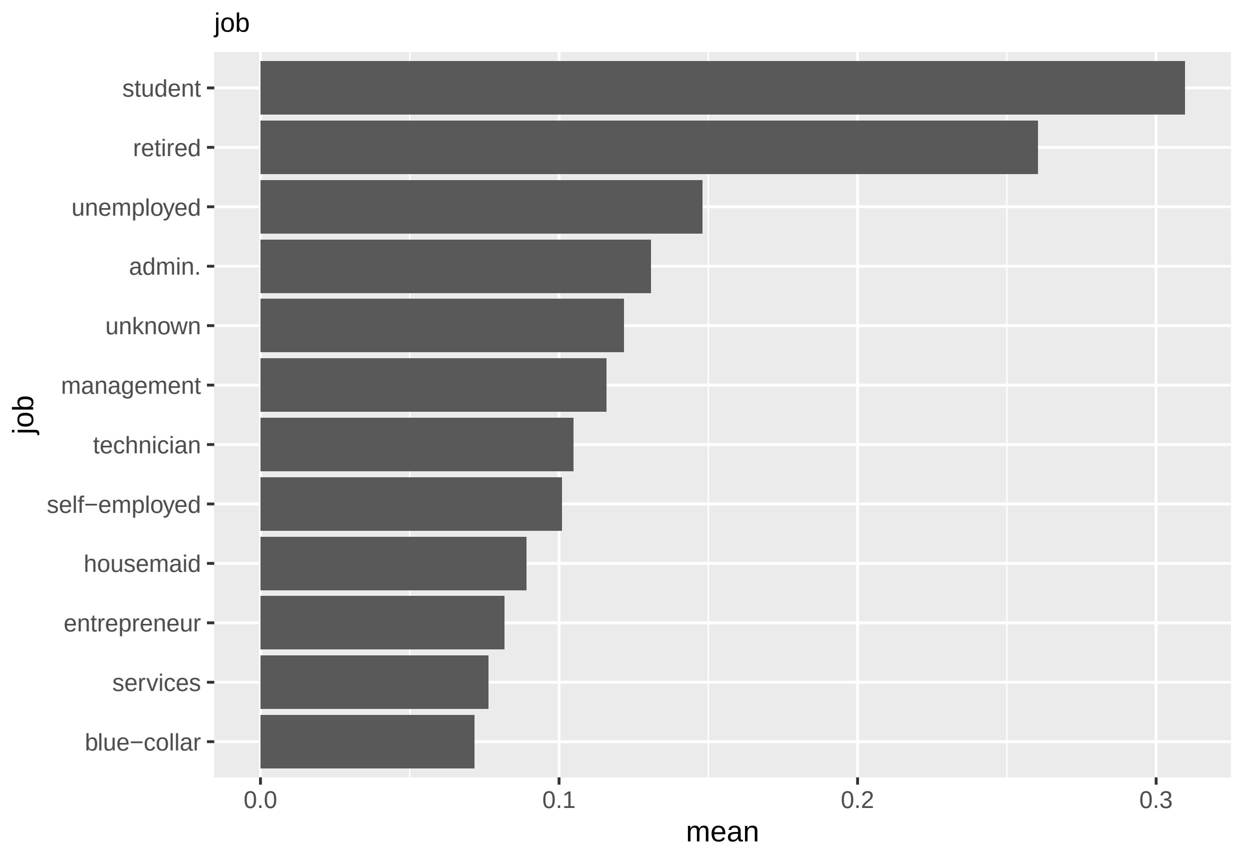
The next barplot shows the appearance for each category of the job variable. This plot shows that most people in our data set work in administration or blue-collar. We can also see that student and retired aren't a lot of people, this confirms what we saw when interpreting the age plot.

```
job_plot <- train_set %>%
  group_by(job) %>%
    summarise(count = n()) %>%
  mutate(job=reorder(job,(-count)))%>%
    ggplot(aes(x =count , y = job)) +
    geom_bar(stat = 'identity')+ggtitle("job")+theme(plot.title = element_text(size=10))
job_plot
```



The plot bellow shows the average of y being equal to "yes" for each of the categories of the job variable. As shown in the plot, students and retired clients are most likely to subscribe to a term deposit. This seems to be very similar to what we found out from the Age-effect plot.

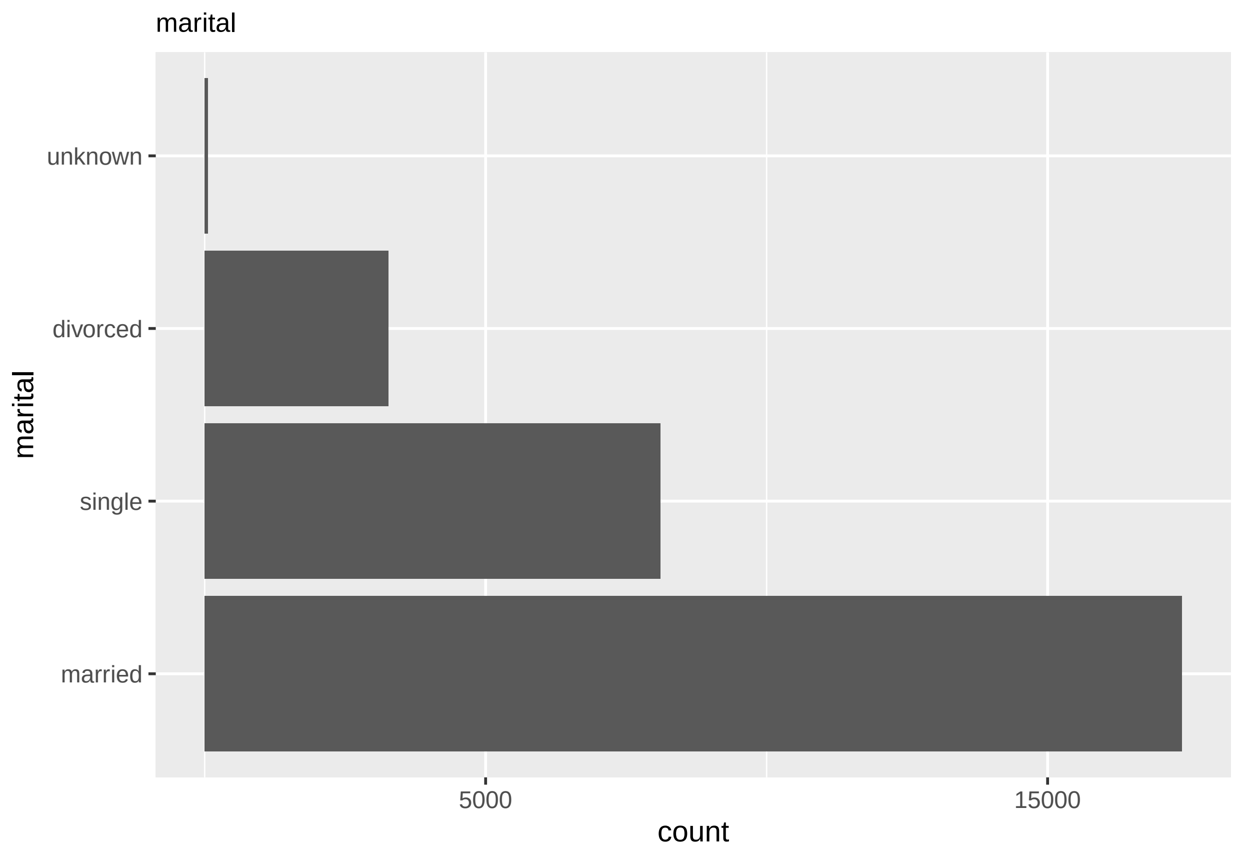
```
job_rate_plot <- train_set %>%
  group_by(job) %>%
    summarise(mean=mean(y)) %>%
  mutate(job=reorder(job,(mean)))%>%
    ggplot(aes(x =mean , y = job),size=10) +
    geom_bar(stat = 'identity')+ggtitle("job")+theme(plot.title = element_text(size=10))
job_rate_plot
```



marital plot

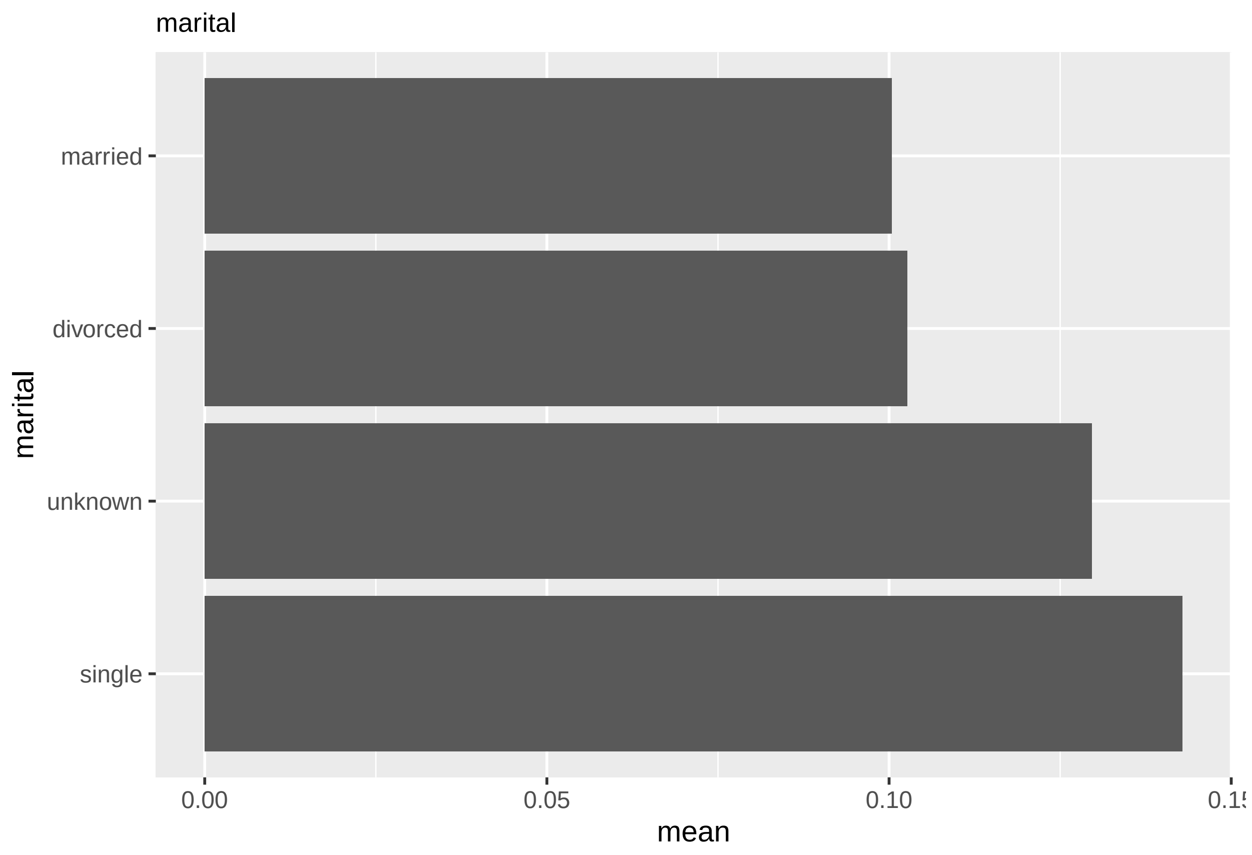
The next Barplot shows the appearance for each category of the marital variable. This shows that by far the most clients that were called were married. The least represented marital status in the dataset is divorced.

```
marital_plot <- train_set %>%
  group_by(marital) %>%
    summarise(count = n()) %>%
  mutate(marital=reorder(marital,(-count)))%>%
    ggplot(aes(x =count , y = marital)) +
    geom_bar(stat = 'identity')+ggtitle("marital")+
  theme(plot.title = element_text(size=10))+scale_x_continuous(breaks=c(5000,15000,25000))
marital_plot
```



The next plot compares the success rate of different categories of the marital variable. When interpreting the plot, it becomes clear that married and divorced clients are less likely to subscribe to the term deposits, than clients who are single.

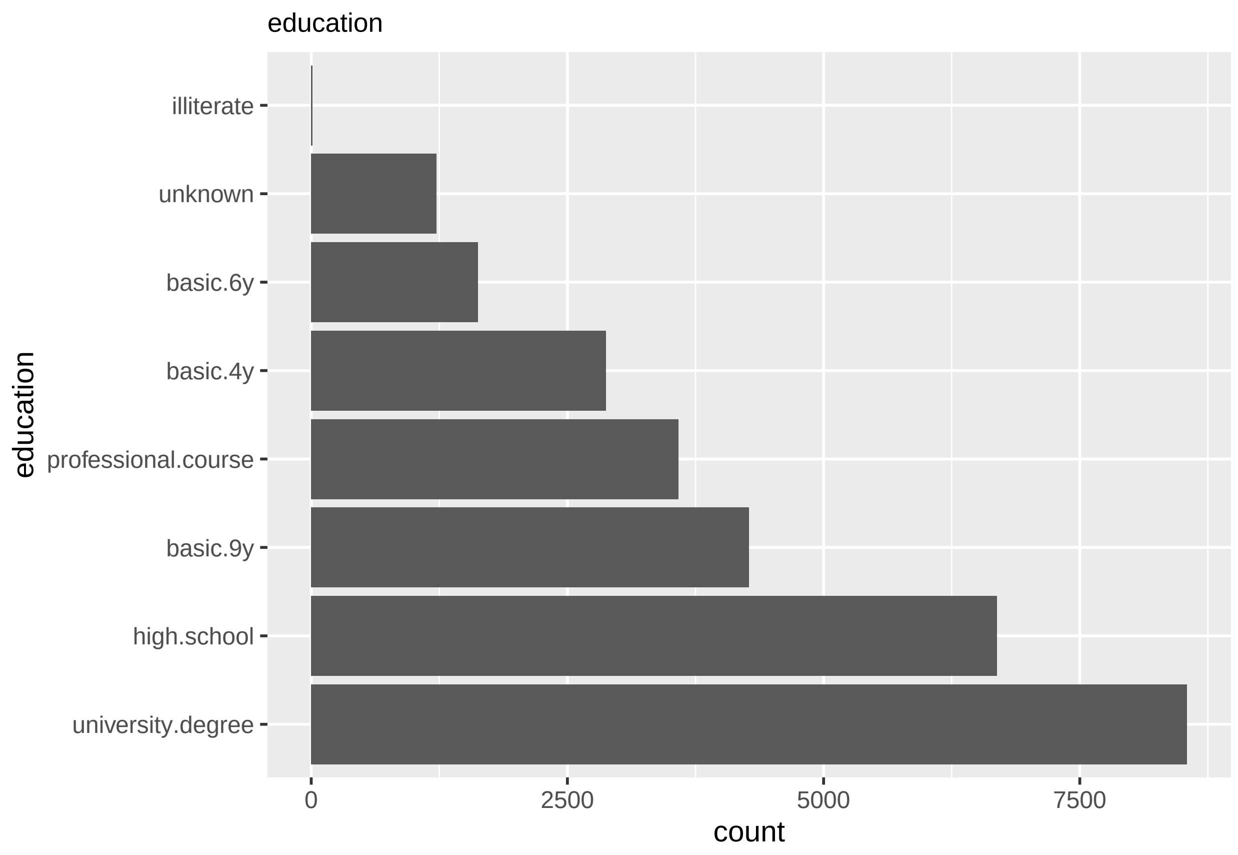
```
marital_rate_plot <- train_set %>%
  group_by(marital) %>%
    summarise(mean=mean(y)) %>%
  mutate(marital=reorder(marital,(-mean)))%>%
    ggplot(aes(x =mean , y = marital),size=10) +
    geom_bar(stat = 'identity')+ggtitle("marital")+theme(plot.title = element_text(size=10))
marital_rate_plot
```



education plot

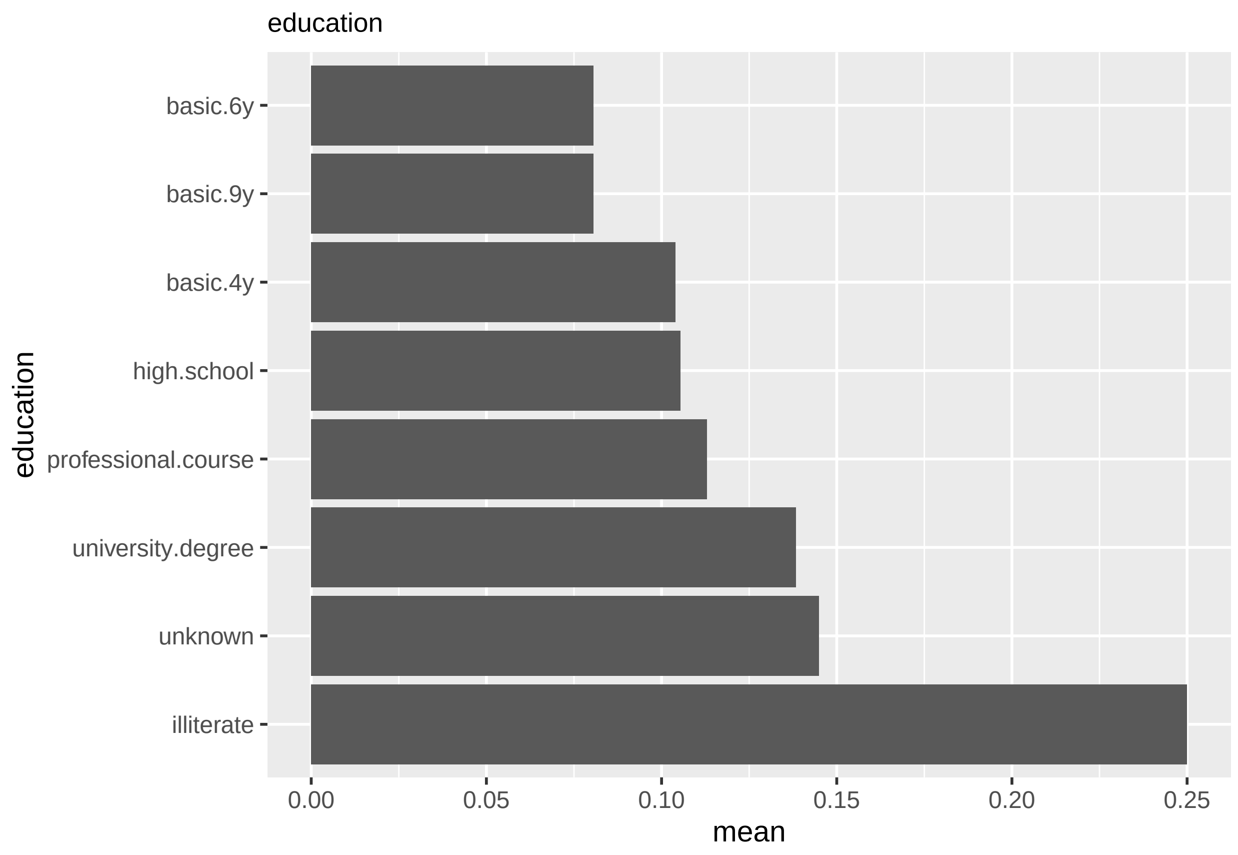
The next Barplot shows the appearance for each category of the education variable. We see that people with university degrees and High.school degrees are most represented in the data set. Illiterate people are almost not represented in the dataset at all.

```
education_plot <- train_set %>%
  group_by(education) %>%
    summarise(count = n()) %>%
  mutate(education=reorder(education,(-count)))%>%
    ggplot(aes(x =count , y = education)) +
    geom_bar(stat = 'identity')+ggtitle("education")+
  theme(plot.title = element_text(size=10))
education_plot
```



The plot bellow displays the effect the education feature has on the output variable. The plot compares seven different categories: basic.9y, basic.6y, basic.4y, high.school, professional.course, university.degree and illiterate. The basic.9y, basic.6y and basic.4y categories are the least probable to subscribe to a term deposit. The high.school and professional.course categories are a bit more likely to to subscribe to a term deposit. The university.degree and illiterate categories are by far the most likely to subscribe to the term deposits. Although we must keep in mind that there are almost no illiterate people in our data.

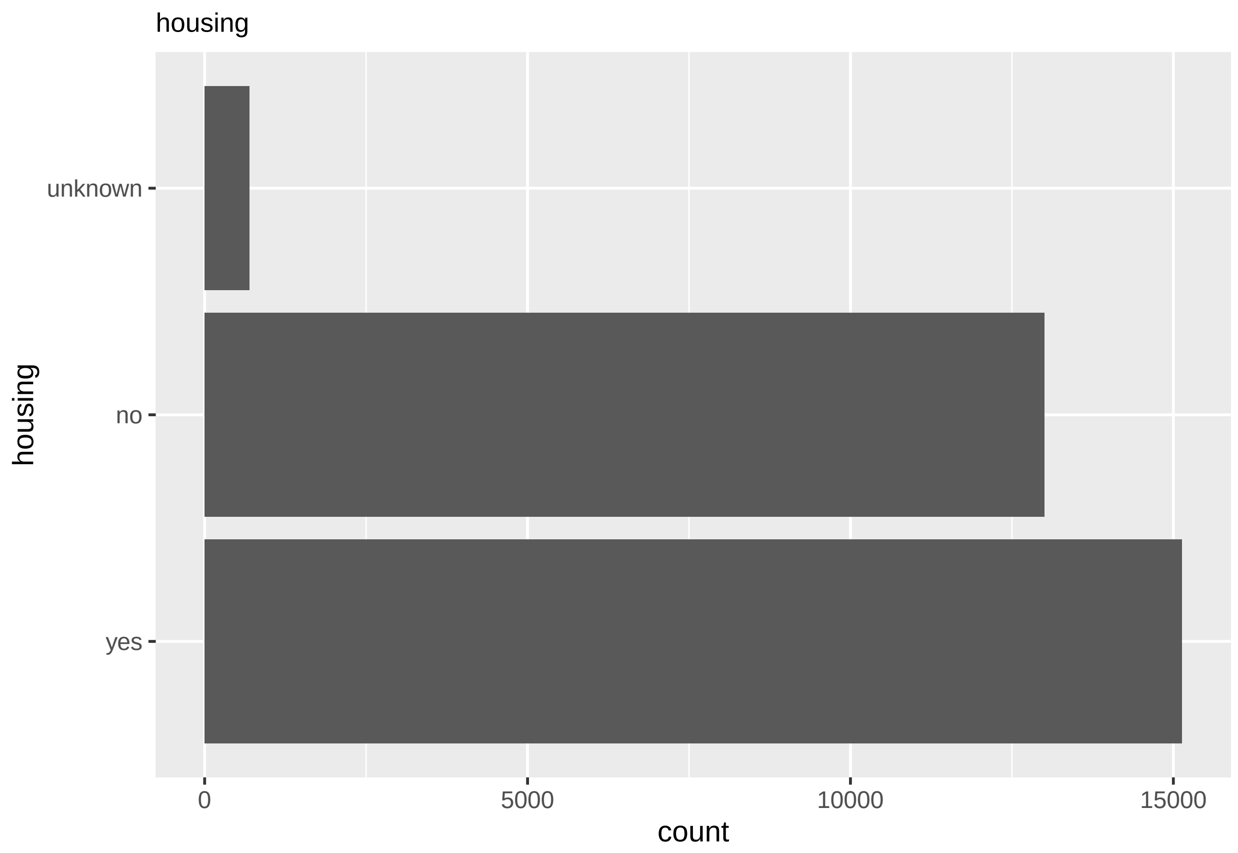
```
education_rate_plot <- train_set %>%
  group_by(education) %>%
    summarise(mean=mean(y)) %>%
  mutate(education=reorder(education,(-mean)))%>%
    ggplot(aes(x =mean , y = education),size=10) +
    geom_bar(stat = 'identity')+ggtitle("education")+theme(plot.title = element_text(size=10))
education_rate_plot
```



Housing plot

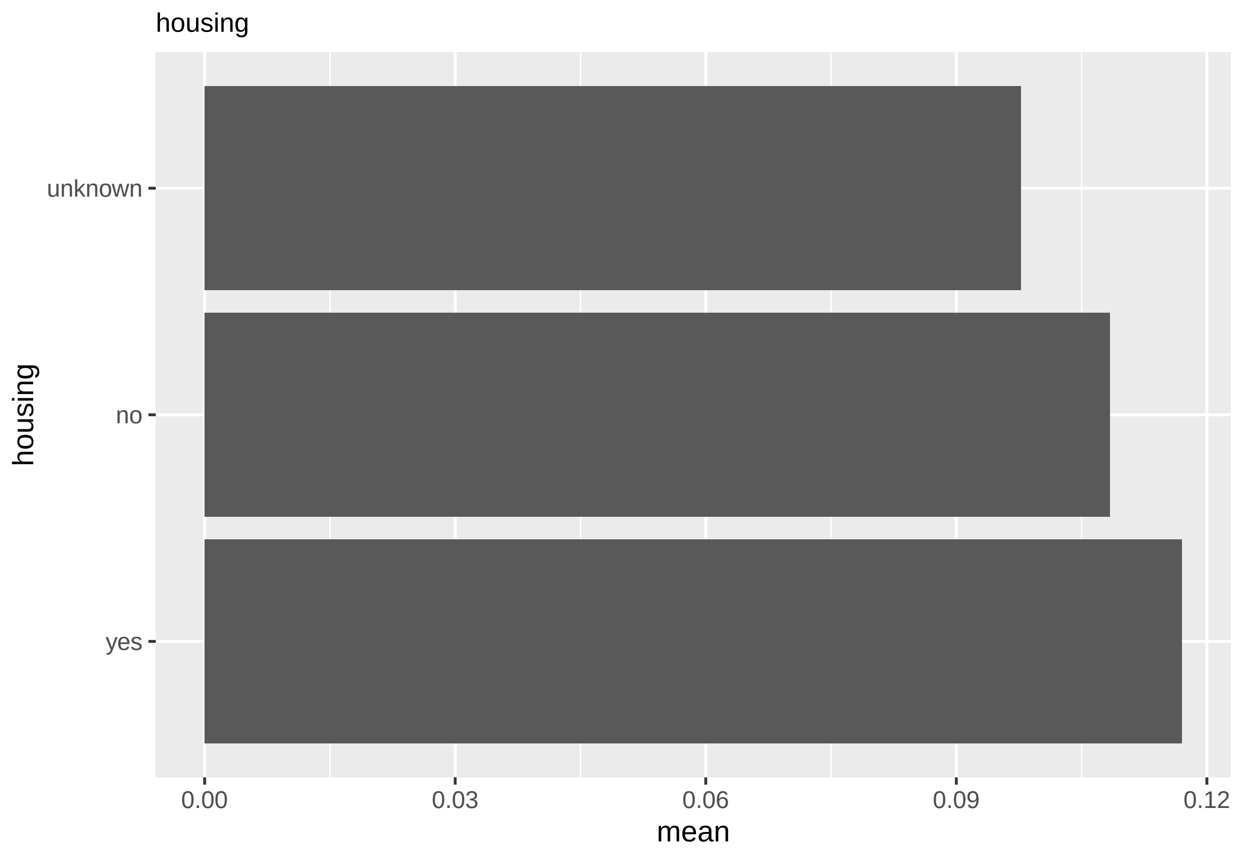
The next barplot shows the appearance for each category of the housing variable. This plot shows that although most people in the data set do have housing loans, but people without housing loans are also well represented.

```
housing_plot <- train_set %>%
  group_by(housing) %>%
    summarise(count = n()) %>%
  mutate(housing=reorder(housing,(-count)))%>%
    ggplot(aes(x =count , y = housing)) +
    geom_bar(stat = 'identity')+ggtitle("housing")+
  theme(plot.title = element_text(size=10))
housing_plot
```



This plot shows the difference of the success rate between people with a housing loan and people without a housing loan. As can be seen in the plot there is almost no difference between those two categories.

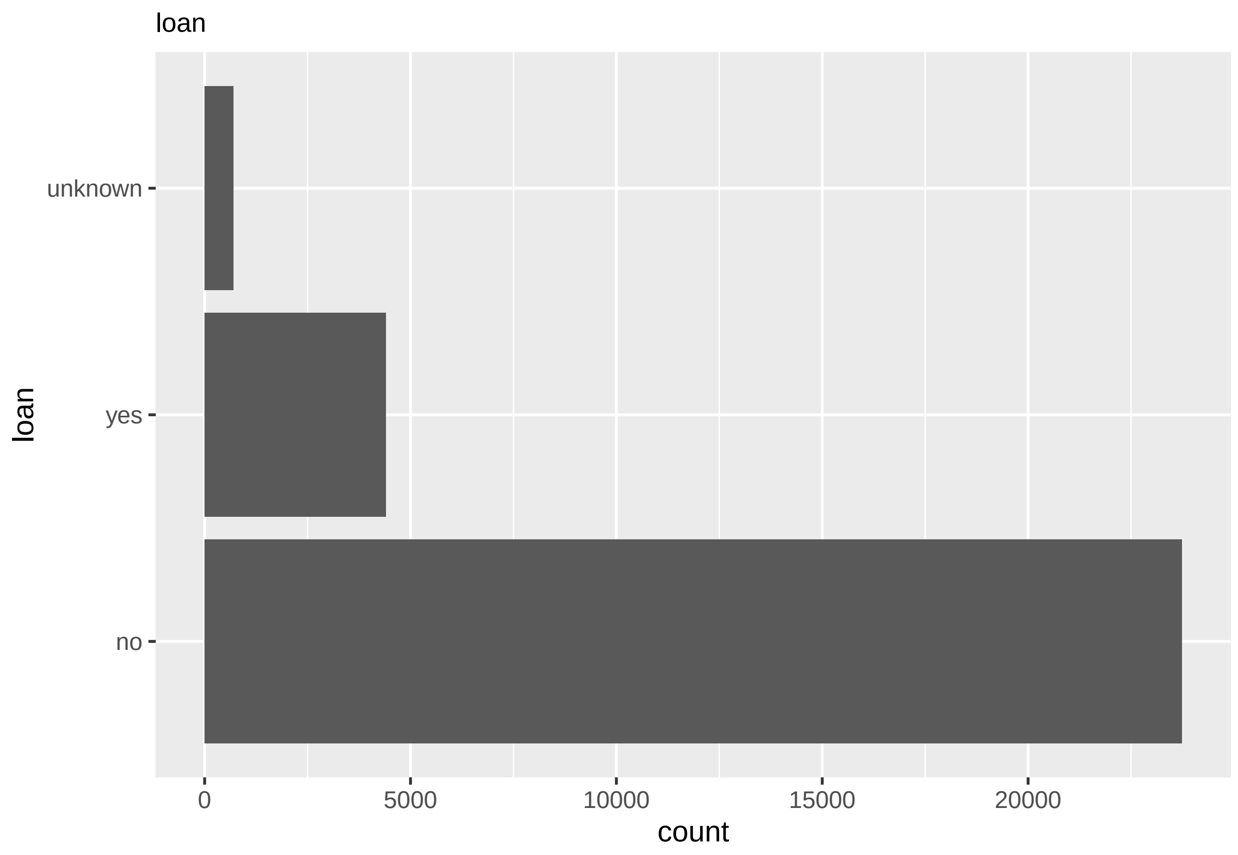
```
housing_rate_plot <- train_set %>%
  group_by(housing) %>%
    summarise(mean=mean(y)) %>%
  mutate(housing=reorder(housing,(-mean)))%>%
    ggplot(aes(x =mean , y = housing),size=10) +
    geom_bar(stat = 'identity')+ggtitle("housing")+theme(plot.title = element_text(size=10))
housing_rate_plot
```



loan plot

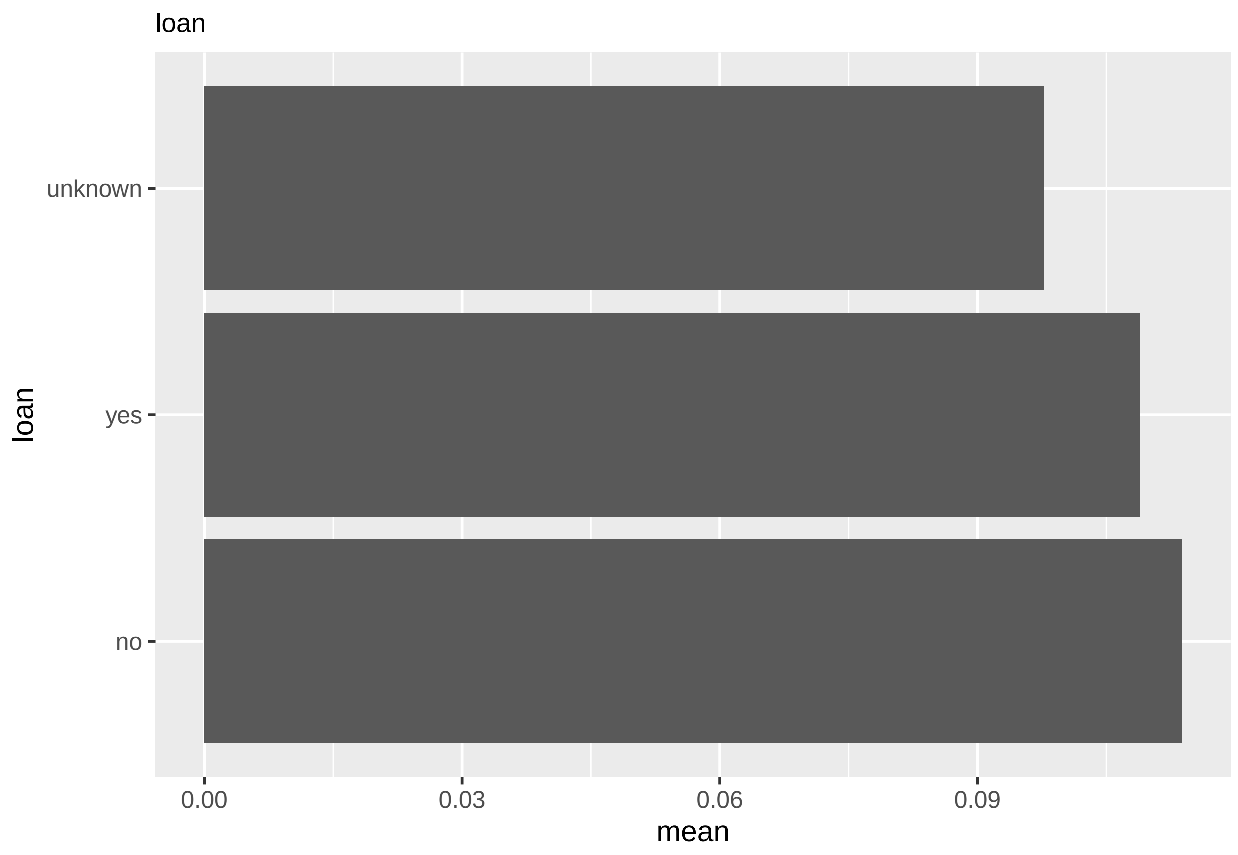
The next Barplot shows the appearance for each category of the loan variable. Contrary to what we found out about the housing loans, most people do not have any personal loan and people with personal loans are not well represented in the dataset.

```
loan_plot <- train_set %>%
  group_by(loan) %>%
    summarise(count = n()) %>%
  mutate(loan=reorder(loan,(-count)))%>%
    ggplot(aes(x =count , y = loan)) +
    geom_bar(stat = 'identity')+ggtitle("loan")+
  theme(plot.title = element_text(size=10))
loan_plot
```



The following plot compares the rate of successful calls of clients with a personal loan and clients without a personal loan. When looking at the plot we realize that like the housing loans the personal loans have almost no effect on the success rate of the calls.

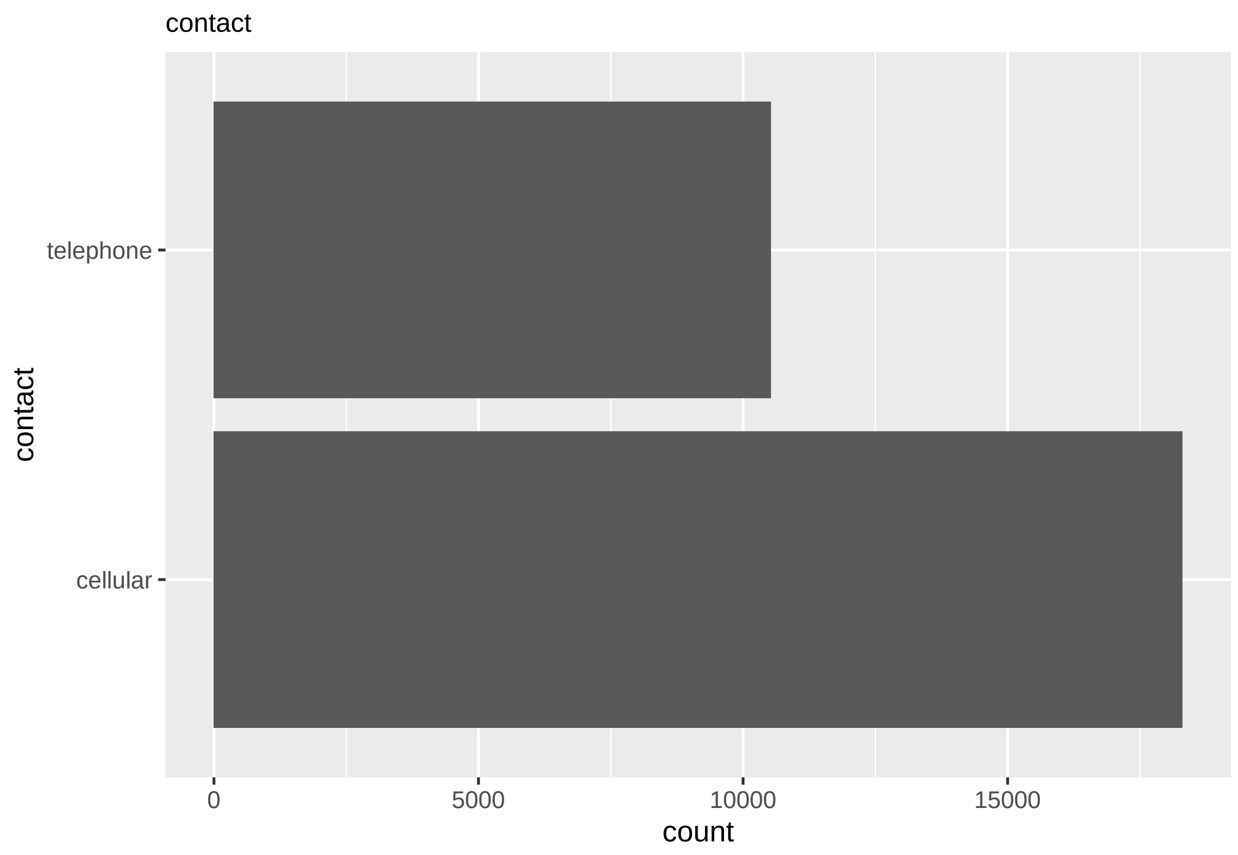
```r
loan_rate_plot <- train_set %>%
  group_by(loan) %>%
    summarise(mean=mean(y)) %>%
  mutate(loan=reorder(loan,(-mean)))%>%
    ggplot(aes(x =mean , y = loan),size=10) +
    geom_bar(stat = 'identity')+ggtitle("loan")+theme(plot.title = element_text(size=10))
loan_rate_plot
```



contact plot

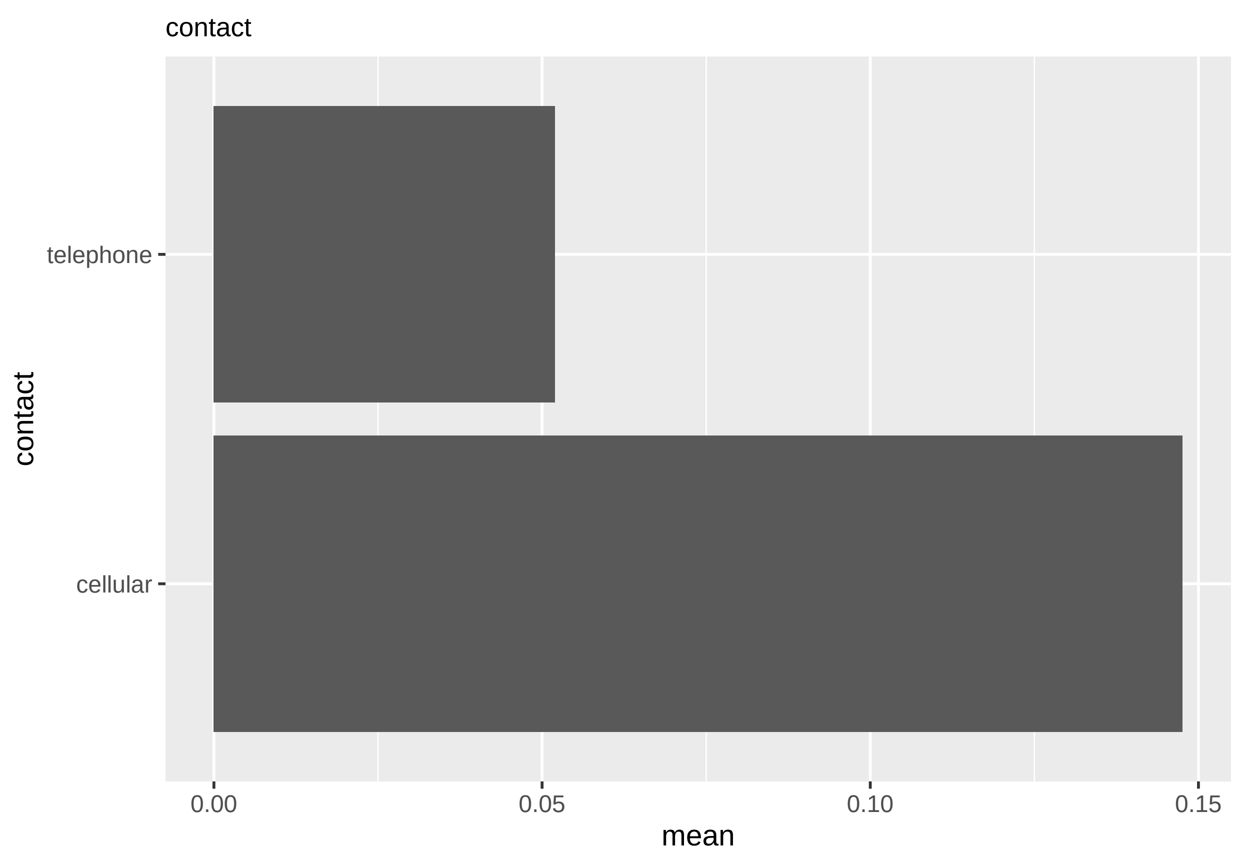
The next Barplot shows the appearance for each category of the contact variable. We can see that most calls were made via mobile phone(cellular), although calls via telephone are also enough represented in the data set.

```
contact_plot <- train_set %>%
  group_by(contact) %>%
    summarise(count = n()) %>%
  mutate(contact=reorder(contact,(-count)))%>%
    ggplot(aes(x =count , y = contact)) +
    geom_bar(stat = 'identity')+ggtitle("contact")+
  theme(plot.title = element_text(size=10))
contact_plot
```



The contact-effect plot shows the effect the contact variable has on the output variable y. It compares whether the client is called via telephone or mobile phone (cellular). By looking at the plot it becomes clear that there is a much higher chance that the client subscribes to a term deposit if the call is via mobile phone (cellular).

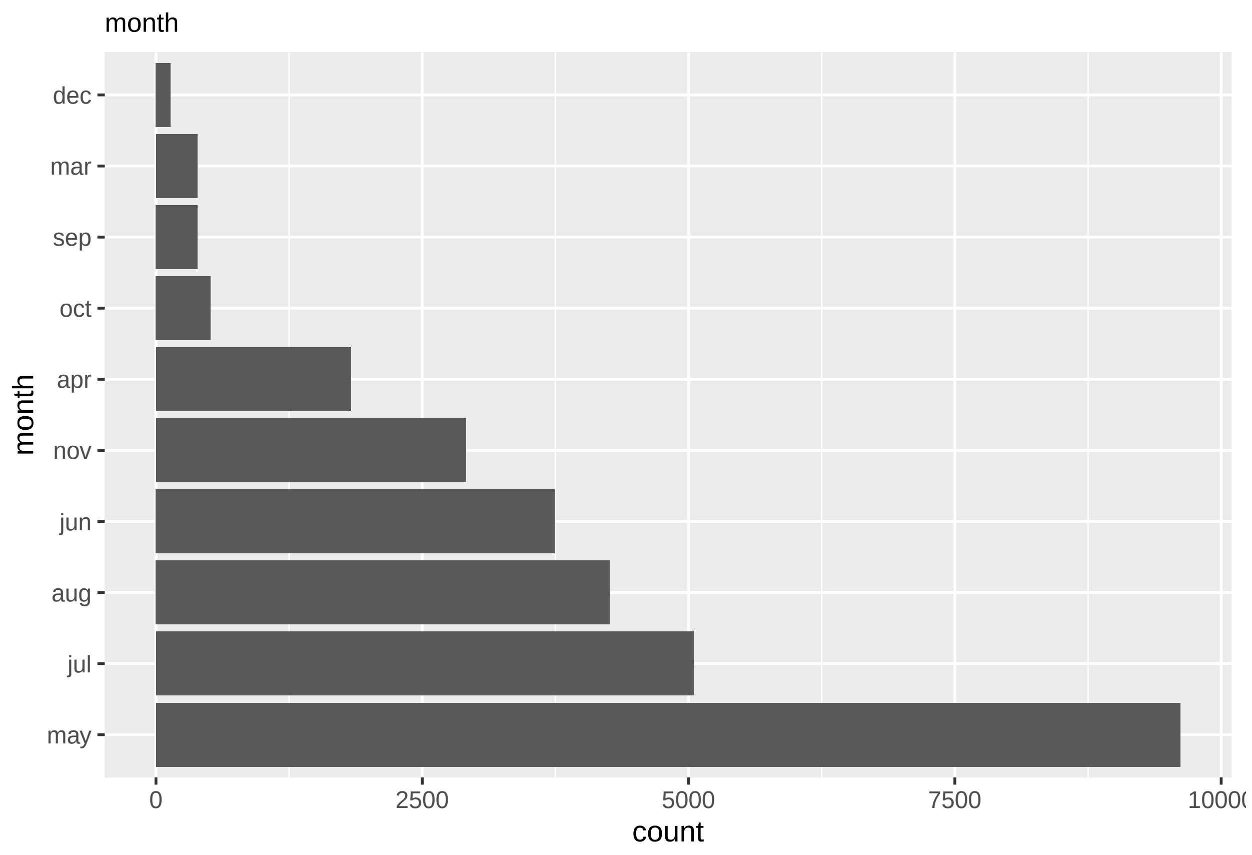
```
contact_rate_plot <- train_set %>%
  group_by(contact) %>%
    summarise(mean=mean(y)) %>%
  mutate(contact=reorder(contact,(-mean)))%>%
    ggplot(aes(x =mean , y = contact),size=10) +
    geom_bar(stat = 'identity')+ggtitle("contact")+theme(plot.title = element_text(size=10))
contact_rate_plot
```



month plot

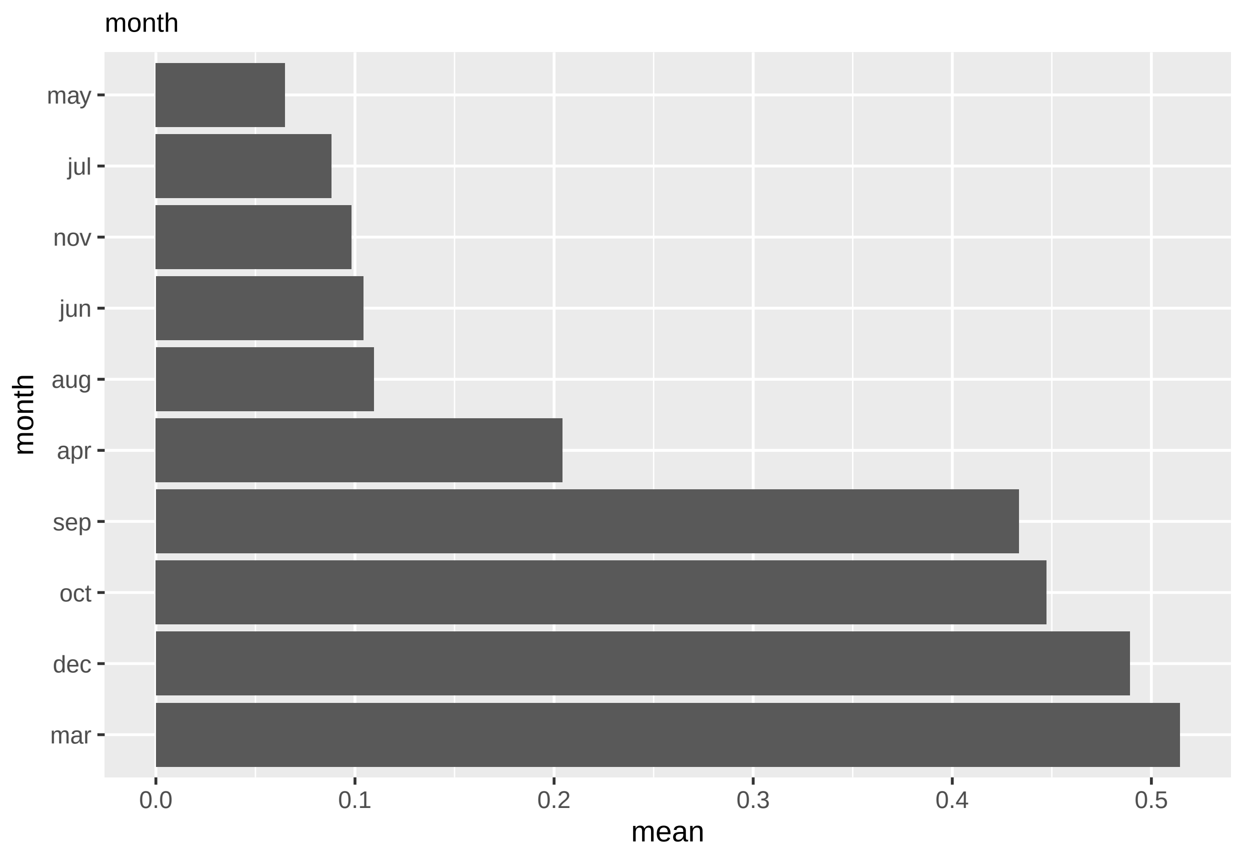
The next barplot shows the appearance for each category of the month variable. In this plot we can see that the majority of the calls were made during the summer.

```
month_plot <- train_set %>%
  group_by(month) %>%
    summarise(count = n()) %>%
  mutate(month=reorder(month,(-count)))%>%
    ggplot(aes(x =count , y = month)) +
    geom_bar(stat = 'identity')+ggtitle("month")+
  theme(plot.title = element_text(size=10))
month_plot
```



The month-effect plot displays the effect the month the call was made has on the success of the call. In May, July November June and August the rate of success is by far the lowest. In October, September, December and March are the months in which the success rate is the highest.

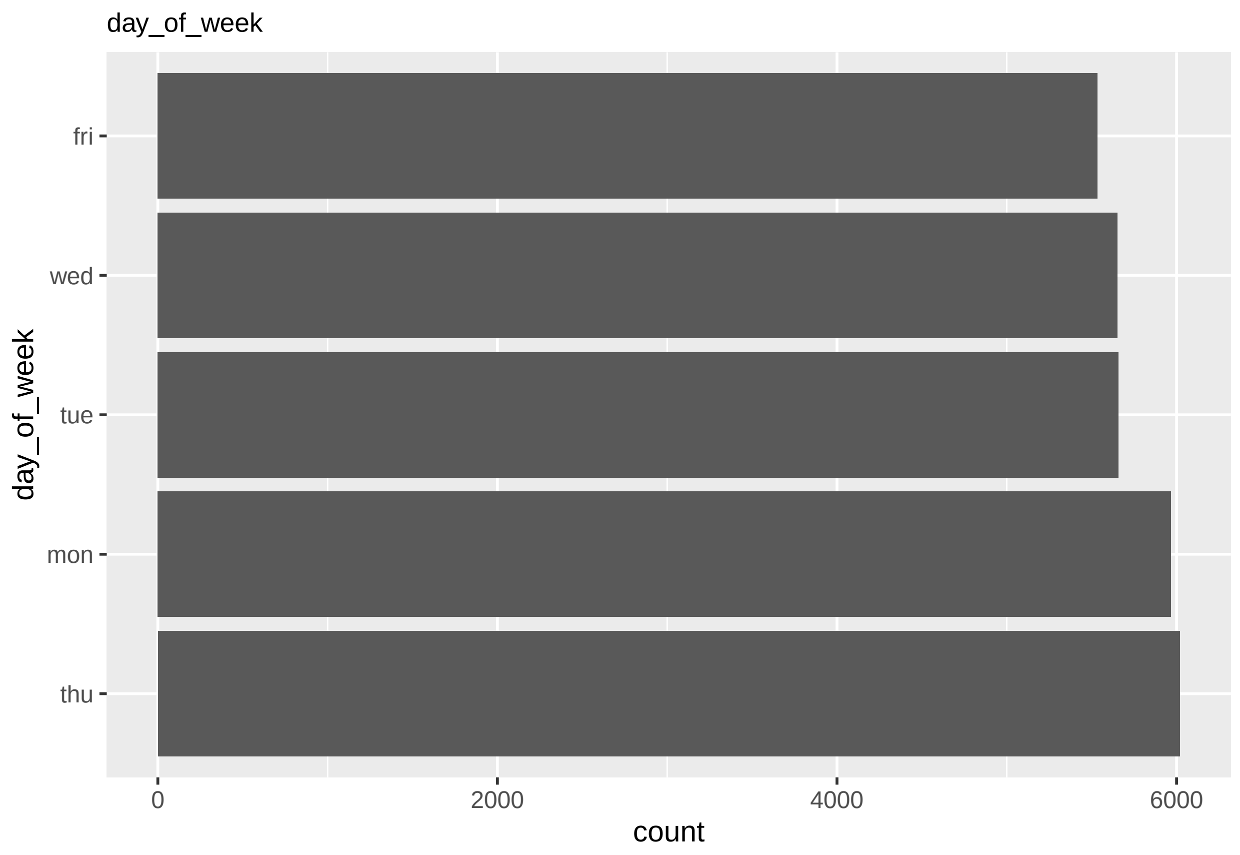
```
month_rate_plot <- train_set %>%
  group_by(month) %>%
    summarise(mean=mean(y)) %>%
  mutate(month=reorder(month,(-mean)))%>%
    ggplot(aes(x =mean , y = month),size=10) +
    geom_bar(stat = 'identity')+ggtitle("month")+theme(plot.title = element_text(size=10))
month_rate_plot
```



days plot

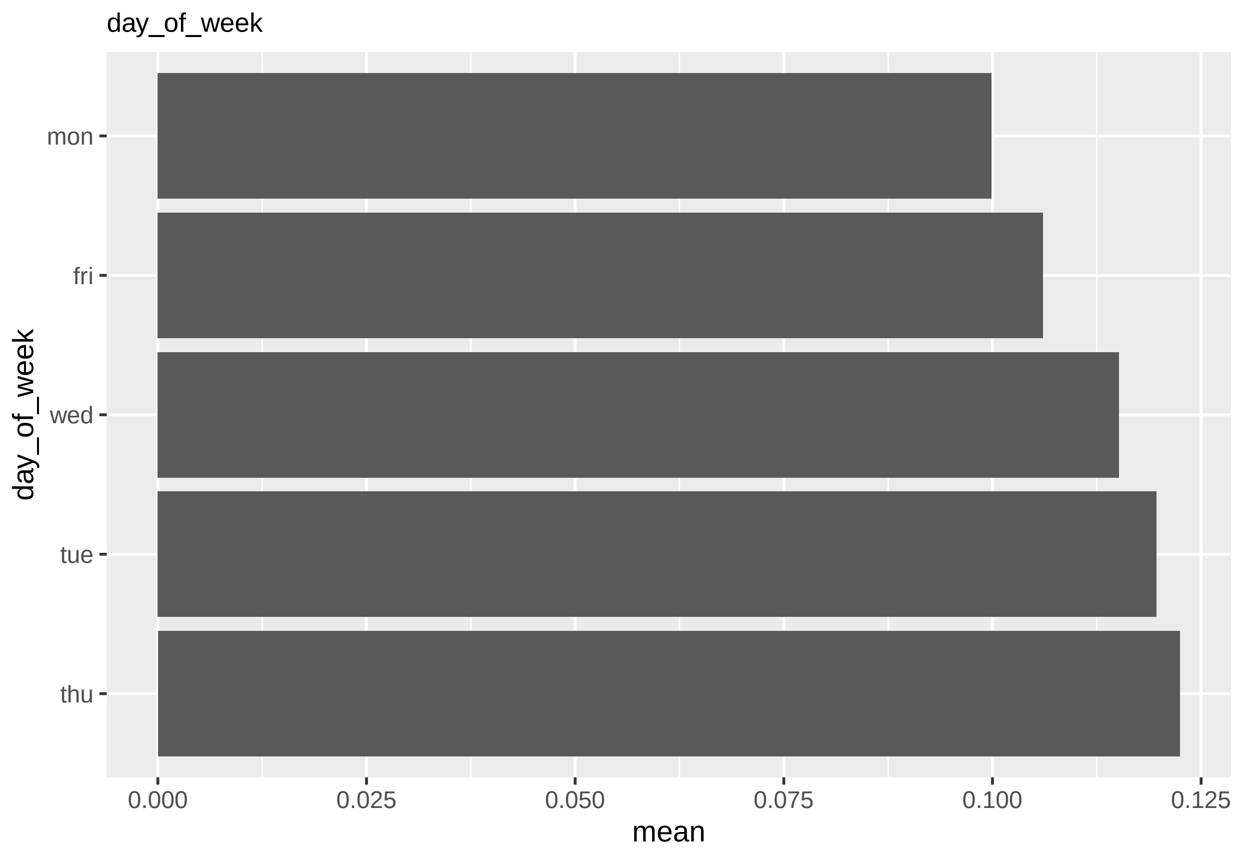
The next Barplot shows the appearance for each category of the day_of_week variable. Here we clearly see that the data is evenly distributed among the days of the week.

```
day_of_week_plot <- train_set %>%
  group_by(day_of_week) %>%
    summarise(count = n()) %>%
  mutate(day_of_week=reorder(day_of_week,(-count)))%>%
    ggplot(aes(x =count , y = day_of_week)) +
    geom_bar(stat = 'identity')+ggtitle("day_of_week")+
  theme(plot.title = element_text(size=10))
day_of_week_plot
```



The following plot shows the effect that the day of the week on which the client is called has an effect on the success rate of the call. By looking at the plot it seems like there is not a big difference between the days and therefore the predictor isn't going to be as strong for predicting the output variable y.

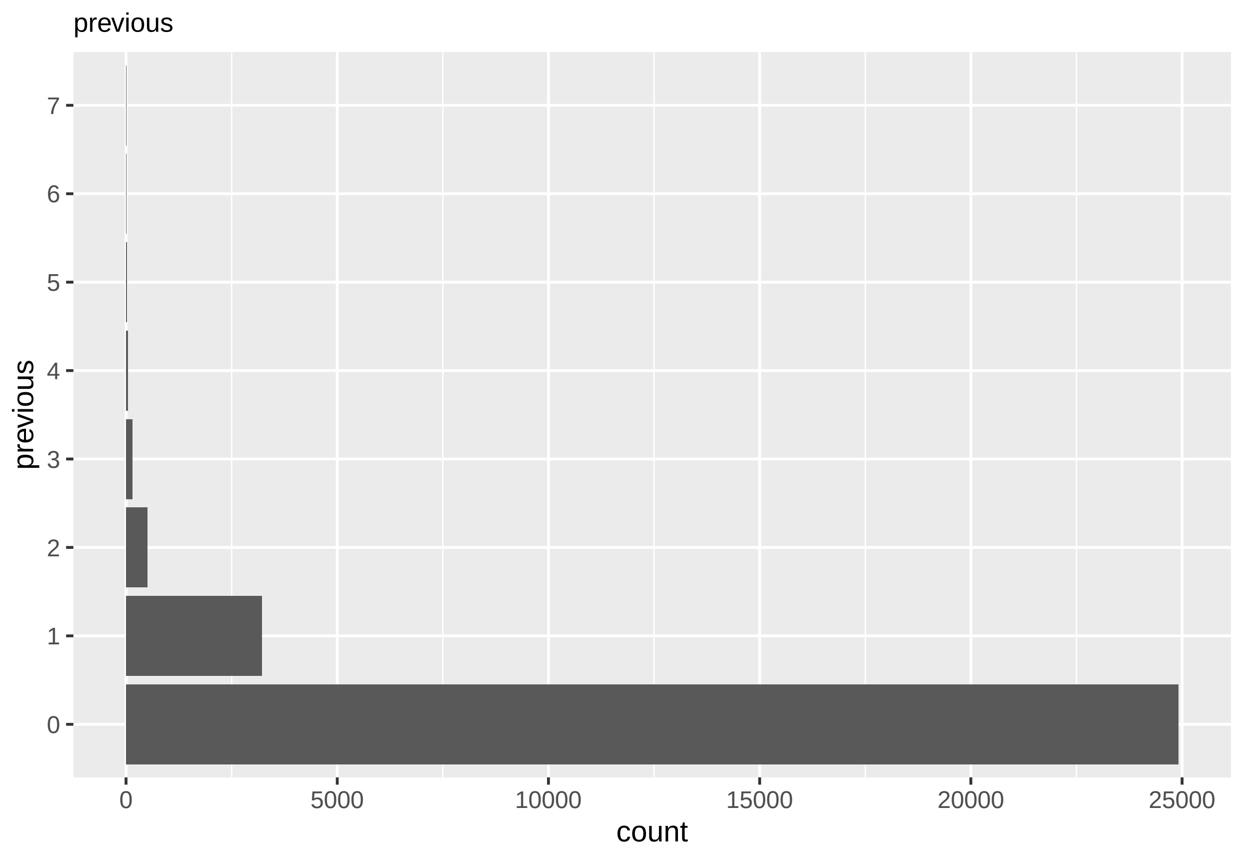
```
day_of_week_rate_plot <- train_set %>%
  group_by(day_of_week) %>%
    summarise(mean=mean(y)) %>%
  mutate(day_of_week=reorder(day_of_week,(-mean)))%>%
    ggplot(aes(x =mean , y = day_of_week),size=10) +
    geom_bar(stat = 'identity')+ggtitle("day_of_week")+theme(plot.title = element_text(size=10))
day_of_week_rate_plot
```



Previous plot

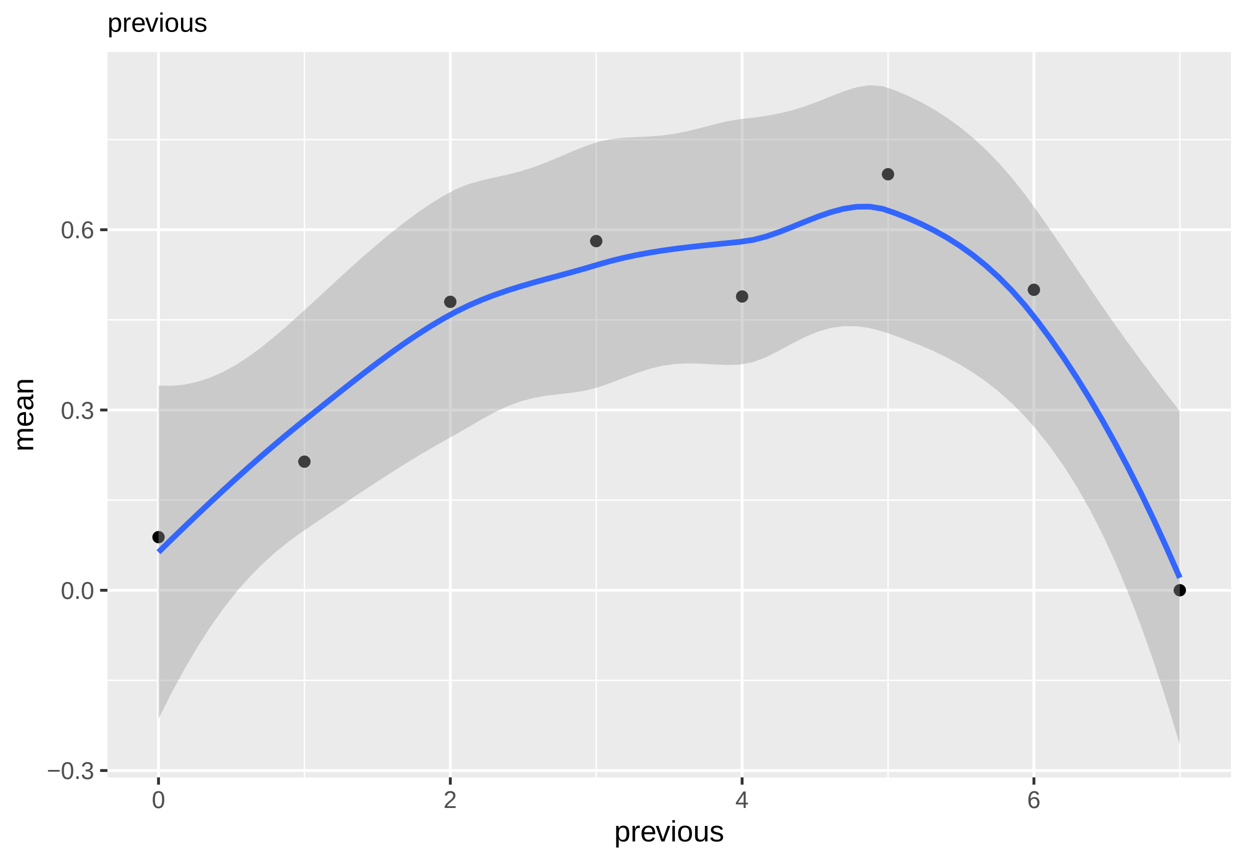
The next Barplot shows the appearance for each category of the previous variable.

```
previous_plot <- train_set %>%
  group_by(previous) %>%
    summarise(count = n()) %>%
  mutate(previous=reorder(previous,(-count)))%>%
    ggplot(aes(x =count , y =previous)) +
    geom_bar(stat = 'identity')+ggtitle("previous")+
  theme(plot.title = element_text(size=10))
previous_plot
```



This plot compares the number of contacts performed before this campaign with the success rate of those contacts. It shows a clear relationship between the variable previous and the output variable y. This relationship is mostly linear except for the variable previous being seven. Therefore the previous variable is an important factor for the model we built in the result section.

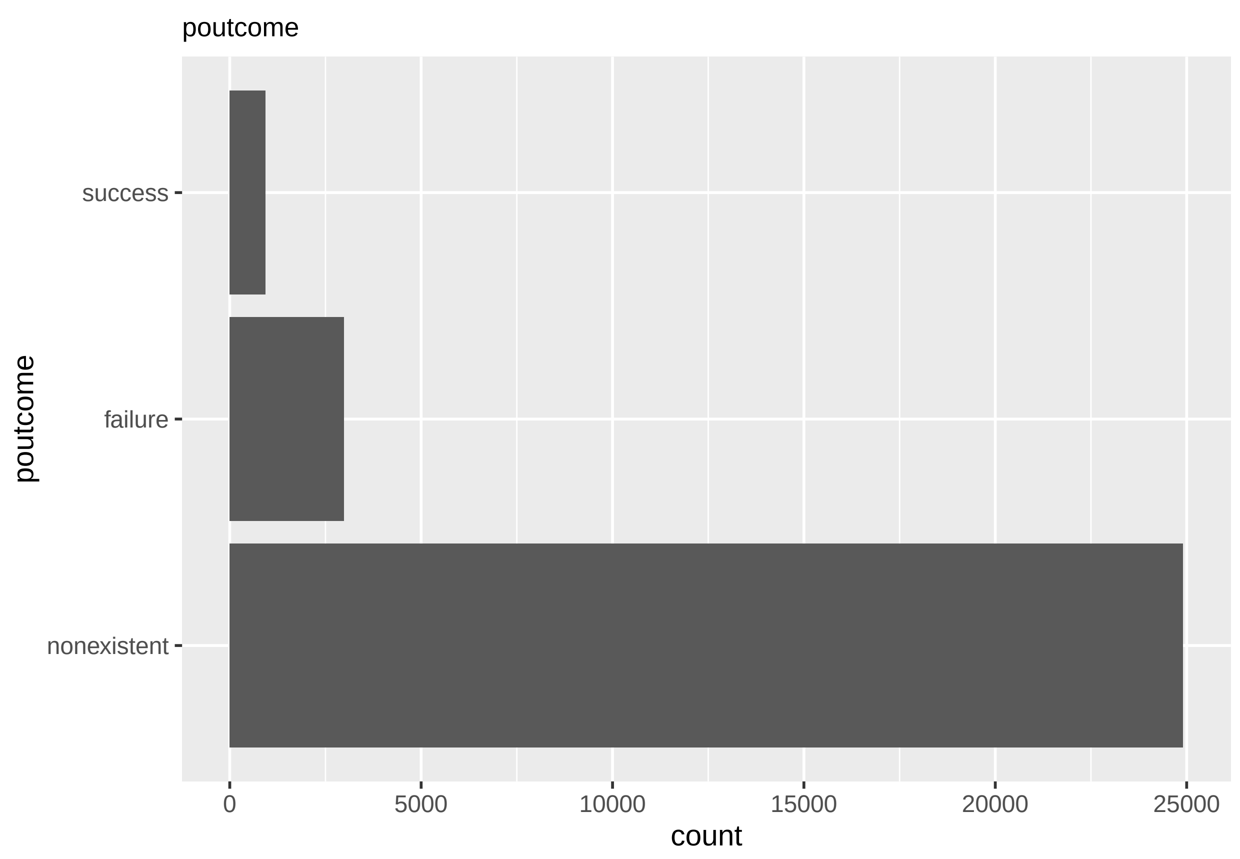
```
previous_rate_plot <- train_set %>%
  group_by(previous) %>%
    summarise(mean=mean(y)) %>%
    ggplot(aes(y =mean , x = previous),size=10) +
    geom_point()+geom_smooth()+ggtitle("previous")+theme(plot.title = element_text(size=10))
previous_rate_plot
```



Poutcome plot

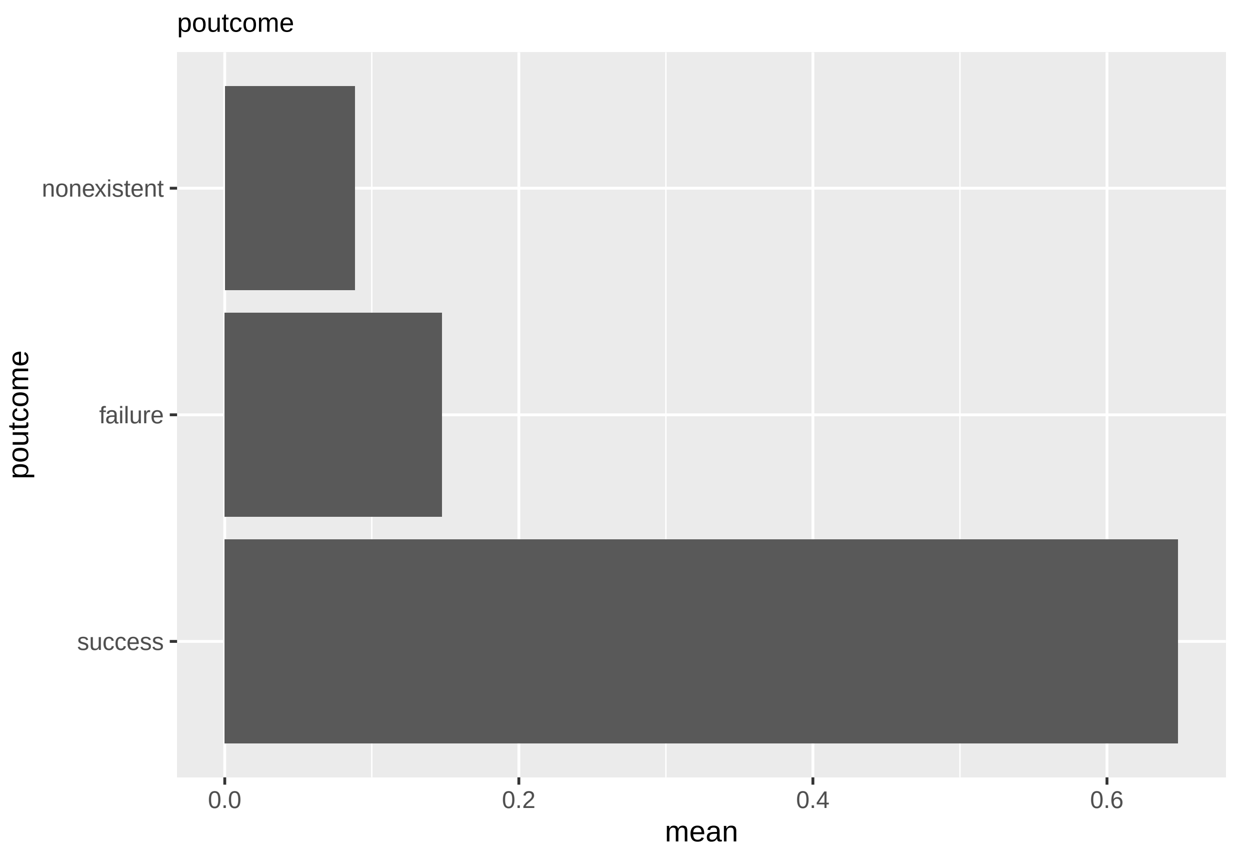
The next Barplot shows the appearance for each category of the poutcome variable.

```
poutcome_plot <- train_set %>%
  group_by(poutcome) %>%
    summarise(count = n()) %>%
  mutate(poutcome=reorder(poutcome,(-count)))%>%
    ggplot(aes(x =count , y =poutcome)) +
    geom_bar(stat = 'identity')+ggtitle("poutcome")+
  theme(plot.title = element_text(size=10))
poutcome_plot
```



This plot shows the effect the outcome of previous campaigns has on the success rate of a call. There are three categories that are being compared: nonexistent, failure and success. The nonexistent category is the least probable to subscribe to a term deposit, this is also what common sense would suggest, because it is unlikely to sell something to a client with who the bank doesn't have a long relationship with. The second category is the failure category, which represents the clients who the previous calls had no success. And the last category shows the clients who the bannk calls were successful in the past.

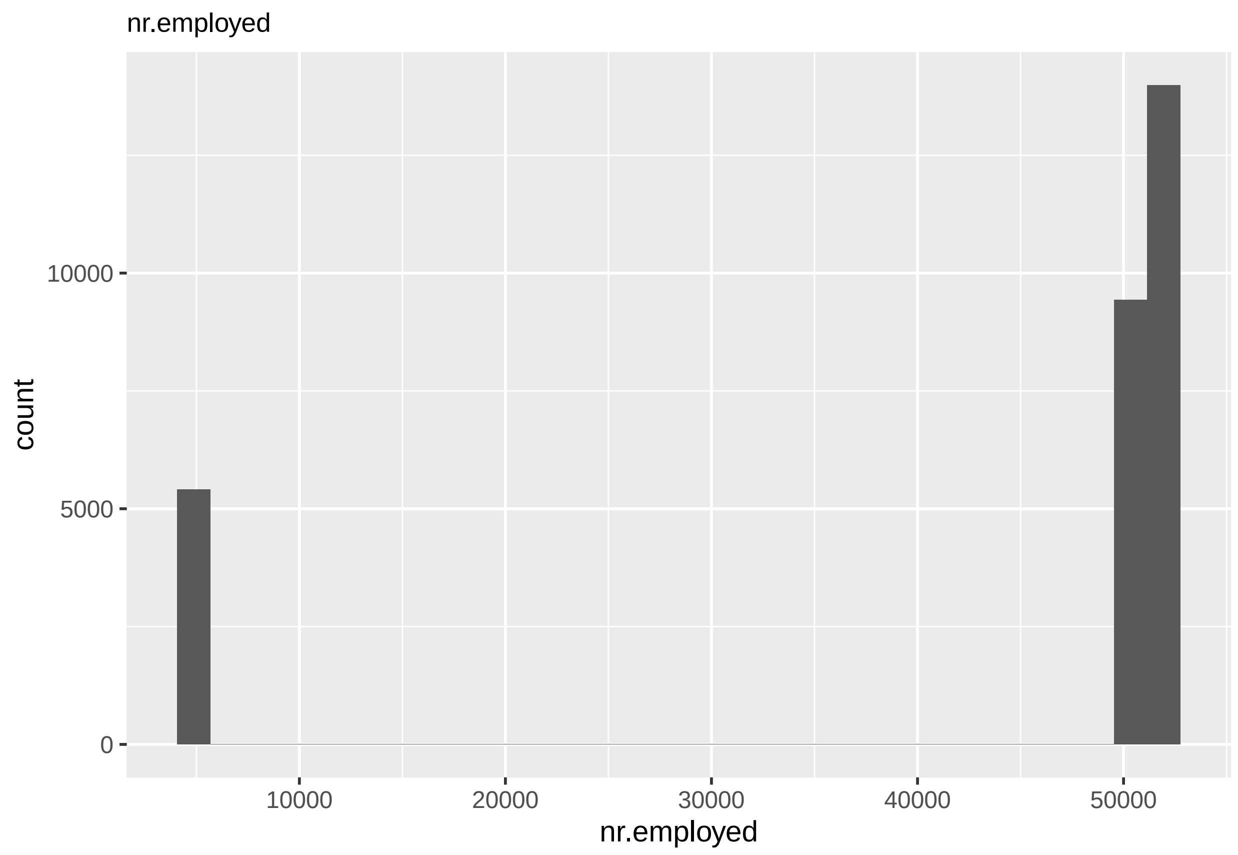
```
poutcome_rate_plot <- train_set %>%
  group_by(poutcome) %>%
    summarise(mean=mean(y)) %>%
  mutate(poutcome=reorder(poutcome,(-mean)))%>%
    ggplot(aes(x =mean , y = poutcome),size=10) +
    geom_bar(stat = 'identity')+ggtitle("poutcome")+theme(plot.title = element_text(size=10))
poutcome_rate_plot
```



nr.employed plot

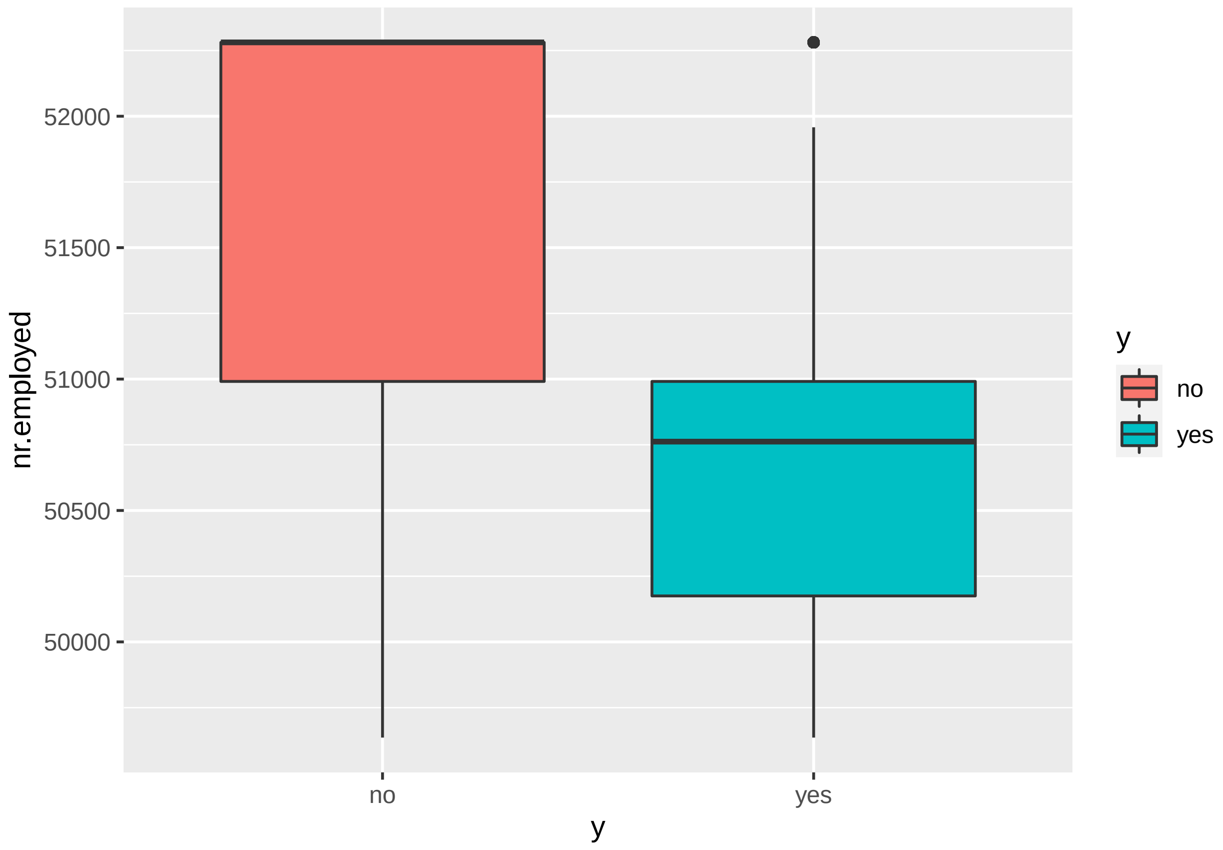
Here we explore the nr.employed feature, we can see that most of the data around 50000 and some of the data is around 0.

```
nr.employed_plot <- train_set %>% ggplot(aes(x=nr.employed))+geom_histogram()+ggtitle("nr.employed")+the
nr.employed_plot
```



Now we look at the effect this feature has on the output variable. This boxplot shows us that lower nr.emploeyd variables leads to higher closure rates.

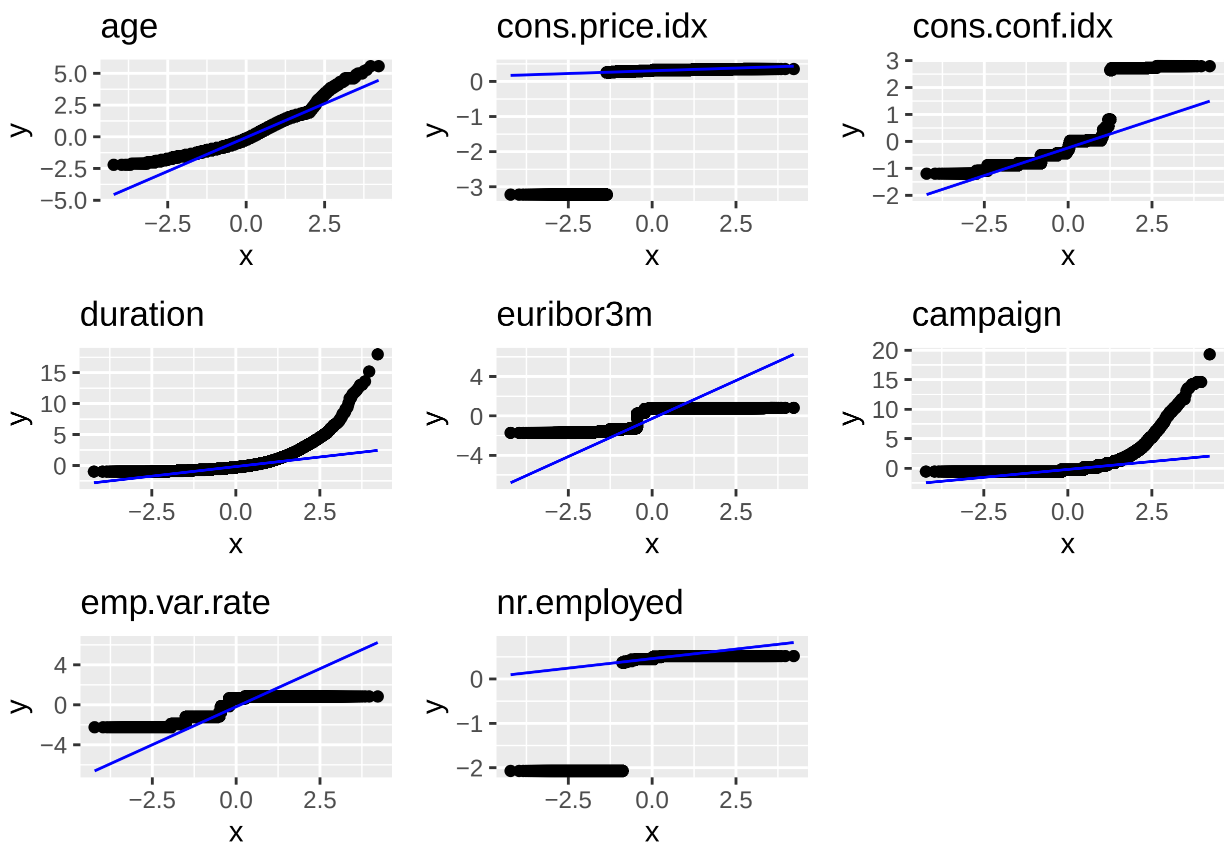
```
train_set$y<- ifelse(train_set$y == 1, "yes","no")
train_set  %>% filter(nr.employed>10000) %>% ggplot(aes(y=nr.employed,x=y,fill=y)) +
  geom_boxplot()
```



QQ-Plots

Since some models like LDA and QDA, require the features to be normally distributed, we compute the QQ-plot. As seen in the plots the numeric features are not at all normally distributed and therefore we can not use a method that requires a normal distribution of the feature data.

```
age_qq<-
  data%>%
  select(age) %>%
  ggplot(data = ., aes(sample = scale(.))) +
  stat_qq() +
  stat_qq_line(col = "blue") +
  ggtitle("age")

cons.price.idx_qq<-
  data%>%
  select(cons.price.idx) %>%
  ggplot(data = ., aes(sample = scale(.))) +
  stat_qq() +
  stat_qq_line(col = "blue")  +
  ggtitle("cons.price.idx")

cons.conf.idx_qq<-
  data%>%
  select(cons.conf.idx) %>%
  ggplot(data = ., aes(sample = scale(.))) +
  stat_qq() +
  stat_qq_line(col= "blue") +
  ggtitle("cons.conf.idx")


duration_qq<-
  data%>%
  select(duration) %>%
  ggplot(data = ., aes(sample = scale(.))) +
  stat_qq() +
  stat_qq_line(col = "blue")  +
  ggtitle("duration")

euribor3m_qq<-
  data%>%
  select(euribor3m) %>%
  ggplot(data = ., aes(sample = scale(.))) +
  stat_qq() +
  stat_qq_line(col = "blue") +
  ggtitle("euribor3m")

campaign_qq<-
  data%>%
  dplyr::select(campaign) %>%
  ggplot(data = ., aes(sample = scale(.))) +
  stat_qq() +
  stat_qq_line(col= "blue")+
  ggtitle("campaign")

emp.var.rate_qq<-
  data%>%
  select(emp.var.rate) %>%
  ggplot(data = ., aes(sample = scale(.))) +
  stat_qq() +
  stat_qq_line(col = "blue") +
  ggtitle("emp.var.rate")

nr.employed_qq<-
  data%>%
  select(nr.employed) %>%
  ggplot(data = ., aes(sample = scale(.))) +
  stat_qq() +
  stat_qq_line(col = "blue") +
  ggtitle("nr.employed")

ggarrange(age_qq,cons.price.idx_qq,cons.conf.idx_qq,duration_qq,euribor3m_qq,campaign_qq,emp.var.rate_q
```

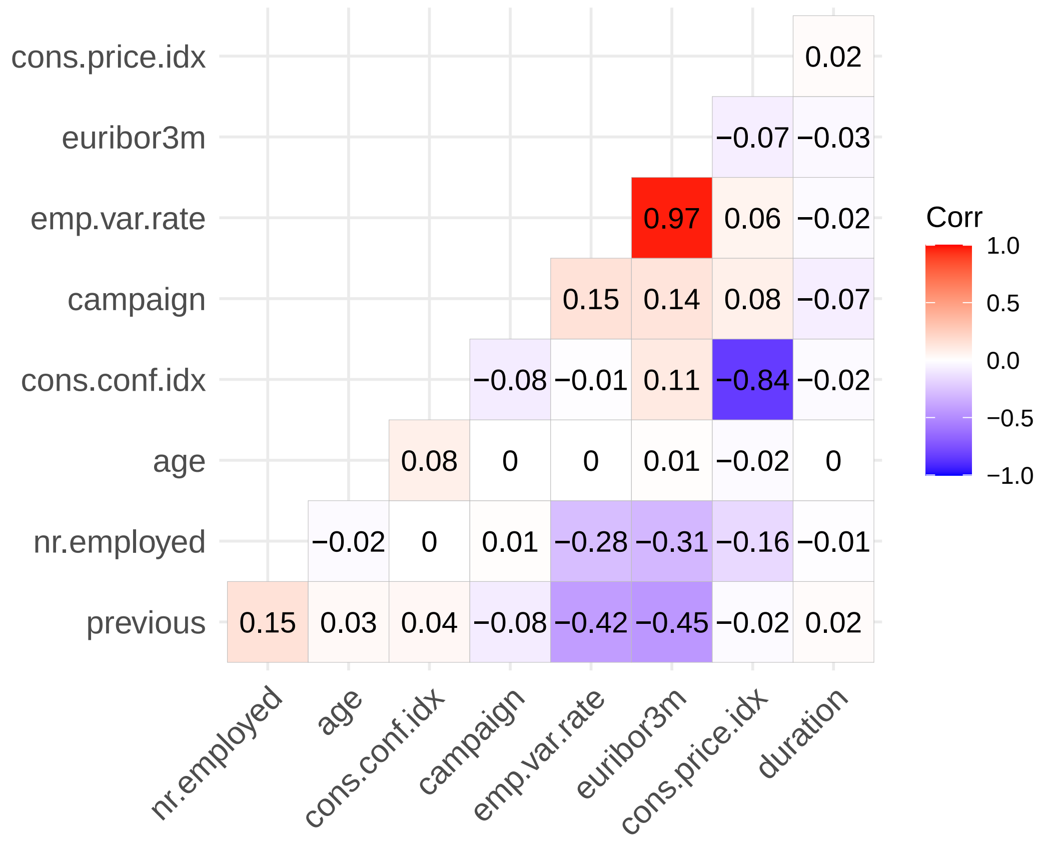


Correlation plot

So far we have seen that several features might be good predictors for our model. But the quality of the prediction can be lower, when the features are highly correlated. The following plot shows the correlation of each numeric feature with each other numeric feature.

In the analysis, we choose 0.5 and -0.5 as cutoff values to determine what is a high correlation. As we can see none of the features we choose in the variable importance part are highly correlated with each other, but we can see that other features like the cons.price.idx and the cons.conf.idx feature are with a correlation of -0.84 and the correlation between emp.var.rate and euribor3m of 0.97 are to high correlated.

```
library(ggcorrplot)
ggcorrplot(cor(train_set[,c("age", "campaign","cons.conf.idx", "cons.price.idx","previous","duration","
           type = "lower",
           lab = TRUE)
```



Modeling

In the Modeling section we will use several machine learning algorithms to classify output variable. First we train the model on the train_set dataset and then we test the model on the test_set dataset. After this is done we select the best performing model and use this model for the Validation dataset.

Reference

The reference model predicts every call to be a “no”. This leads to a very high accuracy and specificity, because most of the output values are “no”. But the Sensitivity is 0, because no client is being predicted to subscribe the term deposit i.e. the output variable being “yes”. This model would make no sense to use in the real world, since the bank wouldn’t call any clients at all. Therefore this model serves as a reference point to compare the different classification algorithms.

KNN-Algorithm

The first algorithm we use is the K-nearest-neighbors algorithm. The knn algorithm computes the distance between points of the training data and a points of the test data. The k nearest points of the training data, from a point in the test data determine by majority vote the output class of the test data point. The idea behind this is that points that are close together should be similar and therefore are likely to have the same output class. K is an optimization parameter that can be optimized using techniques like cross-validation.

Logistic regression

Logistic regression is a type of linear regression that is used often for classification problems, because unlike normal linear regression, logistic regression model estimates values from 0 to 1 and simple linear regression can potentially return prediction greater than 1 or smaller than 0. Since the classification problems deals with estimating conditional probabilities, they can also only range from 0 to 1, this is why logistic regression is much more suitable for this task than simple linear regression.

decision trees

Decision trees built hierarchical models using predictors that are important for classifying the output variable. The decision tree consists of several nodes that use simple yes or no type of questions to separate one class of the output variable from the other. The decision tree is very easy to interpret and uses an easy decision process that is similar to how humans make decisions.

Random Forest

Random forests are as the name suggests based on decision trees. They essentially compute multiple decision trees and accumulate the results of those decision trees to make one prediction. Because they evolve a lot of decision trees the hope is that a random forest is more accurate than a single decision tree, but they are definitely not as interpretable as a simple decision tree.

Ensemble

The last model we use, is using each of the previously discussed models to form a better prediction. It uses the predictions of all the algortihms and decides by majority vote how to classify the output variable.

Results

In this section we apply the above discussed machine learning algorithms to the data set, use the function threshoulder() to optimize the probability threshold and evaluate them based on the confusion matrix.

Reference

We can see that reference model achieved, because of the low prevalence, a relatively high accuracy of 0.8873. But we can also see that since the model never predicted the output variable to be "yes",this is why the sensitivity is 0.

```
test_set$y <- ifelse(test_set$y==1,"yes","no")
reference <- test_set %>% mutate(reference="no") %>% .$reference %>% as.vector()
reference[2] <- "yes"
confusionMatrix(as.factor(reference),as.factor(test_set$y),positive = "yes")
```

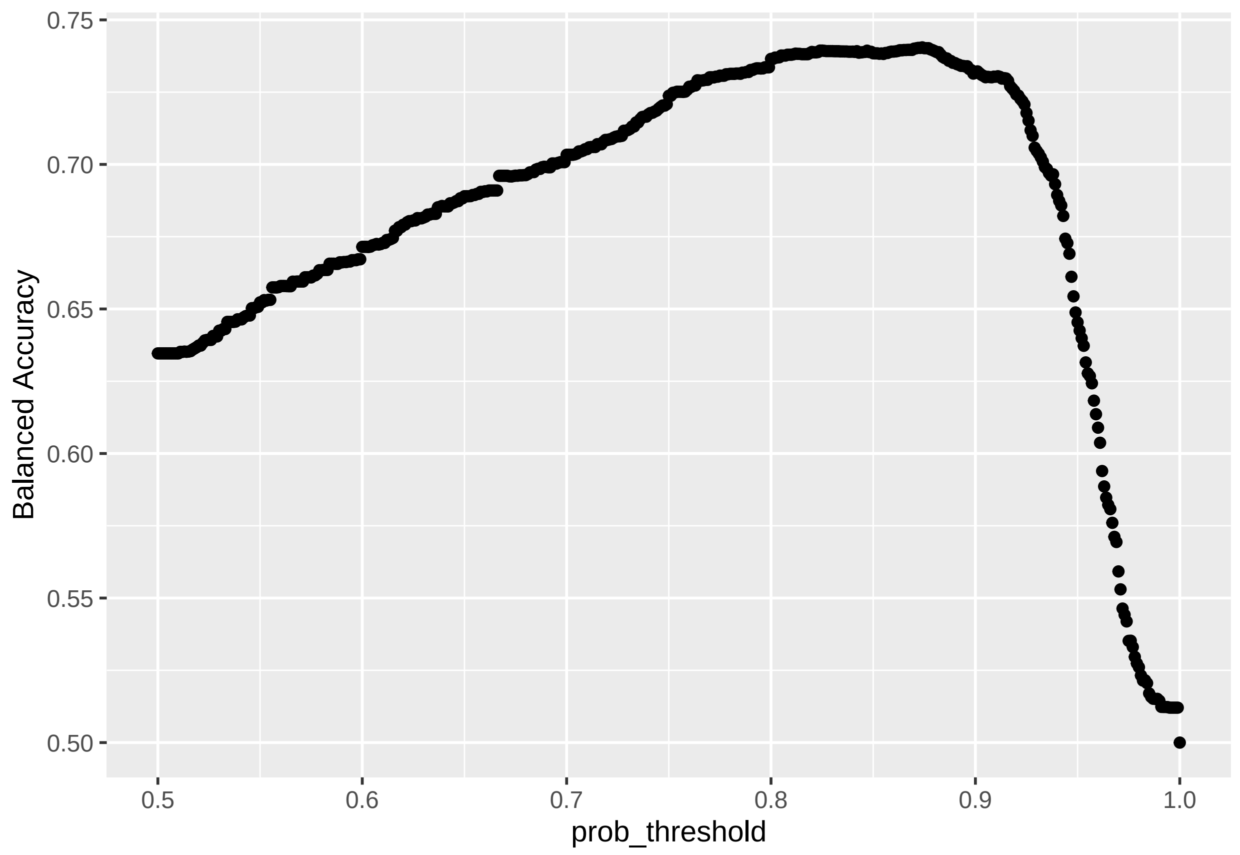
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    no   yes
##        no  10964  1392
##        yes     1     0
```

```
##
##                Accuracy : 0.8873
##                  95% CI : (0.8816, 0.8928)
##     No Information Rate : 0.8874
##     P-Value [Acc > NIR] : 0.5185
##
##                   Kappa : -2e-04
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.000e+00
##             Specificity : 9.999e-01
##          Pos Pred Value : 0.000e+00
##          Neg Pred Value : 8.873e-01
##              Prevalence : 1.126e-01
##          Detection Rate : 0.000e+00
##    Detection Prevalence : 8.093e-05
##       Balanced Accuracy : 5.000e-01
##
##        'Positive' Class : yes
##
```

Knn algortihm

Here we use the KNN algorithm and optimize the probability threshold. We can see that this method achieves a balanced accuracy of 0.7367. We have a Sensitivity of 0.6027 and a Specificity of 0.8706. It is clear that this model performed considerably better than the reference model. The plot bellow shows the optimization of the probability threshold based on the balanced accuracy.

```
set.seed(1)
train_set$y <- as.factor(train_set$y)
test_set$y <- as.factor(test_set$y)
fitControl <- trainControl(method = "cv",
                           classProbs = TRUE,
                           summaryFunction = twoClassSummary,
                           savePredictions="all")

fit_knn <- train(y~nr.employed+contact+cons.price.idx+previous+default+month+job+poutcome,method="knn",

stats_knn <- thresholder(fit_knn ,threshold = seq(0.5,1,0.001),final=TRUE)
stats_knn %>% ggplot(aes(x=prob_threshold,`Balanced Accuracy`))+geom_point()
```



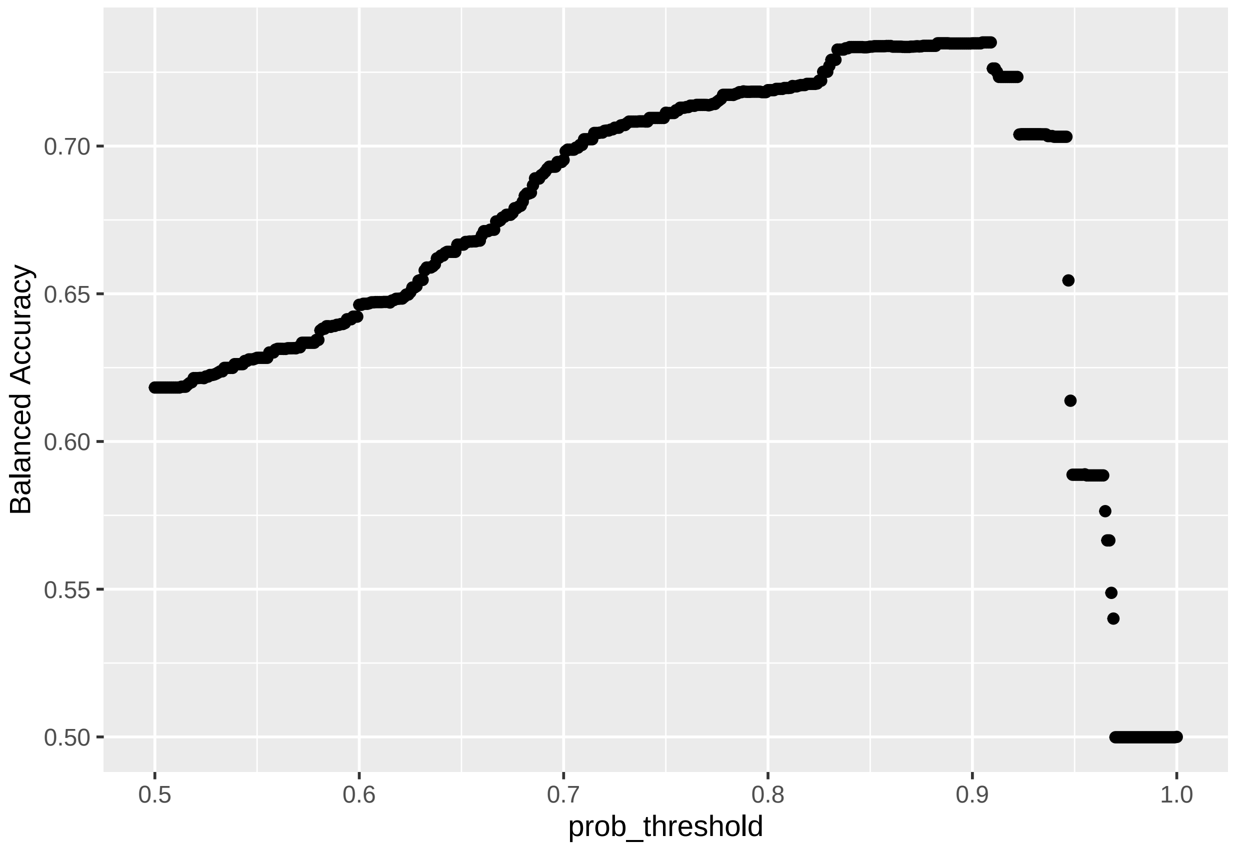
```
knn_pred <- as.factor(ifelse(predict(fit_knn,as.data.frame(test_set),type="prob")$no>=stats_knn$prob_thr
confusionMatrix(knn_pred,as.factor(test_set$y),positive = "yes")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   no  yes
##        no  9546  553
##        yes 1419  839
##
##                Accuracy : 0.8404
##                  95% CI : (0.8338, 0.8468)
##     No Information Rate : 0.8874
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3722
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.6027
##             Specificity : 0.8706
##          Pos Pred Value : 0.3716
##          Neg Pred Value : 0.9452
##              Prevalence : 0.1126
##          Detection Rate : 0.0679
##    Detection Prevalence : 0.1827
##       Balanced Accuracy : 0.7367
##
##        'Positive' Class : yes
##
```

Decision tree algorithm

Next we apply the decision tree model to the data. We see from the confusion matrix, that this model has a balanced accuracy of 0.73537, a Sensitivity of 0.60991 and a Specificity of 0.86083. Therefore the decision tree model preformed slightly worse. The plot bellow shows the optimization of the probability threshold based on the balanced accuracy.

```
library(yardstick)
library(caret)


fit_rpart <- train(y~nr.employed+contact+cons.price.idx+previous+default+month+job+poutcome,data = trai
                   method="rpart",metric="ROC",trControl=fitControl,tuneLength=20)

stats_rpart <- thresholder(fit_rpart,threshold = seq(0.5,1,0.001),final=TRUE)
stats_rpart %>% ggplot(aes(x=prob_threshold,`Balanced Accuracy`))+geom_point()
```



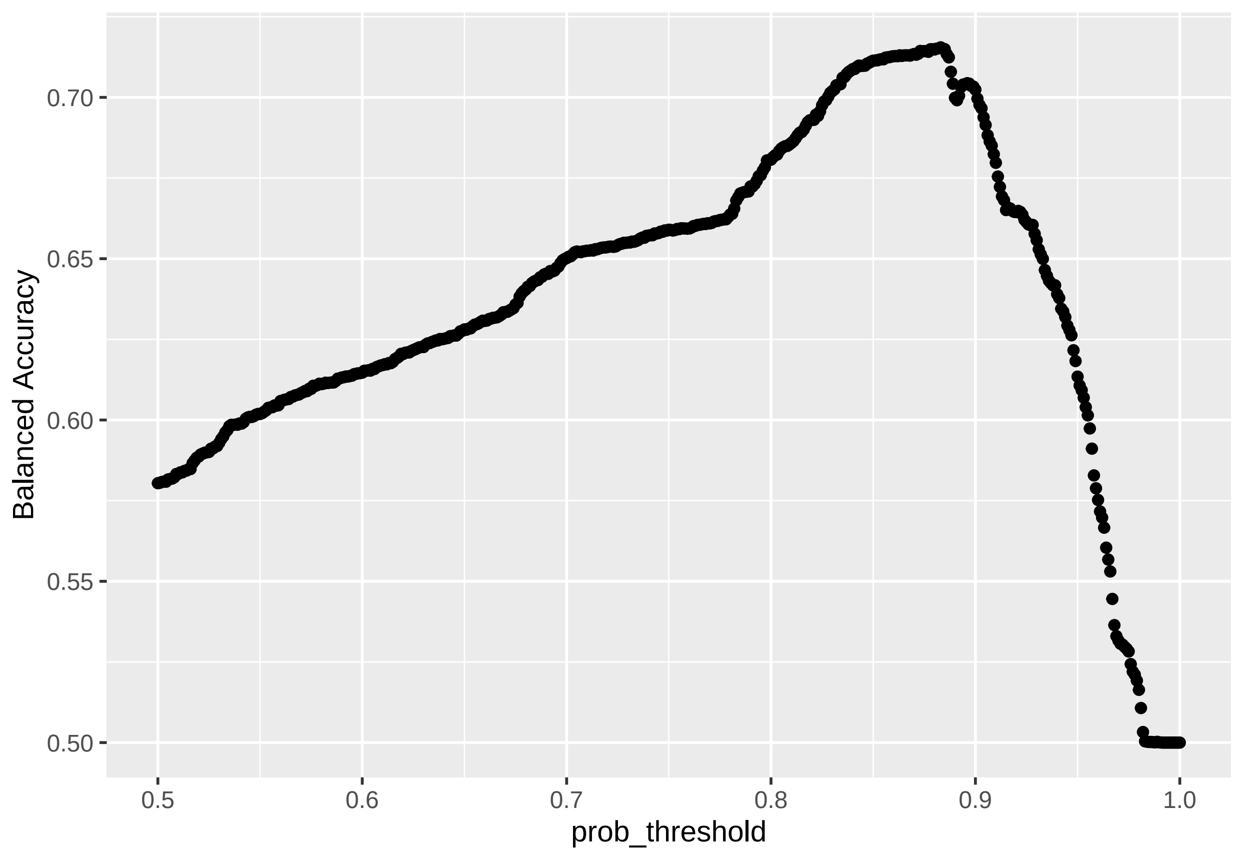
```
rpart_pred <- as.factor(ifelse(predict(fit_rpart,as.data.frame(test_set),type="prob")$no>stats_rpart$pr

confusionMatrix(data=rpart_pred,reference=as.factor(test_set$y),positive = "yes")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   no  yes
##        no  9439  543
##        yes 1526  849
##
##                Accuracy : 0.8326
##                  95% CI : (0.8259, 0.8391)
##     No Information Rate : 0.8874
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3598
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.60991
##             Specificity : 0.86083
##          Pos Pred Value : 0.35747
##          Neg Pred Value : 0.94560
##              Prevalence : 0.11265
##          Detection Rate : 0.06871
##    Detection Prevalence : 0.19220
##       Balanced Accuracy : 0.73537
##
##        'Positive' Class : yes
##
```

glm model

Now we use the logistic regression. We see that it performed worse than the decision tree model and than the knn model. The plot bellow shows the optimization of the probability threshold based on the balanced accuracy.

```
fit_glm <- train(y~nr.employed+contact+cons.price.idx+previous+default+month+job+poutcome,data=train_se

stats <- thresholder(fit_glm ,threshold = seq(0.5,1,0.001),final=TRUE)
stats%>% ggplot(aes(x=prob_threshold,`Balanced Accuracy`))+geom_point()
```



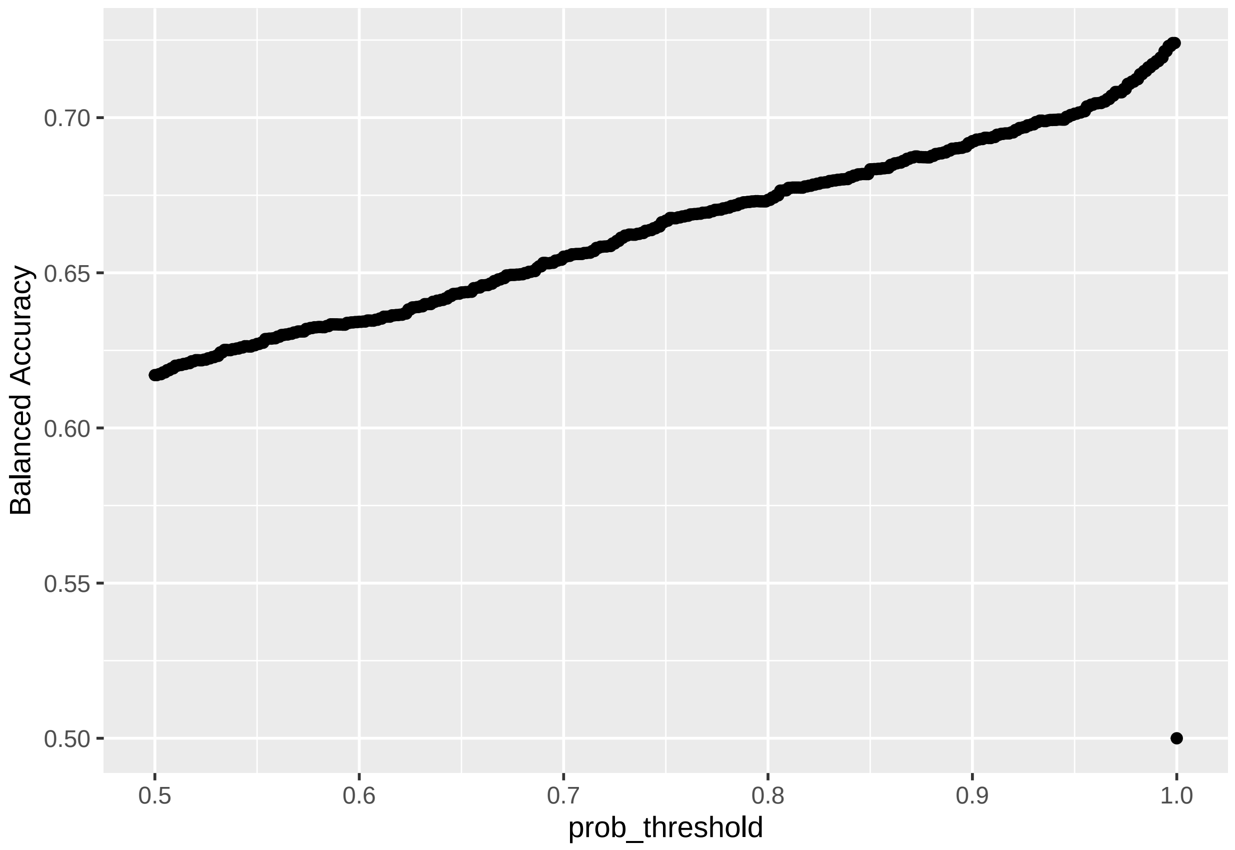
```
glm_pred <- as.factor(ifelse(predict(fit_glm,as.data.frame(test_set),type="prob")$no>stats$prob_threshol

confusionMatrix(glm_pred,as.factor(test_set$y) ,positive = "yes")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   no  yes
##        no  9509  644
##        yes 1456  748
##
##                Accuracy : 0.8301
##                  95% CI : (0.8233, 0.8366)
##     No Information Rate : 0.8874
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3225
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.53736
##             Specificity : 0.86721
##          Pos Pred Value : 0.33938
##          Neg Pred Value : 0.93657
##              Prevalence : 0.11265
##          Detection Rate : 0.06053
##    Detection Prevalence : 0.17836
##       Balanced Accuracy : 0.70229
##
##        'Positive' Class : yes
##
```

Random Forest model

Here we use the Random Forest model for the classification task. We observe that this model performed not as we expected worse than the decision tree, but better than the logistic regression. The plot bellow shows the optimization of the probability threshold based on the balanced accuracy.

```
fit_rf<- train(y~nr.employed+contact+cons.price.idx+previous+default+month+job+poutcome,method="rf",data

stats_rf <- thresholder(fit_rf ,threshold = seq(0.5,1,0.001),final=TRUE)
stats_rf %>% ggplot(aes(x=prob_threshold,`Balanced Accuracy`))+geom_point()
```



```
rf_pred <- as.factor(ifelse(predict(fit_rf,as.data.frame(test_set),type="prob")$no>stats_rf$prob_thresh
class(rf_pred )
```

```
## [1] "factor"
```

```r
confusionMatrix(rf_pred,as.factor(test_set$y) ,positive = "yes")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   no  yes
##        no  9862  626
##        yes 1103  766
##
##                Accuracy : 0.8601
##                  95% CI : (0.8538, 0.8662)
##     No Information Rate : 0.8874
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3912
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.55029
##             Specificity : 0.89941
##          Pos Pred Value : 0.40984
##          Neg Pred Value : 0.94031
##              Prevalence : 0.11265
##          Detection Rate : 0.06199
##    Detection Prevalence : 0.15125
##       Balanced Accuracy : 0.72485
##
##        'Positive' Class : yes
##
```

Ensemble

The last model is the combination of all previous models and it preformed with a balanced accuracy of 0.73801, a Sensitivity of 0.61638 and a Specificity of 0.85964, better than all the other algorithms.

```r
Ensemble_data <- data.frame(glm=ifelse(glm_pred=="yes",1,0), rpart=ifelse(rpart_pred=="yes",1,0), rf=if

ensemble_pred <- ifelse(rowMeans(Ensemble_data)>= 0.5,"yes","no")
confusionMatrix(data=as.factor(ensemble_pred),reference=as.factor(test_set$y) ,positive = "yes")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   no  yes
##        no  9426  534
##        yes 1539  858
##
##                Accuracy : 0.8322
##                  95% CI : (0.8255, 0.8388)
##     No Information Rate : 0.8874
##     P-Value [Acc > NIR] : 1
##
```

```
##                  Kappa : 0.362
##
##  Mcnemar's Test P-Value : <2e-16
##
##            Sensitivity : 0.61638
##            Specificity : 0.85964
##         Pos Pred Value : 0.35795
##         Neg Pred Value : 0.94639
##             Prevalence : 0.11265
##         Detection Rate : 0.06943
##   Detection Prevalence : 0.19398
##      Balanced Accuracy : 0.73801
##
##       'Positive' Class : yes
##
```

Model Evaluation

In machine learning, a model is evaluated by comparing the predicted value with the actual outcome. In this document a confusion matrix is used in order to do that.

Confusion Matrix

A Confusion Matrix is a table that compares the actual positives and the actual negatives with the predicted positives and the predicted negatives.

	Positive	Negative
Positive	True Positive(TP)	False Negative(FN)
Negative	False Positive(FP)	True Negative(TN)

The Confusion matrix is especially useful, because it is possible to compute several very important metrics based on it.

Accuracy: the proportion of predicted values that match the actual values

$$\frac{TP + TN}{TP + FP + TN + FN}$$

Sensitivity: the proportion of positive predicted values that are actual positive values

$$\frac{TP}{TP + FN}$$

specificity: the proportion of negative values that are actual negative values

$$\frac{TN}{TN + FP}$$

Prevalence: the proportion of positive values in the sample

Precision: the probability that an actual positive value occurs given a positive prediction or in other words the proportion of positive predictions that are positive

$$\frac{TP}{TP + FP}$$

Balanced accuracy:

$$\frac{Sensitivity + Specificity}{2}$$

Since our data set has very low prevalence using overall accuracy for evaluating the performance of the algorithms is not good. This is why we decided to use the Balanced accuracy for deciding which algorithm to use for the validation. By comparing the algorithms balanced accuracy we can see that the Ensemble model performed the best.

Validation

Finally we validate the result of the Ensemble model by using it on the validation data set. Here we can see that the model achieved a balanced accuracy of 0.74071.

```
knn_pred_Validation <- as.factor(ifelse(predict(fit_knn,as.data.frame(Validation),type="prob")$no>=stat

rpart_pred_Validation <- as.factor(ifelse(predict(fit_rpart,as.data.frame(Validation),type="prob")$no>s

glm_pred_Validation <- as.factor(ifelse(predict(fit_glm,as.data.frame(Validation),type="prob")$no>stats

rf_pred_Validation <- as.factor(ifelse(predict(fit_rf,as.data.frame(Validation),type="prob")$no>stats_r
```

```
Ensemble_data <- data.frame(glm=ifelse(glm_pred_Validation=="yes",1,0), rpart=ifelse(rpart_pred_Validat

ensemble_pred <- ifelse(rowMeans(Ensemble_data)>= 0.5,"yes","no")
confusionMatrix(data=as.factor(ensemble_pred),reference=as.factor(Validation$y) ,positive = "yes")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   no  yes
##        no  3181  174
##        yes  487  277
##
##                Accuracy : 0.8395
##                  95% CI : (0.828, 0.8506)
##     No Information Rate : 0.8905
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3691
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.61419
##             Specificity : 0.86723
##          Pos Pred Value : 0.36257
##          Neg Pred Value : 0.94814
##              Prevalence : 0.10949
##          Detection Rate : 0.06725
##    Detection Prevalence : 0.18548
##       Balanced Accuracy : 0.74071
##
##        'Positive' Class : yes
##
```

Conclusion

The goal of this project was to develop an algorithm, which classify if the output variable y is either "yes" or "no". We explored the data and selected features that are important for such a classification model. We also visualized the effect of each feature individually. Then we built the models,optimized the probability threshold and compare the performance among each of the model performances. At the end we selected the best performing model, in this case the Ensemble model, and used this model for the validation data set. The Ensemble model achieved a Balanced Accuracy of 0.74071, a Sensitivity of 0.61419 and a Specificity of 0.86723 on the validation dataset. Although the model performed better than the reference, we can see that predicting the decision of a client to subscribe to a term deposit is quit difficult with the predictors we were given.

Reference

https://rafalab.github.io/dsbook/

filterVarImp() function:

https://rdrr.io/cran/caret/man/filterVarImp.html

threshoulder() function:

https://www.rdocumentation.org/packages/caret/versions/6.0-88/topics/thresholder

Data:

https://archive.ics.uci.edu/ml/datasets/Bank+Marketing

[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