Modeling problem report

Jakub_Kubiś

05 06 2021

```
data <- read.csv('C:\\Kuba\\magisterka\\II semestr\\SLM\\raport\\bank-additional\\bank-additional-full.csv',sep =';', stringsAsFactors = F)
```

Introduction

The main objective of this report is to build models that will predict whether the client will subscribe or not a term deposit. Term deposit is an agreement between client and banking institution. The client locks away an amount of money for agreed period of time. In return he receives the guaranteed rate of interest at the end of agreed period. It is important for the banks to receive capital from these term deposits, which they can use to give loans and credits which have higher rates. It allows banking institution to gain benefits.

Problem description

Nowadays banks all over the world use direct marketing campaigns in order to gain new clients who will subscribe term deposits. They all face the same problem of choosing the right people who will become potential clients. If the people who are phoned are chosen randomly there is a high probability that the person will not be interested in the offer and it will be a waste of time for the employee. On the other hand, people who can be potentially interested in subscribing term deposit may not even be phoned. Thanks to the predictive models there is a possibility to increase the knowledge whether the person will subscribe or not a term deposit which will result in the optimization of time of work of employees of the banking institution and will result in increase of benefits for the company. In this report 3 different models will be built and compared: classification tree, random forest and neural network. Hyperparameters for each model will be tuned in order to increase their predictive power. From the business point of view it is equally important to increase the true positive rate and accuracy of the model. It is very important to detect clients that will subscribe a term deposit, because it gives capital for bank that can be used for giving loans and credits. But, it is also very important to know that a certain client will not subscribe a term deposit. It can save a lot of time spent on preparing direct marketing campaign for this person. Thus, it is also essential that model can accurately predict both positive and negative values. Due to this reasons for each of 3 types of models 2 distinct models will be built. First one with objective to increase accuracy and second one with objective to increase sensitivity. Then the results of different types of models will be compared.

Data Set Description

The data comes from the Portuguese banking institution and is related with direct marketing campaigns (phone calls). It was collected between 2008 and 2010. It contains 41188 observations, 20 explanatory variables and a binary target variable. The independent variables are the following:

age - numeric variable

job - categorical variable with 11 categories

marital - categorical variable which describes the marital status of the person (3 categories: “divorced”, “married”, “single”)

education - categorical variable which describes level of education of the person, (7 categories: “basic.4y”, “basic.6y”, “basic.9y”, “high.school”, “illiterate”, “professional.course”, “university.degree”)

default - categorical variable which describes whether the person has credit in default

housing - categorical variable which describes whether the person has housing loan

loan - categorical variable which describes whether the person has personal loan

contact - categorical variable which describes how the person was contacted (2 categories: cellular and telephone)

month - categorical variable which describes during which month was the last contact with the client (10 categories, there is no observations for January and February)

day_of_the_week - categorical variable which describes which day of the week was the last contact with the client (5 categories, the contacts where not performed during weekends)

duration - duration (in seconds) of the last contact

campaign - number of contacts performed during this campaign and for this client

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

previous - number of contacts performed before this campaign and for this client

poutcome - categorical variable which describes what was the outcome of the previous marketing campaign (3 categories: “failure”, “nonexistent”, “success”)

emp.var.rate - the employment variation rate (quarterly indicator)

cons.price.idx - consumer price index (monthly indicator)

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

euribor3m - euribor 3 month rate (daily indicator)

nr.employed - number of employees (quarterly indicator)

Target variable:

y - describes whether the client has subscribed a term deposit (binary variable: “yes”, “no”)

*missing values in this dataset are coded with the “unknown” label

Cleaning and preprocessing data

```
summary(as.factor(data$default))
```

```
##      no unknown     yes
##   32588    8597       3
```

```
sum(data$pdays==999)/nrow(data)
```

```
## [1] 0.9632174
```

```
data <- data[-c(5,11,13)]
```

After the initial exploration of variables 3 of them where decided to be removed from the dataset.

Variable default has only 3 values of category “yes” and over 32000 values of the other category “no”. Variable which has basically only one value does not give any information for the model that is why it has been removed.

Variable pdays has over 96% of observations coded with “999” which stands for clients who were not previously contacted. These observations cannot be left as it because it would significantly disturb the results. Removing these would result in loosing almost all observations that is why it was decided to remove this variable

Variable duration was used in this dataset only for benchmark purposes. The duration of the contact is not known before a call so it is not possible to use this variable for the model, that is why it has been removed.

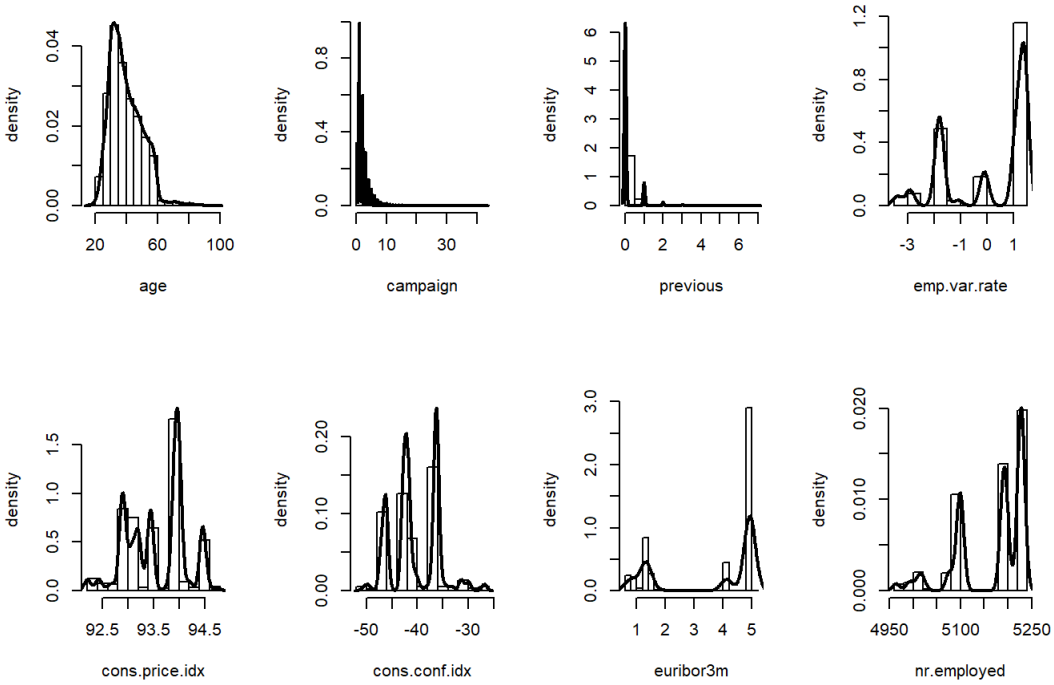
```
data <- data[!(apply(data,1,function(x) any(x=="unknown"))),]
```

All missing values (labeled with “unknown”) has been removed from the dataset

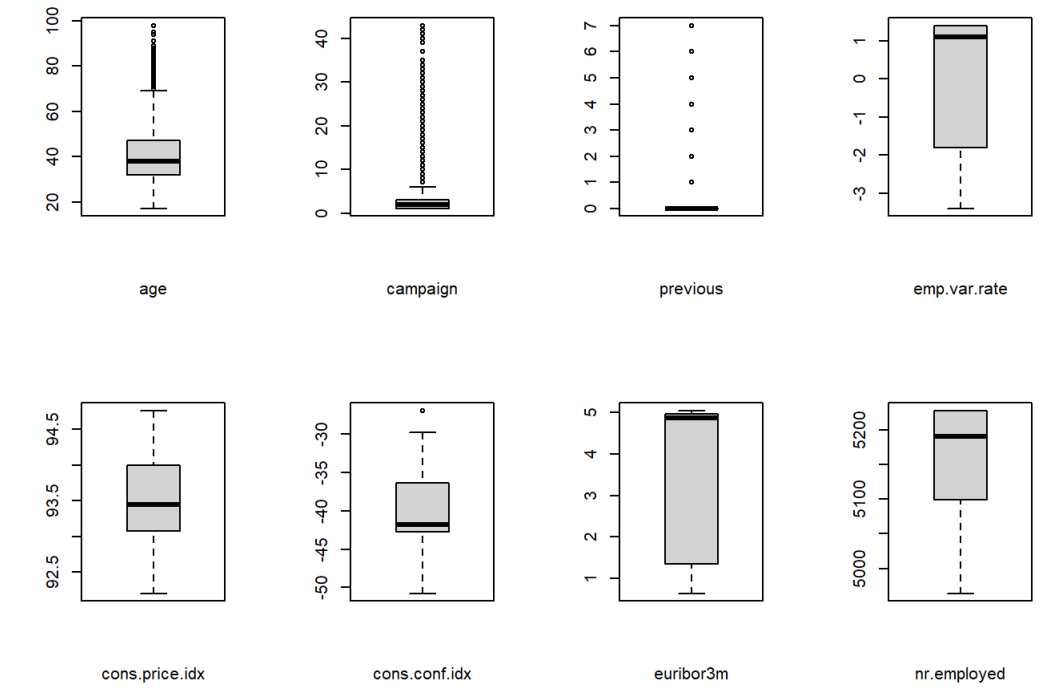
```
data$y <- ifelse(data$y=='yes', 1, 0)
factor_variables <- c(2:9,12)
data[,factor_variables] <- lapply(data[,factor_variables] , factor)
```

Target variable has been changed from character to numeric and all the categorical variables have been transformed from character to factor. These changes have been done in the purpose of the following data exploration.

```
Histogram <- function(values, values.name) {
  histogram <- hist(values, plot = FALSE)
  density.estimate <- density(values)
  y.maximum <- max(histogram$density, density.estimate$y)
  plot(histogram, freq = FALSE, ylim = c(0, y.maximum),
       xlab = values.name, ylab = "density", main = NULL)
  lines(density.estimate, lwd = 2)
}

par(mfrow=c(2,4))

Histogram(data$age, 'age')
Histogram(data$campaign, 'campaign')
Histogram(data$previous, 'previous')
Histogram(data$emp.var.rate, 'emp.var.rate')
Histogram(data$cons.price.idx, 'cons.price.idx')
Histogram(data$cons.conf.idx, 'cons.conf.idx')
Histogram(data$euribor3m, 'euribor3m')
Histogram(data$nr.employed, 'nr.employed')
```




```
par(mfrow=c(2,4))

boxplot(data$age, xlab = "age")
boxplot(data$campaign, xlab="campaign")
boxplot(data$previous, xlab="previous")
boxplot(data$emp.var.rate, xlab= 'emp.var.rate')
boxplot(data$cons.price.idx, xlab="cons.price.idx")
boxplot(data$cons.conf.idx, xlab='cons.conf.idx')
boxplot(data$euribor3m, xlab='euribor3m')
boxplot(data$nr.employed, xlab='nr.employed')
```




```
#outliers age
out_age <- boxplot.stats(data$age)$out
length(out_age)/nrow(data)
```

```
## [1] 0.01077265
```

```
#outliers emp.price.idx
boxplot.stats(data$emp.var.rate)$out
```

```
## numeric(0)
```

```
#outliers cons.price.idx
boxplot.stats(data$cons.price.idx)$out
```

```
## numeric(0)
```

```
#outliers cons.conf.idx
out_cons.conf.idx <- boxplot.stats(data$cons.conf.idx)$out
length(out_cons.conf.idx)/nrow(data)
```

```
## [1] 0.01069421
```

```
#outliers euribor3m
boxplot.stats(data$euribor3m)$out
```

```
## numeric(0)
```

```
#outliers nr.employed
boxplot.stats(data$nr.employed)$out
```

```
## numeric(0)
```

```
#outliers campaign
out_campaign <- boxplot.stats(data$campaign)$out
length(out_campaign)/nrow(data)
```

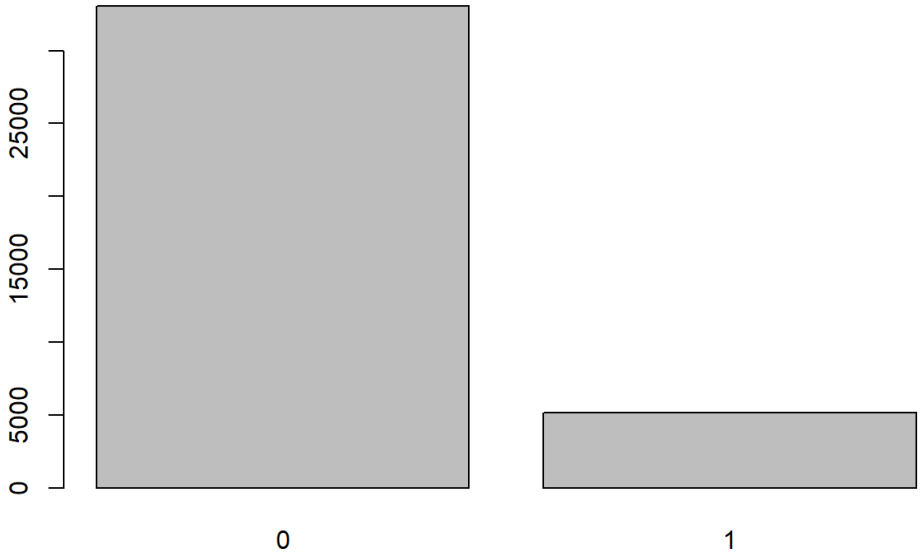
```
## [1] 0.0580468
```

```
#outliers previous
out_previous <- boxplot.stats(data$previous)$out
length(out_previous)/nrow(data)
```

```
## [1] 0.1354164
```

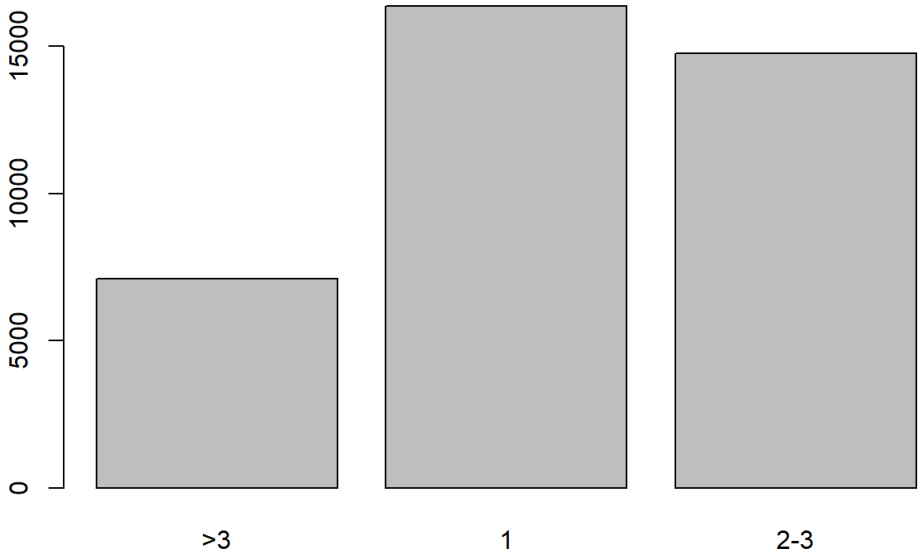
The next step of data cleaning and preprocessing was creating histograms and boxplots in order to check the distributions of all continuous variables. Moreover, the outliers detection has been performed using the IQR criterion. This criterion means that all observations above the value of third quartile + 1.5 x interquartile range and below the value of first quartile - 1.5 x interquartile range are considered as outliers. No outliers where found for variables: emp.price.idx, cons.price.idx, euribor3m, nr.employed. The fraction of outliers for variables age and cons.conf.idx are about 1% which is not much (this amount is acceptable). Also, their distributions do not show any significant anomaly, thus it was decided not to transform these variables. On the other hand, variables campaign and previous has the fraction of outliers respectively: 5% and 13%. Furthermore, their distribution are very skewed with the high majority of values below mean. It was decided to transform these variables.

```
data$previous <- ifelse(data$previous>0, 1, 0)
data$previous <- as.factor(data$previous)
barplot(table(data$previous))
```



It was decided to transform variable previous into the binary variable, where 0 means that no contacts were performed before this campaign for this client and 1 means that at least one contact was performed. The bar plot of transformed variable is presented above.

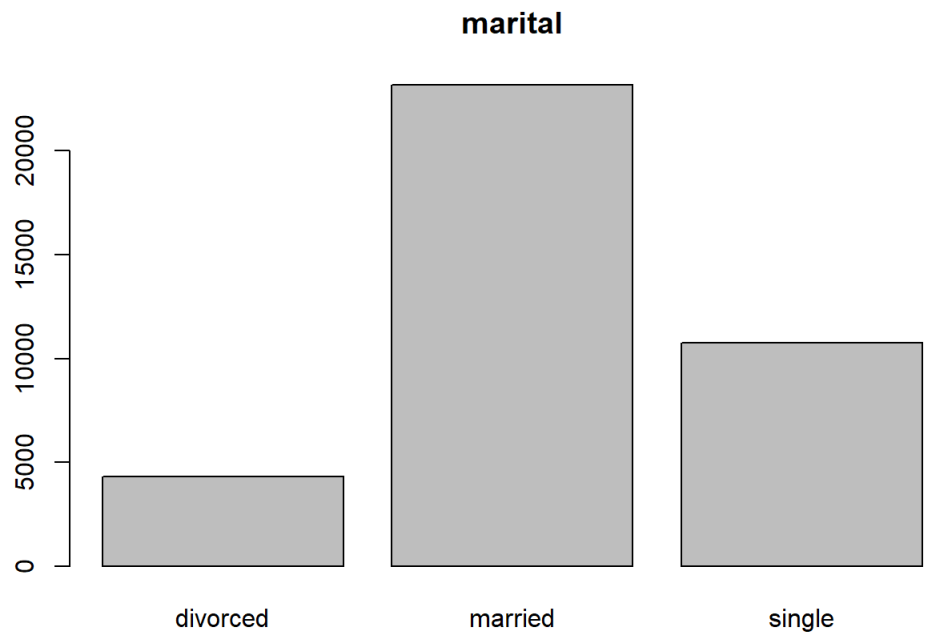
```
data$campaign <- ifelse(data$campaign>3, '>3', ifelse(data$campaign >1, '2-3', '1'))
data$campaign <- as.factor(data$campaign)
barplot(table(data$campaign))
```



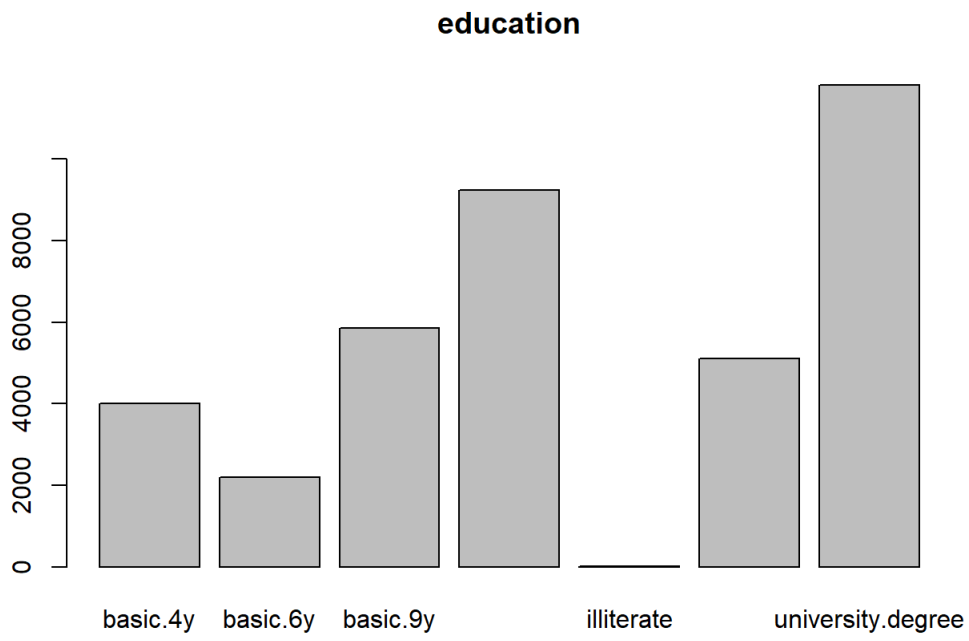
Variable campaign was decided to be transformed into categorical variable with 3 categories: “>3” meaning that more than 3 contacts were performed during this campaign and for this client, “2-3” meaning that between 2 and 3 contacts were performed and “1” meaning that 1 contact was performed. The bar plot of transformed variable is presented above.

```
par(mfrow=c(1,1))

barplot(table(data$job), main="job")
```

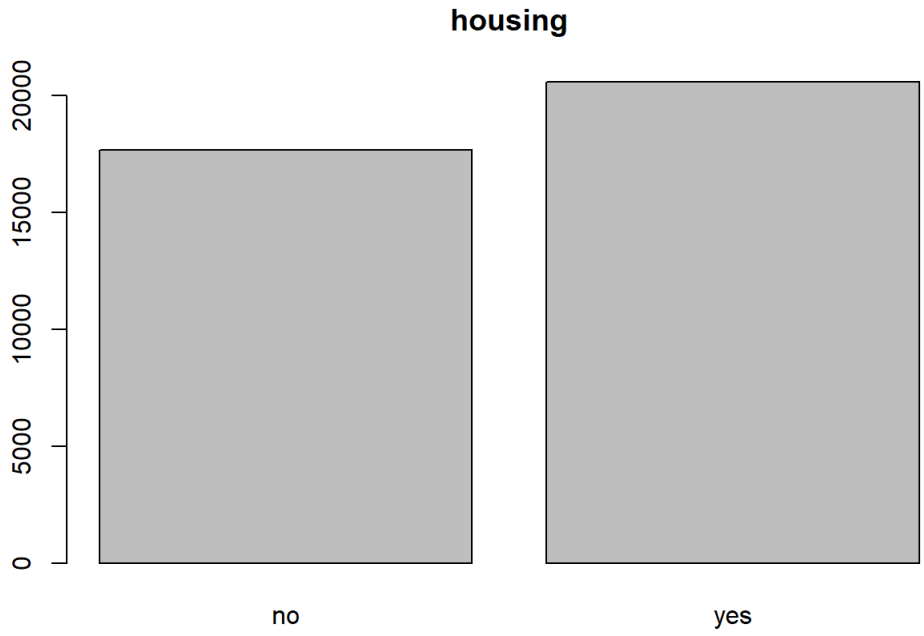

```
barplot(table(data$marital),main="marital" )
```



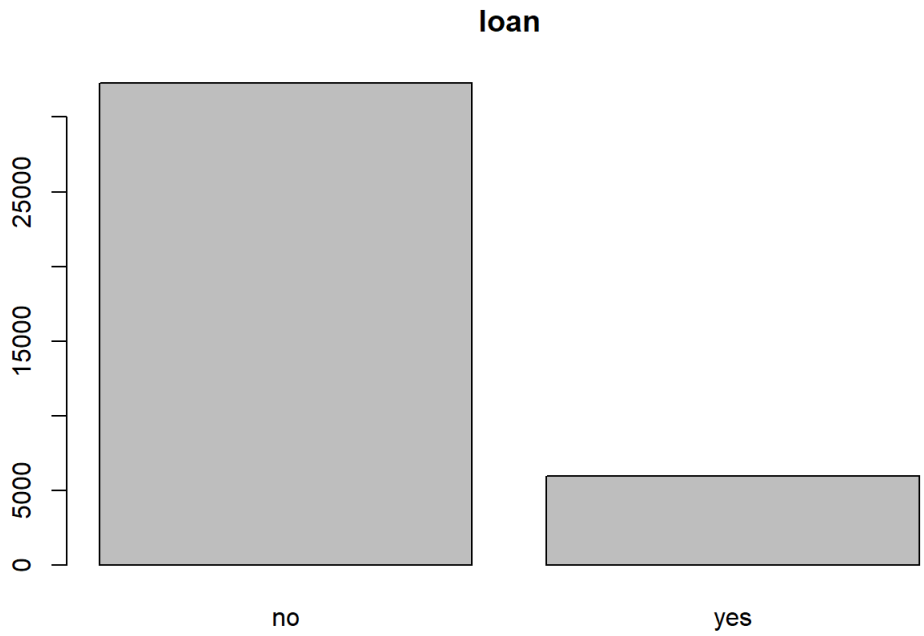

```
barplot(table(data$education),main="education")
```



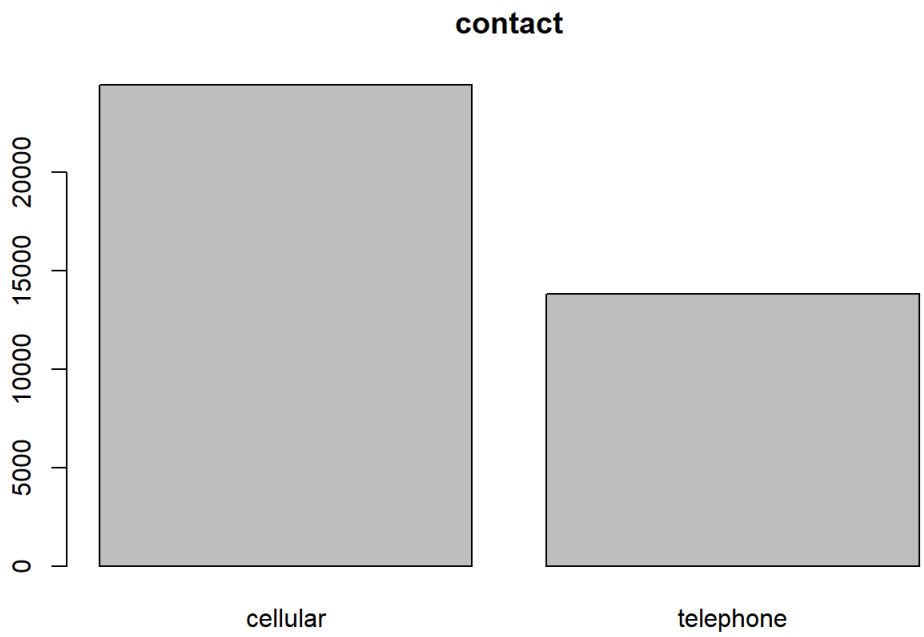

```
barplot(table(data$housing), main="housing")
```



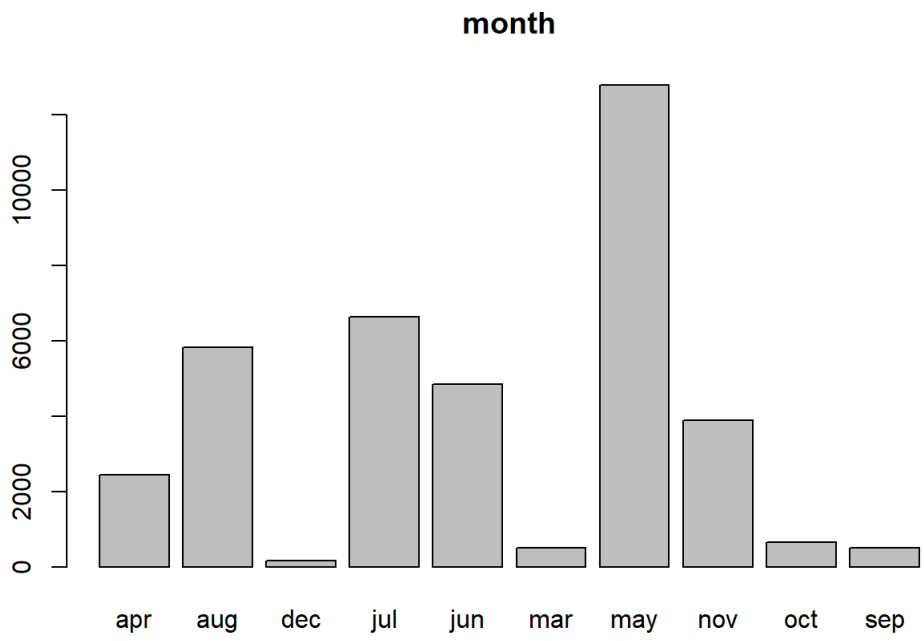

```
barplot(table(data$loan), main="loan")
```



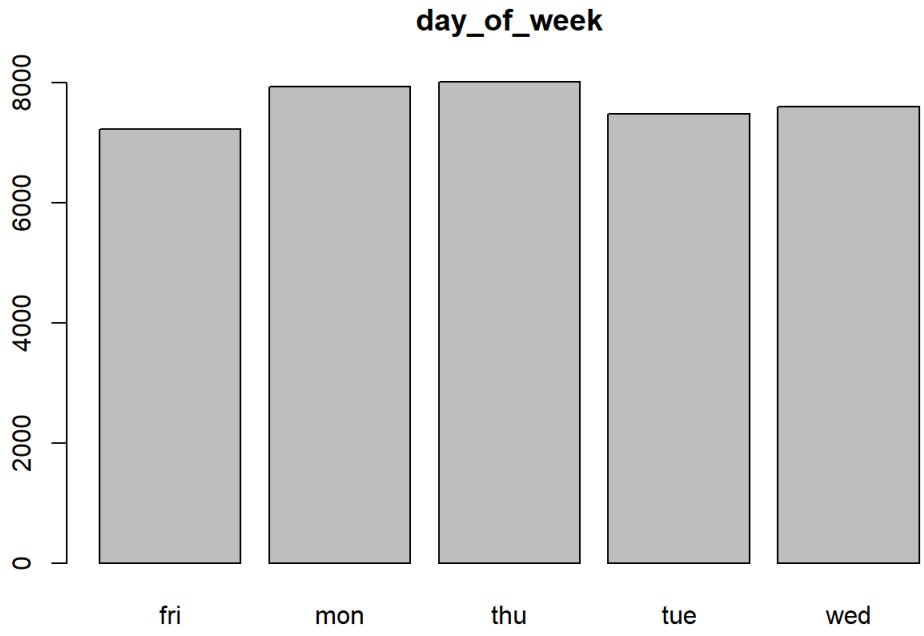

```
barplot(table(data$contact), main="contact")
```



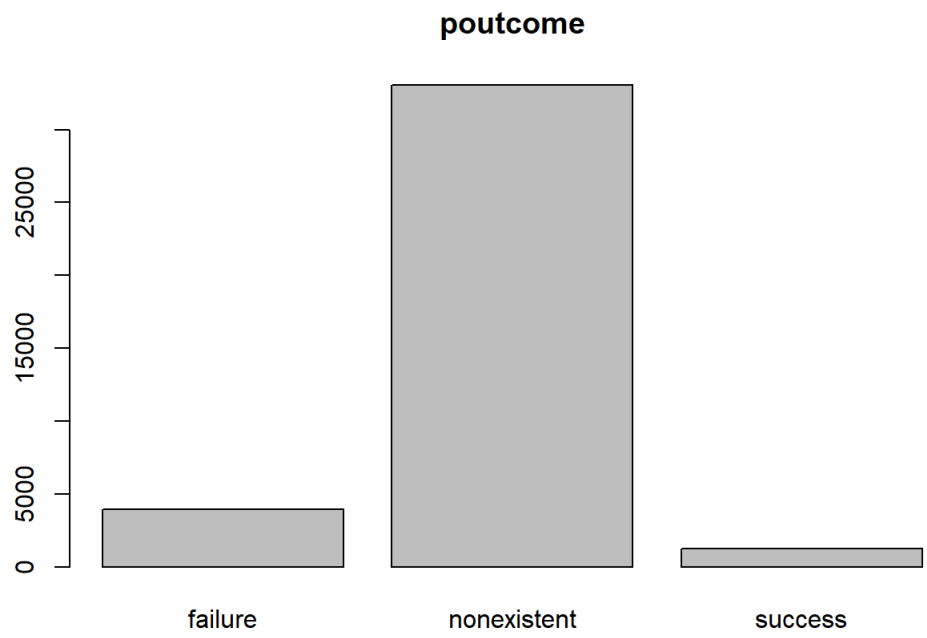

```
barplot(table(data$month), main="month")
```




```
barplot(table(data$day_of_week), main="day_of_week")
```

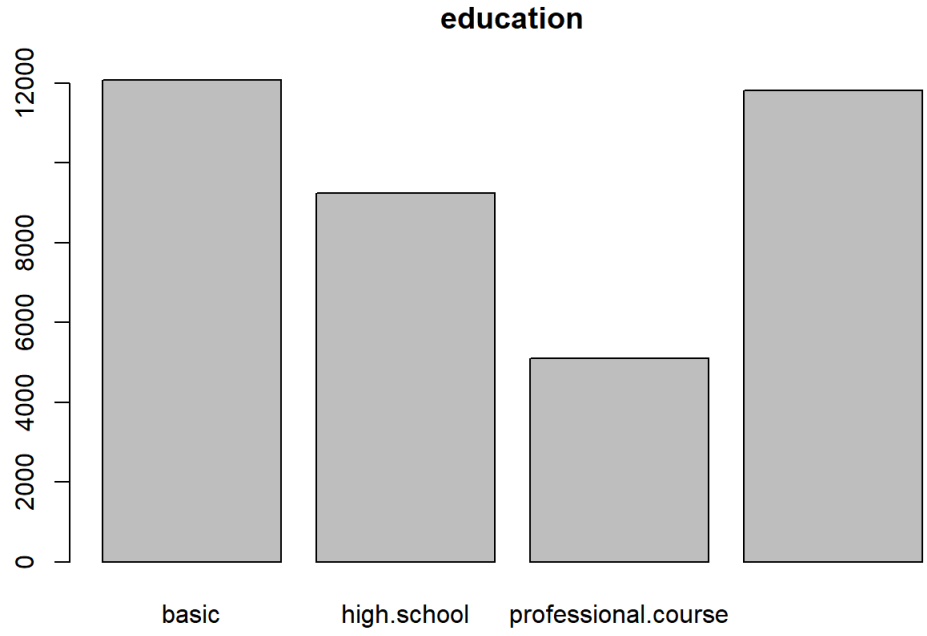



```
barplot(table(data$poutcome), main="poutcome")
```



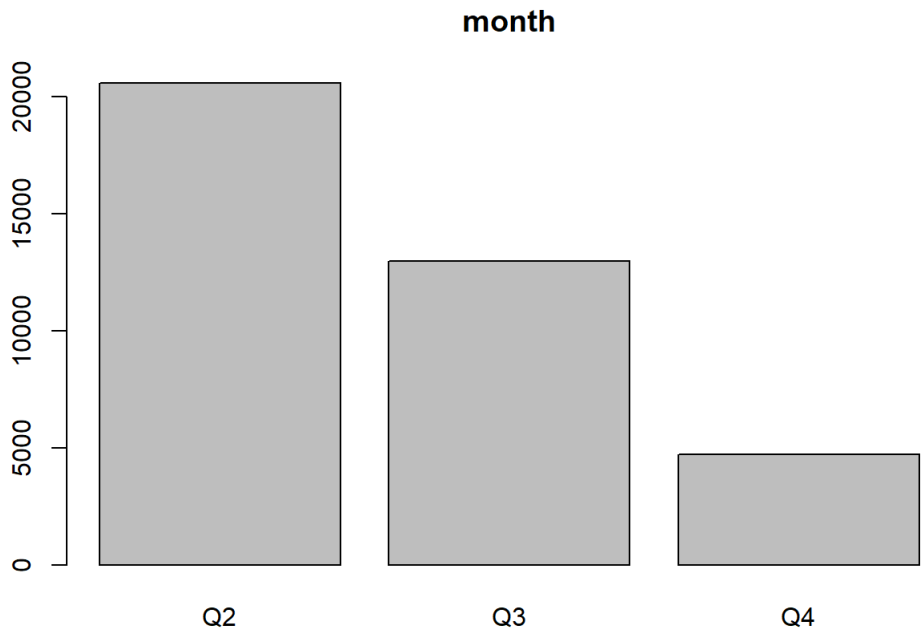
Above there are presented bar plots of all categorical variables. The profound analysis of each variable has been performed. It was checked whether the number of observation in certain category is not too little. Everywhere where it was needed and where it was substantively possible the categories merger was performed.

```
library(plyr)

data$education <- revalue(data$education, c("basic.4y"="basic", "basic.6y"="basic", "basic.9y"='basic', 'illiterate'='basic'))
par(mfrow=c(1,1))
barplot(table(data$education),main="education")
```



For variable education all categories describing basic level of education + the “illiterate” category has been merged into one. As the result the new bar plot is presented above, where the number of observations in each category is much more balanced.

```
data$month <- revalue(data$month, c("mar"="Q2", "apr" = "Q2", "may"="Q2", "jun"="Q2", "jul"="Q3", "aug"="Q3", "sep"="Q3", "oct"="Q4", "nov"="Q4", "dec"="Q4"))
barplot(table(data$month), main="month")
```



Months has been merged into quarters. There are no observations for January and February and there is very little observation for March. These observations has been added to Q2 (second quarter). As a result the new bar plot is presented above, where the number of observations in each category is much more balanced.

```
data$job <- as.character(data$job)
job <- strsplit(data$job,split="\\|")
job_binary <- sapply(unique(data$job), function(x) sapply(data$job, function(y) x%in%y))
data<-cbind(data, job_binary)

data$marital <- as.character(data$marital)
marital_binary <- sapply(unique(data$marital), function(x) sapply(data$marital, function(y) x%in%y))
data<-cbind(data, marital_binary)

data$education <- as.character(data$education)
education_binary <- sapply(unique(data$education), function(x) sapply(data$education, function(y) x%in%y))
data<-cbind(data, education_binary)

data$housing <- ifelse(data$housing == "yes", "housing_yes", "housing_no")
housing_binary <- sapply(unique(data$housing), function(x) sapply(data$housing, function(y) x%in%y))
data<-cbind(data, housing_binary)

data$loan <- ifelse(data$loan == "yes", "loan_yes", "loan_no")
loan_binary <- sapply(unique(data$loan), function(x) sapply(data$loan, function(y) x%in%y))
data<-cbind(data, loan_binary)

data$contact <- as.character(data$contact)
contact_binary <- sapply(unique(data$contact), function(x) sapply(data$contact, function(y) x%in%y))
data<-cbind(data, contact_binary)

data$month <- as.character(data$month)
month_binary <- sapply(unique(data$month), function(x) sapply(data$month, function(y) x%in%y))
data<-cbind(data, month_binary)

data$day_of_week <- as.character(data$day_of_week)
day_of_week_binary <- sapply(unique(data$day_of_week), function(x) sapply(data$day_of_week, function(y) x%in%y))
data<-cbind(data, day_of_week_binary)

data$campaign <- revalue(data$campaign, c('1'="campaign_1", "2-3"= "campaign_2_3", ">3"="campaign_3"))
data$campaign <- as.character(data$campaign)
campaign_binary <- sapply(unique(data$campaign), function(x) sapply(data$campaign, function(y) x%in%y))
data<-cbind(data, campaign_binary)

data$previous <- revalue(data$previous, c('1'="previous_1", "0"= "previous_0"))
data$previous <- as.character(data$previous)
previous_binary <- sapply(unique(data$previous), function(x) sapply(data$previous, function(y) x%in%y))
data<-cbind(data, previous_binary)

data$poutcome <- as.character(data$poutcome)
poutcome_binary <- sapply(unique(data$poutcome), function(x) sapply(data$poutcome, function(y) x%in%y))
data<-cbind(data, poutcome_binary)

colnames(data) <- gsub("-","",colnames(data))

##removing variables changed to binary using one-hot encoding
data<- data[,-c(2,3,4,5,6,7,8,9,10,11,12)]
```

The next step was to convert all categorical variables using one-hot encoding. As a result from each variable the n number of new binary variables were created (where n stands for number of categories for certain variable). Each new variable has the value TRUE when the observation belongs to certain category and value FALSE if not. Machine Learning algorithms have in general better results when the variables are presented in this way, that is why one-hot encoding has been performed and variables before transformations have been removed from the dataset.

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

The target variable has been transformed into factor variable in order to be able to build classification models.

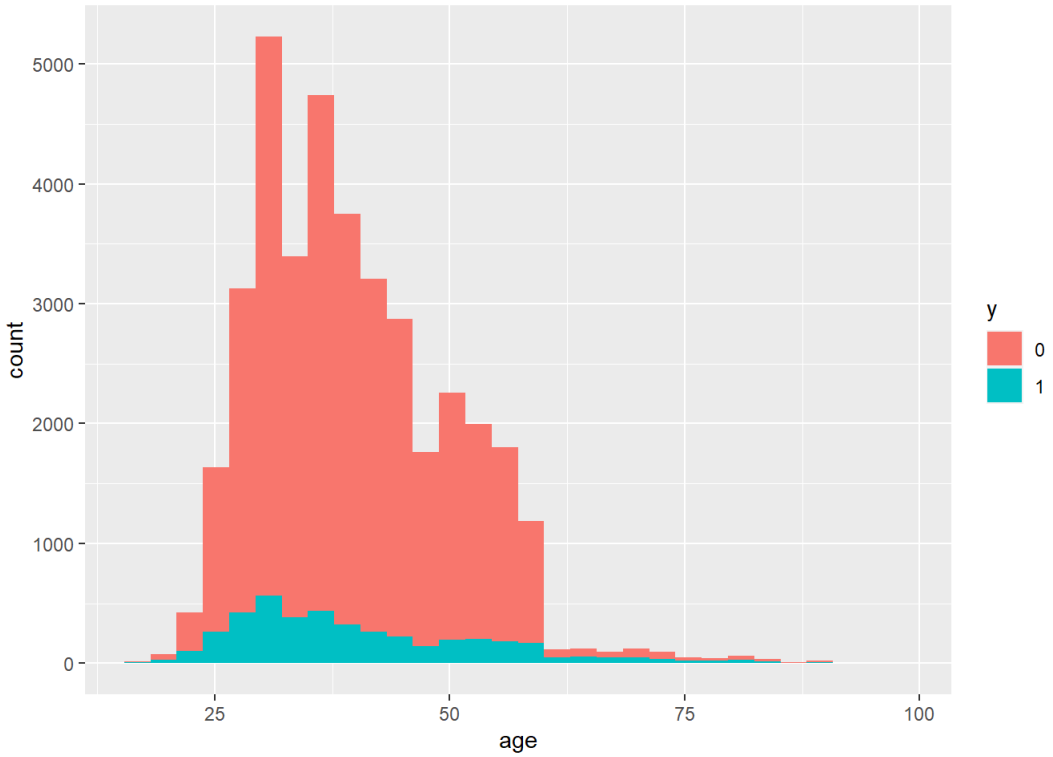
```
SCALED_DATA <- scale(data[c(1:6)])
FINAL_SCALED_DATA <- cbind(SCALED_DATA, data[-c(1:6)])
```

For the purpose of creating neural network the data has been standardized using z-score. In this case the new value of each observation is calculated as the mean of the variable subtracted from the value of observation and divided by standard deviation of the variable. However, for classification tree and random forest data before standardization will be used as these methods are not sensitive to the magnitude of variables.

Exploratory Data Analysis

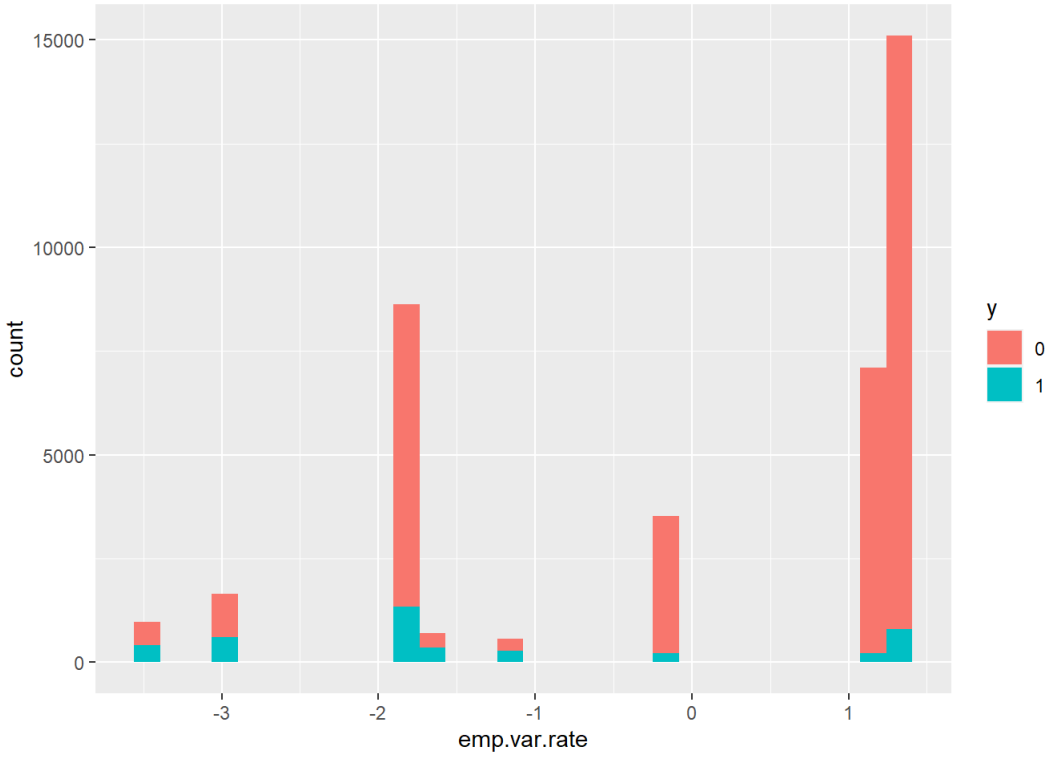
```
library(gridExtra)
library(ggplot2)

ggplot(data, aes(x=age, fill=y)) +
  geom_histogram()
```

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



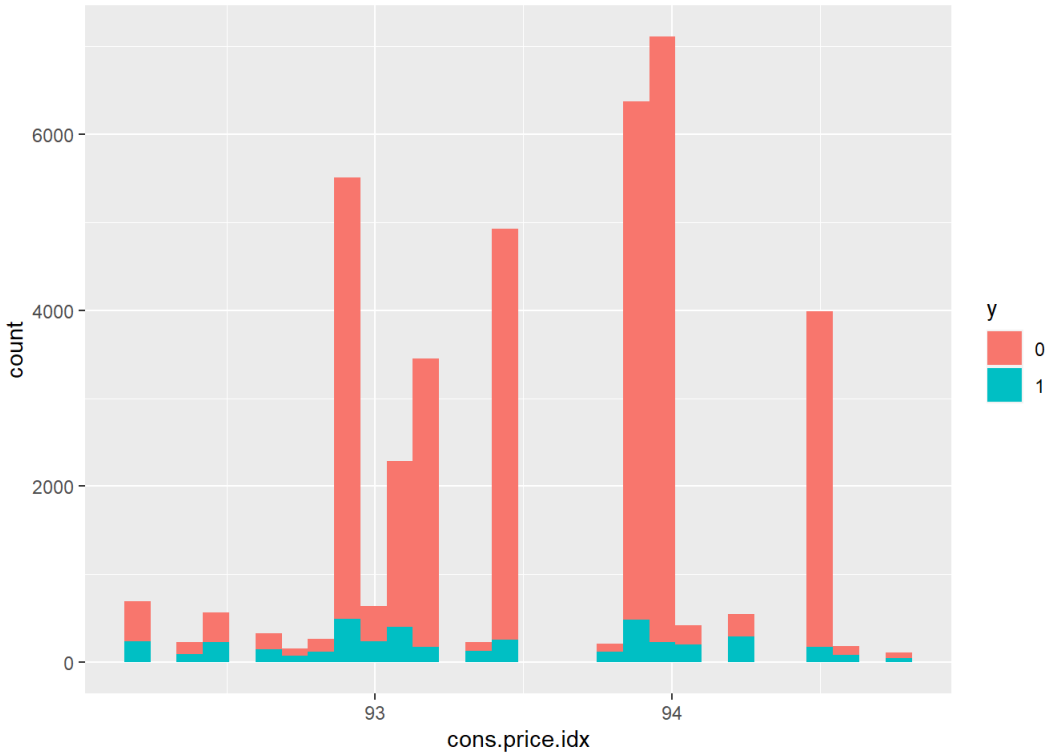

```
ggplot(data, aes(x=emp.var.rate, fill=y)) +
  geom_histogram()
```

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



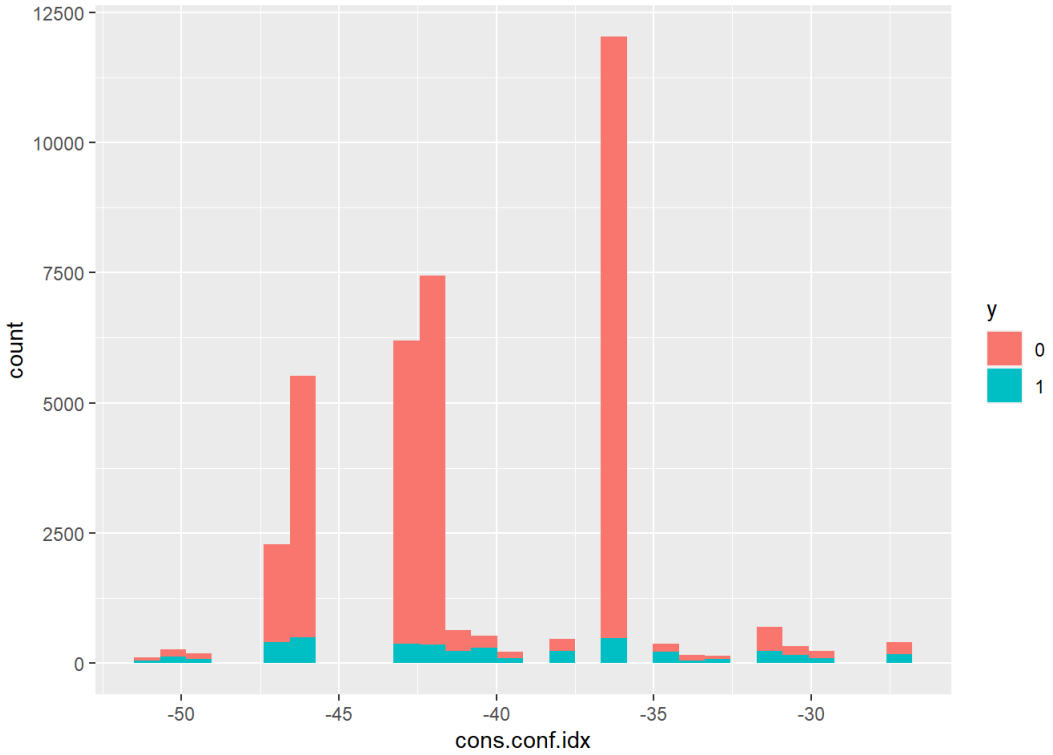

```
ggplot(data, aes(x=cons.price.idx, fill=y)) +
  geom_histogram()
```

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



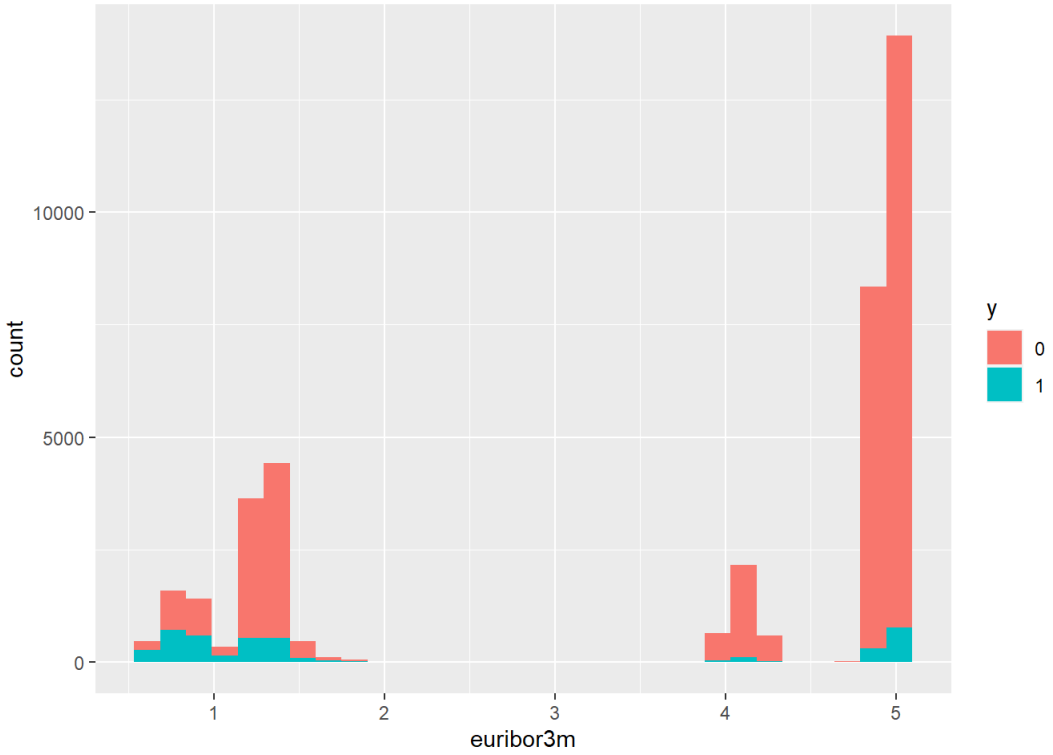

```
ggplot(data, aes(x=cons.conf.idx, fill=y)) +
  geom_histogram()
```

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



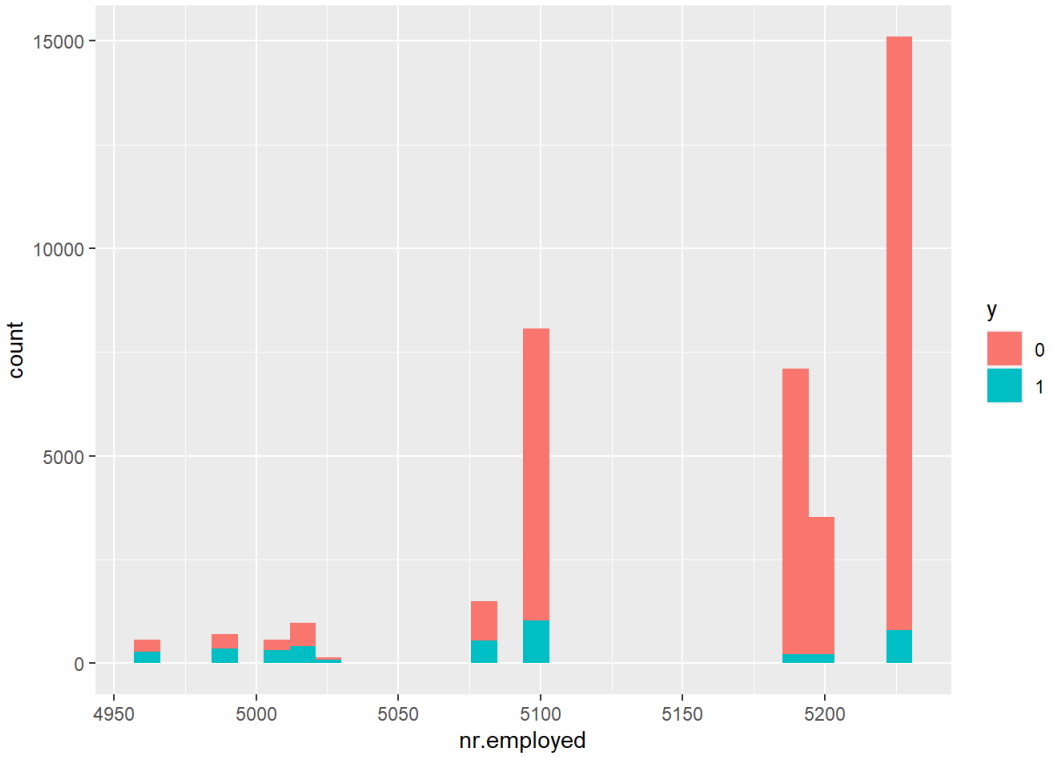

```
ggplot(data, aes(x=euribor3m, fill=y)) +
  geom_histogram()
```

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



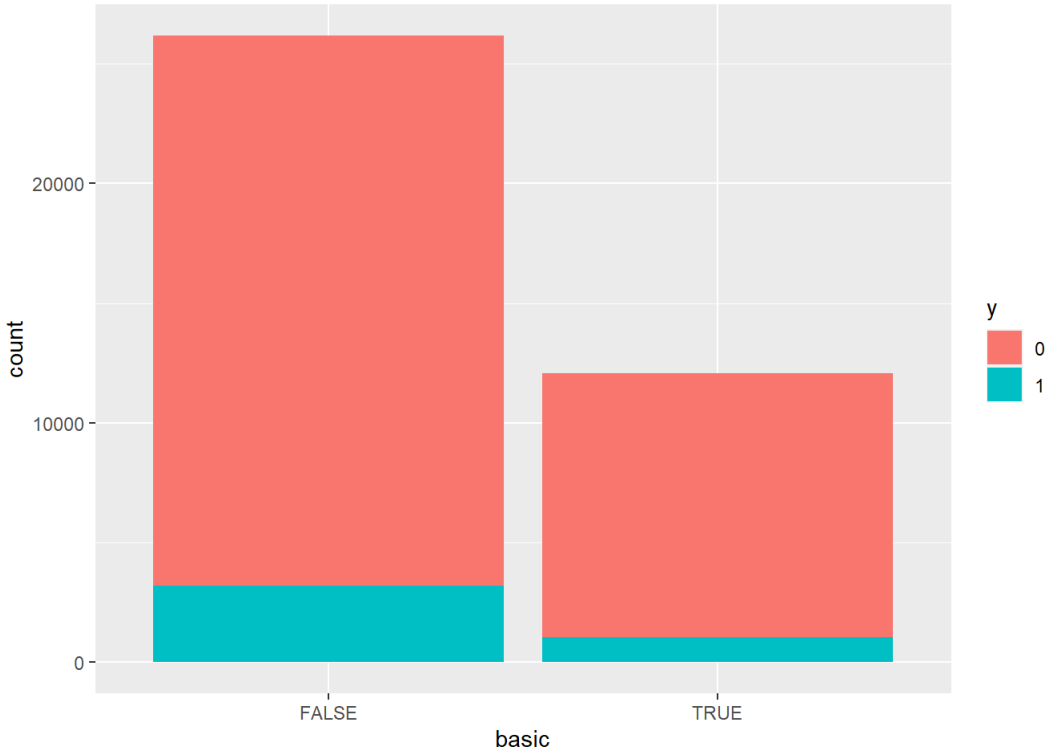

```
ggplot(data, aes(x=nr.employed, fill=y)) +
  geom_histogram()
```

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



Above there can be observed stacked histograms for numeric variables. Each histogram bar is split into 2 parts corresponding with the number of observations belonging to one of 2 values of target variable. These plots are useful when we want to check what was the fraction of positive/negative values of target variable in each range of explanatory variables.

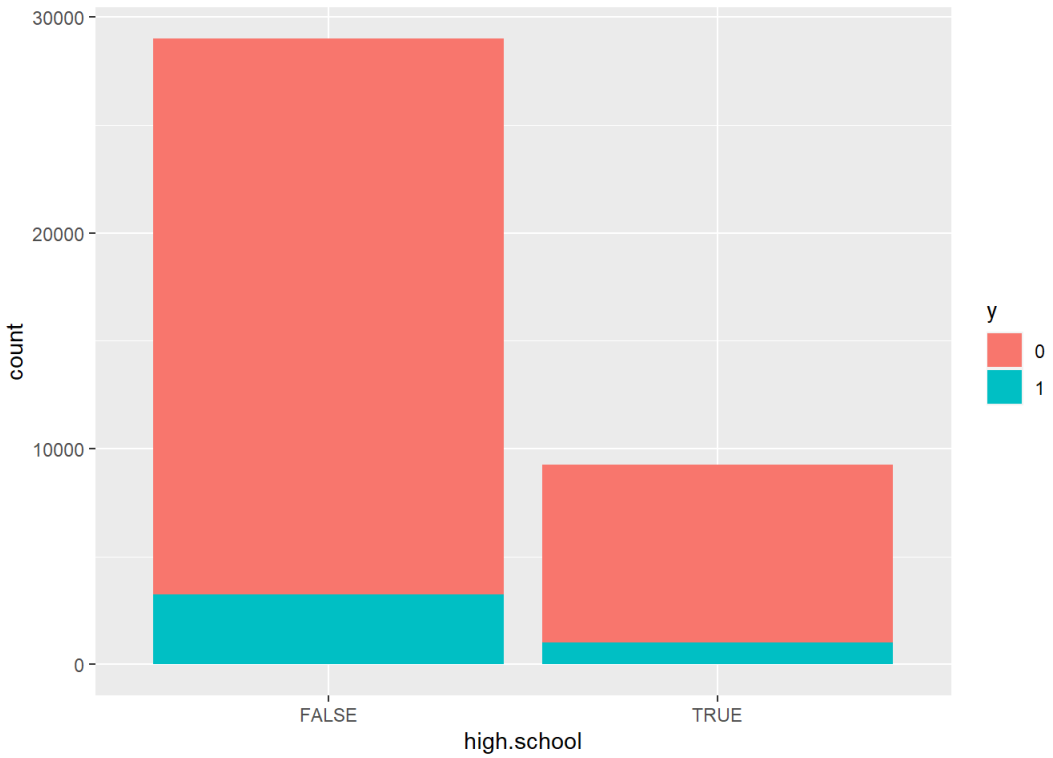
```
ggplot(data, aes(x=basic, fill=y)) +
  geom_bar()
```




```
e1<- table(data$basic, data$y)
prop.table(e1, margin = 1)
```

```
##
##                0         1
##   FALSE 0.87750812 0.12249188
##   TRUE  0.91283113 0.08716887
```

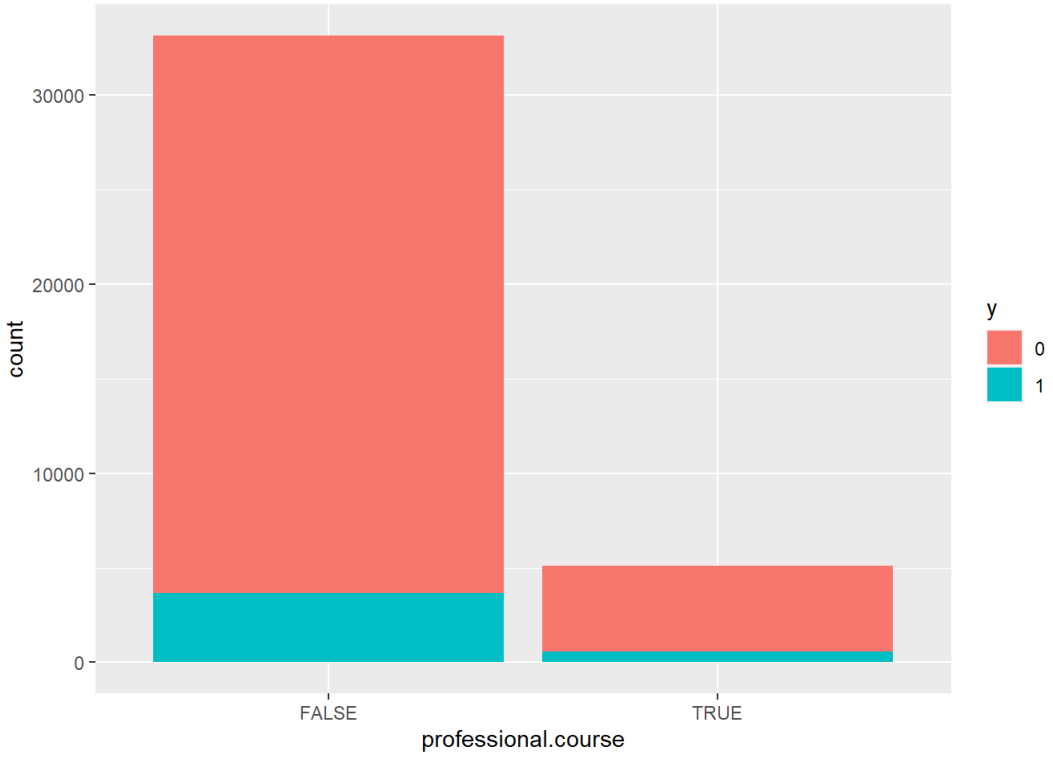
```
ggplot(data, aes(x=high.school, fill=y)) +
  geom_bar()
```




```
e2<- table(data$high.school, data$y)
prop.table(e2, margin = 1)
```

```
##
##                0         1
##   FALSE 0.8879004 0.1120996
##   TRUE  0.8910645 0.1089355
```

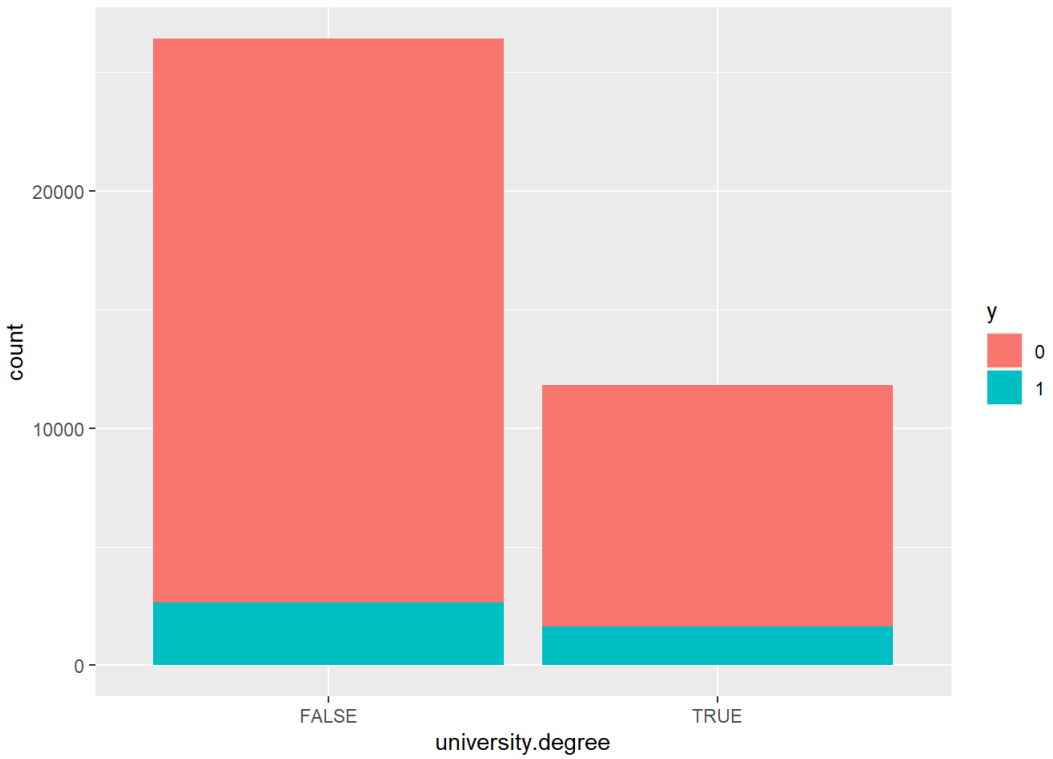
```
ggplot(data, aes(x=professional.course, fill=y)) +
 geom_bar()
```




```
e3<- table(data$professional.course, data$y)
prop.table(e3, margin = 1)
```

```
##
##                0         1
##   FALSE 0.8889727 0.1110273
##   TRUE  0.8866667 0.1133333
```

```
ggplot(data, aes(x=university.degree, fill=y)) +
 geom_bar()
```



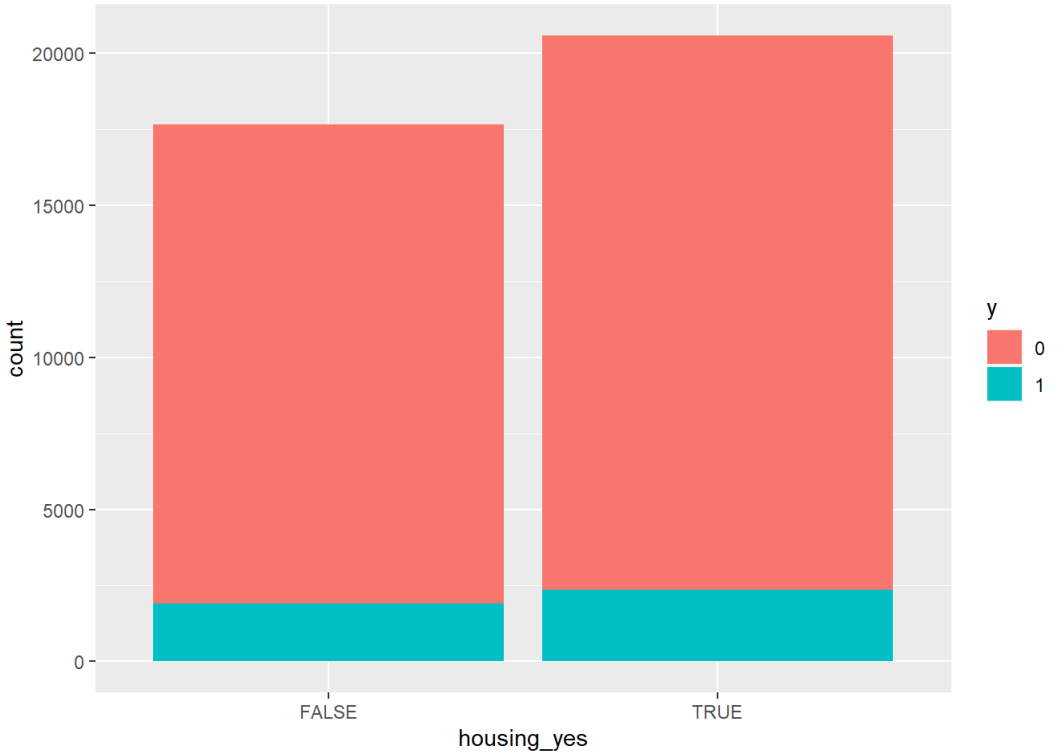

```
e4<- table(data$university.degree, data$y)
prop.table(e4, margin = 1)
```

```
##
##                0         1
##   FALSE 0.90016652 0.09983348
##   TRUE  0.86295576 0.13704424
```

In the next part of exploratory data analysis variables created by one-hot encoding has been presented on the stacked bar charts. Colors of bars corresponds with the fraction of positive/negative values of target variables in each group of binary variable. Below each chart there is also presented the table which shows the exact fraction values of positive/negative observations in each group.

When it comes to education it can be observed that people who have higher than basic education tend to subscribe term deposits more often than people who have basic education (12,2% compared to 8,7%). On the other hand people who have university degree subscribe term deposit more often than people who do not have university degree (13,7%, compared to 9,9%)

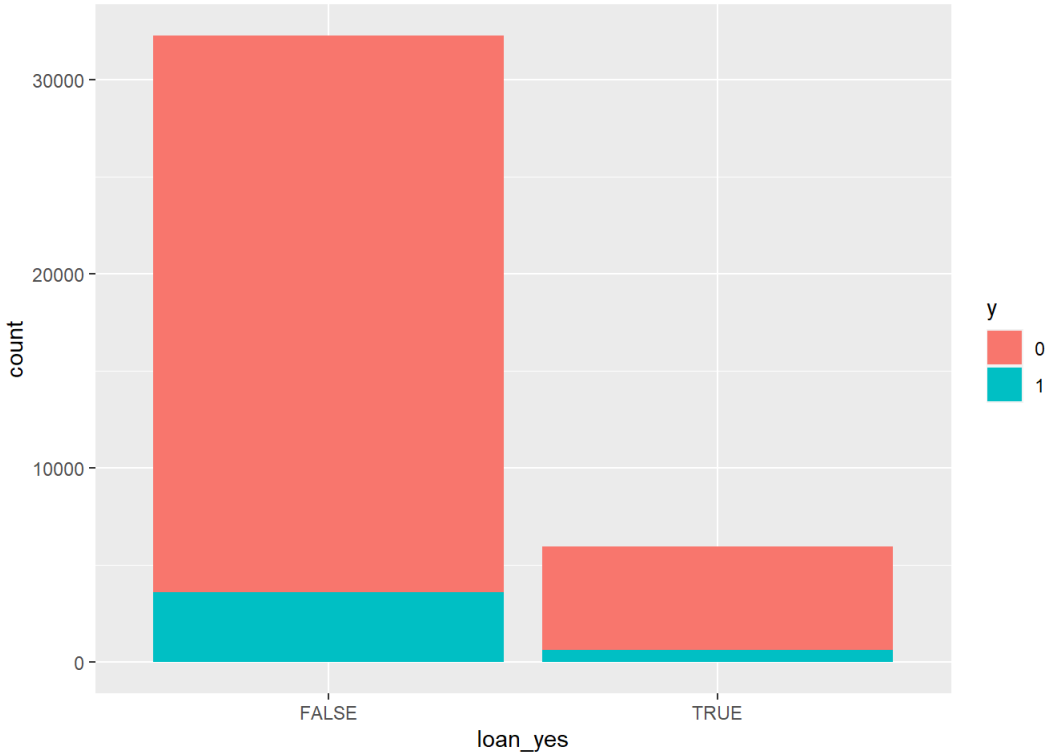
```
ggplot(data, aes(x=housing_yes, fill=y)) +
 geom_bar()
```




```
h1<- table(data$housing_yes, data$y)
prop.table(h1, margin = 1)
```

```
##
##                0         1
##   FALSE 0.8920586 0.1079414
##   TRUE  0.8857518 0.1142482
```

```
ggplot(data, aes(x=loan_yes, fill=y)) +
 geom_bar()
```

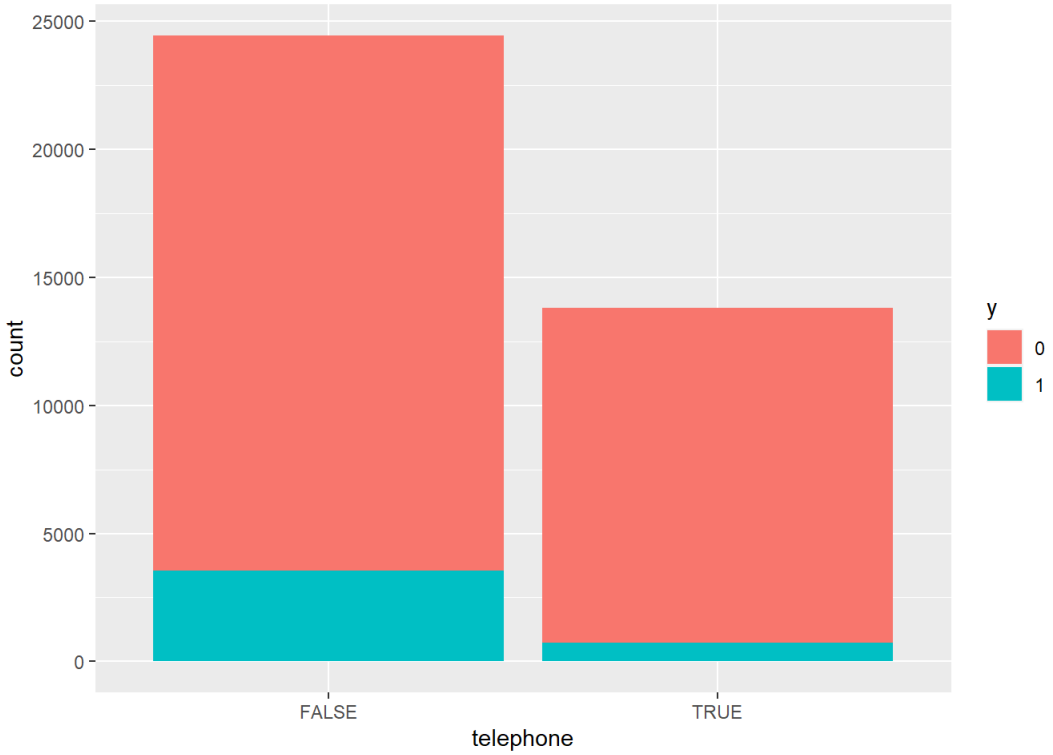



```
l1<- table(data$loan_yes, data$y)
prop.table(l1, margin = 1)
```

```
##
##               0         1
##   FALSE 0.8879081 0.1120919
##   TRUE  0.8927672 0.1072328
```

In case of people who have housing loan and personal loan there is no significant difference in the willing of subscribing term deposit in comparison with people who do not have such loan. The percentage values in all groups are very similar.

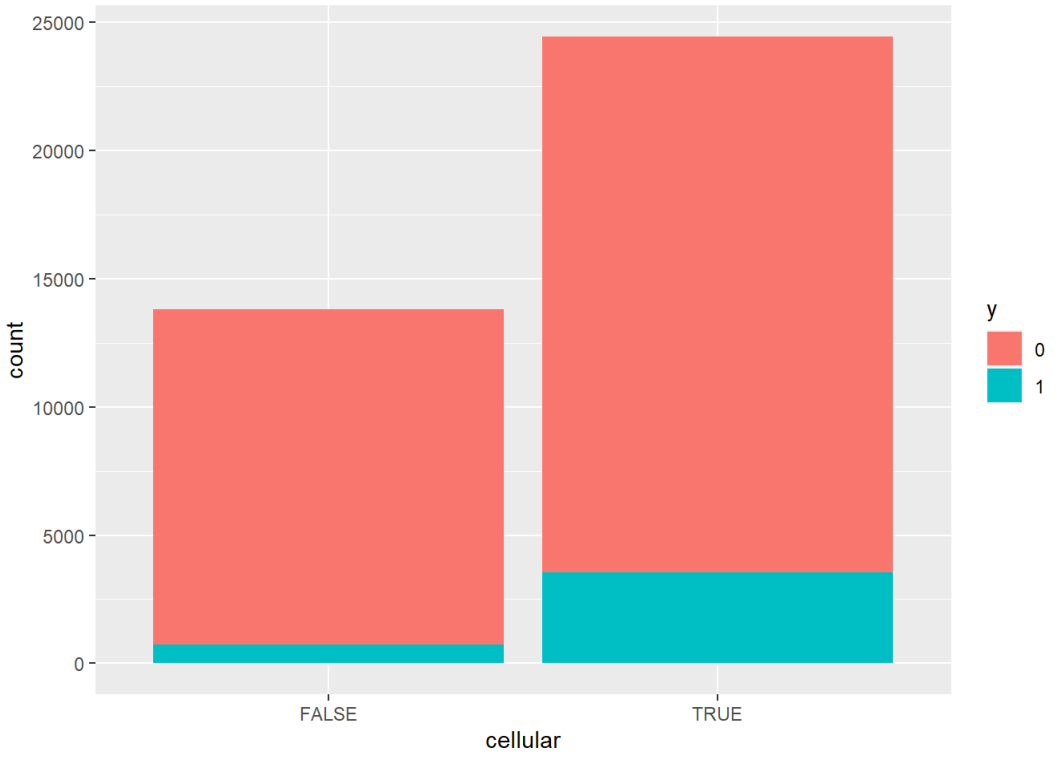
```
ggplot(data, aes(x=telephone, fill=y)) +
  geom_bar()
```




```
c1<- table(data$telephone, data$y)
prop.table(c1, margin = 1)
```

```
##
##                0         1
##   FALSE 0.85536598 0.14463402
##   TRUE  0.94762388 0.05237612
```

```
ggplot(data, aes(x=cellular, fill=y)) +
  geom_bar()
```




```
c2<- table(data$cellular, data$y)
prop.table(c2, margin = 1)
```

```
##
##                0         1
##   FALSE 0.94762388 0.05237612
##   TRUE  0.85536598 0.14463402
```

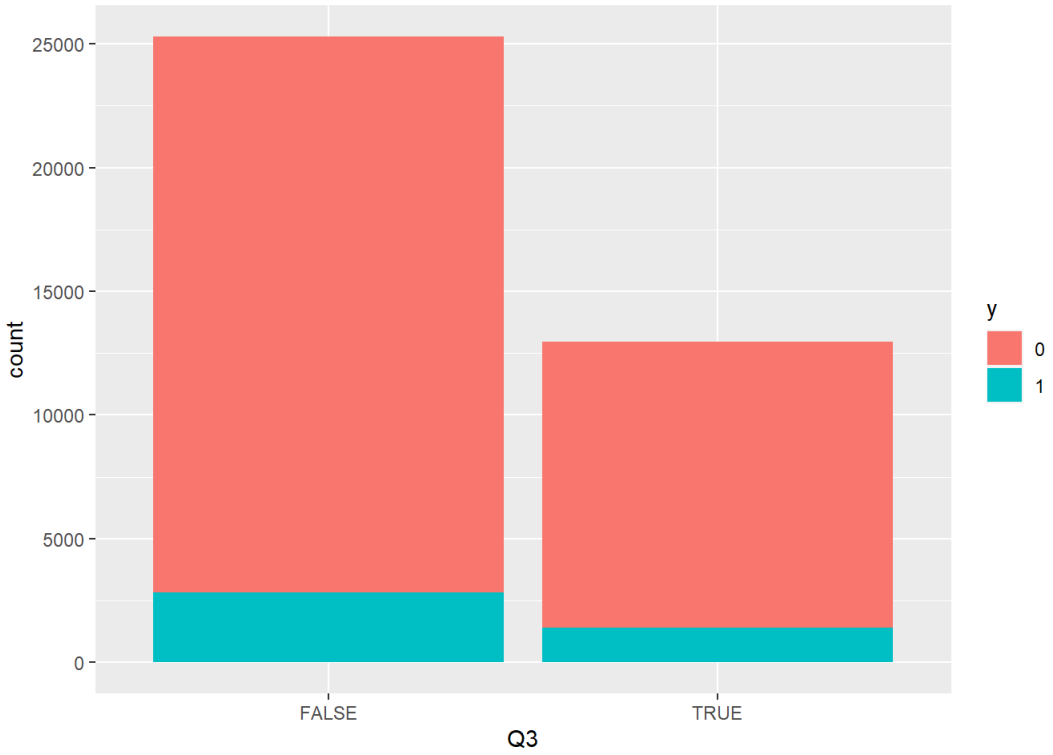
Way of communication seems to have huge importance on the target variable. Over 14% of people contacted by cellular have subscribed a term deposit compared with only 5% of people contacted by telephone.

```
ggplot(data, aes(x=Q2, fill=y)) +
  geom_bar()
```


```
m1<- table(data$Q2, data$y)
prop.table(m1, margin = 1)
```

```
##
##              0         1
##   FALSE 0.8766976 0.1233024
##   TRUE  0.8989452 0.1010548
```

```
ggplot(data, aes(x=Q3, fill=y)) +
  geom_bar()
```




```
m2<- table(data$Q3, data$y)
prop.table(m2, margin = 1)
```

```
##
##              0         1
##   FALSE 0.8877862 0.1122138
##   TRUE  0.8903813 0.1096187
```

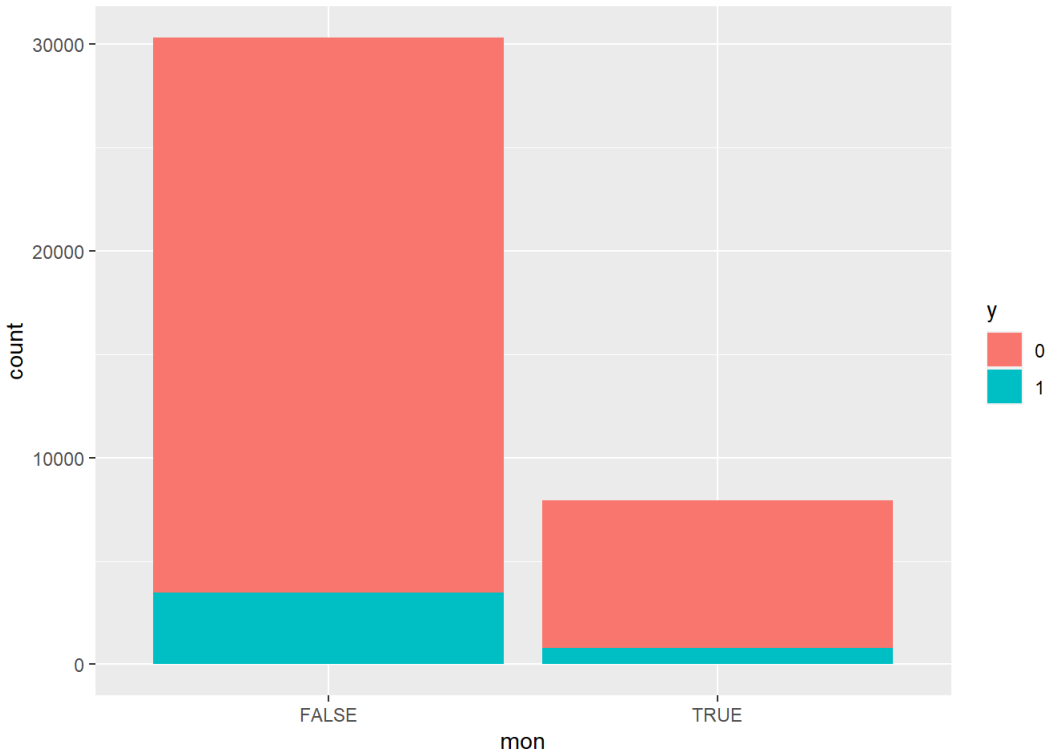
```
ggplot(data, aes(x=Q4, fill=y)) +
  geom_bar()
```


```
m3<- table(data$Q4, data$y)
prop.table(m3, margin = 1)
```

```
##
##                0         1
##   FALSE 0.8956364 0.1043636
##   TRUE  0.8391267 0.1608733
```

The period of year when the last contact with client was performed is important in determining whether the client will subscribe a term deposit. People contacted during last quarter of the year have a chances of positive value of target variable of 16%, while people contacted during first and second quarters have chances of around 10%.

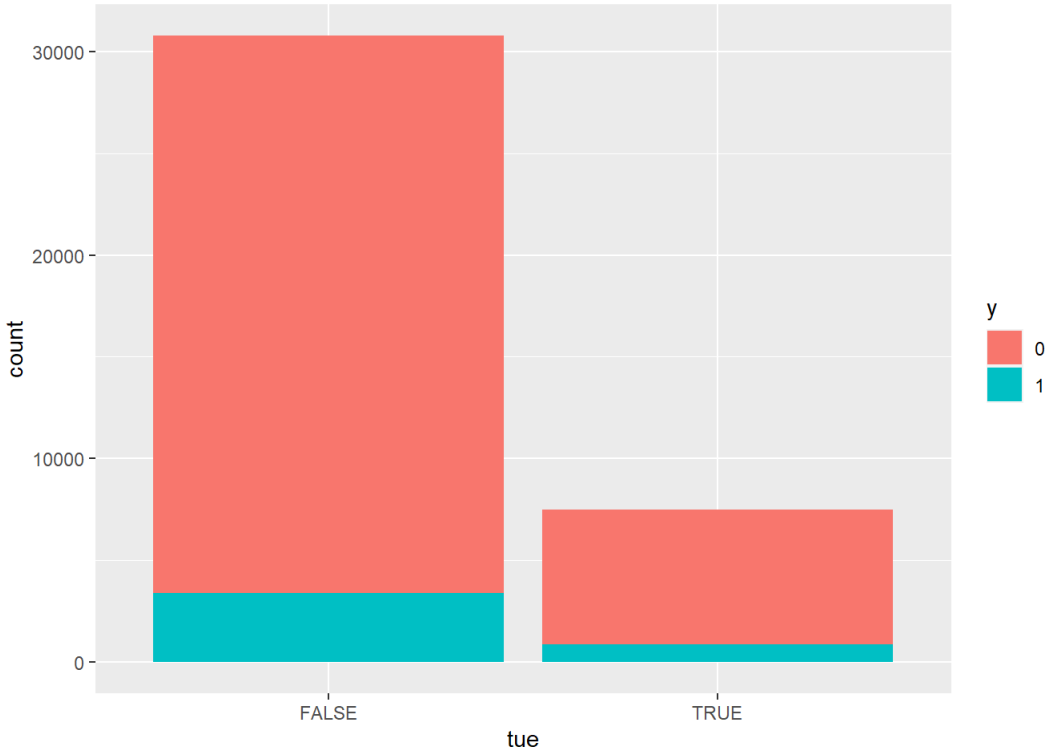
```
ggplot(data, aes(x=mon, fill=y)) +
  geom_bar()
```



```
day1<- table(data$mon, data$y)
prop.table(day1, margin = 1)
```

```
##
##                0         1
##   FALSE 0.88548057 0.11451943
##   TRUE  0.90084521 0.09915479
```

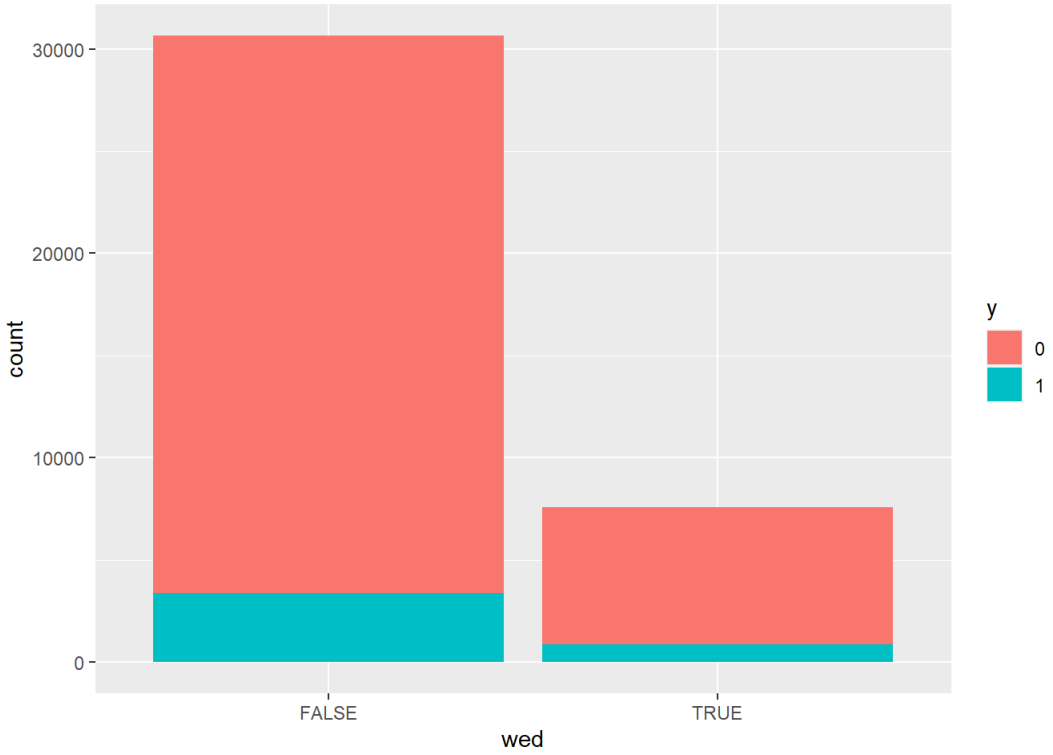
```
ggplot(data, aes(x=tue, fill=y)) +
 geom_bar()
```




```
day2<- table(data$tue, data$y)
prop.table(day2, margin = 1)
```

```
##
##                0         1
##   FALSE 0.8896762 0.1103238
##   TRUE  0.8845074 0.1154926
```

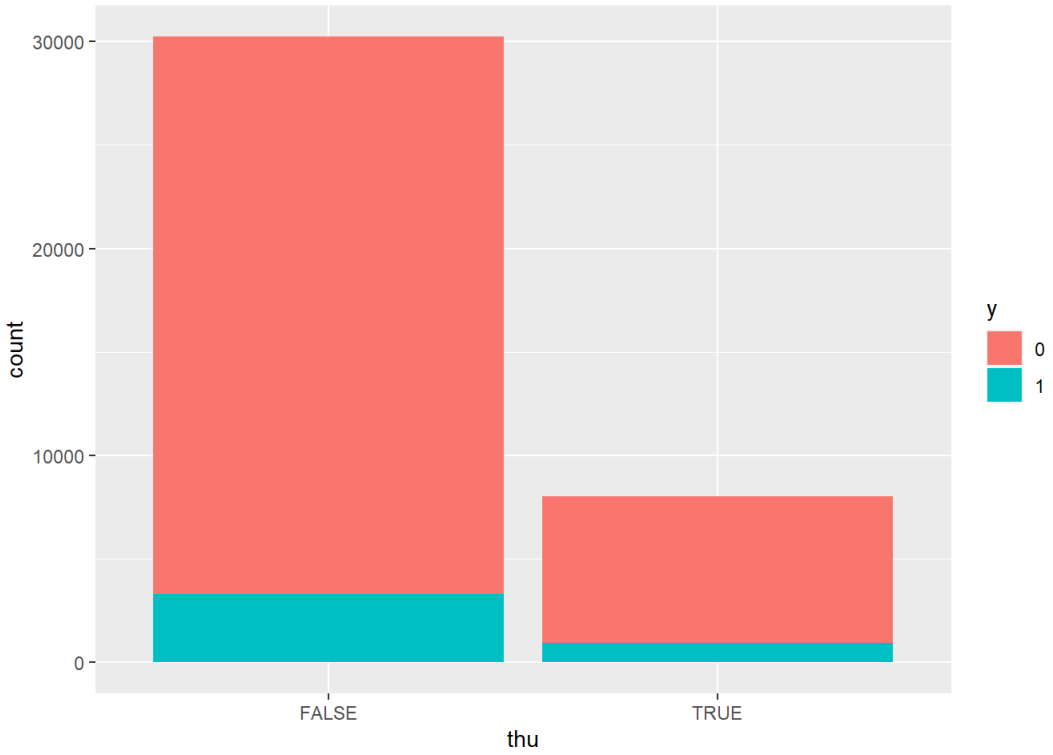
```
ggplot(data, aes(x=wed, fill=y)) +
 geom_bar()
```




```
day3<- table(data$wed, data$y)
prop.table(day3, margin = 1)
```

```
##
##                0         1
##   FALSE 0.8895996 0.1104004
##   TRUE  0.8848987 0.1151013
```

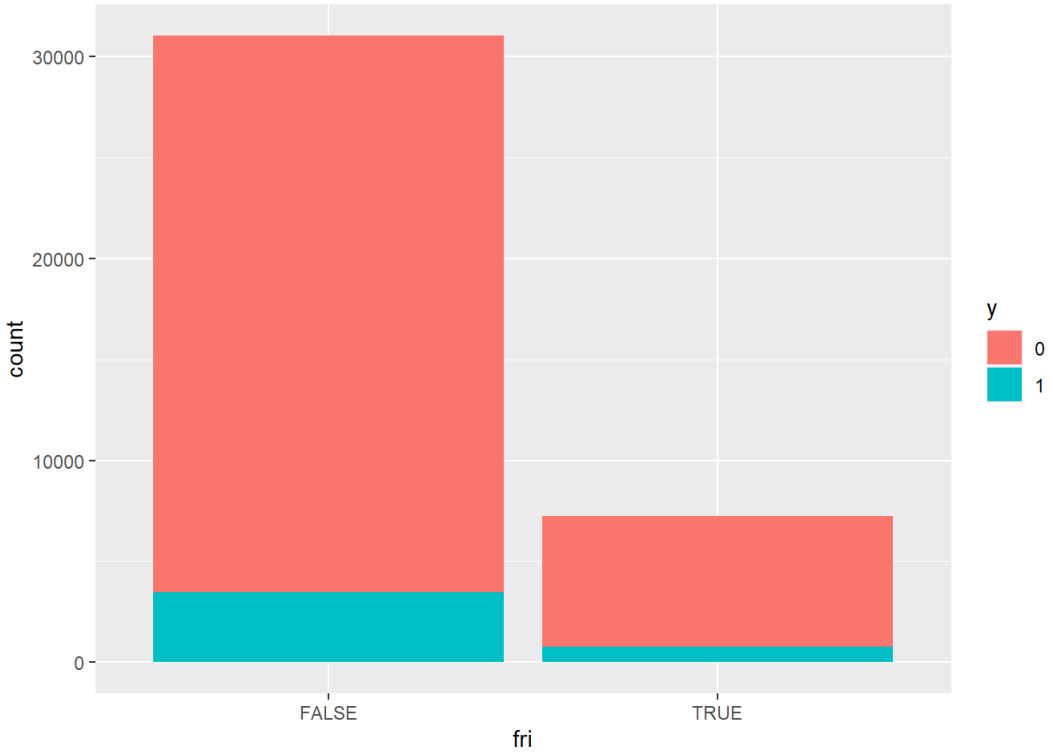
```
ggplot(data, aes(x=thu, fill=y)) +
  geom_bar()
```




```
day4<- table(data$thu, data$y)
prop.table(day4, margin = 1)
```

```
##
##                0         1
##   FALSE 0.8909837 0.1090163
##   TRUE  0.8799151 0.1200849
```

```
ggplot(data, aes(x=fri, fill=y)) +
 geom_bar()
```

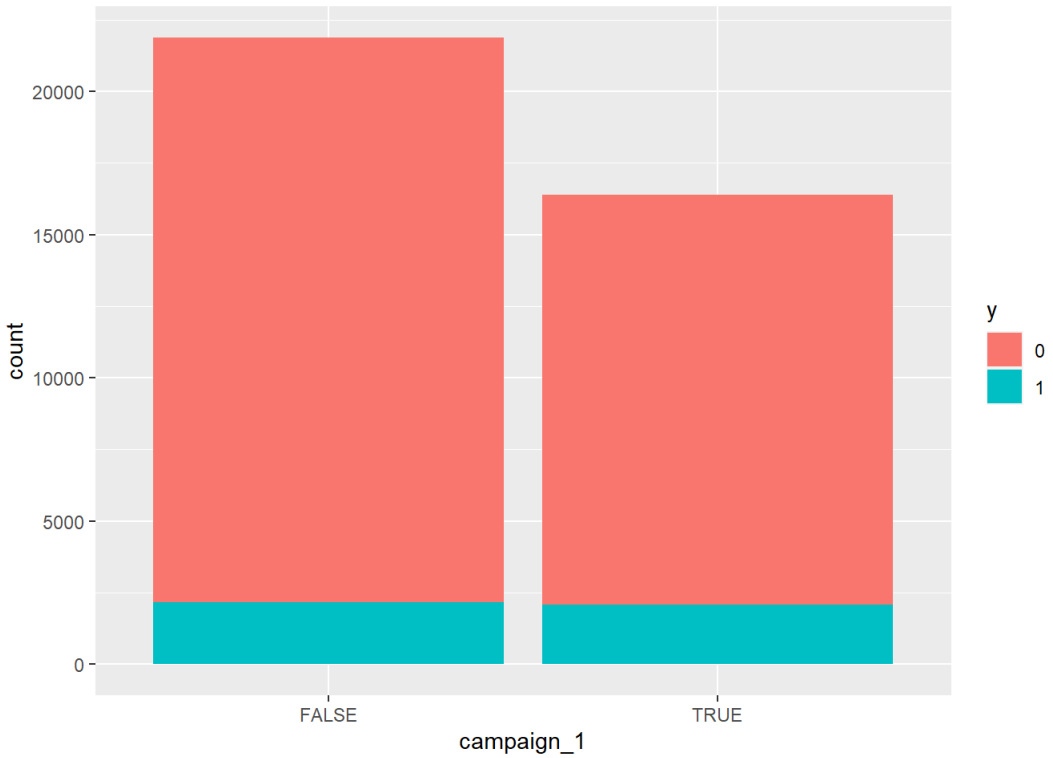



```
day5<- table(data$fri, data$y)
prop.table(day5, margin = 1)
```

```
##
##              0         1
##   FALSE 0.8875923 0.1124077
##   TRUE  0.8932724 0.1067276
```

On other side day of week when last contact was performed does not have significant impact on chances that certain person will subscribe term deposit. Contacts during all days have similar fractions with the slightly advantage for contacts during Thursday.

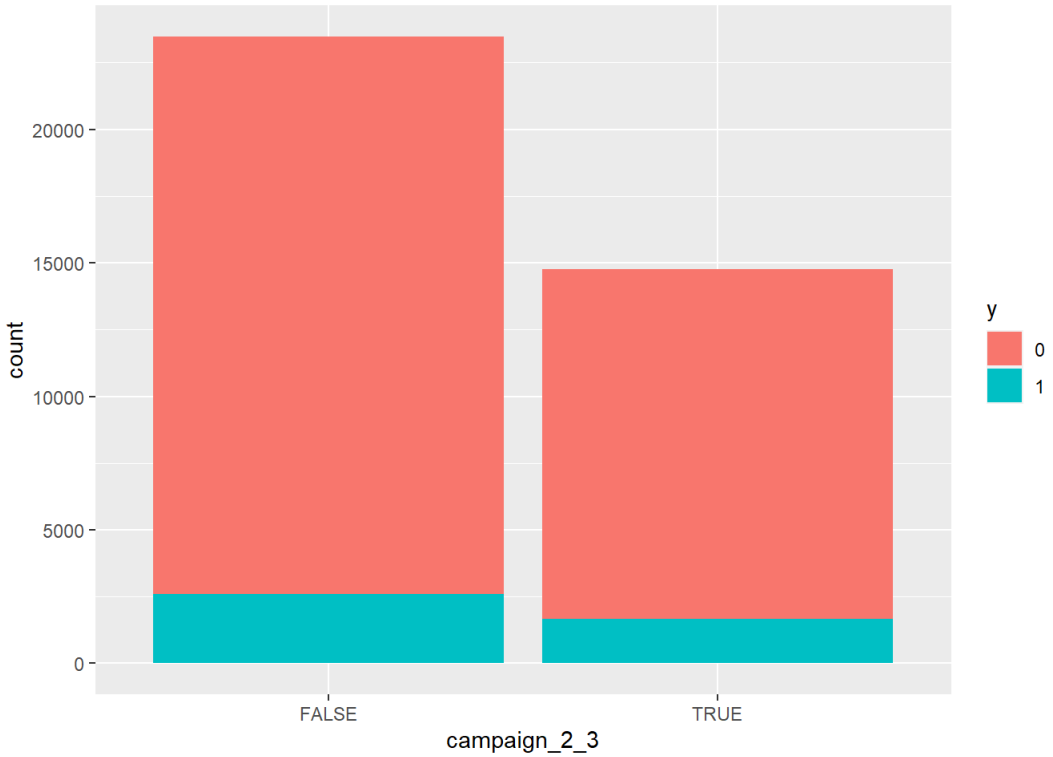
```
ggplot(data, aes(x=campaign_1, fill=y)) +
 geom_bar()
```




```
cp1<- table(data$campaign_1, data$y)
prop.table(cp1, margin = 1)
```

```
##
##                0         1
##   FALSE 0.90091449 0.09908551
##   TRUE  0.87230534 0.12769466
```

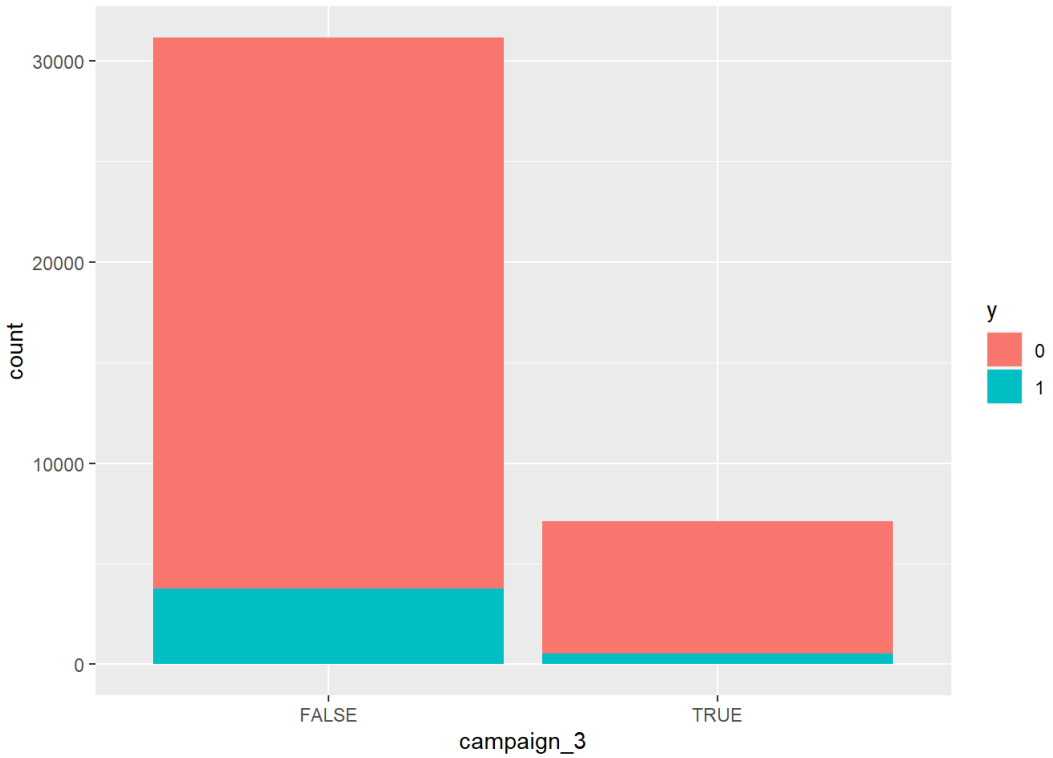
```
ggplot(data, aes(x=campaign_2_3, fill=y)) +
 geom_bar()
```




```
cp2<- table(data$campaign_2_3, data$y)
prop.table(cp2, margin = 1)
```

```
##
##                0         1
##   FALSE 0.8890120 0.1109880
##   TRUE  0.8881143 0.1118857
```

```
ggplot(data, aes(x=campaign_3, fill=y)) +
 geom_bar()
```

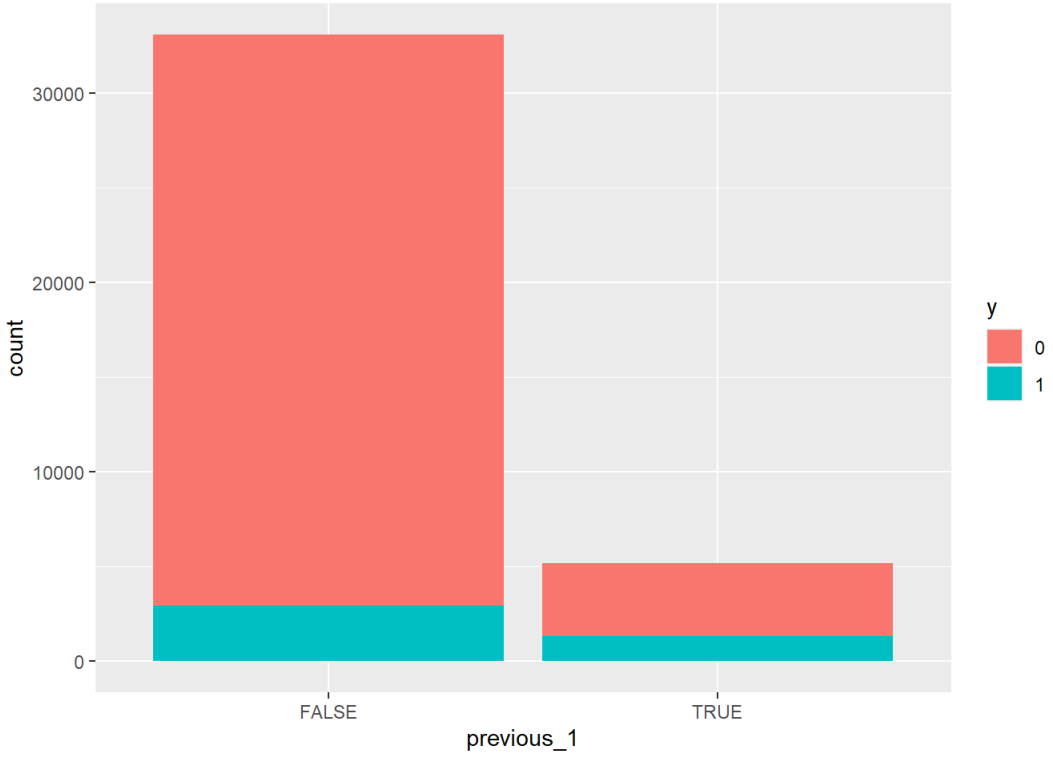


```
cp3<- table(data$campaign_3, data$y)
prop.table(cp3, margin = 1)
```

```
##
##                0         1
##   FALSE 0.87980352 0.12019648
##   TRUE  0.92756483 0.07243517
```

It seems that the higher number of contacts performed during current campaign for a client has a negative impact of chances that this person will subscribe term deposit. Among people who were contacted only once 12% decided for term deposit, among people who were contacted 2-3 times - 11%, and among people who were contacted 3 or more times it is only 7%.

```
ggplot(data, aes(x=previous_1, fill=y)) +
  geom_bar()
```

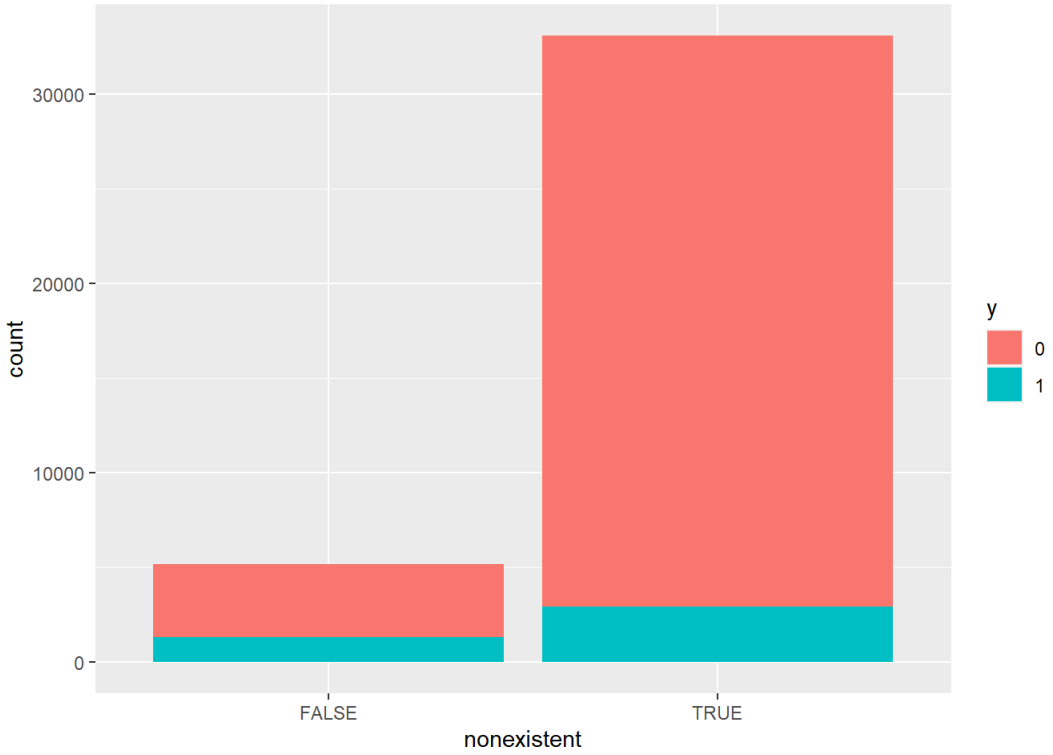



```
p1<- table(data$previous_1, data$y)
prop.table(p1, margin = 1)
```

```
##
##                0         1
##   FALSE 0.91163128 0.08836872
##   TRUE  0.74203514 0.25796486
```

25% of people who were contacted before current campaign have subscribed a term deposit in comparison with only 8% of people who were not contacted. It means that this variable has a significant impact on target variable.

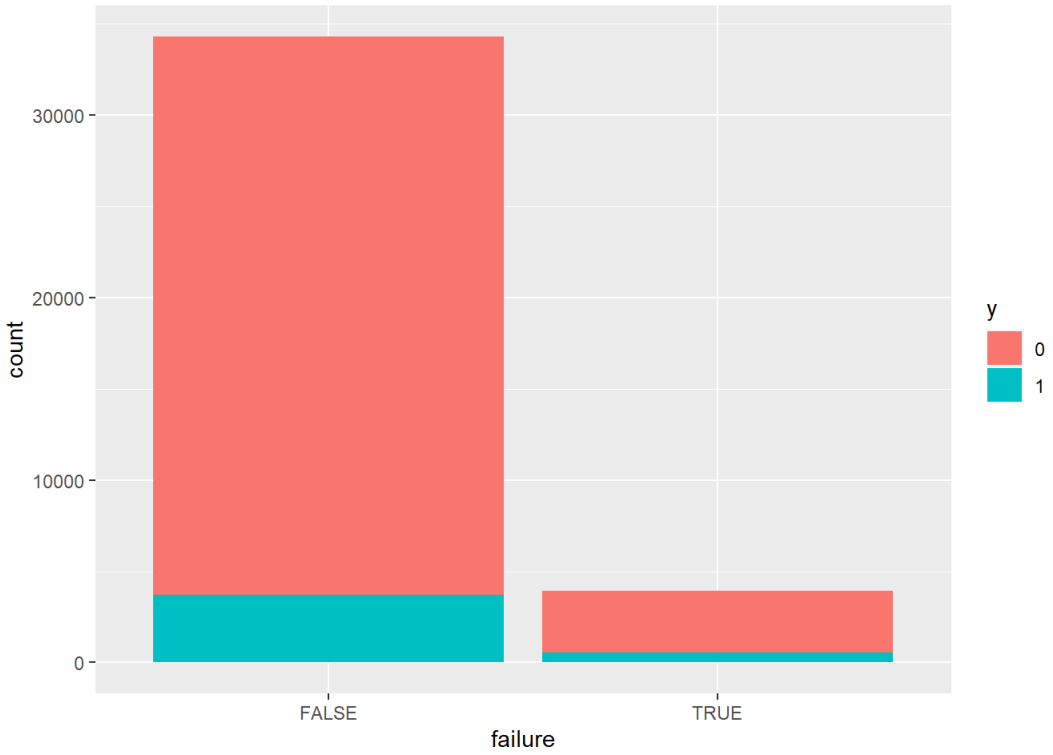
```
ggplot(data, aes(x=nonexistent, fill=y)) +
  geom_bar()
```




```
out1<- table(data$nonexistent, data$y)
prop.table(out1, margin = 1)
```

```
##
##                0         1
##   FALSE 0.74203514 0.25796486
##   TRUE  0.91163128 0.08836872
```

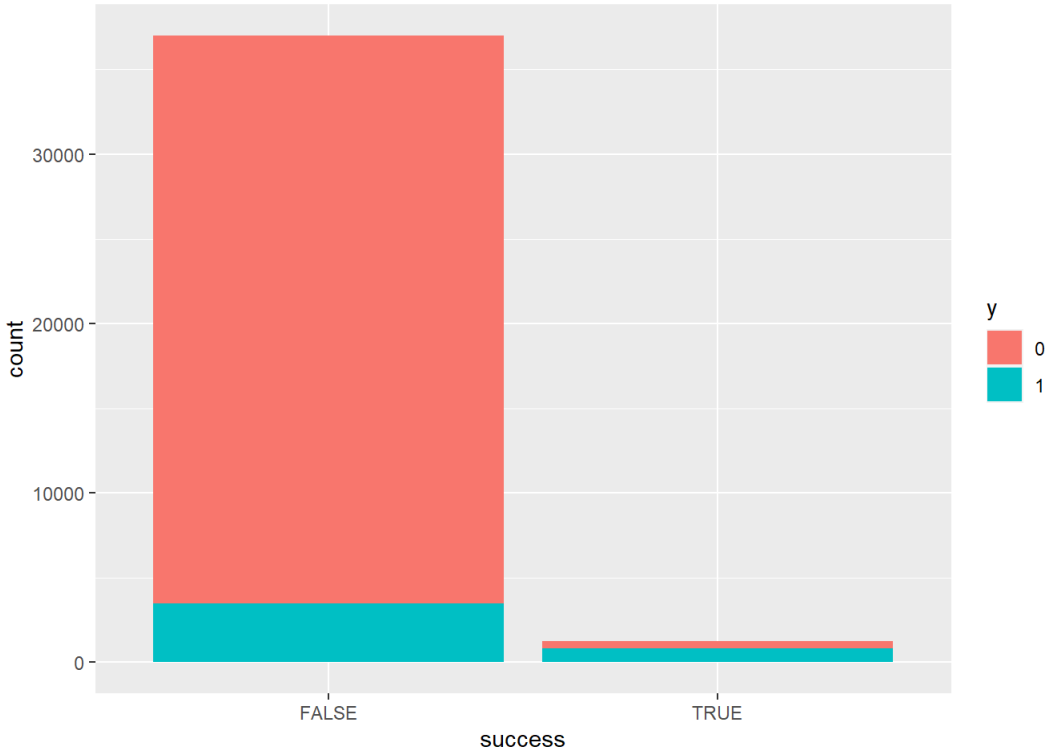
```
ggplot(data, aes(x=failure, fill=y)) +
  geom_bar()
```




```
out2<- table(data$failure, data$y)
prop.table(out2, margin = 1)
```

```
##
##                0         1
##   FALSE 0.8914279 0.1085721
##   TRUE  0.8645833 0.1354167
```

```
ggplot(data, aes(x=success, fill=y)) +
  geom_bar()
```




```
out3<- table(data$success, data$y)
prop.table(out3, margin = 1)
```

```
##
##                0         1
##   FALSE 0.90662667 0.09337333
##   TRUE  0.35398230 0.64601770
```

The variable that describes the outcome of the previous marketing camping has a huge impact on a target variable. 64% of clients for whom the previous marketing campaign was successful have also subscribed term deposit in current campaign.

```
library(corrplot)
```

```
## corrplot 0.87 loaded
```

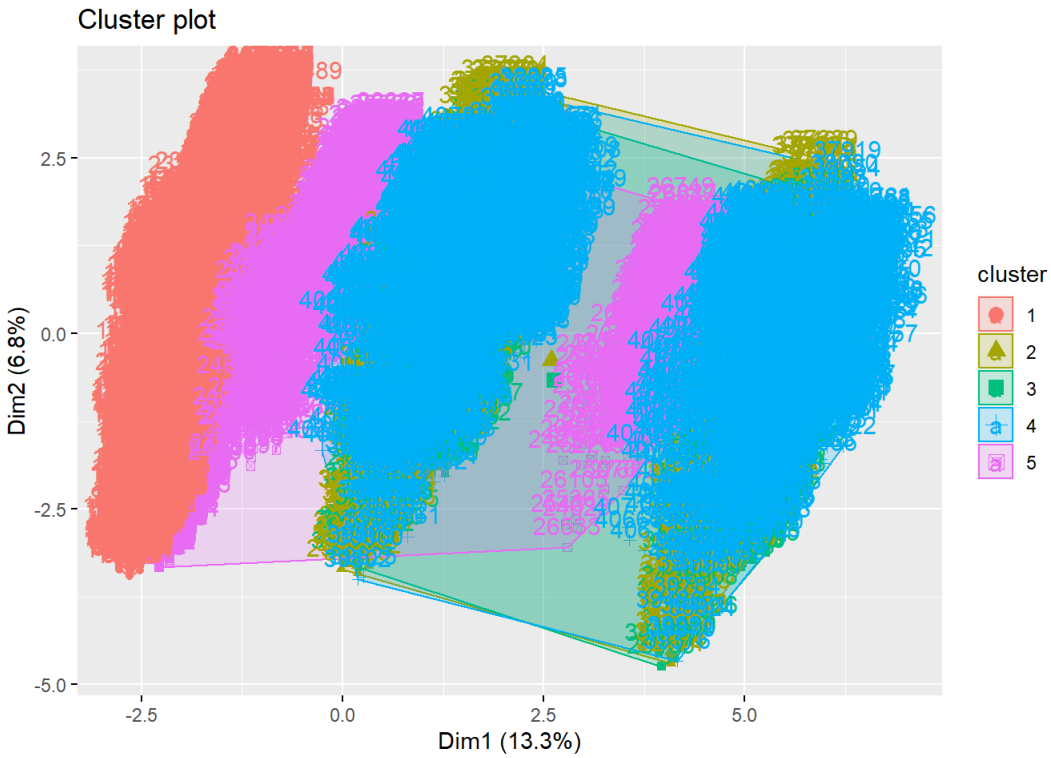
```
par(mfrow=c(1,1))


correlation<-data.frame(data[c(1:6)])
x<-cor(correlation)
corrplot(x, method="number")
```

Above there can be observed the values of correlation between all numerical variables. Age is not correlated with other variables (all the correlation values are close to 0). The highest positive values of correlations are between variables euribor3m and nr.employed, euribor3m and emp.var.rate and between nr.employed and emp.var.rate. All these values are above 0.90.

```
library(cluster)
library(factoextra)
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```
set.seed(123)

data_clustering <- kmeans(data[-7], 5)
fviz_cluster(data_clustering, data = data[-7])
```




```
summary_ds = data_clustering$centers
summary_ds
```

```
##        age emp.var.rate cons.price.idx cons.conf.idx euribor3m nr.employed
## 1 39.82185    1.4000000       93.90789     -40.31059 4.9546742    5228.100
## 2 32.83445   -1.9404984       92.89399     -45.23409 1.3019059    5096.175
## 3 51.62775   -2.0482612       92.85462     -44.00688 1.2740447    5093.932
## 4 43.36391   -2.2274543       93.48265     -35.94391 0.7850267    4999.555
## 5 40.27916    0.7018247       93.73373     -38.25190 4.6155081    5192.574
##     housemaid   services    admin. technician bluecollar     retired management
## 1 0.032450331 0.09112583 0.2711258 0.20430464 0.21754967 0.031059603 0.05841060
## 2 0.008010681 0.11778668 0.2787420 0.15502151 0.26064382 0.001780151 0.05444296
## 3 0.024485451 0.08623137 0.2139815 0.09936125 0.23491838 0.133427963 0.09723208
## 4 0.031482735 0.04840894 0.3212593 0.14691943 0.06601219 0.153012864 0.06973595
## 5 0.026429646 0.10910459 0.2268623 0.14446953 0.25028217 0.025206922 0.09396163
##   unemployed selfemployed entrepreneur     student   married    single
## 1 0.02205298   0.03450331   0.03145695 0.005960265 0.6201987 0.2658940
## 2 0.01869159   0.03352618   0.02447708 0.046877318 0.4842012 0.4460762
## 3 0.02661462   0.03229241   0.05145493 0.000000000 0.7136267 0.1096522
## 4 0.04603927   0.03114421   0.01658768 0.069397427 0.5324983 0.3612051
## 5 0.02755831   0.03940933   0.04947329 0.007242287 0.6555681 0.2221595
##     divorced     basic high.school professional.course university.degree
## 1 0.11390728 0.3054967   0.2272185          0.1480795         0.3192053
## 2 0.06972259 0.2944667   0.2830441          0.1197152         0.3027741
## 3 0.17672108 0.3921221   0.2214336          0.1089425         0.2775018
## 4 0.10629655 0.2078538   0.2329045          0.1472580         0.4119838
## 5 0.12227239 0.3539315   0.2438864          0.1236832         0.2784989
##   housing_no housing_yes   loan_no  loan_yes  telephone  cellular        Q2
## 1  0.4776821   0.5223179 0.8408609 0.1591391 0.33437086 0.6656291 0.2640397
## 2  0.4141819   0.5858181 0.8399347 0.1600653 0.07491470 0.9250853 0.9342827
## 3  0.4268985   0.5731015 0.8456352 0.1543648 0.08090845 0.9190916 0.8537970
## 4  0.4333108   0.5666892 0.8507109 0.1492891 0.16892349 0.8310765 0.2630332
## 5  0.4871144   0.5128856 0.8494169 0.1505831 0.70758089 0.2924191 0.6682656
##           Q3        Q4       mon       tue       wed       thu       fri
## 1 0.73596026 0.0000000 0.2149669 0.1972185 0.1992053 0.2183444 0.1702649
## 2 0.06571725 0.0000000 0.2167334 0.1664441 0.1830589 0.2257825 0.2079810
## 3 0.14620298 0.0000000 0.2544358 0.1855926 0.1745919 0.1997871 0.1855926
## 4 0.33378470 0.4031821 0.2193636 0.2098849 0.1821259 0.2109005 0.1777251
## 5 0.00000000 0.3317344 0.1744733 0.2104966 0.2191497 0.1886757 0.2072047
##   campaign_1 campaign_2_3 campaign_3 previous_0 previous_1 nonexistent
## 1  0.3692715    0.3805960 0.25013245  1.0000000  0.0000000   1.0000000
## 2  0.4632844    0.3799140 0.15680166  0.7031598  0.2968402   0.7031598
## 3  0.4872250    0.3757984 0.13697658  0.6770759  0.3229241   0.6770759
## 4  0.5409614    0.3656060 0.09343263  0.4705484  0.5294516   0.4705484
## 5  0.4425320    0.4069789 0.15048909  0.9337848  0.0662152   0.9337848
##      failure     success
## 1 0.00000000 0.000000000
## 2 0.26049548 0.036344756
## 3 0.26756565 0.055358410
## 4 0.25626269 0.273188896
## 5 0.06292325 0.003291949
```

In the next part of explanatory data analysis k-means clustering has been performed. All observations have been split into 5 clusters. The main objective of k-means algorithm is to minimize sum of distances between the points and their cluster centres. In our case the observations (clients) have been split into groups of people with similar features. The mean value of each feature for each group is presented in the table. These kind of data clustering can be very useful for further analysis. For example it is possible to find customer segments and try prepare distinct marketing campaign for different segments. It can, besides of predictive models which will be used in this report, also improve the work of marketing departments of banking institutions, allowing them to derive more insights.

The chart presented above uses principal component analysis to plot the data points of the first two principal components that explains the majority of variance.

Creating and assessment of the models

```
set.seed(123)

training <- sample(1:nrow(data), 0.75*nrow(data))

trainingset <- data[training,]
testset <- data[-training,]
```

In purpose of creating predictive models data has been split into training set (75% of observations) and test set (25% of observations).

In this part of the report 3 distinct models will be built: Classification Tree, Random Forest and Neural Network. Among each type of models the hyperparameters will be tuned in order to build models that maximizes accuracy and sensitivity. For each model the confusion matrix will be

constructed in order to assess its performance. Moreover, for one of Neural Network models the ROC, Lift and Gain curve will be shown.

Classification Tree

```
set.seed(123)
library(caret)
```

```
## Loading required package: lattice
```

```
cp<- seq(0,0.01, length.out =  100)

accuracy_rt <- list()
sensitivity_rt <- list()
specificity_rt <- list()
random_tree <- list()

set.seed(123)

#for(c in 1:length(cp)){
#  print(c)
#  random_tree[[c]] <- rpart(y~., data=trainingset,cp=cp[c])
#  rpart_prediction <- predict(random_tree[[c]], newdata=testset, type = "class")
#  conf_mat <- table(rpart_prediction, as.factor(testset$y))
#  accuracy_rt[c] <- (conf_mat[1,1] + conf_mat[2,2])/ sum(conf_mat)
#  sensitivity_rt[c] <- conf_mat[2,2]/sum(conf_mat[,2])
#  specificity_rt[c] <- conf_mat[1,1]/sum(conf_mat[,1])
#}

#which.max(accuracy_rt)
#which.max(sensitivity_rt)
#which.max(specificity_rt)
```

First of proposed models is Classification Tree. In this case algorithm starts with the root and on each step it seeks for variable which decreases entropy the most (increases information the most). Moreover, the logarithms looks for the optimal split level (assuming that variable has more than 2 values). The size of the tree is determined by the values of complexity parameter (cp). 100 trees with cp parameters in the range between 0 and 0.01 have been built in order to find trees which maximizes accuracy and sensitivity. The code above has been commented because it took much time to execute it. But it has been executed earlier and the results are the following:

cp which maximizes accuracy = 0.001717172

cp which maximizes sensitivity = 0

Both classification trees with these specifications have been created.

```
library(rpart)

set.seed(123)

rpart_max_accuracy <- rpart(y~., data=trainingset,cp=cp[18])

rpart_max_accuracy_prediction<- predict(rpart_max_accuracy, newdata=testset, type = "class")

confusion_matrix_rprat_1 <- table(rpart_max_accuracy_prediction, as.factor(testset$y))

rpart_maximal_accuracy <- (confusion_matrix_rprat_1[1,1] + confusion_matrix_rprat_1[2,2])/ sum(confusion_matrix_rprat_1)

cf_rpart_optimal_accuracy<- confusionMatrix(rpart_max_accuracy_prediction, testset$y, positive = '1')
cf_rpart_optimal_accuracy
```

```
## Confusion Matrix and Statistics
##
##          Reference
## Prediction    0    1
##         0 8334  900
##         1   99  229
##
##               Accuracy : 0.8955
##                 95% CI : (0.8892, 0.9016)
##    No Information Rate : 0.8819
##    P-Value [Acc > NIR] : 1.513e-05
##
##                  Kappa : 0.2758
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.20283
##             Specificity : 0.98826
##          Pos Pred Value : 0.69817
##          Neg Pred Value : 0.90253
##              Prevalence : 0.11807
##          Detection Rate : 0.02395
##    Detection Prevalence : 0.03430
##       Balanced Accuracy : 0.59555
##
##        'Positive' Class : 1
##
```

The tree which purpose is to maximize accuracy has been built with cp parameter level of 0.001717172. The confusion matrix with all statistics is presented above. The accuracy of this tree is equal to 0.8955.

```
set.seed(123)

rpart_max_sensitivity <- rpart(y~., data=trainingset,cp=cp[1])


rpart_max_sensitivity_prediction<- predict(rpart_max_sensitivity, newdata=testset, type = "class")

confusion_matrix_rprat_2 <- table(rpart_max_sensitivity_prediction, as.factor(testset$y))

rpart_maximal_sensitivity <- confusion_matrix_rprat_2[2,2]/sum(confusion_matrix_rprat_2[,2])

cf_rpart_optimal_sensitivity<- confusionMatrix(rpart_max_sensitivity_prediction, testset$y, positive = '1')
cf_rpart_optimal_sensitivity
```

```
## Confusion Matrix and Statistics
##
##          Reference
## Prediction    0    1
##         0 8141  781
##         1  292  348
##
##               Accuracy : 0.8878
##                 95% CI : (0.8813, 0.894)
##    No Information Rate : 0.8819
##    P-Value [Acc > NIR] : 0.03857
##
##                  Kappa : 0.3368
##
##  Mcnemar's Test P-Value : < 2e-16
##
##             Sensitivity : 0.30824
##             Specificity : 0.96537
##          Pos Pred Value : 0.54375
##          Neg Pred Value : 0.91246
##              Prevalence : 0.11807
##          Detection Rate : 0.03639
##    Detection Prevalence : 0.06693
##       Balanced Accuracy : 0.63681
##
##        'Positive' Class : 1
##
```

The tree which purpose is to maximize sensitivity has been built with cp parameter level of 0. The confusion matrix with all statistics is presented above. The sensitivity of this tree is equal to 0.30824.

Random Forest

```
set.seed(123)
library(randomForest)
```

```
## randomForest 4.6-14
```

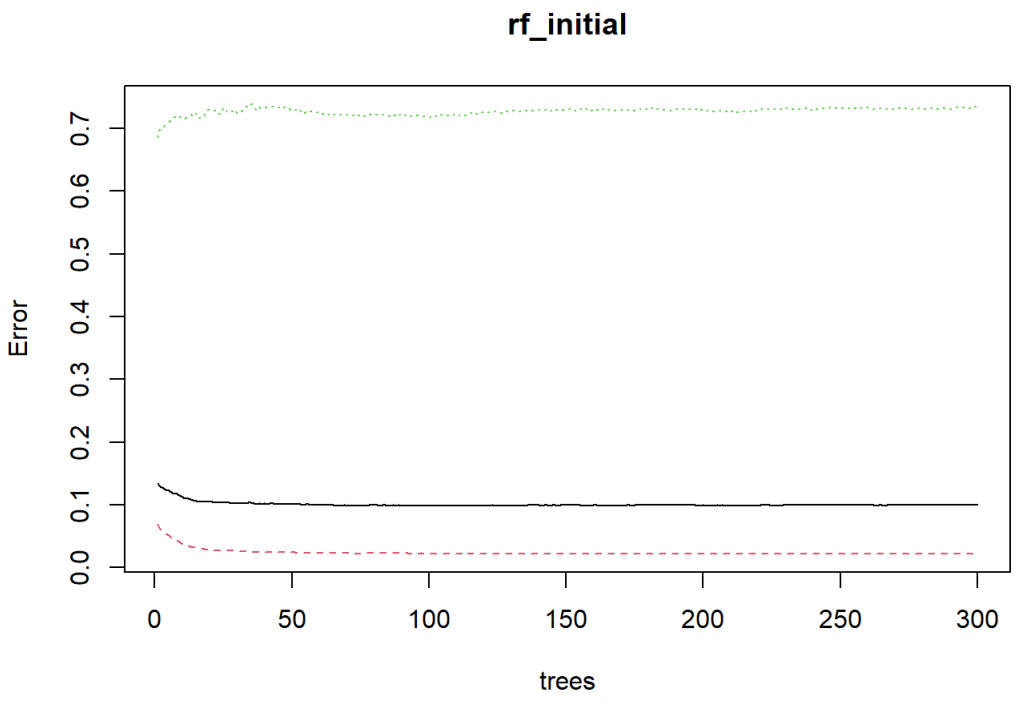
```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

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

```
## The following object is masked from 'package:gridExtra':
##
##     combine
```

```
rf_initial <- randomForest(y~., data=trainingset, ntree=300, do.trace=F)

plot(rf_initial)
```



Second of the models used in this report is random forest. In this case instead of building one tree, the number of trees are built and the average of their prediction is taken into consideration. In random forest model the idea is to build many simple trees as opposed to Classification Tree method where one very complex tree was built. Moreover, trees are uncorrelated with each other. In random forest you have to specify the number of trees built and the number of variables randomly sampled at each split. The first random forest was created in order to find the optimal number of trees. On the above chart it can be observed that above 150 trees the error almost does not decrease, thus this is the chosen number of trees for the following Random Forest models.

```
mtry<-c(1:ncol(trainingset))
accuracy_rf <- list()
sensitivity <- list()
specificity <- list()
random_forest <- list()

#for(i in 1:length(mtry)){
#  print(i)
# random_forest[[i]] <- randomForest(y~., data=trainingset, ntree=150, mtry=mtry[i])
# test.prediction <- predict(random_forest[[i]], newdata = testset)
# conf_mat <- table(test.prediction, as.factor(testset$y))
# accuracy_rf[i] <- (conf_mat[1,1] + conf_mat[2,2])/ sum(conf_mat)
# sensitivity[i] <- conf_mat[2,2]/sum(conf_mat[,2])
# specificity[i] <- conf_mat[1,1]/sum(conf_mat[,1])
#  }

#which.max(accuracy_rf)
#which.max(sensitivity)
```

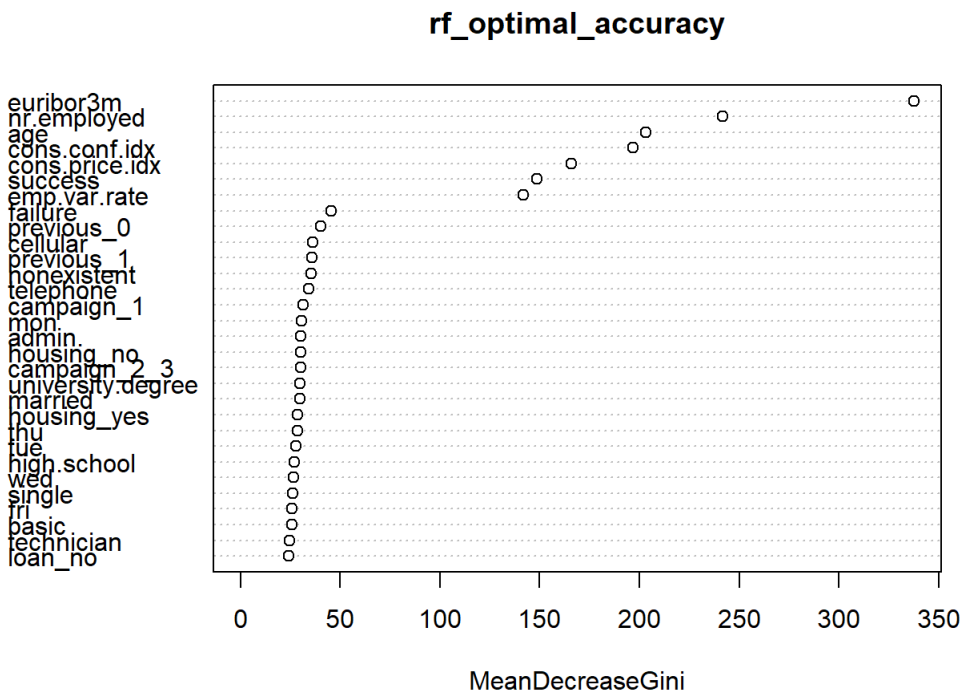
In order to find the optimal number of variables randomly chosen at each split 47 different random forests have been created (corresponding with the number of variables in the dataset). In each forest different value of mtry was specified. The goal was to find the mtry which maximizes accuracy and maximizes sensitivity. The code above has been commented because it took much time to execute it. But it has been executed earlier and the results are the following:

mtry which maximizes accuracy = 3

mtry which maximizes sensitivity = 36

Both random forests with these specifications have been created.

```
set.seed(123)
library(randomForest)
library(caret)

rf_optimal_accuracy <- randomForest(y~., data=trainingset, ntree=150, mtry=3, do.trace=F)

varImpPlot(rf_optimal_accuracy)
```

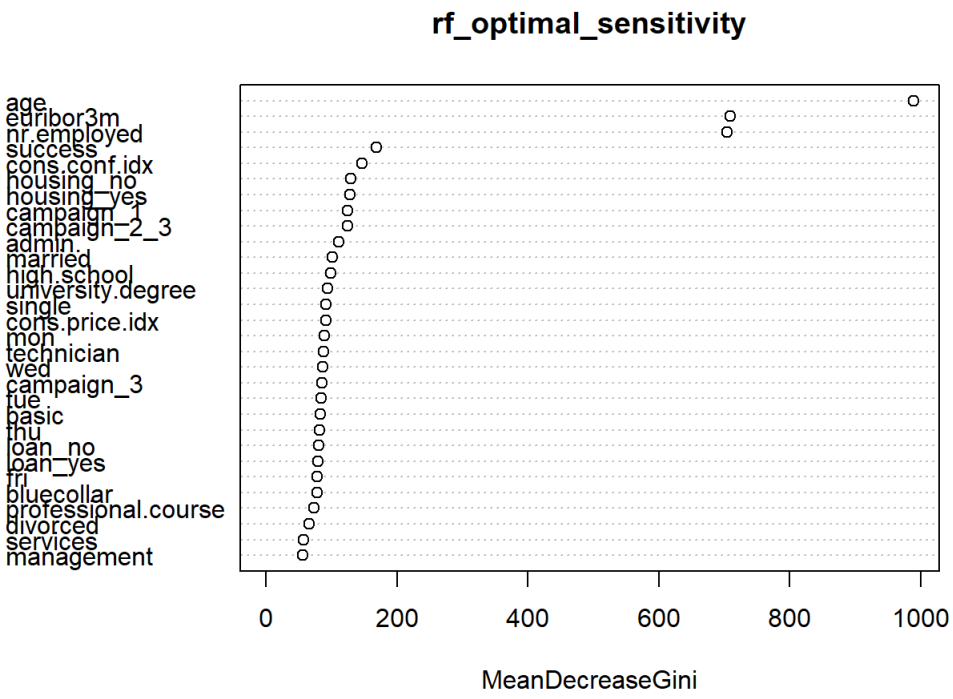



```
rf_prediction <- predict(rf_optimal_accuracy, newdata=testset)

confusion_matrix_rf_1 <- table(rf_prediction, testset$y)
random_forest_maximal_acuracy <- (confusion_matrix_rf_1[1,1] + confusion_matrix_rf_1[2,2])/ sum(confusion_matrix_rf_1)

cf_rf_optimal_accuracy<- confusionMatrix(rf_prediction, testset$y, positive = '1')
cf_rf_optimal_accuracy
```

```
## Confusion Matrix and Statistics
##
##          Reference
## Prediction    0    1
##         0 8309  900
##         1  124  229
##
##               Accuracy : 0.8929
##                 95% CI : (0.8865, 0.899)
##    No Information Rate : 0.8819
##    P-Value [Acc > NIR] : 0.0003986
##
##                  Kappa : 0.2679
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.20283
##             Specificity : 0.98530
##          Pos Pred Value : 0.64873
##          Neg Pred Value : 0.90227
##              Prevalence : 0.11807
##          Detection Rate : 0.02395
##    Detection Prevalence : 0.03692
##       Balanced Accuracy : 0.59407
##
##        'Positive' Class : 1
##
```

The Random Forest which purpose is to maximize accuracy has been built with mtry parameter equal to 3. The confusion matrix with all statistics is presented above. The accuracy of this model is equal to 0.8929. The chart presented above shows which variables where the most important for building the model. In other words, higher the value of a variable, the more times it appeared in trees of the forest and more information it gave. In this case the most important variables are: euribor3m, nr.employed, age, cons.conf.idx, cons.price.idx.

```
set.seed(123)
rf_optimal_sensitivity <- randomForest(y~., data=trainingset, ntree=150, mtry=36, do.trace=F)

varImpPlot(rf_optimal_sensitivity)
```




```
rf_prediction_2 <- predict(rf_optimal_sensitivity, newdata=testset)

confusion_matrix_rf_2 <- table(rf_prediction_2, testset$y)
random_forest_maximal_sensitivity <- confusion_matrix_rf_2[2,2]/sum(confusion_matrix_rf_2[,2])

cf_rf_optimal_sensitivity <- confusionMatrix(rf_prediction_2, testset$y, positive = '1')

cf_rf_optimal_sensitivity
```

```
## Confusion Matrix and Statistics
##
##          Reference
## Prediction    0    1
##         0 8110  801
##         1  323  328
##
##               Accuracy : 0.8825
##                 95% CI : (0.8758, 0.8888)
##    No Information Rate : 0.8819
##    P-Value [Acc > NIR] : 0.4449
##
##                  Kappa : 0.3088
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.29052
##             Specificity : 0.96170
##          Pos Pred Value : 0.50384
##          Neg Pred Value : 0.91011
##              Prevalence : 0.11807
##          Detection Rate : 0.03430
##    Detection Prevalence : 0.06808
##       Balanced Accuracy : 0.62611
##
##        'Positive' Class : 1
##
```

The Random Forest which purpose is to maximize sensitivity has been built with mtry parameter equal to 36. The confusion matrix with all statistics is presented above. The sensitivity of this model is equal to 0.29052. Again chart showing which variables were the most important for building the model has been presented. In this case the most important variables are: age, euribor3m, nr.employed, success, cons.conf.idx.

Neural Network

```
##podzial zbiorow na treningowy i testowy

set.seed(123)
training_neural <- sample(1:nrow(FINAL_SCALED_DATA), 0.75*nrow(FINAL_SCALED_DATA))

trainingset_neural <- FINAL_SCALED_DATA[training_neural,]
testset_neural <- FINAL_SCALED_DATA[-training_neural,]
```

The third type of model is Neural Network. In this case previously standardized data will be used. Dataset again has been split into train set (75% of observations) and test set (25 % of observations).

```
library(nnet)
set.seed(123)
NEURONS <- 5
wts.parameter <-  2 * runif(NEURONS * ncol(FINAL_SCALED_DATA) + NEURONS + 1) - 1

DECAYS <- seq(0, 20, length.out = 100)
neural.nets <- list()
accuracy_net <- list()
sensitivity_net <- list()
specificity_net <- list()

##for (d in 1:length(DECAYS)){
#  print(d)
#  neural.nets[[d]] <- nnet(y ~ ., data = trainingset_neural, size = 5,
#                           decay = DECAYS[d], linout = F, maxit = 1000,
#                           trace = TRUE, Wts = wts.parameter)
#  test.prediction <- predict(neural.nets[[d]], newdata = testset_neural)
#  score_neural <- ifelse(test.prediction>0.5, 1, 0)
#  conf_mat <- table(score_neural, as.factor(testset_neural$y))
#  accuracy_net[d] <- (conf_mat[1,1] + conf_mat[2,2])/ sum(conf_mat)
#  sensitivity_net[d] <- conf_mat[2,2]/sum(conf_mat[,2])
#  specificity_net[d] <- conf_mat[1,1]/sum(conf_mat[,1])

#}
```

The number of neurons in hidden layer has been set up to 5. The initial weights have been randomly chosen. In order to find the optimal value for decay parameter 100 neural networks have been trained. Each one with different decay value from range (0,20). The cutoff threshold has been set up to 0,5. The code above has been commented because it took much time to execute it. But it has been executed earlier and the results are the

following:

value of decay parameter that maximizes accuracy = 11.51515

value of decay parameter that maximizes sensitivity = 0.4040404

```
set.seed(123)
neural.nets_optimal_accuracy <- nnet(y ~ ., data = trainingset_neural, size = 5,
                     decay = DECAYS[9], linout = F, maxit = 1000,
                     trace = FALSE, Wts = wts.parameter)
test.prediction_optimal_accuracy <- predict(neural.nets_optimal_accuracy, newdata = testset_neural)
score_neural_optimal_accuracy <- ifelse(test.prediction_optimal_accuracy>0.5, 1, 0)
score_neural_optimal_accuracy <- as.factor(score_neural_optimal_accuracy)


confusion_matrix_nnet_1 <- table(score_neural_optimal_accuracy, testset_neural$y)

neural_network_maximal_accuracy <- (confusion_matrix_nnet_1[1,1] + confusion_matrix_nnet_1[2,2])/ sum(confusion_matrix_nnet_1)

confusionMatrix(score_neural_optimal_accuracy, testset_neural$y, positive='1')
```

```
## Confusion Matrix and Statistics
##
##          Reference
## Prediction    0    1
##         0 8302  868
##         1  131  261
##
##               Accuracy : 0.8955
##                 95% CI : (0.8892, 0.9016)
##    No Information Rate : 0.8819
##    P-Value [Acc > NIR] : 1.513e-05
##
##                  Kappa : 0.3006
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.2312
##             Specificity : 0.9845
##          Pos Pred Value : 0.6658
##          Neg Pred Value : 0.9053
##              Prevalence : 0.1181
##          Detection Rate : 0.0273
##    Detection Prevalence : 0.0410
##       Balanced Accuracy : 0.6078
##
##        'Positive' Class : 1
##
```

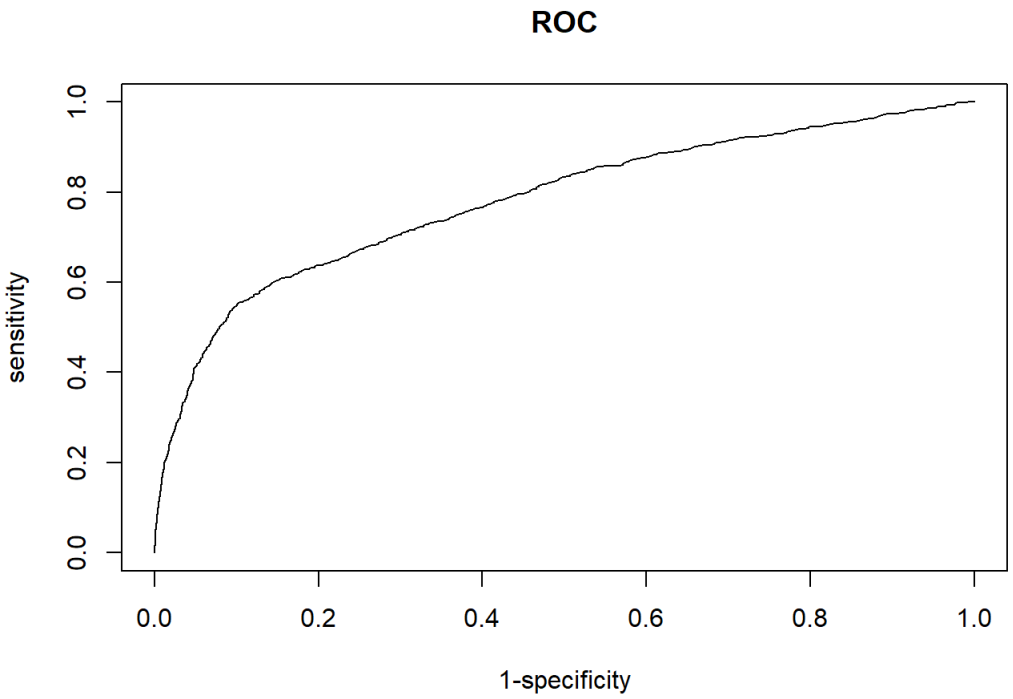
The Neural Network which purpose is to maximize accuracy has been built with decay parameter equal to 11.51515. The confusion matrix with all statistics is presented above. The accuracy of this tree is equal to 0.8955.

```
set.seed(123)
neural.nets_optimal_sensitivity <- nnet(y ~ ., data = trainingset_neural, size = 5,
                                decay = DECAYS[3], linout = F, maxit = 1000,
                                trace = FALSE, Wts = wts.parameter)
test.prediction_optimal_sensitivity <- predict(neural.nets_optimal_sensitivity, newdata = testset_neural)
score_neural_optimal_sensitivity <- ifelse(test.prediction_optimal_sensitivity>0.5, 1, 0)
score_neural_optimal_sensitivity <- as.factor(score_neural_optimal_sensitivity)


confusion_matrix_nnet_2 <- table(score_neural_optimal_sensitivity, testset$y)
neural_network_maximal_sensitivity <- confusion_matrix_nnet_2[2,2]/sum(confusion_matrix_nnet_2[,2])

confusionMatrix(score_neural_optimal_sensitivity, testset_neural$y, positive='1')
```

```
## Confusion Matrix and Statistics
##
##          Reference
## Prediction    0    1
##         0 8275  854
##         1  158  275
##
##               Accuracy : 0.8942
##                 95% CI : (0.8878, 0.9003)
##    No Information Rate : 0.8819
##    P-Value [Acc > NIR] : 9.002e-05
##
##                  Kappa : 0.3067
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.24358
##             Specificity : 0.98126
##          Pos Pred Value : 0.63510
##          Neg Pred Value : 0.90645
##              Prevalence : 0.11807
##          Detection Rate : 0.02876
##    Detection Prevalence : 0.04528
##       Balanced Accuracy : 0.61242
##
##        'Positive' Class : 1
##
```

The Neural Network which purpose is to maximize sensitivity has been built with decay parameter equal to 0.4040404. The confusion matrix with all statistics is presented above. The sensitivity of this tree is equal to 0.24358.

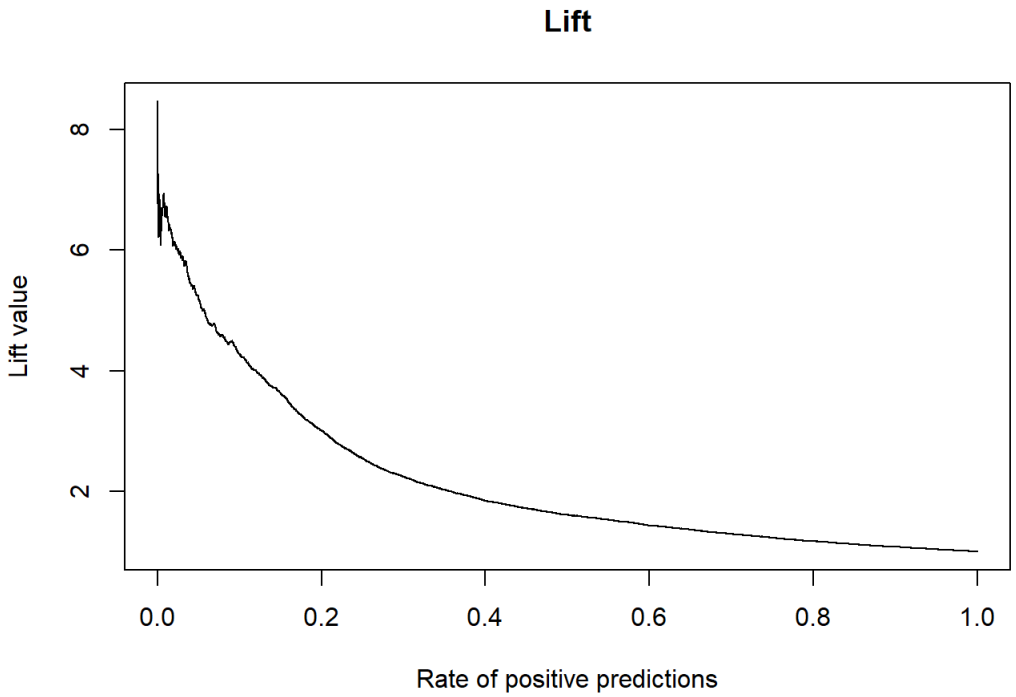
```
library(ROCR)
set.seed(123)

prediction.object_nnet <- prediction(predict(neural.nets_optimal_sensitivity, newdata = testset_neural), testset_neural$y)

roc_nnet <- performance(prediction.object_nnet, "tpr", "fpr")
plot(roc_nnet, main="ROC", ylab="sensitivity", xlab="1-specificity")
```



For neural network which maximizes sensitivity some plots assessing its performance have been constructed.

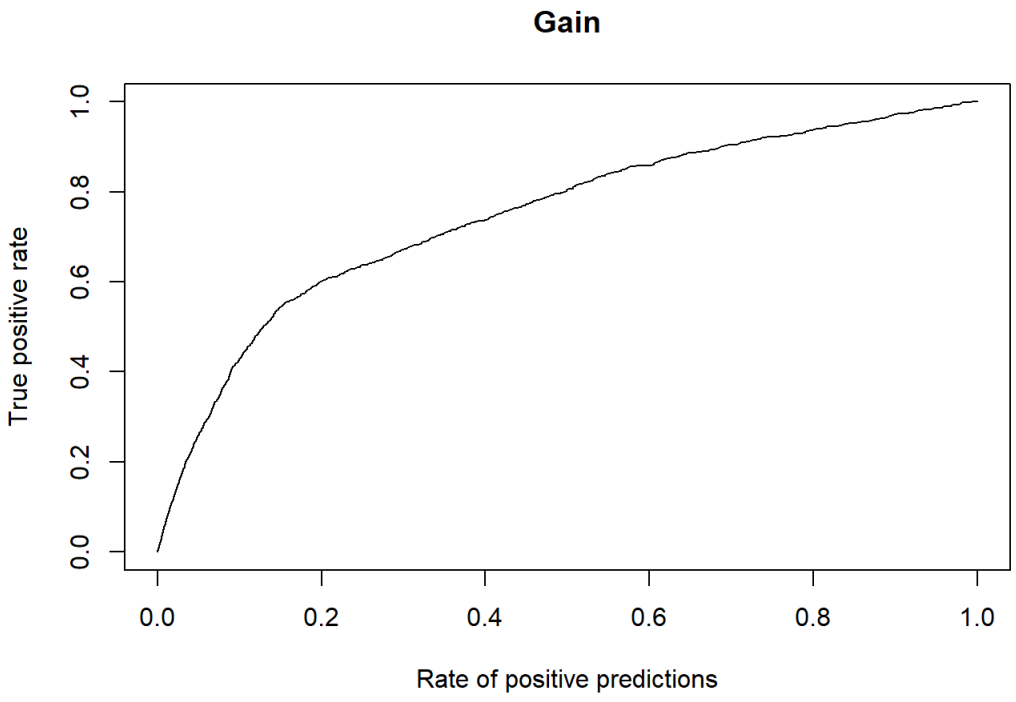
First one is ROC curve, which allows to compare the value of sensitivity and specificity for different cutoff thresholds.

```
lift_nnet <- performance(prediction.object_nnet, "lift", "rpp")
plot(lift_nnet, main="Lift")
```



Second plot is Lift curve which compare value of lift (which is true positive rate divided by rate of positive predictions) on y-axis with rate of positive predictions on y-axis. In other words lift curve shows how much better the constructed model is in comparison with random model.

```
gain_nnet <- performance(prediction.object_nnet, "tpr", "rpp")
plot(gain_nnet, main="Gain")
```



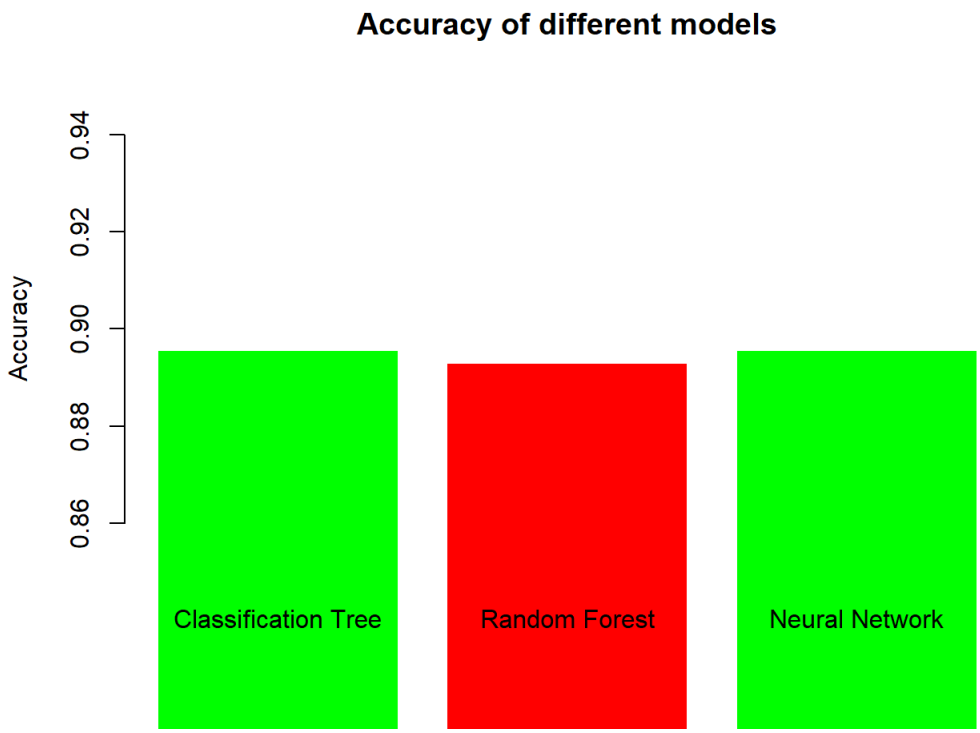
The third plot is Gain curve which compares value of True Positive rate with value of Rate of positive predictions. In other words this curve shows how good the model is in detecting positive values.

Comparison of different models

```
results_accuracy<-matrix(c(rpart_maximal_accuracy, random_forest_maximal_acuracy, neural_network_maximal_accuracy), ncol=1,dimnames=list(c("Classification Tree","Random Forest","Neural Network"),"accuracy"))
print(results_accuracy)
```

```
##                    accuracy
## Classification Tree 0.8955239
## Random Forest       0.8929094
## Neural Network      0.8955239
```

```
barplot(results_accuracy[,1],ylim=c(0.85,0.95),border=F,main="Accuracy of different models",ylab = "Accuracy",col=c("green", "red", "green"))
```

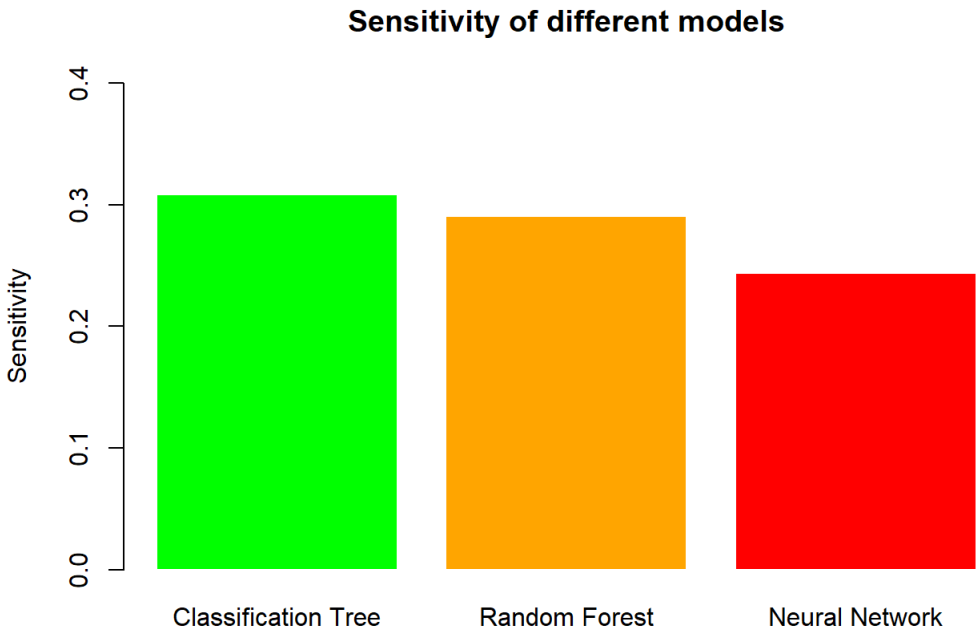


3 types of models which were tuned in order to maximize accuracy have been compared in above bar chart. Classification Tree and Neural Network have the same value of accuracy equal to 0.8955239. This value is slightly better than for Random Forest, where the value of accuracy is equal to 0.8929094.

```
results_sensitivity<-matrix(c(rpart_maximal_sensitivity, random_forest_maximal_sensitivity, neural_network_maximal_sensitivity), ncol=1,dimnames=list(c(
"Classification Tree", "Random Forest", "Neural Network"),"sensitivity"))
results_sensitivity
```

```
##                    sensitivity
## Classification Tree   0.3082374
## Random Forest         0.2905226
## Neural Network        0.2435784
```

```
barplot(results_sensitivity[,1],ylim=c(0,0.40),border=F,main="Sensitivity of different models",ylab = "Sensitivity",col=c("green", "orange", "red"))
```



Also models which were tuned in order to maximize sensitivity have been compared in above bar chart. In this case Classification Tree has the best results with the value of sensitivity equal to 0.3082374. Random Forest is the second best model with the value of sensitivity equal to 0.2905226. The worst model when the objective is to maximize sensitivity is Neural Network. It has the value of sensitivity equal to 0.2435784.

Summary

The main objective of this report was to build and compare 3 different models that predict whether a person will subscribe or not a term deposit. However, a big part of work was devoted for the initial data cleaning and preprocessing which turned out to be quite a challenge. Firstly, observations containing missing values have been removed from the data set. Some variables which were too homogeneous or had too many missing values had to be extracted. All categorical variables have been investigated in order to check whether in some categories there are not too little observations. Everywhere where it was needed and where it was substantively possible the categories merger was performed. Moreover, 2 numeric variables which had skewed distributions and huge fraction of outliers have been transformed into categorical ones. Then, all categorical variables have been transformed using one-hot encoding. In the next part of report profound explanatory data analysis have been performed. Stacked bar plots/histograms of all variables have been presented in order to explore relations between certain features and target variable. Also, clustering using k-means has been performed. The next part of report was building predictive models. 3 distinct types of models have been tuned in order to find the ones which maximizes accuracy and sensitivity. The small problem encountered in this part was huge computer power needed to build many models with different values of hyperparameters. For example in case of Neural Network, 100 models with different values of decay parameter have been built. That is why this part of code took a long time to proceed. But, at the end final models have good predictive statistics and could be used in banking financial institutions in order to improve results of direct marketing campaigns.