Applying statistical analysis and modeling for Bank marketing campaign

Maksim Kokot (ID: 2072065), Yelnur Shauketbek (ID: 2078709)

Contents

Introduction

This project is dedicated to the binary classification problem. The classification goal is aimed to predict if the client will accept a term deposit. For the project we used real dataset with social and economic context provided by Portuguese banking institution. The dataset is available by THIS LINK. There are four datasets, however we are using the largest one called "bank-additional-full.csv" with additional features and data.

While working on this project, we expect to achieve following results:

1. Understanding the processes which lay behind the marketing campaign by analyzing the data;
2. Estimating the impact of predictors on the target variable;
3. Building a statistical model which will help us to select the clients who will probably accept a term deposit.

1. Dataset

First, we import required libraries.

```
library(pROC)
library(ROSE)
library(ggplot2)
library(cowplot)
library(dplyr)
library(Rmisc)
library(corrplot)
library(MASS)
library(caret)
library(regclass)
library(fastDummies)
library(class)
library(glmnet)
library(outliers)
library(Hmisc)
library(dplyr)
library(igraph)
library(gRbase)
library(FactoMineR)
library(e1071)

# set random seed
set.seed(1)
```

Then we read the dataset file.

```
# reading data
data <- read.csv('data/bank-additional-full.csv', sep=';',
                 na.strings=c("unknown"))

# initial info
nrow(data)
```

```
## [1] 41188
```

```
ncol(data)
```

```
## [1] 21
```

```
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.   :-3.40000
##  1st Qu.:999.0   1st Qu.:0.000   Class :character   1st Qu.:-1.80000
##  Median :999.0   Median :0.000   Mode  :character   Median : 1.10000
##  Mean   :962.5   Mean   :0.173                      Mean   : 0.08189
##  3rd Qu.:999.0   3rd Qu.:0.000                      3rd Qu.: 1.40000
##  Max.   :999.0   Max.   :7.000                      Max.   : 1.40000
##  cons.price.idx  cons.conf.idx     euribor3m      nr.employed
##  Min.   :92.20   Min.   :-50.8   Min.   :0.634   Min.   :4964
##  1st Qu.:93.08   1st Qu.:-42.7   1st Qu.:1.344   1st Qu.:5099
##  Median :93.75   Median :-41.8   Median :4.857   Median :5191
##  Mean   :93.58   Mean   :-40.5   Mean   :3.621   Mean   :5167
##  3rd Qu.:93.99   3rd Qu.:-36.4   3rd Qu.:4.961   3rd Qu.:5228
##  Max.   :94.77   Max.   :-26.9   Max.   :5.045   Max.   :5228
##       y
##  Length:41188
##  Class :character
##  Mode  :character
##
##
##
```

```
# class disbalance
table(data$y)
```

```
##
##    no   yes
## 36548  4640
```

```
# missing values
ceiling(colSums(is.na(data)) / nrow(data) * 100)
```

```
##            age            job        marital      education        default
##              0              1              1              5             21
##        housing           loan        contact          month    day_of_week
##              3              3              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
```

Dataset consist of 41188 instances and 20 features. It contains missing values, later we will take care of them.

1.1 Initial features

The dataset's source provides us with following feature description. The features related to client:

Feature	Description
Age (numeric)	client's age.
Job (categorical)	client's type of job
Marital (categorical)	client's marital status
Education (categorical)	client's education level
Default (categorical)	client's default status (has credit in default?)
Housing (categorical)	client's house loan status (has client a house loan?)
Loan (categorical)	client's loan status (has client a loan?)

The features related to the current marketing campaign:

Feature	Description
Contact (categorical)	contact type
Month (categorical)	last contact month
Day_of_week (categorical)	last contact day of the week
Duration (numeric)	last contact duration. **In purpose of preventing data leakage, we will not use that feature, because we can't retrieve this feature for future clients.**
Campaign (numeric)	shows number of contacts during current marketing campaign

The features related to the previous marketing campaign:

Feature	Description
Pdays (numeric)	shows number of days passed after last contact in current campaign (999 – means this client wasn't contacted before)
Previous (numeric)	shows number of contacts in previous marketing campaign
Poutcome (categorical)	shows the results of previous marketing campaign

Features related to the bank:

Feature	Description
Emp.var.rate (numeric)	employment variation rate - quarterly indicator
Nr.employed (numeric)	number of employees - quarterly indicator

Social and economic context features:

Feature	Description
Cons.price.idx (numeric)	consumer price index - monthly indicator
Cons.conf.idx (numeric)	consumer confidence index - monthly indicator
Euribor3m (numeric)	the Euro Interbank offered 3 month rate - daily indicator

The target variable:

Target variable	Description
The target variable is y	has the client subscribed a term deposit? (binary: 'yes','no').

First, we're going to look at unique values for every categorical feature. We might need this information for making factor representation for these features.

```
# unique values of categorical features
categorical_feats <- c("default", "housing", "loan", "contact", "education",
                       "job", "marital", "month", "day_of_week", "poutcome")
for (feat in categorical_feats) {
  print(feat)
  print(unique(data[, feat]))
  print("_____________________________________________________________________")
}
```

```
## [1] "default"
## [1] "no"  NA    "yes"
## [1] "_____________________________________________________________________"
## [1] "housing"
## [1] "no"  "yes" NA
## [1] "_____________________________________________________________________"
## [1] "loan"
## [1] "no"  "yes" NA
## [1] "_____________________________________________________________________"
## [1] "contact"
## [1] "telephone" "cellular"
## [1] "_____________________________________________________________________"
## [1] "education"
## [1] "basic.4y"            "high.school"         "basic.6y"
## [4] "basic.9y"            "professional.course" NA
## [7] "university.degree"   "illiterate"
## [1] "_____________________________________________________________________"
## [1] "job"
##  [1] "housemaid"     "services"      "admin."        "blue-collar"
##  [5] "technician"    "retired"       "management"    "unemployed"
##  [9] "self-employed" NA              "entrepreneur"  "student"
## [1] "_____________________________________________________________________"
## [1] "marital"
## [1] "married"  "single"   "divorced" NA
## [1] "_____________________________________________________________________"
## [1] "month"
##  [1] "may" "jun" "jul" "aug" "oct" "nov" "dec" "mar" "apr" "sep"
## [1] "_____________________________________________________________________"
```

```
## [1] "day_of_week"
## [1] "mon" "tue" "wed" "thu" "fri"
## [1] "----------------------------------------------------------------------"
## [1] "poutcome"
## [1] "nonexistent" "failure"     "success"
## [1] "----------------------------------------------------------------------"
```

According to given and obtained information, we can group the features by 4 categories:

- Binary features ("default", "housing", "loan", "contact");
- Ordinal features ("education");
- Nominal features ("job", "marital", "month", "day_of_week", "poutcome");
- Continuous features ("age", "duration", "campaign", "pdays", "previous", "emp.var.rate", "cons.price.idx", "cons.conf.idx", "euribor3m", "nr.employed").

```
# group features
bin_feats <- c("default", "housing", "loan", "contact")
ordinal_feats <- c("education")
nominal_feats <- c("job", "marital",
                   "month", "day_of_week", "poutcome")
continuous_feats <- c("age", "duration", "campaign", "pdays", "previous",
                      "emp.var.rate", "cons.price.idx", "cons.conf.idx",
                      "euribor3m", "nr.employed")
```

1.2 Initial data preprocessing

Here we create a factor representation for categorical features.

```
data$y <- as.factor(data$y)
data$marital <- factor(data$marital, levels = c("single", "divorced",
                                                "married"), ordered = FALSE)
data$education <- factor(data$education, levels = c('illiterate',
                                                    'basic.4y','basic.6y',
                                                    'basic.9y','high.school',
                                                    'professional.course',
                                                    'university.degree'),
                         ordered = TRUE)
data$default <- as.factor(data$default)
data$housing <- as.factor(data$housing)
data$loan <- as.factor(data$loan)
data$contact <- as.factor(data$contact)

data$day_of_week <- factor(data$day_of_week,
                           levels = c("mon", "tue", "wed", "thu", "fri"),
                           ordered = FALSE)
data$poutcome <- factor(data$poutcome, levels = c("nonexistent", "failure",
                                                  "success"), ordered = FALSE)
data$job <- factor(data$job, levels = c('admin.','blue-collar','entrepreneur',
                                        'housemaid', 'management','retired',
                                        'self-employed','services',
```

```
                                          'student','technician','unemployed'),
                    ordered = FALSE)

# temporarily treat month as an ordered factor
data$month <- factor(data$month, levels = c("jan", "feb", "mar", "apr", "may",
                                            "jun", "jul", "aug", "sep", "oct",
                                            "nov", "dec"), ordered = TRUE)
```

1.3 Feature engineering

As all the examples in dataset are ordered by date and for each example we know the month of call, we can calculate year column since we know that the campaign has been started in 2008. Also we can construct date feature. It would be useful for further data analysis.

```
# adding year
data$year <- NA
year = 2008
for (i in 1:(nrow(data)-1)) {
  data[i, 'year'] <- year
  if (data[i, 'month'] > data[i+1, 'month']) {
    year <- year + 1
  }
}
data[i+1, 'year'] <- year
continuous_feats <- c(continuous_feats, "year")

# temporal variable for creating barplots
data$y_num <- as.integer(as.integer(data$y) - 1)

# adding date
data$date <- paste(as.character(data$year), "01",
                   as.character(as.numeric(data$month)),
                   sep = "-")
data$date <- as.Date(data$date)
levels = as.character(unique(data$date))
data$date <- as.character(data$date)
data$date <- factor(data$date, levels=levels, ordered = TRUE)
```

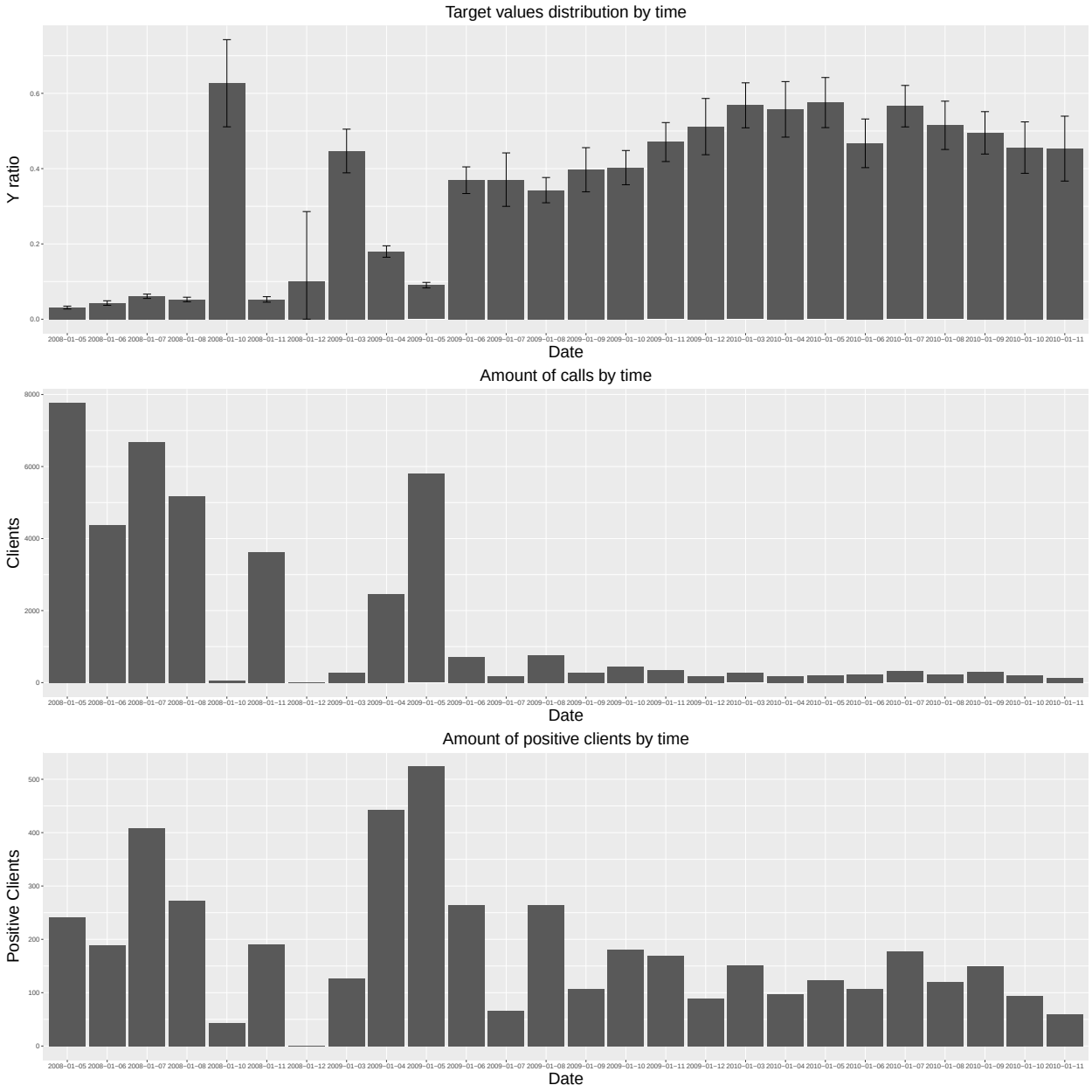
2 Exploratory Data Analysis

First, we want to know several things:

1. How the quality of campaign changed during the time;
2. How the amount and proportion of the clients changed during the time;
3. How the number of calls changed during the time.

For that purpose, we plot following barplots and histograms.

```
# Let's see how the quality of campaign changed during the time
# Preparing data for plots
data_sse <- summarySE(data, measurevar="y_num", groupvars="date")
# alpha for building confidence interval
a <- .05
interval_value <- qnorm(1 - a / 2) * sqrt((1 / data_sse$N) * data_sse$y_num *
                                            (1 - data_sse$y_num))
data_sse$upper_bound <- data_sse$y_num + interval_value
data_sse$lower_bound <- data_sse$y_num - interval_value
data_sse$lower_bound <- ifelse(data_sse$lower_bound > 0, data_sse$lower_bound,
                               0)
settings <- theme(plot.title = element_text(hjust = 0.5, size = 22),
                  axis.title.x = element_text(size = 22),
                  axis.title.y = element_text(size = 22))

c1 <- ggplot(data_sse, aes(x=date, y=y_num)) +
  geom_bar(position=position_dodge(), stat="identity") +
  geom_errorbar(aes(ymin=lower_bound, ymax=upper_bound),
                width=.2,
                position=position_dodge(.9)) +
  labs(x = "Date", y = "Y ratio") +
  ggtitle("Target values distribution by time") +
  settings

c2 <- ggplot(data, aes(x = date)) +
  geom_bar() +
  labs(x = "Date", y = "Clients") +
  ggtitle("Amount of calls by time") +
  settings

c3 <- ggplot(data, aes(x = date, y=y_num)) +
  geom_bar(stat = "identity", na.rm=TRUE) +
  labs(x = "Date", y = "Positive Clients") +
  ggtitle("Amount of positive clients by time") +
  settings

cowplot::plot_grid(c1, c2, c3, labels = NULL, ncol = 1, align = "v")
```



By looking at this plot, we can make the following conclusions:

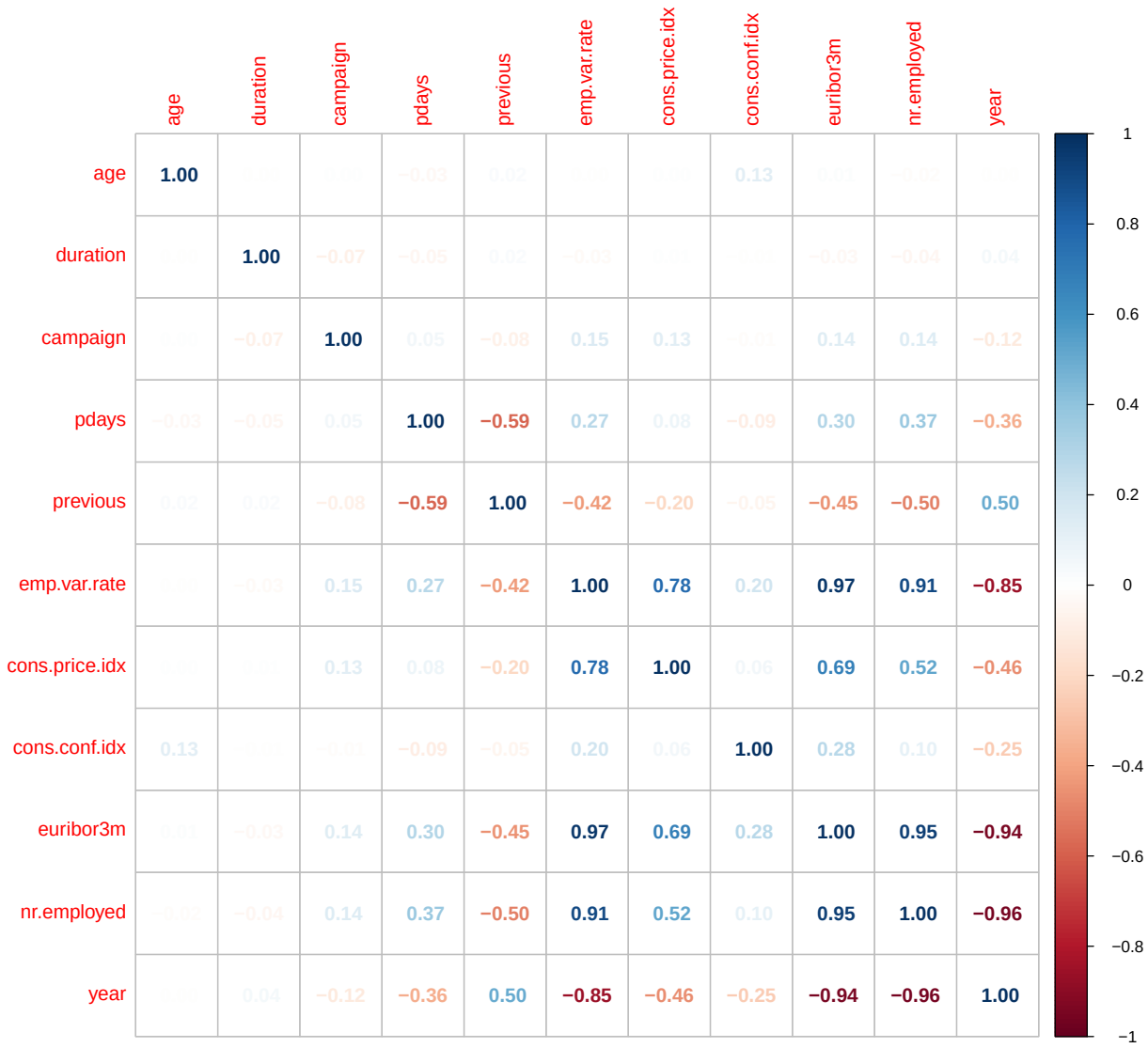
1. Before June 2009, the bank used to make the calls to many clients. After this date, number of people, who was contacted by bank, per month has been decreased.
2. Number of attracted clients per month has also been decreased.
3. However, percentage of attracted clients per month has been sufficiently increased.

Probably, the bank had been changed its client searching policy after June 2009. Further we are going to make an investigation and see how the bank could increase the percentage of attracted clients per month. But before going deeper, we will plot correlation matrix.

2.1 Correlation matrix and independence graph

Here we plot correlation matrix for continuous features of the dataset.

```
# plotting correlation matrix for continuous features
corrmat <- cor(data[, continuous_feats],
               use="pairwise.complete.obs")
corrplot(corrmat, method="number")
```



Then, in order to better understand the relationship between continuous features, we plot independence graph based on partial correlation matrix.

```r
# plotting independence graph
S <- var(data[, continuous_feats], use="pairwise.complete.obs")
P <- cov2cor(S)
R <- -cov2cor(solve(S))
thr <- 0.2
G <- abs(R)>thr
diag(G) <- 0
Gi <- as(G, "igraph")
tkplot(Gi, vertex.color="white")
```

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

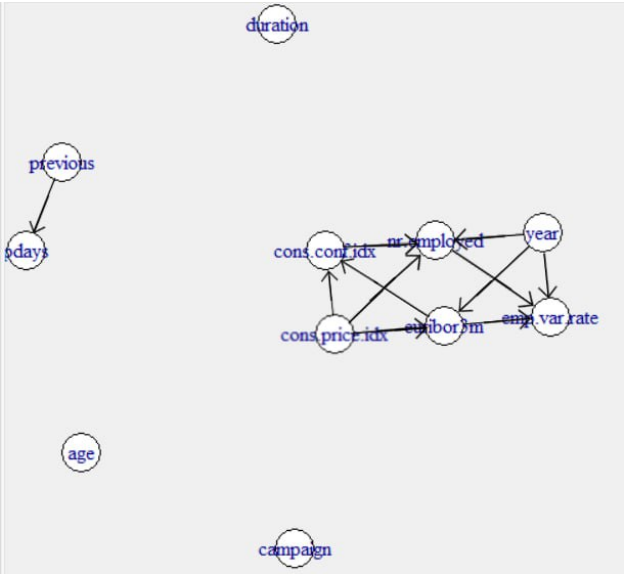


Figure 1: Independence graph

Here we can clearly see that features “year”, emp.var.rate”, “cons.price.idx”, “cons.conf.idx”, “euribor3m”, “nr.employed” are highly correlated. A high correlation between social and economic context features seems to be natural. Those features depend on different aspects (historical and social events, crises etc) which are not available in this dataset. However they can be reflected by year. In our opinion, that’s why feature “year” correlates to social and economic context features.

We can also notice a correlation between “pdays” and “previous”, but it can be explained by the fact that if the client hadn’t been contacted during previous campaign (“previous” = 0), pdays equals 999.

While applying statistical models, we have to handle correlating features in order to avoid multicollinearity.

After plotting independence graph, we manually added causality according to our assumptions. For, example, “year” can’t depend on social and economic context. Also, features related to the bank can’t affect features related to global market.

2.2 Features’ distridution

Next, we plot the distribution of available predictors. For each predictor we plot distribution of its values split by target variable and by dates.

2.2.1 Continuous variables. Here we plot distribution of continuous features.

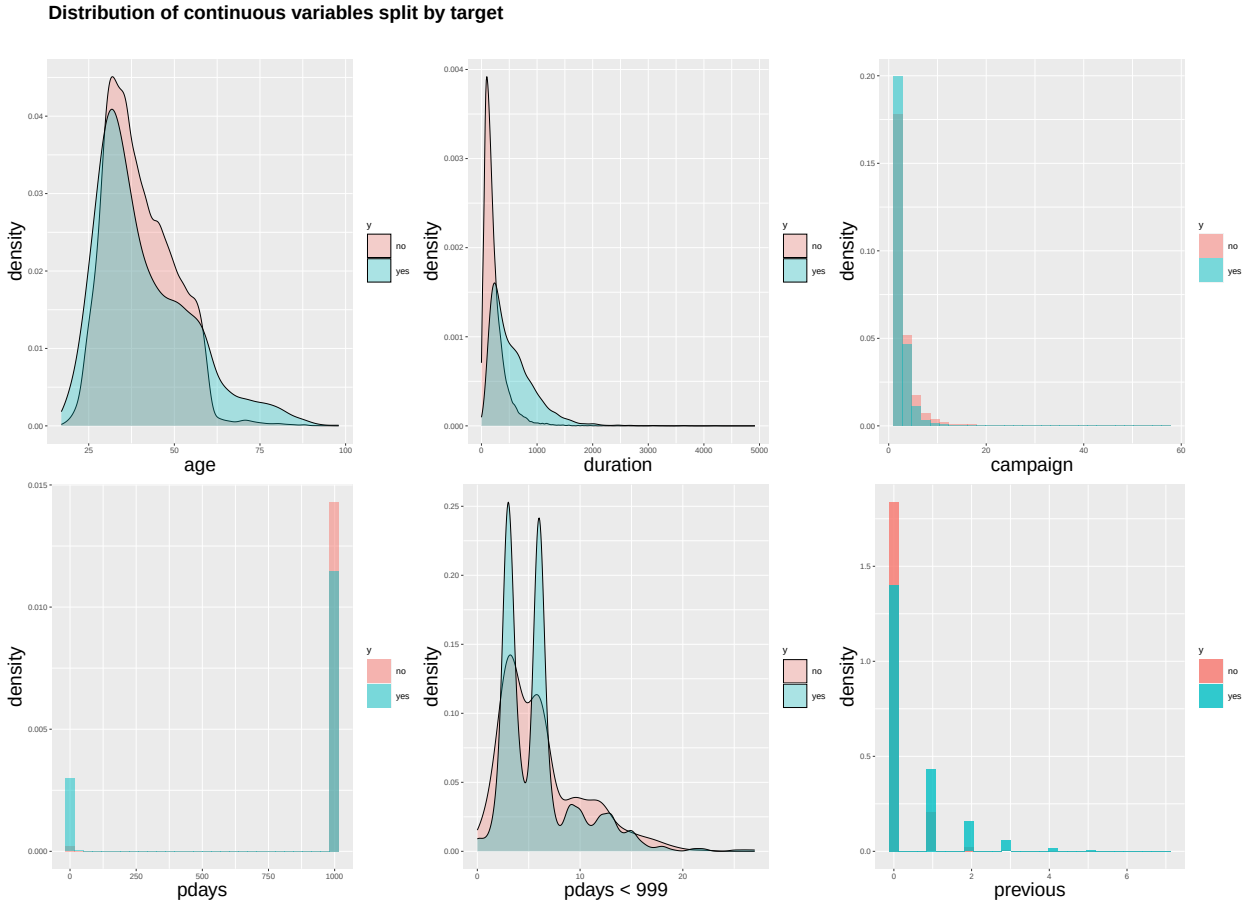
First, we plot distribution of continuous variables split by target value.

```
# distribution of continuous variables split by target pt 1

settings <- theme(axis.title.x = element_text(size = 22),
                  axis.title.y = element_text(size = 22),
                  legend.key.size = unit(1, 'cm'),
                  legend.text = element_text(size=10))

g_age <- ggplot(data,aes(x=age,fill=y))+
  ggplot2::geom_density(alpha=.3) + settings

g_duration <- ggplot(data,aes(x=duration,fill=y))+
  ggplot2::geom_density(alpha=.3) + settings

g_campaign <- ggplot(data,aes(x=campaign,fill=y))+
  ggplot2::geom_histogram(aes(y=0.5*..density..),
                          alpha=0.5,position='identity') + labs(y = "density")+
  settings

g_pdays <- ggplot(data,aes(x=pdays,fill=y))+
  ggplot2::geom_histogram(aes(y=0.5*..density..),
                          alpha=0.5,position='identity') + labs(y = "density")+
  settings

# also we plot distribution of real pdays (<999)
g_pdays_f <- ggplot(data[data$pdays<999, ],aes(x=pdays,fill=y))+
  ggplot2::geom_density(alpha=.3)+ labs(x = "pdays < 999") + settings

g_previous <- ggplot(data,aes(x=previous,fill=y))+
  ggplot2::geom_histogram(aes(y=0.5*..density..),
                          alpha=0.8,position='identity') + labs(y = "density")+
  settings

plot <- plot_grid(g_age, g_duration, g_campaign,
                  g_pdays, g_pdays_f, g_previous,
                  labels = NULL,
                  nrow = 2, ncol = 3)
```

```
## Warning: The dot-dot notation (`..density..`) was deprecated in ggplot2 3.4.0.
## i Please use `after_stat(density)` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

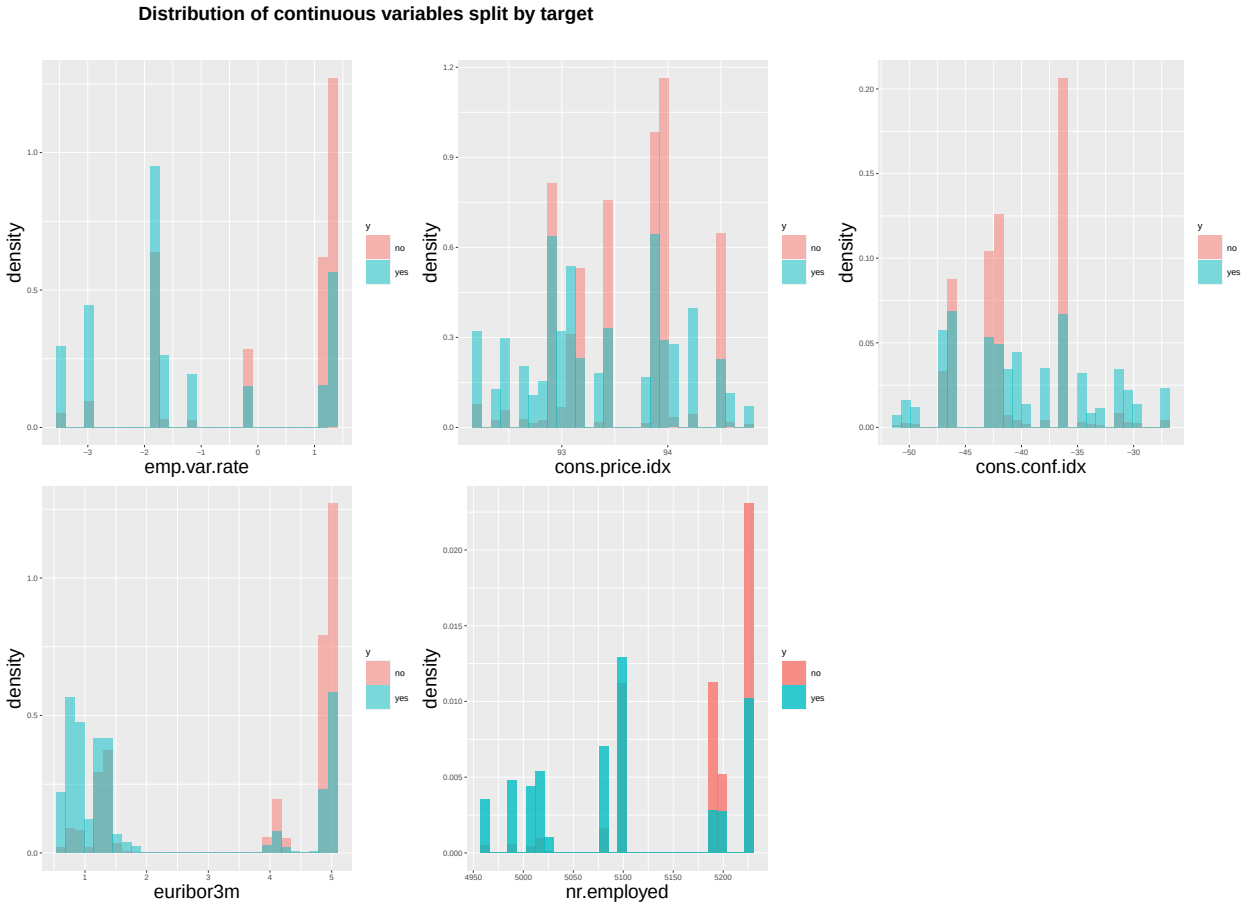
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
title <- ggdraw() +
  draw_label(
    "Distribution of continuous variables split by target",
    fontface = 'bold',
    x = 0,
    hjust = -0.1,
    size = 22
  )
plot_grid(
  title, plot,
  ncol = 1,
  rel_heights = c(0.1, 1)
)
```




```
# distribution of continuous variables split by target pt 2

g_emp.var.rate <- ggplot(data,aes(x=emp.var.rate,fill=y))+
  geom_histogram(aes(y=0.5*..density..),
                 alpha=0.5,position='identity') + labs(y = "density") + settings

g_cons.price.idx <- ggplot(data,aes(x=cons.price.idx,fill=y))+
  geom_histogram(aes(y=0.5*..density..),
```

```
                alpha=0.5,position='identity') + labs(y = "density") + settings

g_cons.conf.idx <- ggplot(data,aes(x=cons.conf.idx,fill=y))+
  geom_histogram(aes(y=0.5*..density..),
                alpha=0.5,position='identity') + labs(y = "density") + settings

g_euribor3m <- ggplot(data,aes(x=euribor3m,fill=y))+
  geom_histogram(aes(y=0.5*..density..),
                alpha=0.5,position='identity') + labs(y = "density") + settings

g_nr.employed <- ggplot(data,aes(x=nr.employed,fill=y))+
  geom_histogram(aes(y=0.5*..density..),
                alpha=0.8,position='identity') + labs(y = "density") + settings

plot <- plot_grid(g_emp.var.rate, g_cons.price.idx, g_cons.conf.idx,
                  g_euribor3m, g_nr.employed,
                  labels = NULL,
                  nrow = 2, ncol = 3)

title <- ggdraw() +
  draw_label(
    "Distribution of continuous variables split by target",
    fontface = 'bold',
    x = 0,
    hjust = -0.3,
    size = 22
  )
plot_grid(
  title, plot,
  ncol = 1,
  rel_heights = c(0.1, 1)
)
```

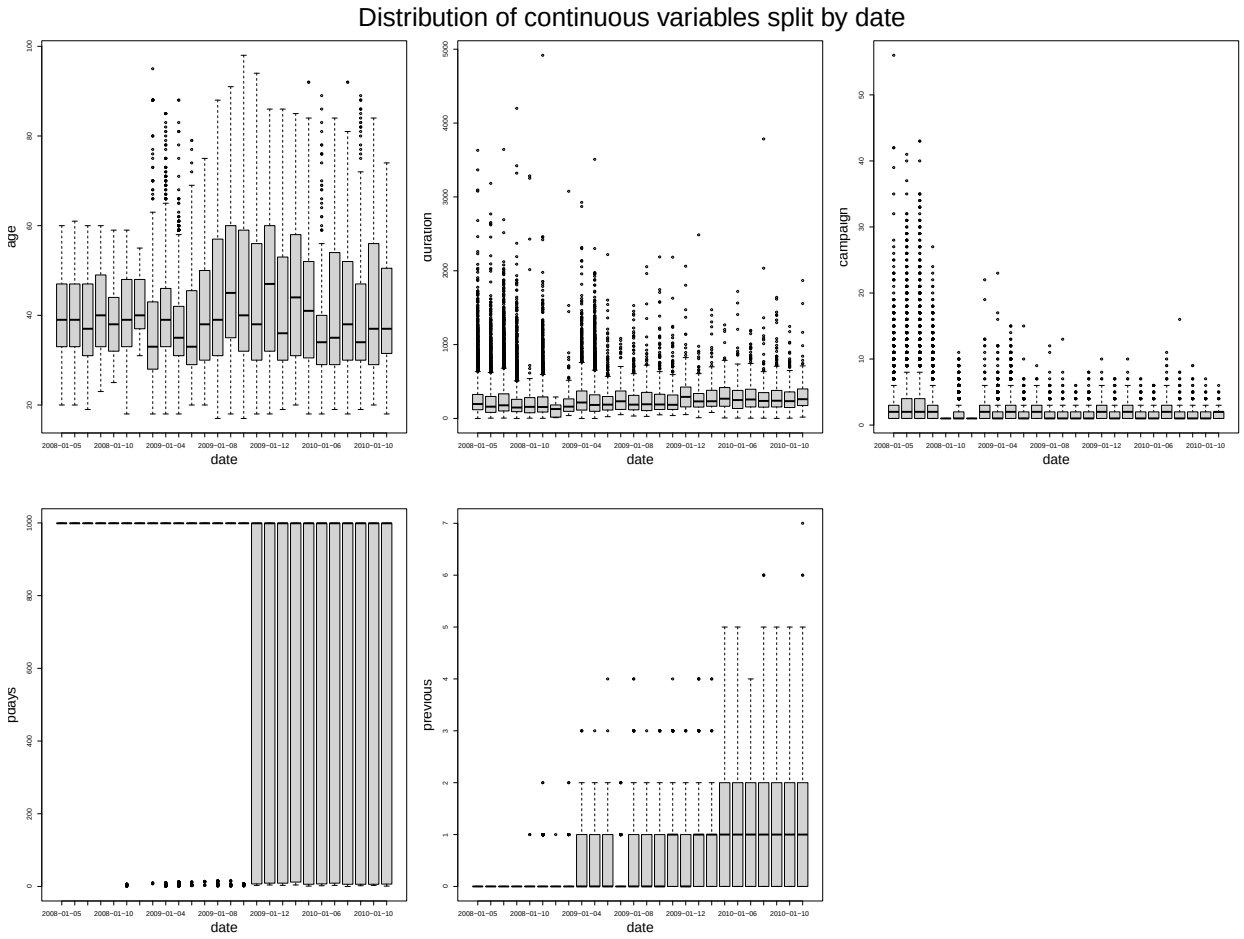


Considering 2 groups, we can observe that following features have different distributions:

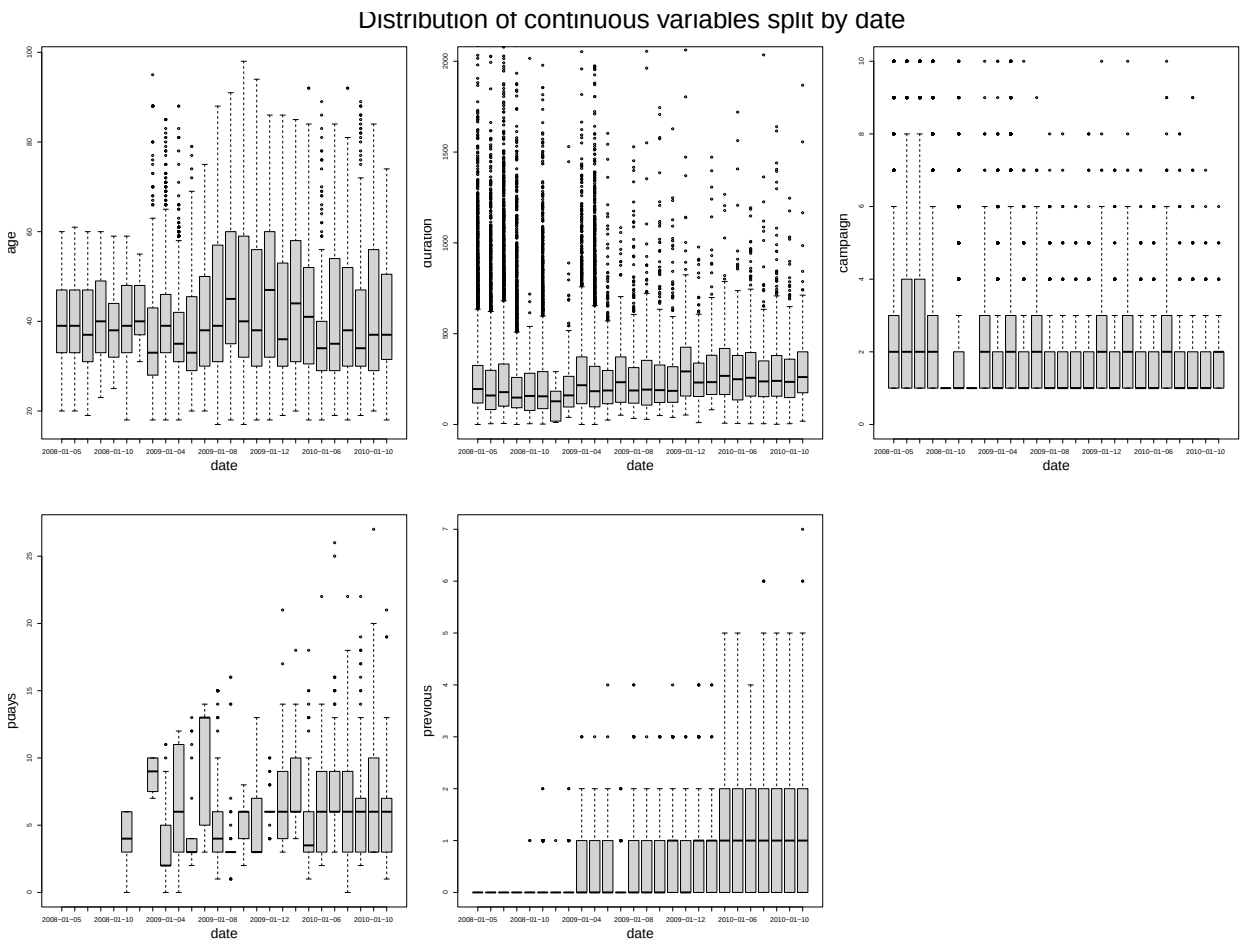
- Duration – it simply could be explained by the fact that we need some time to persuade the costumer to take a term deposit and explain him all of details. In case if costumer is not interested, the conversation ends pretty soon.
- Pdays – it's clearly seen that if a costumer wasn't contacted during the previous campaign, the probability of rejection is higher.
- Previous – the same situation as we observed for pdays.

Second, we plot distribution of continuous variables split by date.

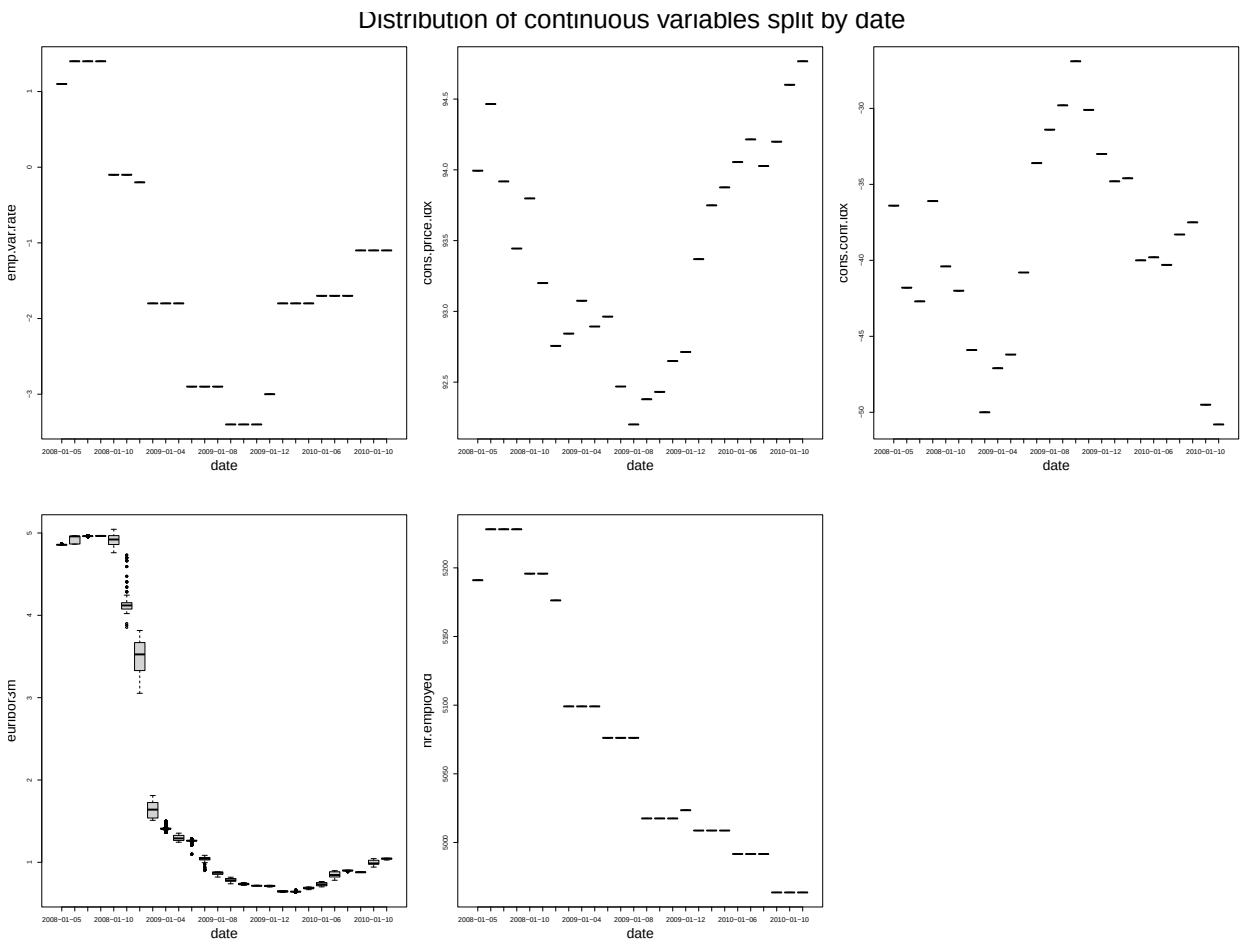
```
# distribution of continuous variables split by date pt 1
par(mfrow=c(2, 3))
for (feat in continuous_feats[1:5]) {
  boxplot(data[,feat] ~ data[,"date"], xlab="date", ylab=feat,
          cex.lab=2)
}
mtext("Distribution of continuous variables split by date", side = 3,
      line = - 2.5, outer = TRUE, cex=2.5)
```




```
# Let's set ylim for some plots in order to eliminate some outliers with
# extremely huge values
par(mfrow=c(2, 3))
boxplot(data$age ~ data$date, xlab="date", ylab="age", cex.lab=2)
boxplot(data$duration ~ data$date, xlab="date",
        ylab="duration", ylim = c(0, 2000), cex.lab=2)
boxplot(data$campaign ~ data$date, xlab="date", ylab="campaign",
        ylim = c(0, 10), cex.lab=2)
# here we're considering only real pdays
boxplot(data[data$pdays<999, 'pdays'] ~ data[data$pdays<999, 'date'],
        xlab='date', ylab='pdays', cex.lab=2)
boxplot(data$previous ~ data$date, xlab="date", ylab="previous", cex.lab=2)
mtext("Distribution of continuous variables split by date", side = 3,
      line = - 2, outer = TRUE, cex=2.5)
```

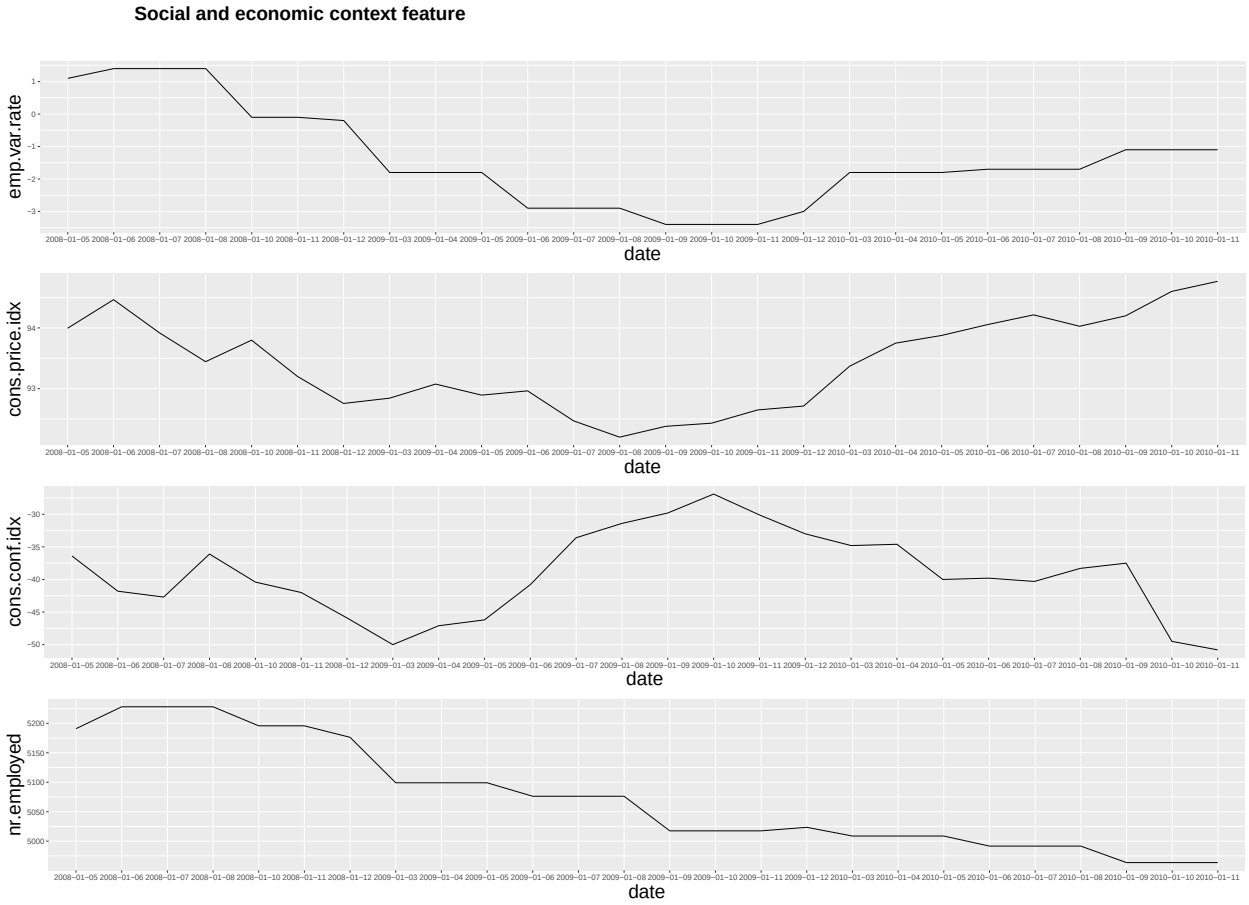



```
# distribution of continuous variables split by date pt 2
par(mfrow=c(2, 3))
for (feat in continuous_feats[6:10]) {
  boxplot(data[,feat] ~ data[,"date"], xlab="date", ylab=feat, cex.lab=2)
}
mtext("Distribution of continuous variables split by date", side = 3,
      line = - 2, outer = TRUE, cex=2.5)
```




```
# some of these features are constant during a month or three months
# let's use other kind of plot

agg_tbl <- data %>% dplyr::group_by(date) %>%
  dplyr::summarise(emp.var.rate=unique(emp.var.rate))
agg_tbl$cons.price.idx <- as.data.frame(data %>% dplyr::group_by(date) %>%
  dplyr::summarise(cons.price.idx=unique(cons.price.idx)))$cons.price.idx
agg_tbl$cons.conf.idx <-as.data.frame(data %>% dplyr::group_by(date) %>%
  dplyr::summarise(cons.conf.idx=unique(cons.conf.idx)))$cons.conf.idx
agg_tbl$nr.employed <-as.data.frame(data %>% dplyr::group_by(date) %>%
  dplyr::summarise(nr.employed=unique(nr.employed)))$nr.employed
list_plots <- list()
list_plots <- outlist <- append(list_plots, list(ggplot(agg_tbl,
                                                        aes(x = date,
                                                            y = emp.var.rate,
                                                        group = 1)) + geom_line() +
                                                        settings))
list_plots <- outlist <- append(list_plots, list(ggplot(agg_tbl,
                                                        aes(x = date,
                                                            y = cons.price.idx,
                                                        group = 1)) + geom_line() +
                                                        settings))
```

```
list_plots <- outlist <- append(list_plots, list(ggplot(agg_tbl,
                                                        aes(x = date,
                                                            y = cons.conf.idx,
                                                        group = 1)) + geom_line() +
                                                        settings))
list_plots <- outlist <- append(list_plots, list(ggplot(agg_tbl,
                                                        aes(x = date,
                                                            y = nr.employed,
                                                        group = 1)) + geom_line() +
                                                        settings))
plot <- plot_grid(plotlist=list_plots,
                  labels = NULL,
                  nrow = 4, ncol = 1)
title <- ggdraw() +
  draw_label(
    "Social and economic context feature",
    fontface = 'bold',
    x = 0,
    hjust = -0.4,
    size = 22
  )
plot_grid(
  title, plot,
  ncol = 1,
  rel_heights = c(0.1, 1)
)
```



Here we can see how bank, social and economic context features changed their values starting from a beginning of the campaign. By looking at emp.var.rate's trend line, it's clearly seen that the bank's administration decided to decrease the number of employees involved in this campaign after August 2008. Perhaps, the goal of that decision was saving man hours and labor costs.

Moreover, we can estimate inflation rate by looking at cons.price.idx's trend line. At first, it shows reducing inflation. As the rule, reducing inflation make interest rate (euribor3m) become lower. This could be another reason why bank's administration decided to decrease the number of employees involved in this campaign: during the periods characterized by low interest rate, it's more profitable to use other kinds of investments.

Also we can observe decreasing Euribor rate which coincides to decreasing of total number of bank's employees.

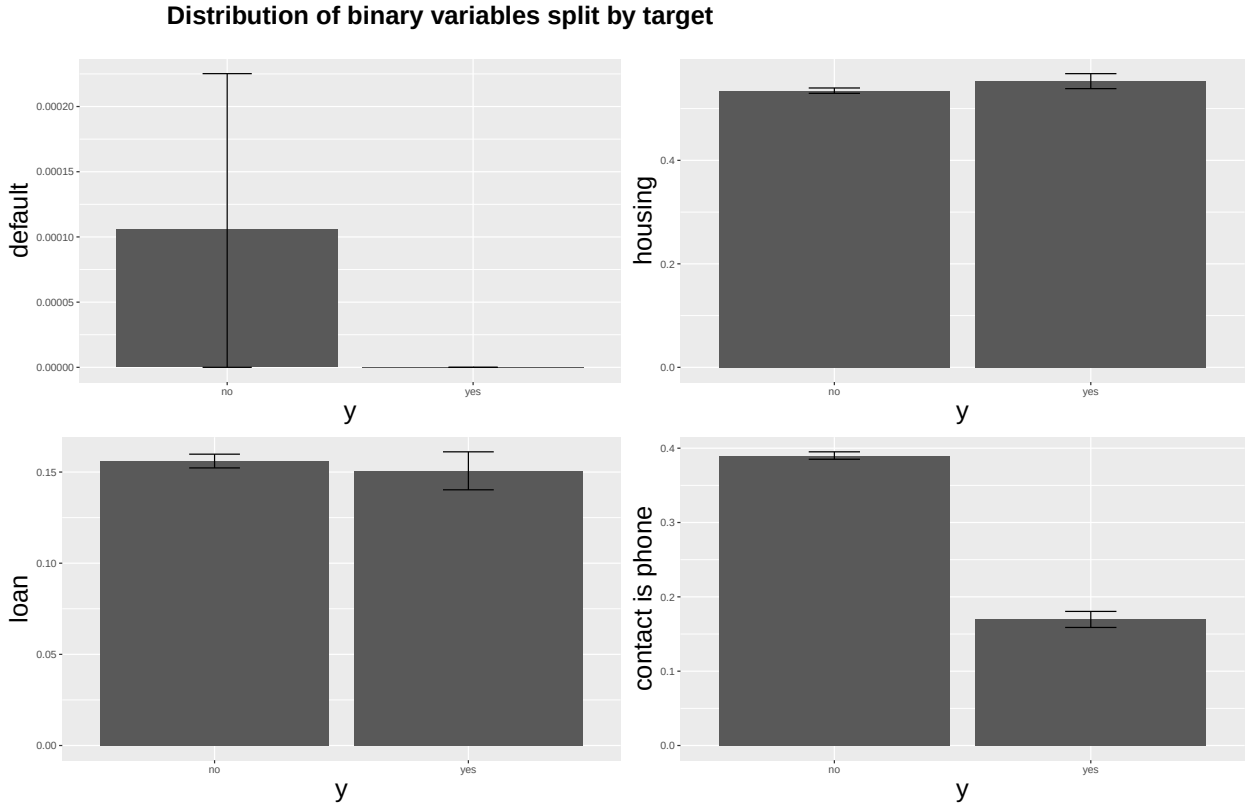
Previously we have seen that the bank reduced number of contacts after the middle of 2009. If we look at the boxplot for "Previous", we can see, that after this date bank started to make an emphasis on the clients who was contacted before. Probably, those clients are loyal one and there's a high probability that they'll accept the proposition again. This could explain the raising of acceptance ratio that we have seen before.

2.2.2 Binary variables. Here we plot distribution of binary features.

As before, we plot distribution of the variables split by target value.

```
list_plots <- list()
# alpha for building confidence interval
```

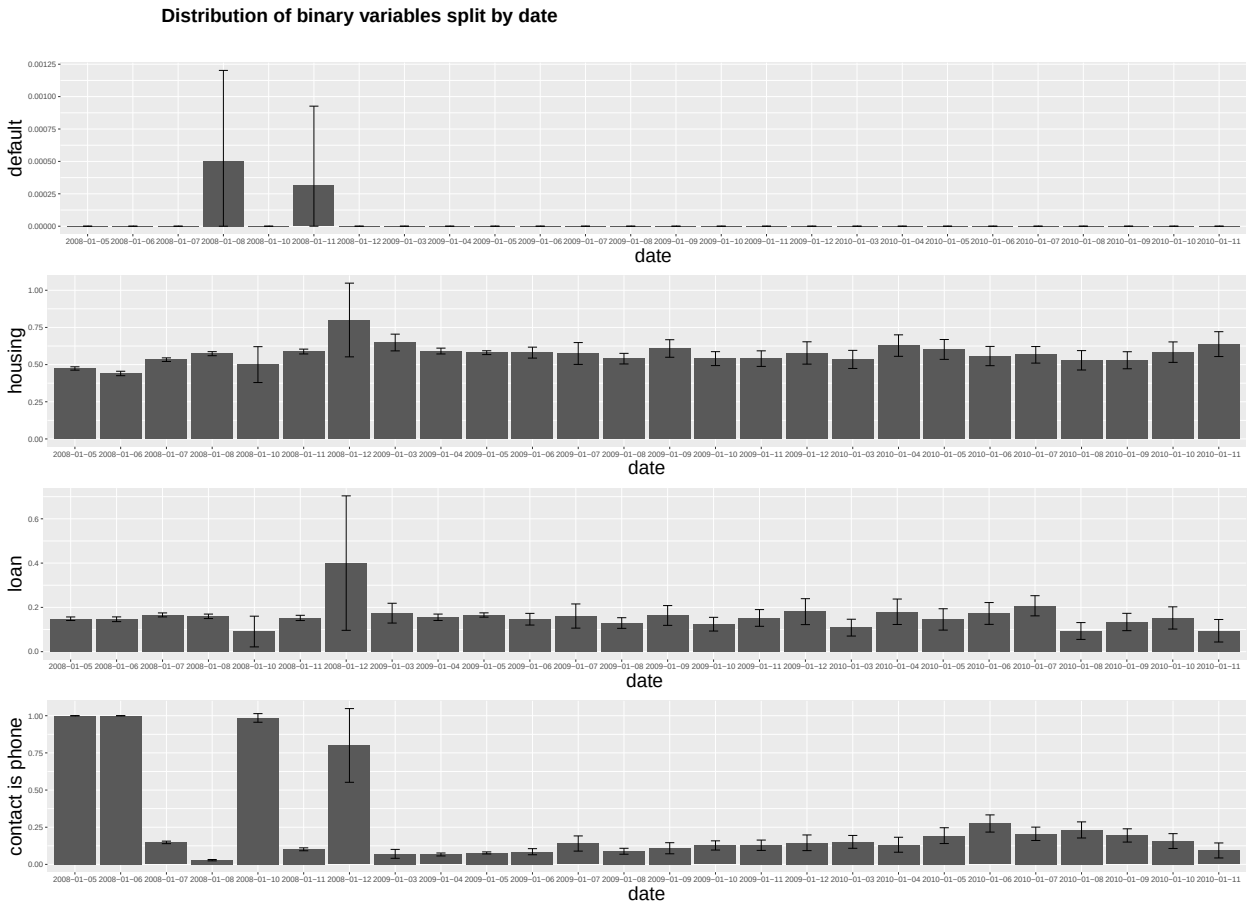
```
a <- .05
for (feat in bin_feats) {
  if (feat == "contact") {lab = "contact is phone"} else {lab = feat}
  data_feat <- data[!is.na(data[, feat]), ]
  data_feat[, feat] <- as.factor(data_feat[, feat])
  data_feat[, feat] <- as.numeric(data_feat[, feat]) - 1
  data_temp_sse <- summarySE(data_feat, measurevar=feat, groupvars="y")
  interval_value <- qnorm(1 - a / 2) * sqrt((1 / data_temp_sse$N) *
                                            data_temp_sse[, feat] *
                                            (1 - data_temp_sse[, feat]))
  interval_value[is.na(interval_value)] <- 0
  data_temp_sse$upper_bound <- data_temp_sse[, feat] + interval_value
  data_temp_sse$lower_bound <- data_temp_sse[, feat] - interval_value
  data_temp_sse$lower_bound <- ifelse(data_temp_sse$lower_bound > 0,
                                      data_temp_sse$lower_bound, 0)
  curr_plot <- ggplot(data_temp_sse, aes(x=.data[["y"]], y=.data[[feat]])) +
    geom_bar(position=position_dodge(), stat="identity") +
    geom_errorbar(aes(ymin=lower_bound, ymax=upper_bound),
                  width=.2,                    # Width of the error bars
                  position=position_dodge(.9)) +
    settings + ylab(lab)
  list_plots <- outlist <- append(list_plots,list(curr_plot))
}

plot <- plot_grid(plotlist=list_plots,
          labels = NULL,
          nrow = 2, ncol = 2)

title <- ggdraw() +
  draw_label(
    "Distribution of binary variables split by target",
    fontface = 'bold',
    x = 0,
    hjust = -0.3,
    size = 22
  )
plot_grid(
  title, plot,
  ncol = 1,
  rel_heights = c(0.1, 1)
)
```



Now we plot distribution of binary variables split by date.

```
# distribution of binary variables split by date
list_plots <- list()
for (feat in bin_feats) {
  if (feat == "contact") {lab = "contact is phone"} else {lab = feat}
  data_temp <- data[!is.na(data[, feat]), ]
  data_temp[, feat] <- as.factor(data_temp[, feat])
  data_temp[, feat] <- as.numeric(data_temp[, feat]) - 1
  data_temp_sse <- summarySE(data_temp, measurevar=feat, groupvars="date")
  interval_value <- qnorm(1 - a / 2) * sqrt((1 / data_temp_sse$N) *
                                            data_temp_sse[, feat] *
                                            (1 - data_temp_sse[, feat]))
  interval_value[is.na(interval_value)] <- 0
  data_temp_sse$upper_bound <- data_temp_sse[, feat] + interval_value
  data_temp_sse$lower_bound <- data_temp_sse[, feat] - interval_value
  data_temp_sse$lower_bound <- ifelse(data_temp_sse$lower_bound > 0,
                                      data_temp_sse$lower_bound, 0)
  curr_plot <- ggplot(data_temp_sse, aes(x=.data[["date"]], y=.data[[feat]])) +
    geom_bar(position=position_dodge(), stat="identity") +
    geom_errorbar(aes(ymin=lower_bound, ymax=upper_bound),
                  width=.2,                    # Width of the error bars
                  position=position_dodge(.9)) +
    settings + ylab(lab)
  list_plots <- outlist <- append(list_plots,list(curr_plot))
```

```
}

plot <- plot_grid(plotlist=list_plots,
                  labels = NULL,
                  nrow = 4, ncol = 1)
title <- ggdraw() +
  draw_label(
    "Distribution of binary variables split by date",
    fontface = 'bold',
    x = 0,
    hjust = -0.4,
    size = 22
  )
plot_grid(
  title, plot,
  ncol = 1,
  rel_heights = c(0.1, 1)
)
```



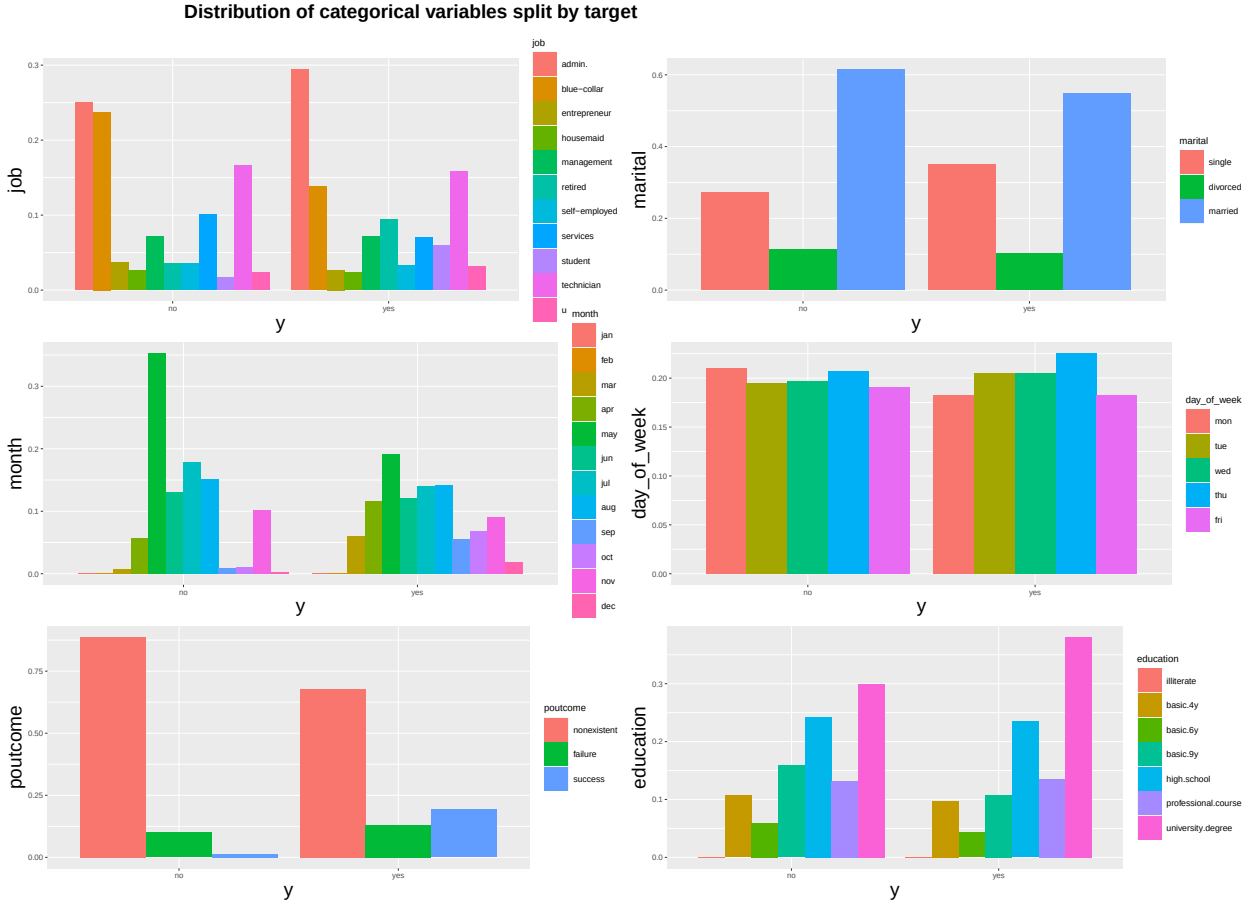
We can provide obtained plots with following comments:

- The boxplots for "Default" are not informative because there are only 3 people who has credit in default.
- By looking at the boxplot for "Contact" (split by target), we may notice that the acceptance rate for

cellular contact is higher. But if we look at other boxplot for "Contact" (split by date), we can see that bank simply decided to give a preference to cellular contact after the changes of the policy, which has increased acceptance rate. It explains observed effect.

2.2.3 Categorical variables. Here we plot distribution of categorical features. Again, we plot distribution of the variables split by target value.

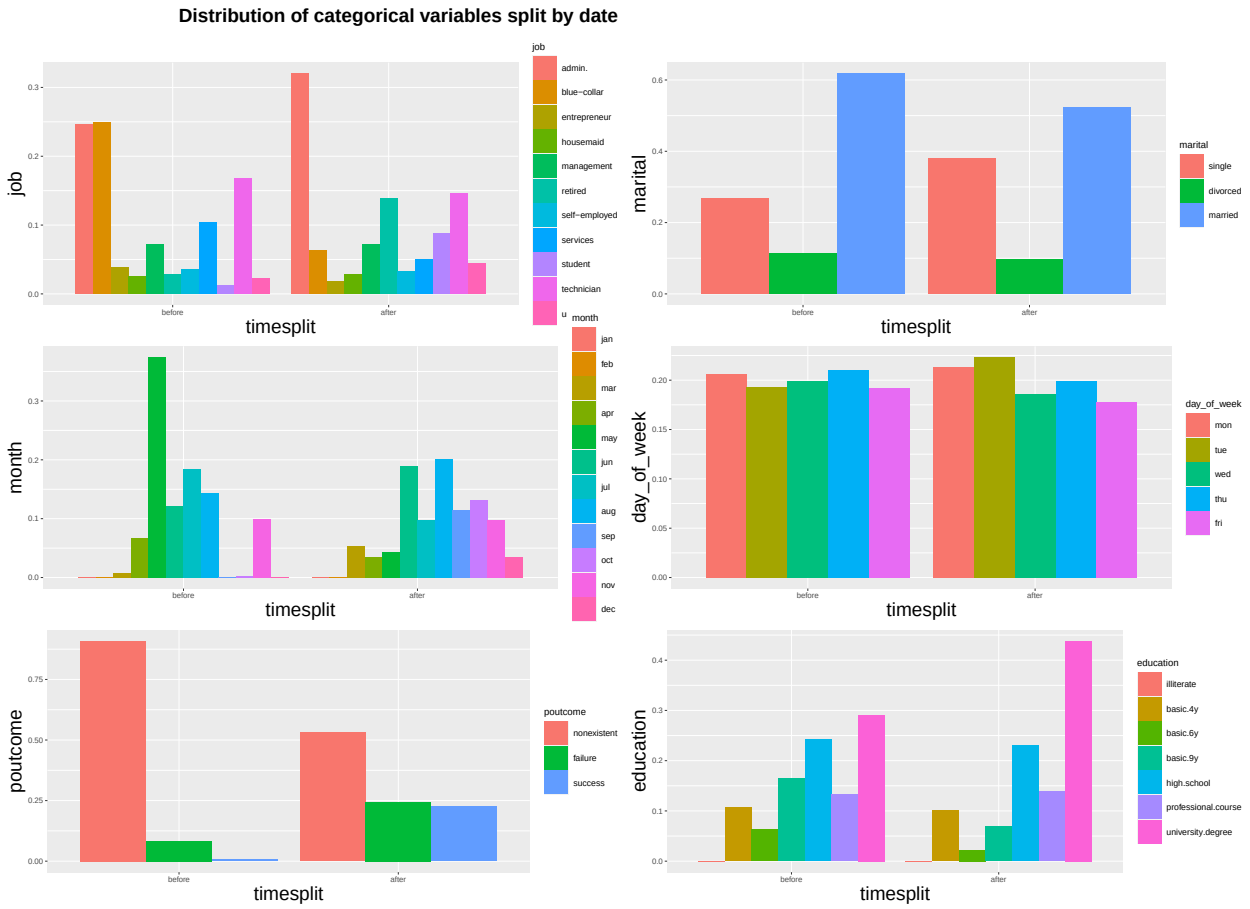
```
# distribution of categorical variables split by target
list_plots <- list()
for (feat in c(nominal_feats, ordinal_feats)) {
  data_feat <- with(data, table(y, data[, feat], dnn = c('y', feat)))
  data_feat <- as.data.frame(data_feat)
  # For each group we normalize the frequencies separately
  data_feat$NormFreq <- data_feat$Freq /
    sapply(data_feat$y, function(x) sum(data_feat[data_feat$y == x, 'Freq']))
  curr_plot <- ggplot(data_feat, aes(x=.data[["y"]], y=.data[["NormFreq"]],
                                     fill = .data[[feat]])) +
    geom_col(position = 'dodge') +
    ylab(feat) +
    settings
  list_plots <- outlist <- append(list_plots,list(curr_plot))
}
plot <- plot_grid(plotlist=list_plots, labels = NULL, nrow = 3, ncol = 2)
title <- ggdraw() +
  draw_label(
    "Distribution of categorical variables split by target",
    fontface = 'bold',
    x = 0,
    hjust = -0.4,
    size = 22
  )
plot_grid(
  title, plot,
  ncol = 1,
  rel_heights = c(0.1, 1)
)
```



Now we plot distribution of categorical variables split by date. In case of categorical features, we can't plot the features' distribution for each date. That's why we will consider only 2 periods: before changing of policy and after.

```
# distribution of categorical variables split by date
list_plots <- list()
cut_date <- as.Date("2009-01-05")
cut_date <- as.character(cut_date)
for (feat in c(nominal_feats, ordinal_feats)) {
  data_timesplit <- data
  data_timesplit$timesplit <- data_timesplit$date > cut_date
  data_feat <- with(data_timesplit, table(timesplit, data_timesplit[, feat],
                                          dnn = c('timesplit', feat)))
  data_feat <- as.data.frame(data_feat)
  data_feat$NormFreq <- data_feat$Freq /
    sapply(data_feat$timesplit,
           function(x) sum(data_feat[data_feat$timesplit == x, 'Freq']))
  curr_plot <- ggplot(data_feat, aes(x=.data[["timesplit"]],
                                     y=.data[["NormFreq"]],
                                     fill = .data[[feat]])) +
    geom_col(position = 'dodge') +
```

```
    ylab(feat)  +
    settings + scale_x_discrete(labels=c("TRUE" = "after", "FALSE" = "before"))
  list_plots <- outlist <- append(list_plots,list(curr_plot))
}
plot <- plot_grid(plotlist=list_plots, labels = NULL, nrow = 3, ncol = 2)
title <- ggdraw() +
  draw_label(
    "Distribution of categorical variables split by date",
    fontface = 'bold',
    x = 0,
    hjust = -0.4,
    size = 22
  )
plot_grid(
  title, plot,
  ncol = 1,
  rel_heights = c(0.1, 1)
)
```



The distribution built for "Poutcome" confirms the assumption about changing the policy. Indeed, bank started to make an emphasis on the clients who had accepted the proposition during the previous campaign.

2.3 General conclusions

We can point out and gather main conclusions obtained above:

1. After June 2009, the bank policy has changed. It started making less calls and the calls became more aimed on clients which had accepted previous propositions.
2. This solution could be made because of reducing inflation and interest rate (euribor3m). The bank probably expected demand for term deposit to become lower.
3. This is also the reasom why the bank decreased number of employees involved in the campaign.

3 Building classification model

After data analysis and understanding the processes which lay behind the marketing campaign, we are going to fit test several predictive models which was covered during the course:

1. Logistic regression
2. Ridge classifier
3. Lasso classifier
4. Linear discriminant analysis
5. Quadratic discriminant analysis
6. Naive Bayes
7. K-NN

3.1 Initial feature selection

Before applying mentioned above methods, we must remove some features, that can't be used for particular reasons. For example, we must delete "Duration" in order to avoid data leakage. Obviously, there are some useless features like "Default", which is almost constant. Some features can't be used considering validation strategy described further. For example, we can't use "Month" because R doesn't allow as to use unseen during training factor levels.

```
# remove leak feats
data <- subset(data, select = -c(duration))

# remove useless feats
# save default values for feature needs
default <- data$default
default[is.na(default)] <- "no"

# remove useless feats
data <- subset(data, select = -c(default))

# remove feats which can't be used for prediction
data <- subset(data, select = -c(month, date, y_num))

# update groups of feats
continuous_feats <- continuous_feats[! continuous_feats %in% c('duration')]
nominal_feats <- nominal_feats[! nominal_feats %in% c('month')]
bin_feats <- bin_feats[! bin_feats %in% c('default')]
```

3.2 Validation strategy

As we deal with temporal data, we can't use such techniques as random train test split and k-fold cross-validation. Instead, we are going to split the data in the way that the most recent examples go to train set and the last ones go to test set. Also, in case if we have to select models' hyperparameters, we are going to use validation set, which could be obtained by selecting the latest examples from training set.

```
# time series train test split
# leave 25% of examples for a test set
cut_idx = round(nrow(data) * 0.75)
train <- data[1:cut_idx, ]
test <- data[(cut_idx+1):nrow(data), ]

# check target distribution in the datasets
table(train$y) / nrow(train) * 100
```

```
##
##        no       yes
## 93.583892  6.416108
```

```
table(test$y) / nrow(test) * 100
```

```
##
##       no      yes
## 74.18666 25.81334
```

Unfortunately, this splitting strategy doesn't guarantee us preserving the same class proportion in obtained datasets. Moreover, in our case we can see that most of the train samples belong to the period before changing the policy and most of the test samples belong to the period after changing the policy. For this reason, it could be hard to achieve a good quality for test set.

Despite those difficulties, we will anyway act according to chosen validation strategy.

3.3 Filling missing values

As we have seen before, missing values preserves only in categorical features. We are going to fill them with mode values obtained from train dataset.

```
# function for calcilating mode
calc_mode <- function(x){

  vals <- unique(x)
  counts <- tabulate(match(x, vals))
  mode <- vals[which.max(counts)]
  return(mode)
}

# function for filling train and test sets
fill_na_tr_tst <- function(tr, tst, feats) {
  tr_filled <- tr
```

```
  tst_filled <- tst
  for (feat in feats) {
    mode_val <- calc_mode(tr[, feat])
    tr_filled[is.na(tr_filled[, feat]), feat] <- mode_val
    tst_filled[is.na(tst_filled[, feat]), feat] <- mode_val
  }
  return(list(train = tr_filled, test = tst_filled))
}

# fill NA with mode
missing_feats <- c("job", "education", "marital", "housing", "loan")
filled_dfs <-fill_na_tr_tst(train, test, missing_feats)
train <- filled_dfs$train
test <- filled_dfs$test
```

3.4 Removing outliers

Some methods such as LDA and QDA are sensitive to outliers. For this reason, we are going to detect and remove them.

First, we are going to remove those rare clients who has credit in default. Also we have seen that "Campaign" has several large values, for example 50. It's difficult to imagine that it's possible to call the client so many times. We're going to remove those larger values by applying IQR method of outlier detection.

```
# remove outliers from train set
train <- train[default[0:cut_idx] != "yes", ]
max_campaign <- quantile(train$campaign)[4] + 1.5*IQR(train$campaign)
train <- train[train$campaign <= max_campaign, ]

# create a validation set
val_cut_idx = round(nrow(train) * 0.85)
val_train <- train[0:val_cut_idx, ]
val_val <- train[(val_cut_idx+1):nrow(train), ]
```

3.5 Metric choice

As we work with unbalanced dataset, we are going to use area under the ROC Curve as an evaluation metric. In a business scenario, we have a certain set of the potential clients and our task is to search people among given set for participating in the campaign. As the rule, we don't have enough time to call every client. This is the reason why we want to start with people who would probably accept the proposition. Basically, we need to order given set by estimated probability of acceptance and area under the ROC Curve is very suitable metric for that kind of task.

Also we are going to take a look at confusion matrices in order to understand what kind of errors models make.

3.6 Data preprocessing for Ridge classifier, Lasso classifier and K-NN

As Ridge classifier, Lasso classifier and K-NN are not able to handle factors in R, we should transform them into numeric format. The most appropriate approach for those method is One Hot Encoding. Moreover, as

K-NN based on distance computation, it's highly recommended to scale the features. For this purpose, we will use Min Max Scaling like this

$$Xscaled = \frac{X - min(x)}{max(X) - min(X)}$$

```
# additional data preprocessing for Ridge, Lasso, KNN
# create dummies
data_for_dummies <- rbind(train, test)
data_dummies <- dummy_cols(data_for_dummies,
                           select_columns=c(bin_feats, ordinal_feats,
                                            nominal_feats),
                           remove_selected_columns=TRUE)
# replace "-" with "_" for avoiding further errors
colnames(data_dummies)[colnames(data_dummies) ==
                         "job_blue-collar"] <- "job_blue_collar"
colnames(data_dummies)[colnames(data_dummies) ==
                         "job_self-employed"] <- "job_self_employed"
# again, train test split
train_dummies <- data_dummies[1:nrow(train), ]
test_dummies <- data_dummies[(nrow(train_dummies)+1):nrow(data_dummies), ]

# min max scaling
mins <- sapply(subset(train_dummies, select = -c(y)), min)
maxs <- sapply(subset(train_dummies, select = -c(y)), max)
maxs_m_mins <- maxs - mins

# function for min max scaling
min_max_scaling <- function(df, mins, maxs_m_mins) {
  df_sc <- sweep(subset(df, select = -c(y)),
                 2,
                 mins)
  df_sc <- sweep(df_sc,
                 2,
                 maxs_m_mins, FUN='/')
  df_sc$y <- df$y
  return(df_sc)
}

train_dummies_sc <- min_max_scaling(train_dummies, mins, maxs_m_mins)
test_dummies_sc <- min_max_scaling(test_dummies, mins, maxs_m_mins)

# leave some samples for validation set
val_train_dummies_sc <- train_dummies_sc[0:val_cut_idx, ]
val_val_dummies_sc <- train_dummies_sc[(val_cut_idx+1):nrow(train_dummies), ]
```

3.7 Modeling

3.7.1 Logistic regression. We start with simple logistic regression. Here we are not able to adjust any hyperparameters, hence we use the whole train set for fitting the model. For this step, we will use all the

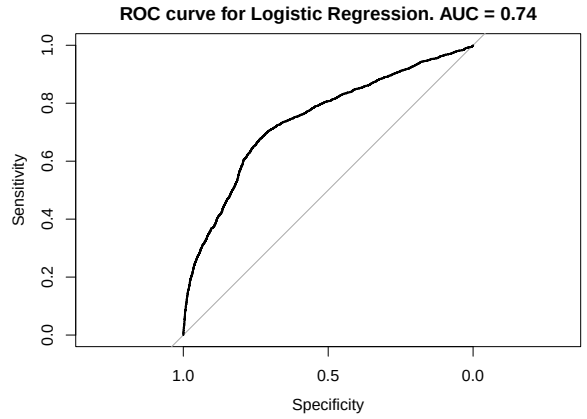
features in order to estimate later multicollinearity by using Variance Inflation Factor (VIF).

```
# Logistic regression
# set default threshold for computing binary predictions
threshold <- 0.5
# modeling
model_lr <- glm(y ~., data=train, family = binomial)
summary(model_lr)
```

```
##
## Call:
## glm(formula = y ~ ., family = binomial, data = train)
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -1.781e+04  1.183e+03 -15.059  < 2e-16 ***
## age              -2.607e-03  3.076e-03  -0.848 0.396660
## jobblue-collar   -4.455e-02  9.010e-02  -0.494 0.620976
## jobentrepreneur  -1.442e-03  1.348e-01  -0.011 0.991463
## jobhousemaid     -3.208e-02  1.810e-01  -0.177 0.859340
## jobmanagement    -5.429e-02  1.041e-01  -0.522 0.601922
## jobretired        5.753e-01  1.378e-01   4.173 3.00e-05 ***
## jobself-employed  3.636e-02  1.348e-01   0.270 0.787352
## jobservices      -1.008e-01  1.014e-01  -0.995 0.319917
## jobstudent        5.122e-01  1.715e-01   2.986 0.002827 **
## jobtechnician    -4.170e-02  8.649e-02  -0.482 0.629701
## jobunemployed    -2.781e-02  1.747e-01  -0.159 0.873499
## maritaldivorced  -1.186e-01  9.284e-02  -1.277 0.201538
## maritalmarried   -1.163e-01  6.250e-02  -1.861 0.062709 .
## education.L      -3.897e-01  4.647e-01  -0.839 0.401693
## education.Q       6.064e-01  4.459e-01   1.360 0.173890
## education.C      -3.754e-01  3.384e-01  -1.109 0.267318
## education^4       3.233e-01  2.103e-01   1.537 0.124198
## education^5      -1.733e-01  1.216e-01  -1.425 0.154129
## education^6       1.735e-01  8.344e-02   2.079 0.037582 *
## housingyes       -2.156e-02  4.999e-02  -0.431 0.666201
## loanyes           3.313e-02  6.828e-02   0.485 0.627512
## contacttelephone -3.554e-01  1.048e-01  -3.393 0.000692 ***
## day_of_weektue    1.790e-01  8.097e-02   2.211 0.027035 *
## day_of_weekwed    4.202e-01  7.931e-02   5.298 1.17e-07 ***
## day_of_weekthu    3.965e-01  7.598e-02   5.218 1.81e-07 ***
## day_of_weekfri    1.618e-01  8.265e-02   1.957 0.050309 .
## campaign          7.951e-03  1.958e-02   0.406 0.684747
## pdays            -2.038e-03  1.057e-03  -1.928 0.053889 .
## previous          1.123e-01  3.319e-01   0.338 0.735212
## poutcomefailure  -8.489e-01  3.728e-01  -2.277 0.022771 *
## poutcomesuccess  -7.196e-01  9.965e-01  -0.722 0.470259
## emp.var.rate     -2.526e+00  2.086e-01 -12.107  < 2e-16 ***
## cons.price.idx    1.818e-01  1.797e-01   1.012 0.311483
## cons.conf.idx    -2.785e-02  1.695e-02  -1.643 0.100445
## euribor3m         3.824e+00  2.571e-01  14.873  < 2e-16 ***
## nr.employed       1.535e-02  3.875e-03   3.960 7.49e-05 ***
## year              8.814e+00  5.827e-01  15.125  < 2e-16 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 13974  on 28784  degrees of freedom
## Residual deviance: 12689  on 28747  degrees of freedom
## AIC: 12765
##
## Number of Fisher Scoring iterations: 6
```

```
# prediction
probabilities_lr <- predict(model_lr, test, type="response")
preds_lr <- ifelse(probabilities_lr > threshold, "yes", "no")
preds_lr <- factor(preds_lr, levels=c("no", "yes"))

# evaluation
roc_score_lr <- roc(test$y, probabilities_lr)
plot(roc_score_lr ,main =sprintf("ROC curve for Logistic Regression. AUC = %s",
                                 round(roc_score_lr$auc, 2)))
```




```
confusionMatrix(data=preds_lr, reference = test$y)
```

```
## Confusion Matrix and Statistics
##
##          Reference
## Prediction   no  yes
##        no  6277 1283
##        yes 1362 1375
##
##                Accuracy : 0.7431
##                  95% CI : (0.7346, 0.7515)
##     No Information Rate : 0.7419
##     P-Value [Acc > NIR] : 0.3898
##
##                   Kappa : 0.3358
##
##  Mcnemar's Test P-Value : 0.1294
##
##             Sensitivity : 0.8217
##             Specificity : 0.5173
```

```
##          Pos Pred Value : 0.8303
##          Neg Pred Value : 0.5024
##              Prevalence : 0.7419
##          Detection Rate : 0.6096
##    Detection Prevalence : 0.7342
##       Balanced Accuracy : 0.6695
##
##        'Positive' Class : no
##
```

Considering the fact that train and test client are treated according to different policies, we've obtained a good quality for the test set. Also we can point several significant for a model features: "job" (is the client student or retired), "day_of_week", " emp.var.rate", "euribor3m", "nr.employed" and "year". However, we haven't eliminated multicollinearity, and that's the reason why we could obtain unstable estimates of coefficients.

For this reason, we're going to use VIF in order to explore the degree of multicollinearity. Due to presence of categorical features, we are going to use a modification of VIF, which can be computed as $GVIF^{\frac{1}{2Df}}$, where $GVIF = \sqrt{VIF}$, Df is a number of levels for categorical feature (it equals one in case of continuous features).

The rule of thumb here is to achieve at most $\sqrt{10}$ for each predictor.

```
# VIF
VIF(model_lr)
```

```
##                      GVIF Df GVIF^(1/(2*Df))
## age              1.620182  1        1.272864
## job              3.929706 10        1.070824
## marital          1.333637  2        1.074631
## education        2.819934  6        1.090234
## housing          1.014403  1        1.007176
## loan             1.004277  1        1.002136
## contact          3.783518  1        1.945127
## day_of_week      1.080587  4        1.009735
## campaign         1.044432  1        1.021975
## pdays           31.860974  1        5.644553
## previous        15.276880  1        3.908565
## poutcome       318.149359  2        4.223357
## emp.var.rate   137.006164  1       11.704963
## cons.price.idx  12.460425  1        3.529933
## cons.conf.idx    8.467515  1        2.909899
## euribor3m      259.083888  1       16.096083
## nr.employed     73.110873  1        8.550490
## year           113.815345  1       10.668427
```

As we may see, the features from 2 groups of correlated predictors mentioned before have large values of modified GVIF:

1. The first group: "year", emp.var.rate", "cons.price.idx", "cons.conf.idx", "euribor3m", "nr.employed"
2. The second group: "pdays" and "previous"

First, we tried to remove those features one by one sequentially until we got small values of modified GVIF.

However if we remove one of the features from the first group, the test metric dramatically reduces its value. This could lead to a conclusion that, despite of the correlation of those features, we might loose an important information by removing those features. As a solution, we can use Principal Component Analysis (PCA). By applying this method, we can substitute those features by their linear combination, and new features won't be correlated. This will help us to reduce multicollinearity without decreasing of model quality. However, it will obscure the interpretation of a model since we can't interpret new features.

```
# PCA + removing pdays and previous
first_g_correlated_feats <- c("year", "nr.employed", "euribor3m",
                              "emp.var.rate", "cons.price.idx")
pca <- PCA(train[,first_g_correlated_feats], graph = FALSE)
pca_train <- subset(train, select = -c(year, nr.employed, euribor3m,
                                       emp.var.rate, cons.price.idx))
pca_test <- subset(test, select = -c(year, nr.employed, euribor3m,
                                     emp.var.rate, cons.price.idx))
pca_comps_train <- predict(pca, train[,first_g_correlated_feats])$coord
pca_comps_test <- predict(pca, test[,first_g_correlated_feats])$coord
pca_train <- cbind(pca_train, pca_comps_train)
pca_test <- cbind(pca_test, pca_comps_test)

model_pca_lr <- glm(y ~.-pdays-previous, data=pca_train, family = binomial)
# now we have reduced multicollinearity
VIF(model_pca_lr)
```

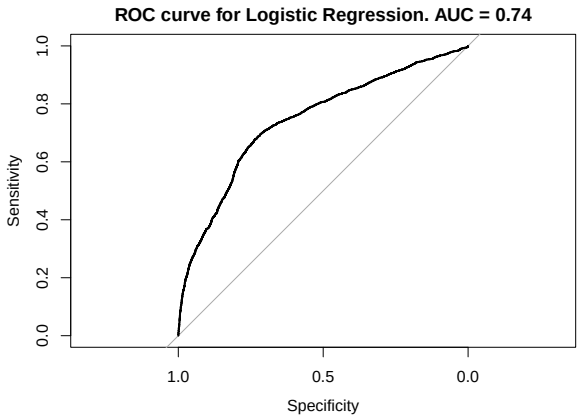
```
##                   GVIF Df GVIF^(1/(2*Df))
## age           1.620953  1        1.273167
## job           3.916529 10        1.070644
## marital       1.333879  2        1.074680
## education     2.817070  6        1.090142
## housing       1.014116  1        1.007033
## loan          1.004068  1        1.002032
## contact       3.781490  1        1.944605
## day_of_week   1.080187  4        1.009688
## campaign      1.044267  1        1.021894
## poutcome      1.163588  2        1.038603
## cons.conf.idx 8.469594  1        2.910257
## Dim.1         5.286495  1        2.299238
## Dim.2         3.447312  1        1.856694
## Dim.3         2.194231  1        1.481294
## Dim.4         2.546159  1        1.595669
## Dim.5         1.200994  1        1.095899
```

```
summary(model_pca_lr)
```

```
##
## Call:
## glm(formula = y ~ . - pdays - previous, family = binomial, data = pca_train)
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -3.766488   0.733417  -5.136 2.81e-07 ***
## age              -0.002578   0.003075  -0.838 0.401908
## jobblue-collar   -0.043373   0.090072  -0.482 0.630138
## jobentrepreneur  -0.002522   0.134814  -0.019 0.985073
```

```
## jobhousemaid     -0.033125   0.181020  -0.183 0.854805
## jobmanagement    -0.055186   0.104073  -0.530 0.595929
## jobretired        0.577782   0.137753   4.194 2.74e-05 ***
## jobself-employed  0.034844   0.134804   0.258 0.796039
## jobservices      -0.101221   0.101363  -0.999 0.317985
## jobstudent        0.533634   0.170424   3.131 0.001741 **
## jobtechnician    -0.044145   0.086497  -0.510 0.609792
## jobunemployed    -0.029796   0.174674  -0.171 0.864552
## maritaldivorced  -0.117363   0.092790  -1.265 0.205935
## maritalmarried   -0.116175   0.062490  -1.859 0.063014 .
## education.L      -0.387681   0.464694  -0.834 0.404128
## education.Q       0.606980   0.445884   1.361 0.173420
## education.C      -0.376716   0.338394  -1.113 0.265603
## education^4       0.319280   0.210235   1.519 0.128842
## education^5      -0.174257   0.121621  -1.433 0.151919
## education^6       0.173384   0.083423   2.078 0.037674 *
## housingyes       -0.020680   0.049968  -0.414 0.678970
## loanyes           0.031529   0.068276   0.462 0.644229
## contacttelephone -0.354883   0.104718  -3.389 0.000702 ***
## day_of_weektue    0.180602   0.080952   2.231 0.025682 *
## day_of_weekwed    0.420806   0.079305   5.306 1.12e-07 ***
## day_of_weekthu    0.398612   0.075966   5.247 1.54e-07 ***
## day_of_weekfri    0.163225   0.082617   1.976 0.048190 *
## campaign          0.008714   0.019565   0.445 0.656059
## poutcomefailure  -0.705304   0.115082  -6.129 8.86e-10 ***
## poutcomesuccess   1.433906   0.194596   7.369 1.72e-13 ***
## cons.conf.idx    -0.028007   0.016945  -1.653 0.098370 .
## Dim.1            -0.181167   0.019573  -9.256  < 2e-16 ***
## Dim.2             0.112858   0.065229   1.730 0.083596 .
## Dim.3             0.379173   0.119276   3.179 0.001478 **
## Dim.4            -0.104844   0.180678  -0.580 0.561723
## Dim.5             5.646514   0.367787  15.353  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 13974  on 28784  degrees of freedom
## Residual deviance: 12694  on 28749  degrees of freedom
## AIC: 12766
##
## Number of Fisher Scoring iterations: 6
```

```r
# prediction
probabilities_pca_lr <- predict(model_pca_lr, pca_test, type="response")
preds_pca_lr <- ifelse(probabilities_pca_lr > threshold, "yes", "no")
preds_pca_lr <- as.factor(preds_pca_lr)

# evaluation
roc_score_pca_lr <- roc(test$y, probabilities_lr)
plot(roc_score_pca_lr ,main =sprintf("ROC curve for Logistic Regression. AUC = %s",
                             round(roc_score_pca_lr$auc, 2)))
```




```
confusionMatrix(data=preds_pca_lr, reference = test$y)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   no  yes
##        no  6289 1300
##        yes 1350 1358
##
##                Accuracy : 0.7426
##                  95% CI : (0.7341, 0.7511)
##     No Information Rate : 0.7419
##     P-Value [Acc > NIR] : 0.4336
##
##                   Kappa : 0.3321
##
##  Mcnemar's Test P-Value : 0.3412
##
##             Sensitivity : 0.8233
##             Specificity : 0.5109
##          Pos Pred Value : 0.8287
##          Neg Pred Value : 0.5015
##              Prevalence : 0.7419
##          Detection Rate : 0.6108
##    Detection Prevalence : 0.7370
##       Balanced Accuracy : 0.6671
##
##        'Positive' Class : no
##
```

Now we can see that we've eliminated multicollinearity without any loss of quality. The significant features are "job" (is the client student or retired), "day_of_week", "poutcome".

Since the dataset is unbalanced, we may try to apply oversampling.

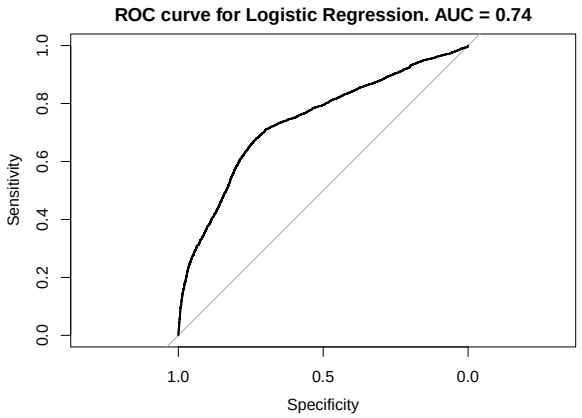
```
# oversampling
pca_over_train <- ovun.sample(y~., data = pca_train, method = "over")$data
```

```r
# modeling
model_pca_over_lr <- glm(y ~.-pdays-previous, data=pca_over_train,
                         family = binomial)
summary(model_pca_over_lr)
```

```
##
## Call:
## glm(formula = y ~ . - pdays - previous, family = binomial, data = pca_over_train)
##
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -0.461876   0.301881  -1.530 0.126019
## age              -0.004130   0.001157  -3.571 0.000356 ***
## jobblue-collar   -0.011431   0.033309  -0.343 0.731469
## jobentrepreneur   0.113548   0.050309   2.257 0.024007 *
## jobhousemaid     -0.124555   0.064911  -1.919 0.055002 .
## jobmanagement     0.013268   0.039239   0.338 0.735271
## jobretired        0.575149   0.056355  10.206  < 2e-16 ***
## jobself-employed  0.005644   0.051585   0.109 0.912873
## jobservices       0.022895   0.037099   0.617 0.537142
## jobstudent        0.616249   0.082246   7.493 6.75e-14 ***
## jobtechnician    -0.033286   0.032225  -1.033 0.301639
## jobunemployed    -0.054076   0.064617  -0.837 0.402661
## maritaldivorced  -0.061861   0.034479  -1.794 0.072788 .
## maritalmarried   -0.106025   0.023895  -4.437 9.12e-06 ***
## education.L      -0.322623   0.212159  -1.521 0.128344
## education.Q       0.429737   0.204157   2.105 0.035297 *
## education.C      -0.254883   0.154222  -1.653 0.098391 .
## education^4       0.295965   0.094106   3.145 0.001661 **
## education^5      -0.202271   0.050816  -3.980 6.88e-05 ***
## education^6       0.139927   0.031152   4.492 7.07e-06 ***
## housingyes       -0.054446   0.018723  -2.908 0.003637 **
## loanyes           0.061461   0.025602   2.401 0.016367 *
## contacttelephone -0.440328   0.042119 -10.454  < 2e-16 ***
## day_of_weektue    0.089525   0.029704   3.014 0.002580 **
## day_of_weekwed    0.305263   0.029422  10.375  < 2e-16 ***
## day_of_weekthu    0.283233   0.028753   9.850  < 2e-16 ***
## day_of_weekfri    0.210051   0.030261   6.941 3.88e-12 ***
## campaign          0.018464   0.007297   2.530 0.011390 *
## poutcomefailure  -0.679857   0.045720 -14.870  < 2e-16 ***
## poutcomesuccess   1.392725   0.124869  11.154  < 2e-16 ***
## cons.conf.idx    -0.014525   0.006930  -2.096 0.036076 *
## Dim.1            -0.181255   0.007954 -22.788  < 2e-16 ***
## Dim.2             0.142936   0.025975   5.503 3.74e-08 ***
## Dim.3             0.412525   0.041554   9.928  < 2e-16 ***
## Dim.4            -0.196112   0.070113  -2.797 0.005157 **
## Dim.5             4.746574   0.179766  26.404  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 74729  on 53905  degrees of freedom
## Residual deviance: 67219  on 53870  degrees of freedom
```

```
## AIC: 67291
##
## Number of Fisher Scoring iterations: 4
```

```
# prediction
probabilities_pca_over_lr <- predict(model_pca_over_lr, pca_test, type="response")
preds_pca_over_lr <- ifelse(probabilities_pca_over_lr > threshold, "yes", "no")
preds_pca_over_lr <- factor(preds_pca_over_lr, levels=c("no", "yes"))

# evaluation
roc_score_pca_over_lr <- roc(test$y, probabilities_pca_over_lr)
plot(roc_score_pca_over_lr ,main =sprintf("ROC curve for Logistic Regression. AUC = %s",
                              round(roc_score_pca_over_lr$auc, 2)))
```




```
confusionMatrix(data=preds_pca_over_lr, reference = test$y)
```

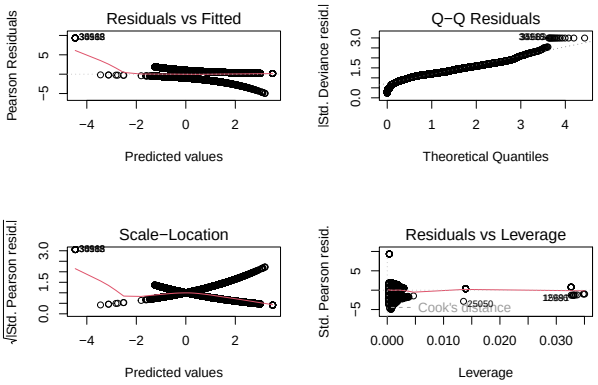
```
## Confusion Matrix and Statistics
##
##          Reference
## Prediction   no  yes
##        no   466   70
##        yes 7173 2588
##
##               Accuracy : 0.2966
##                 95% CI : (0.2878, 0.3055)
##    No Information Rate : 0.7419
##    P-Value [Acc > NIR] : 1
##
##                  Kappa : 0.0185
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.06100
##             Specificity : 0.97366
##          Pos Pred Value : 0.86940
##          Neg Pred Value : 0.26514
##              Prevalence : 0.74187
##          Detection Rate : 0.04526
##    Detection Prevalence : 0.05205
##       Balanced Accuracy : 0.51733
##
```

```
##         'Positive' Class : no
##
```

As result, the metric doesn't change its value, however the model started to be too confident in terms of giving positive labels to test samples.

After resampling we can exploit several plots for error analysis.

```
# model diagnostics
par(mfrow=c(2,2))
plot(model_pca_over_lr)
```




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

By looking at the plots we can make following conclusions:

1. In general, there is no linear trend between predicted values and residuals, however there is resampled point which has a huge pearson residual and it affect the absence of the trend
2. The residuals are distributed almost normaly
3. The point mentioned before creates heteroscedasticity. Also we can see increased standard deviation of errors around zero point
4. There are no leverage points which are influential in our model

Finally, we will try to perform feature selection in order to get rid of useless information. We are going to use Backward and Forward Stepwise Selection. Both of them use Akaike information criterion for selecting features.

```
# Backward Stepwise Selection
model_pca_lr <- glm(y ~.-pdays-previous, data=pca_train, family = binomial)
b_step <- step(model_pca_lr, direction= "backward",
               scope=formula(model_pca_lr), trace=0)
# Removed features
b_step$anova
```

```
##         Step Df  Deviance Resid. Df Resid. Dev      AIC
## 1            NA        NA     28749   12693.60 12765.60
## 2  - housing  1 0.1712043     28750   12693.77 12763.77
## 3 - campaign  1 0.1994780     28751   12693.97 12761.97
```

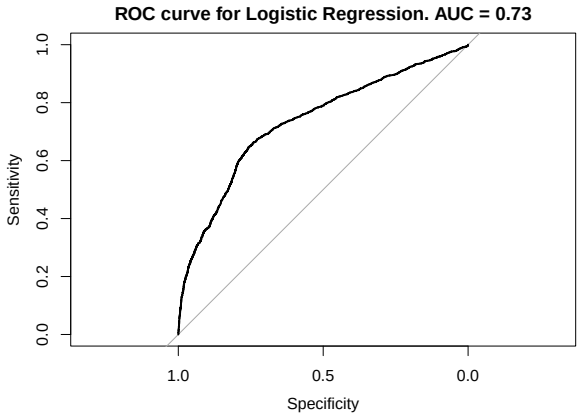
```
## 4     - loan  1 0.2053627     28752   12694.18 12760.18
## 5    - Dim.4  1 0.3064210     28753   12694.48 12758.48
## 6      - age  1 0.6688680     28754   12695.15 12757.15
```

```
model_pca_fsel_lr <- glm(y ~.-pdays-previous-housing-campaign-loan-Dim.4-age,
                         data=pca_train, family = binomial)
summary(model_pca_fsel_lr)
```

```
##
## Call:
## glm(formula = y ~ . - pdays - previous - housing - campaign -
##     loan - Dim.4 - age, family = binomial, data = pca_train)
##
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -4.159905   0.504105  -8.252  < 2e-16 ***
## jobblue-collar   -0.043176   0.089947  -0.480 0.631217
## jobentrepreneur  -0.006368   0.134616  -0.047 0.962271
## jobhousemaid     -0.041882   0.180809  -0.232 0.816821
## jobmanagement    -0.061627   0.103868  -0.593 0.552965
## jobretired        0.528303   0.126790   4.167 3.09e-05 ***
## jobself-employed  0.032876   0.134738   0.244 0.807231
## jobservices      -0.100826   0.101295  -0.995 0.319555
## jobstudent        0.553969   0.168591   3.286 0.001017 **
## jobtechnician    -0.043838   0.086467  -0.507 0.612161
## jobunemployed    -0.032365   0.174653  -0.185 0.852984
## maritaldivorced  -0.139084   0.088445  -1.573 0.115822
## maritalmarried   -0.131457   0.058789  -2.236 0.025347 *
## education.L      -0.372060   0.464278  -0.801 0.422916
## education.Q       0.598875   0.445632   1.344 0.178988
## education.C      -0.375810   0.338305  -1.111 0.266629
## education^4       0.321789   0.210154   1.531 0.125719
## education^5      -0.174776   0.121603  -1.437 0.150643
## education^6       0.174976   0.083416   2.098 0.035936 *
## contacttelephone -0.344643   0.103254  -3.338 0.000844 ***
## day_of_weektue    0.176740   0.080643   2.192 0.028405 *
## day_of_weekwed    0.419414   0.078994   5.309 1.10e-07 ***
## day_of_weekthu    0.396209   0.075497   5.248 1.54e-07 ***
## day_of_weekfri    0.163322   0.082423   1.982 0.047534 *
## poutcomefailure  -0.701086   0.114672  -6.114 9.73e-10 ***
## poutcomesuccess   1.432805   0.194372   7.371 1.69e-13 ***
## cons.conf.idx    -0.035715   0.011371  -3.141 0.001684 **
## Dim.1            -0.172337   0.014396 -11.971  < 2e-16 ***
## Dim.2             0.103227   0.062437   1.653 0.098269 .
## Dim.3             0.352684   0.107659   3.276 0.001053 **
## Dim.5             5.603385   0.361968  15.480  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 13974  on 28784  degrees of freedom
## Residual deviance: 12695  on 28754  degrees of freedom
## AIC: 12757
##
```

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

```
# prediction
probabilities_pca_fsel_lr <- predict(model_pca_fsel_lr, pca_test,
                                     type="response")
preds_pca_fsel_lr <- ifelse(probabilities_pca_fsel_lr > threshold, "yes", "no")
preds_pca_fsel_lr <- factor(preds_pca_fsel_lr, levels=c("no", "yes"))

# evaluation
roc_score_pca_fsel_lr <- roc(test$y, probabilities_pca_fsel_lr)
plot(roc_score_pca_fsel_lr ,main =sprintf("ROC curve for Logistic Regression. AUC = %s",
                               round(roc_score_pca_fsel_lr$auc, 2)))
```




```
confusionMatrix(data=preds_pca_fsel_lr, reference = test$y)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   no  yes
##        no  6336 1340
##        yes 1303 1318
##
##                Accuracy : 0.7433
##                  95% CI : (0.7348, 0.7517)
##     No Information Rate : 0.7419
##     P-Value [Acc > NIR] : 0.3726
##
##                   Kappa : 0.3268
##
##  Mcnemar's Test P-Value : 0.4838
##
##             Sensitivity : 0.8294
##             Specificity : 0.4959
##          Pos Pred Value : 0.8254
##          Neg Pred Value : 0.5029
##              Prevalence : 0.7419
##          Detection Rate : 0.6153
##    Detection Prevalence : 0.7455
##       Balanced Accuracy : 0.6626
##
##        'Positive' Class : no
```

##

Test metric has decreased a little. However, now it's easier to interpret the model.

```
# Forward Stepwise Selection
model_intercept <- glm(y ~ 1, data=pca_train, family = binomial)
f_step <- step(model_intercept, direction= "forward",
               scope=formula(model_pca_lr), trace=0)
# Removed features
f_step$anova
```

```
##               Step  Df   Deviance Resid. Df Resid. Dev      AIC
## 1                   NA         NA     28784   13973.62 13975.62
## 2          + Dim.1  -1 729.528387     28783   13244.10 13248.10
## 3          + Dim.5  -1 215.871287     28782   13028.22 13034.22
## 4       + poutcome  -2 109.132150     28780   12919.09 12929.09
## 5          + Dim.3  -1  79.410310     28779   12839.68 12851.68
## 6    + day_of_week  -4  47.504838     28775   12792.18 12812.18
## 7            + job -10  40.095373     28765   12752.08 12792.08
## 8  + cons.conf.idx  -1  21.193032     28764   12730.89 12772.89
## 9        + contact  -1  12.136198     28763   12718.75 12762.75
## 10     + education  -6  15.444922     28757   12703.31 12759.31
## 11       + marital  -2   5.368495     28755   12697.94 12757.94
## 12         + Dim.2  -1   2.787090     28754   12695.15 12757.15
```

```
# The same results
```

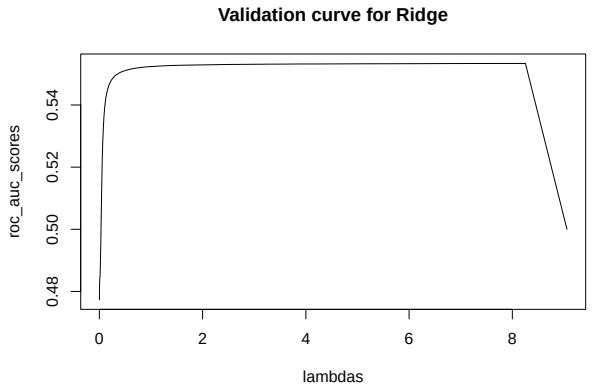
After Forward Stepwise Selection we obtain the same feature set as before.

3.7.2 Ridge classifier. Next we are going to apply Ridge classifier. The regularization allows us to deal directly with datasets which have a problem related to multicollinearity. Also it helps to prevent overfitting.

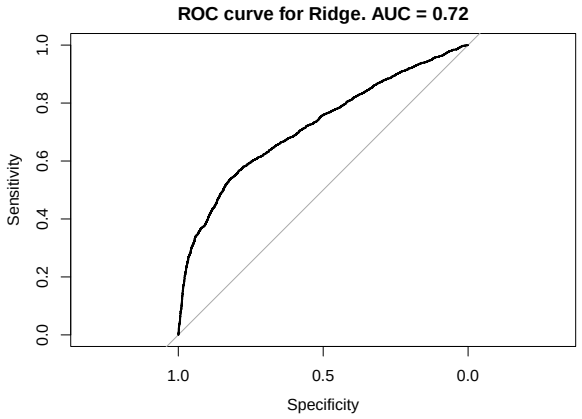
Here we will use validation set for selecting regularization parameter.

```
# Ridge
# convert the data to required format
X_val_train_mat <- as.matrix(subset(val_train_dummies_sc, select = -c(y)))
y_val_train_mat <- as.matrix(as.numeric(val_train_dummies_sc$y)-1)
X_val_val_mat <- as.matrix(subset(val_val_dummies_sc, select = -c(y)))
y_val_val_mat <- as.matrix(as.numeric(val_val_dummies_sc$y)-1)
X_train_mat <- as.matrix(subset(train_dummies_sc, select = -c(y)))
y_train_mat <- as.matrix(as.numeric(train_dummies_sc$y)-1)
X_test_mat <- as.matrix(subset(test_dummies_sc, select = -c(y)))
y_test_mat <- as.matrix(as.numeric(test_dummies_sc$y)-1)

# selecting lambda
model_ridge <- glmnet(x=X_val_train_mat, y=y_val_train_mat,
                      family = binomial, alpha = 0)
probabilities_ridge <- predict(model_ridge, X_val_val_mat, type="response")
roc_auc_scores <- sapply(as.data.frame(probabilities_ridge),
```

```
                          function(x) roc(val_val_dummies_sc$y, x)$auc)
lambdas <- model_ridge$lambda
plot(lambdas, roc_auc_scores, type="l", main ="Validation curve for Ridge")
```




```
lambda = lambdas[roc_auc_scores == max(roc_auc_scores)][1]

# modeling
model_ridge <- glmnet(x=X_train_mat, y=y_train_mat,
                      family = binomial, alpha = 0, lambda = lambda)

# prediction
probabilities_ridge <- predict(model_ridge, X_test_mat,
                               type="response")
probabilities_ridge <- as.vector(probabilities_ridge)
preds_ridge <- ifelse(probabilities_ridge > threshold, "yes", "no")
preds_ridge <- as.factor(preds_ridge)

# evaluation
roc_score_ridge <- roc(test_dummies_sc$y, probabilities_ridge)
plot(roc_score_ridge ,main =sprintf("ROC curve for Ridge. AUC = %s",
                                round(roc_score_ridge$auc, 2)))
```




```
confusionMatrix(data=preds_ridge, reference = test$y)
```

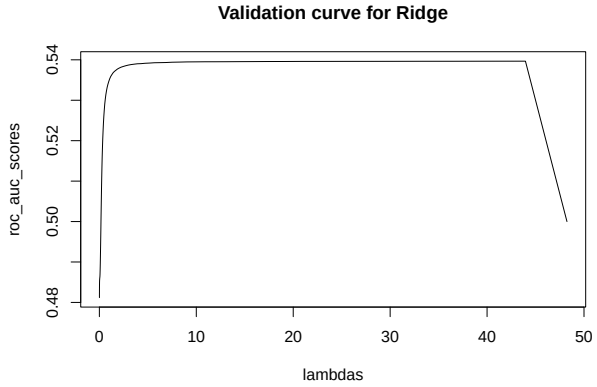
```
## Warning in confusionMatrix.default(data = preds_ridge, reference = test$y):
## Levels are not in the same order for reference and data. Refactoring data to
```

```
## match.
```

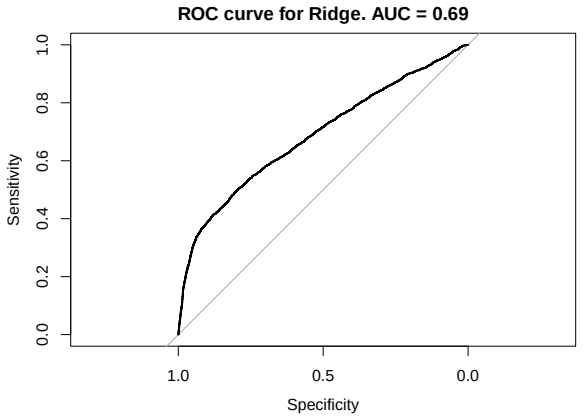
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   no  yes
##        no  7639 2658
##        yes    0    0
##
##                Accuracy : 0.7419
##                  95% CI : (0.7333, 0.7503)
##     No Information Rate : 0.7419
##     P-Value [Acc > NIR] : 0.5052
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 1.0000
##             Specificity : 0.0000
##          Pos Pred Value : 0.7419
##          Neg Pred Value :    NaN
##              Prevalence : 0.7419
##          Detection Rate : 0.7419
##    Detection Prevalence : 1.0000
##       Balanced Accuracy : 0.5000
##
##        'Positive' Class : no
##
```

We can see that model doesn't give any sample a positive class. We may try to use oversampling.

```
# oversampling
val_train_dummies_sc_over <- ovun.sample(y~., data = val_train_dummies_sc,
                                         method = "over")$data
train_dummies_sc_over <- ovun.sample(y~., data = train_dummies_sc,
                                         method = "over")$data

X_val_train_over_mat <- as.matrix(subset(val_train_dummies_sc_over,
                                         select = -c(y)))
y_val_train_over_mat <- as.matrix(as.numeric(val_train_dummies_sc_over$y)-1)
X_train_over_mat <- as.matrix(subset(train_dummies_sc_over, select = -c(y)))
y_train_over_mat <- as.matrix(as.numeric(train_dummies_sc_over$y)-1)

# selecting lambda
model_ridge_over <- glmnet(x=X_val_train_over_mat, y=y_val_train_over_mat,
                      family = binomial, alpha = 0)
probabilities_ridge_over <- predict(model_ridge_over, X_val_val_mat,
                                    type="response")
roc_auc_scores <- sapply(as.data.frame(probabilities_ridge_over),
                         function(x) roc(val_val_dummies_sc$y, x)$auc)
lambdas <- model_ridge_over$lambda
plot(lambdas, roc_auc_scores, type="l", main ="Validation curve for Ridge")
```




```
lambda = lambdas[roc_auc_scores == max(roc_auc_scores)][1]

# modeling
model_ridge_over <- glmnet(x=X_train_over_mat, y=y_train_over_mat,
                       family = binomial, alpha = 0, lambda = lambda)

# prediction
probabilities_ridge_over <- predict(model_ridge_over, X_test_mat,
                                    type="response")
probabilities_ridge_over <- as.vector(probabilities_ridge_over)
preds_ridge_over <- ifelse(probabilities_ridge_over > threshold, "yes", "no")
preds_ridge_over <- as.factor(preds_ridge_over)

# evaluation
roc_score_ridge_over <- roc(test$y, probabilities_ridge_over)
plot(roc_score_ridge_over ,main =sprintf("ROC curve for Ridge. AUC = %s",
                              round(roc_score_ridge_over$auc, 2)))
```




```
confusionMatrix(data=preds_ridge_over, reference = test$y)
```

```
## Warning in confusionMatrix.default(data = preds_ridge_over, reference =
## test$y): Levels are not in the same order for reference and data. Refactoring
## data to match.
```

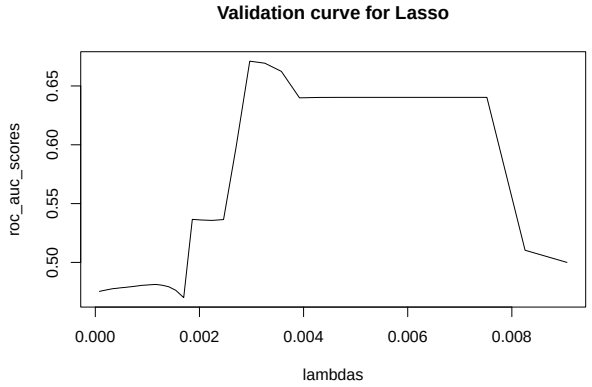
```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction   no  yes
##        no     0    0
##        yes 7639 2658
##
##                Accuracy : 0.2581
##                  95% CI : (0.2497, 0.2667)
##     No Information Rate : 0.7419
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.0000
##             Specificity : 1.0000
##          Pos Pred Value :    NaN
##          Neg Pred Value : 0.2581
##              Prevalence : 0.7419
##          Detection Rate : 0.0000
##    Detection Prevalence : 0.0000
##       Balanced Accuracy : 0.5000
##
##        'Positive' Class : no
##
```

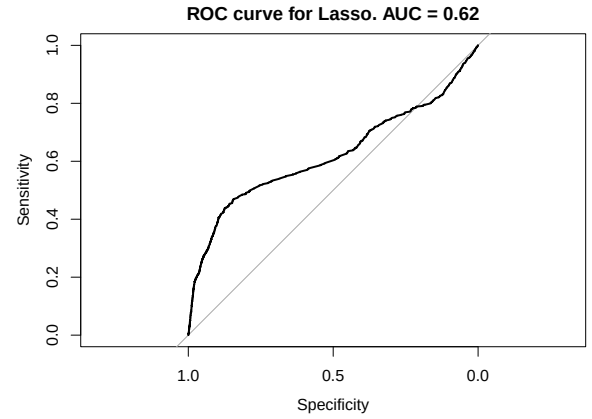
Unfortunately, oversampling made the quality worse and after that model gives only positive labels to test samples.

3.7.3 Lasso classifier. Now we are going to apply Lasso classifier. We will perform the same steps as for Ridge.

```
# Lasso
# selecting lambda
model_lasso <- glmnet(x=X_val_train_mat, y=y_val_train_mat,
                      family = binomial, alpha = 1)
probabilities_lasso <- predict(model_lasso, X_val_val_mat, type="response")
roc_auc_scores <- sapply(as.data.frame(probabilities_lasso),
                         function(x) roc(val_val_dummies_sc$y, x)$auc)
lambdas <- model_lasso$lambda
plot(lambdas, roc_auc_scores, type="l", main ="Validation curve for Lasso")
```




```
lambda = lambdas[roc_auc_scores == max(roc_auc_scores)][1]

# modeling
model_lasso <- glmnet(x=X_train_mat, y=y_train_mat,
                      family = binomial, alpha = 1, lambda = lambda)

# prediction
probabilities_lasso <- predict(model_lasso, X_test_mat,
                               type="response")
probabilities_lasso <- as.vector(probabilities_lasso)
preds_lasso <- ifelse(probabilities_lasso > threshold, "yes", "no")
preds_lasso <- as.factor(preds_lasso)

# evaluation
roc_score_lasso <- roc(test$y, probabilities_lasso)
plot(roc_score_lasso ,main =sprintf("ROC curve for Lasso. AUC = %s",
                            round(roc_score_lasso$auc, 2)))
```



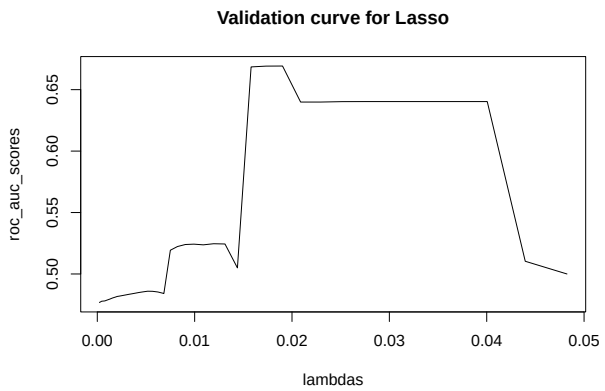

```
confusionMatrix(data=preds_lasso, reference = test$y)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   no  yes
##        no  7523 2300
##        yes  116  358
```

```
##
##                Accuracy : 0.7654
##                  95% CI : (0.7571, 0.7735)
##     No Information Rate : 0.7419
##     P-Value [Acc > NIR] : 1.975e-08
##
##                   Kappa : 0.1632
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9848
##             Specificity : 0.1347
##          Pos Pred Value : 0.7659
##          Neg Pred Value : 0.7553
##              Prevalence : 0.7419
##          Detection Rate : 0.7306
##    Detection Prevalence : 0.9540
##       Balanced Accuracy : 0.5598
##
##        'Positive' Class : no
##
```

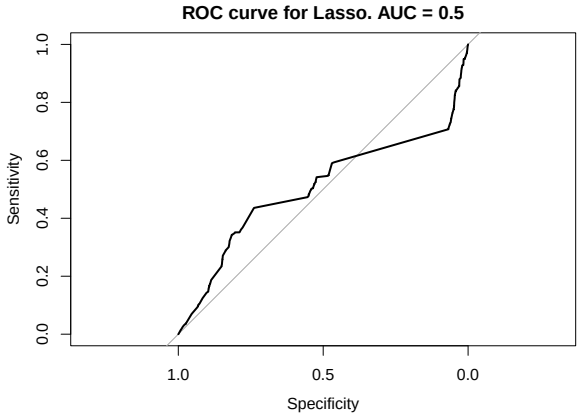
Again, we try oversampling.

```
# oversampling
# selecting lambda
model_lasso_over <- glmnet(x=X_val_train_over_mat, y=y_val_train_over_mat,
                           family = binomial, alpha = 1)
probabilities_lasso_over <- predict(model_lasso_over, X_val_val_mat,
                                    type="response")
roc_auc_scores <- sapply(as.data.frame(probabilities_lasso_over),
                         function(x) roc(val_val_dummies_sc$y, x)$auc)
lambdas <- model_lasso_over$lambda
plot(lambdas, roc_auc_scores, type="l", main ="Validation curve for Lasso")
```




```
lambda = lambdas[roc_auc_scores == max(roc_auc_scores)][1]

# modeling
```

```r
model_lasso_over <- glmnet(x=X_train_over_mat, y=y_train_over_mat,
                           family = binomial, alpha = 1, lambda = lambda)

# prediction
probabilities_lasso_over <- predict(model_lasso_over, X_test_mat,
                                    type="response")
probabilities_lasso_over <- as.vector(probabilities_lasso_over)
preds_lasso_over <- ifelse(probabilities_lasso_over > threshold, "yes", "no")
preds_lasso_over <- as.factor(preds_lasso_over)

# evaluation
roc_score_lasso_over <- roc(test$y, probabilities_lasso_over)
plot(roc_score_lasso_over ,main =sprintf("ROC curve for Lasso. AUC = %s",
                              round(roc_score_lasso_over$auc, 2)))
```




```r
confusionMatrix(data=preds_lasso_over, reference = test$y)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   no  yes
##        no   203  100
##        yes 7436 2558
##
##                Accuracy : 0.2681
##                  95% CI : (0.2596, 0.2768)
##     No Information Rate : 0.7419
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : -0.0058
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.02657
##             Specificity : 0.96238
##          Pos Pred Value : 0.66997
##          Neg Pred Value : 0.25595
##              Prevalence : 0.74187
##          Detection Rate : 0.01971
##    Detection Prevalence : 0.02943
```

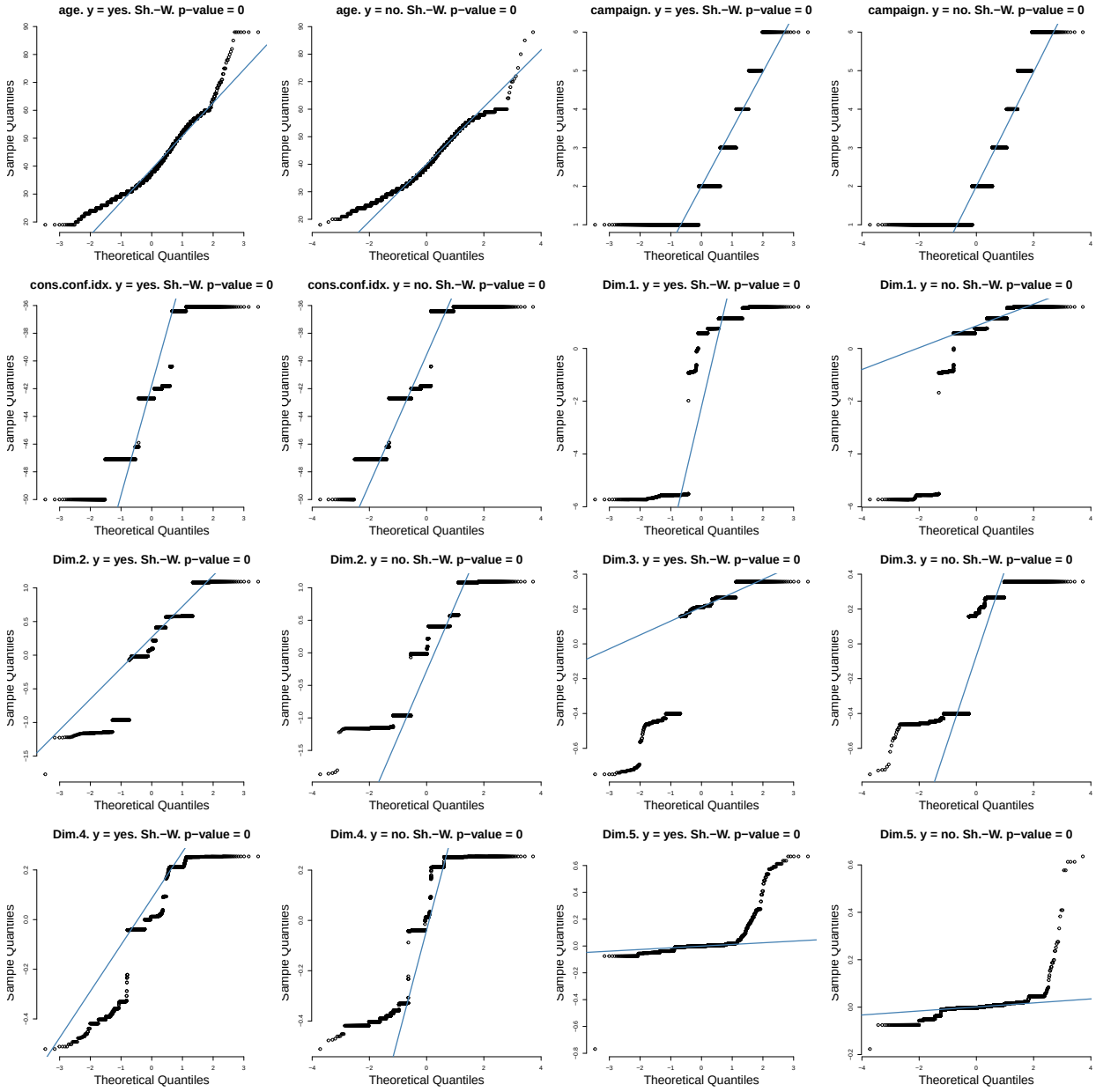
```
##       Balanced Accuracy : 0.49448
##
##        'Positive' Class : no
##
```

Unfortunately, in both cases we have weak quality.

3.7.4 Linear discriminant analysis. Before applying LDA and QDA to our problem, we should check the required assumptions. In particular, we are going to check if data distributions for each class are normal. For that purpose, we'll use Q–Q plot and also the Shapiro–Wilk test of normality.

For both methods mentioned above the absence of multicollinearity is also important. Thus, we keep using dataset with reduced multicollinearity obtained during previous steps.

```
# LDA
# Checking normality assumption
continuous_feats_pca <- c("age", "campaign",
                          "cons.conf.idx", "Dim.1", "Dim.2", "Dim.3",
                          "Dim.4", "Dim.5")
limit <- 5000
par(mfrow=c(4, 4))
for (feat in continuous_feats_pca) {
  for (cl_lab in c("yes", "no")) {
    init_arr <- pca_train[pca_train$y == cl_lab, feat]
    size <- min(limit, length(init_arr))
    arr <- sample(init_arr, size)
    res <- shapiro.test(arr)
    p_val <- res[2]$p.value
    p_val <- round(p_val, 2)
    qqnorm(arr, pch = 1, frame = FALSE,
           main=sprintf("%s. y = %s. Sh.-W. p-value = %s",
                        feat, cl_lab, p_val), cex.lab=2, cex.main=2)
    qqline(arr, col = "steelblue", lwd = 2)
  }
}
```




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

Unfortunately, there are no evidence of data normality. As a solution, we considered applying Box Cox transformation, but it didn't helped enough to normalize data. Moreover, it's not possible to apply this kind of transformation to both group separately since we wouldn't know which class unseen samples belong. Despite those circumstances, we are going to apply LDA and QDA anyway.

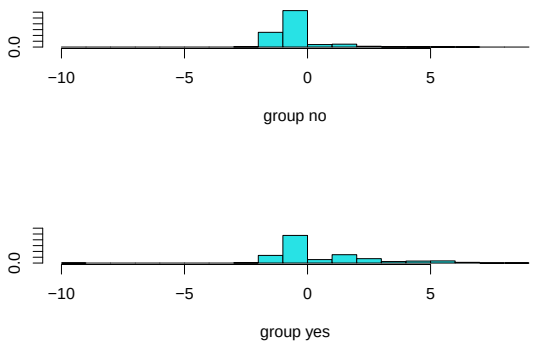
```
# modeling
model_pca_lda <- lda(y ~.-pdays-previous, data=pca_train)
model_pca_lda
```

```
## Call:
```

```
## lda(y ~ . - pdays - previous, data = pca_train)
##
## Prior probabilities of groups:
##        no       yes
## 0.9341671 0.0658329
##
## Group means:
##          age jobblue-collar jobentrepreneur jobhousemaid jobmanagement
## no  40.19483      0.2370398      0.03901078   0.02815173    0.07329862
## yes 39.77625      0.1910290      0.03957784   0.02005277    0.07546174
##     jobretired jobself-employed jobservices  jobstudent jobtechnician
## no  0.02975084       0.03558944  0.10022313 0.009074005     0.1726664
## yes 0.05593668       0.04010554  0.08548813 0.031134565     0.1656992
##     jobunemployed maritaldivorced maritalmarried education.L education.Q
## no     0.02316846       0.1166605      0.6347713   0.2253871  0.01736654
## yes    0.02163588       0.1102902      0.5788918   0.2656721  0.06402594
##     education.C education^4 education^5 education^6 housingyes   loanyes
## no   0.05331975  0.04209818   0.1162584 0.004122903  0.5364820 0.1513574
## yes  0.07001620  0.05915038   0.1156152 0.024026521  0.5572559 0.1551451
##     contacttelephone day_of_weektue day_of_weekwed day_of_weekthu
## no         0.4575307      0.2022685      0.1947564      0.2046486
## yes        0.2839050      0.1836412      0.2058047      0.2496042
##     day_of_weekfri campaign poutcomefailure poutcomesuccess cons.conf.idx
## no       0.1868353 2.093306      0.04856824     0.002789141     -40.12984
## yes      0.1678100 2.024802      0.05277045     0.027968338     -42.48332
##          Dim.1        Dim.2        Dim.3         Dim.4        Dim.5
## no   0.1005394 -0.003502097 -0.005303614  0.0006943157 -0.001354378
## yes -1.4266510  0.049694674  0.075258154 -0.0098523189  0.019218590
##
## Coefficients of linear discriminants:
##                            LD1
## age                -0.001657810
## jobblue-collar     -0.059633710
## jobentrepreneur    -0.044746529
## jobhousemaid       -0.024344661
## jobmanagement      -0.073119320
## jobretired          0.683108802
## jobself-employed    0.038707542
## jobservices        -0.109933568
## jobstudent          1.124283050
## jobtechnician      -0.030975634
## jobunemployed      -0.025196256
## maritaldivorced    -0.129155946
## maritalmarried     -0.127245846
## education.L        -0.672142977
## education.Q         0.877947254
## education.C        -0.549196547
## education^4         0.403662587
## education^5        -0.205894026
## education^6         0.164644907
## housingyes         -0.017809699
## loanyes             0.036574522
## contacttelephone   -0.220136285
## day_of_weektue      0.195904397
```

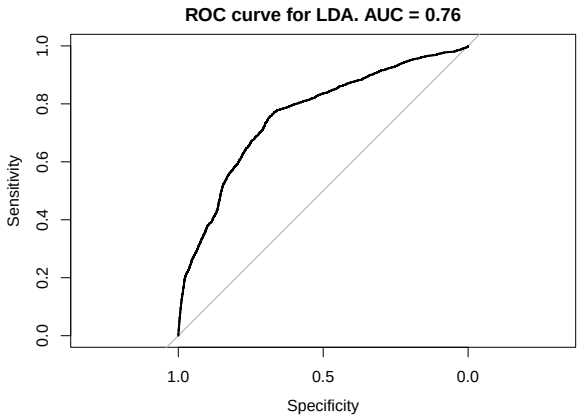
```
## day_of_weekwed    0.420686077
## day_of_weekthu    0.442814834
## day_of_weekfri    0.212756481
## campaign          0.007475876
## poutcomefailure  -1.041476840
## poutcomesuccess   4.161051304
## cons.conf.idx    -0.043555202
## Dim.1            -0.307495189
## Dim.2             0.142365166
## Dim.3             0.325217571
## Dim.4             0.131431258
## Dim.5            11.549721509
```

```r
# distribution of discriminant function values
plot(model_pca_lda)
```



By looking at distribution of discriminant function values, we can notice, that mostly two distribution overlap, but distribution for group “yes” has heavier right tail.

```r
probabilities_pca_lda <- predict(model_pca_lda, pca_test, type="response")
probabilities_pca_lda <- probabilities_pca_lda$posterior[,2]
preds_pca_lda <- ifelse(probabilities_pca_lda > threshold, "yes", "no")
preds_pca_lda <- as.factor(preds_pca_lda)

# evaluation
roc_score_pca_lda <- roc(test$y, probabilities_pca_lda)
plot(roc_score_pca_lda ,main =sprintf("ROC curve for LDA. AUC = %s",
                              round(roc_score_pca_lda$auc, 2)))
```




```
confusionMatrix(data=preds_pca_lda, reference = test$y)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   no  yes
##        no  5787  916
##        yes 1852 1742
##
##                Accuracy : 0.7312
##                  95% CI : (0.7225, 0.7397)
##     No Information Rate : 0.7419
##     P-Value [Acc > NIR] : 0.9934
##
##                   Kappa : 0.3704
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.7576
##             Specificity : 0.6554
##          Pos Pred Value : 0.8633
##          Neg Pred Value : 0.4847
##              Prevalence : 0.7419
##          Detection Rate : 0.5620
##    Detection Prevalence : 0.6510
##       Balanced Accuracy : 0.7065
##
##        'Positive' Class : no
##
```

Now we obtained better results than before. Next, we are going to repeat these steps with oversampling.

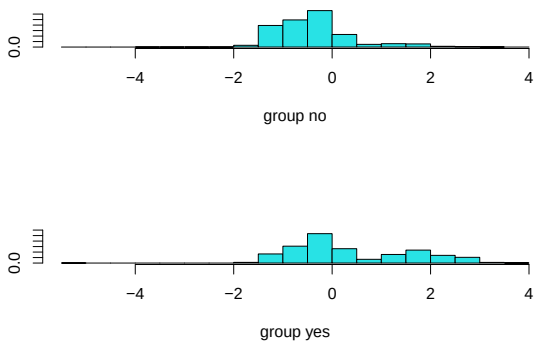
```
# oversampling
# modeling
model_pca_over_lda <- lda(y ~.-pdays-previous, data=pca_over_train)
model_pca_over_lda
```

```
## Call:
```

```
## lda(y ~ . - pdays - previous, data = pca_over_train)
##
## Prior probabilities of groups:
##        no       yes
## 0.4988313 0.5011687
##
## Group means:
##          age jobblue-collar jobentrepreneur jobhousemaid jobmanagement
## no  40.19483      0.2370398      0.03901078   0.02815173    0.07329862
## yes 39.68337      0.1882588      0.03893989   0.01921084    0.07351199
##     jobretired jobself-employed jobservices  jobstudent jobtechnician
## no  0.02975084       0.03558944  0.10022313 0.009074005     0.1726664
## yes 0.05663311       0.04093870  0.08650429 0.031907018     0.1649023
##     jobunemployed maritaldivorced maritalmarried education.L education.Q
## no     0.02316846       0.1166605      0.6347713   0.2253871  0.01736654
## yes    0.02328250       0.1134513      0.5747335   0.2656006  0.06374247
##     education.C education^4 education^5 education^6 housingyes   loanyes
## no   0.05331975  0.04209818   0.1162584 0.004122903  0.5364820 0.1513574
## yes  0.06952733  0.05913922   0.1146864 0.023478618  0.5536349 0.1558706
##     contacttelephone day_of_weektue day_of_weekwed day_of_weekthu
## no         0.4575307      0.2022685      0.1947564      0.2046486
## yes        0.2826473      0.1827065      0.2029168      0.2487415
##     day_of_weekfri campaign poutcomefailure poutcomesuccess cons.conf.idx
## no       0.1868353 2.093306      0.04856824     0.002789141     -40.12984
## yes      0.1706396 2.026503      0.05300563     0.028057447     -42.47735
##          Dim.1        Dim.2        Dim.3         Dim.4        Dim.5
## no   0.1005394 -0.003502097 -0.005303614  0.0006943157 -0.001354378
## yes -1.4177691  0.050718513  0.077059033 -0.0095133594  0.018590460
##
## Coefficients of linear discriminants:
##                            LD1
## age                -0.005975933
## jobblue-collar     -0.008444777
## jobentrepreneur     0.155310996
## jobhousemaid       -0.166242551
## jobmanagement       0.009909415
## jobretired          0.747304091
## jobself-employed    0.008181041
## jobservices         0.025225208
## jobstudent          0.581272741
## jobtechnician      -0.052284667
## jobunemployed      -0.064446846
## maritaldivorced    -0.077978675
## maritalmarried     -0.134053842
## education.L        -0.334615149
## education.Q         0.482789257
## education.C        -0.273545634
## education^4         0.353084378
## education^5        -0.247007572
## education^6         0.190197541
## housingyes         -0.069573091
## loanyes             0.086376851
## contacttelephone   -0.590792071
## day_of_weektue      0.103948806
```

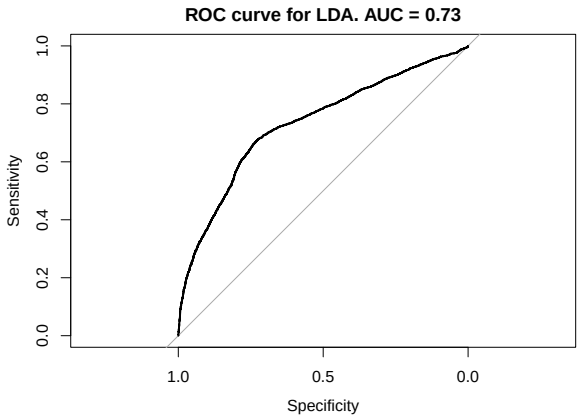
```
## day_of_weekwed    0.391822901
## day_of_weekthu    0.352920738
## day_of_weekfri    0.252543062
## campaign          0.021312239
## poutcomefailure  -0.842318341
## poutcomesuccess   1.162523864
## cons.conf.idx    -0.015112912
## Dim.1            -0.239608231
## Dim.2             0.180934668
## Dim.3             0.576051507
## Dim.4            -0.368370099
## Dim.5             5.258612422
```

```
# distribution of discriminant function values
plot(model_pca_over_lda)
```



Again, distributions have different right tails.

```
# prediction
probabilities_pca_over_lda <- predict(model_pca_over_lda, pca_test, type="response")
probabilities_pca_over_lda <- probabilities_pca_over_lda$posterior[,2]
preds_pca_over_lda <- ifelse(probabilities_pca_over_lda > threshold, "yes", "no")
preds_pca_over_lda <- as.factor(preds_pca_over_lda)

# evaluation
roc_score_pca_over_lda <- roc(test$y, probabilities_pca_over_lda)
plot(roc_score_pca_over_lda ,main =sprintf("ROC curve for LDA. AUC = %s",
                            round(roc_score_pca_over_lda$auc, 2)))
```




```
confusionMatrix(data=preds_pca_over_lda, reference = test$y)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   no  yes
##        no   339   65
##        yes 7300 2593
##
##                Accuracy : 0.2847
##                  95% CI : (0.276, 0.2936)
##     No Information Rate : 0.7419
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0106
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.04438
##             Specificity : 0.97555
##          Pos Pred Value : 0.83911
##          Neg Pred Value : 0.26210
##              Prevalence : 0.74187
##          Detection Rate : 0.03292
##    Detection Prevalence : 0.03923
##       Balanced Accuracy : 0.50996
##
##        'Positive' Class : no
##
```

However after resampling we get lower score and too optimistic predictions.

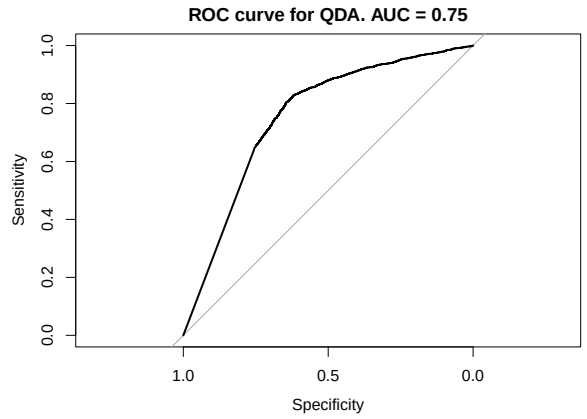
3.7.5 Quadratic discriminant analysis. Here we perform the same steps as before for LDA.

```
# QDA
# modeling
```

```
model_pca_qda <- qda(y ~.-pdays-previous, data=pca_train, family = binomial)
summary(model_pca_qda)
```

```
##         Length Class  Mode
## prior      2   -none- numeric
## counts     2   -none- numeric
## means     70   -none- numeric
## scaling 2450   -none- numeric
## ldet       2   -none- numeric
## lev        2   -none- character
## N          1   -none- numeric
## call       4   -none- call
## terms      3   terms  call
## xlevels    8   -none- list
```

```
# prediction
probabilities_pca_qda <- predict(model_pca_qda, pca_test, type="response")
probabilities_pca_qda <- probabilities_pca_qda$posterior[,2]
preds_pca_qda <- ifelse(probabilities_pca_qda > threshold, "yes", "no")
preds_pca_qda <- as.factor(preds_pca_qda)

# evaluation
roc_score_pca_qda <- roc(test$y, probabilities_pca_qda)
plot(roc_score_pca_qda ,main =sprintf("ROC curve for QDA. AUC = %s",
                              round(roc_score_pca_qda$auc, 2)))
```




```
confusionMatrix(data=preds_pca_qda, reference = test$y)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   no  yes
##        no  2143  158
##        yes 5496 2500
##
##                Accuracy : 0.4509
##                  95% CI : (0.4413, 0.4606)
##     No Information Rate : 0.7419
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1336
```

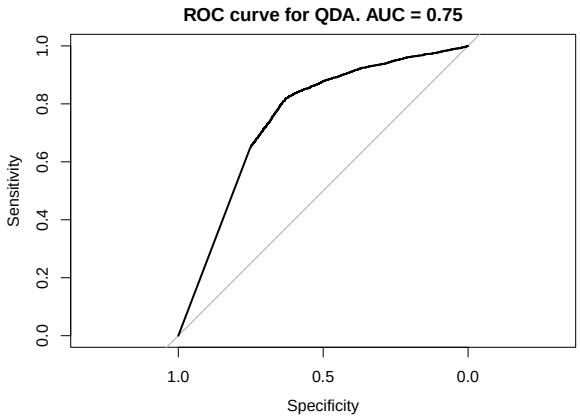
```
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.2805
##             Specificity : 0.9406
##          Pos Pred Value : 0.9313
##          Neg Pred Value : 0.3127
##              Prevalence : 0.7419
##          Detection Rate : 0.2081
##    Detection Prevalence : 0.2235
##       Balanced Accuracy : 0.6105
##
##        'Positive' Class : no
##
```

Despite the fact that QDA has more flexibility, we didn't achieve better performance.

```
# oversampling

# modeling
model_pca_over_qda <- qda(y ~.-pdays-previous, data=pca_over_train, family = binomial)
summary(model_pca_over_qda)
```

```
##         Length Class  Mode
## prior      2   -none- numeric
## counts     2   -none- numeric
## means     70   -none- numeric
## scaling 2450   -none- numeric
## ldet       2   -none- numeric
## lev        2   -none- character
## N          1   -none- numeric
## call       4   -none- call
## terms      3   terms  call
## xlevels    8   -none- list
```

```
# prediction
probabilities_pca_over_qda <- predict(model_pca_over_qda, pca_test, type="response")
probabilities_pca_over_qda <- probabilities_pca_over_qda$posterior[,2]
preds_pca_over_qda <- ifelse(probabilities_pca_over_qda > threshold, "yes", "no")
preds_pca_over_qda <- as.factor(preds_pca_over_qda)

# evaluation
roc_score_pca_over_qda <- roc(test$y, probabilities_pca_over_qda)
plot(roc_score_pca_over_qda ,main =sprintf("ROC curve for QDA. AUC = %s",
                              round(roc_score_pca_over_qda$auc, 2)))
```




```
confusionMatrix(data=preds_pca_over_qda, reference = test$y)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   no  yes
##        no   459   35
##        yes 7180 2623
##
##                Accuracy : 0.2993
##                  95% CI : (0.2905, 0.3083)
##     No Information Rate : 0.7419
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.025
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.06009
##             Specificity : 0.98683
##          Pos Pred Value : 0.92915
##          Neg Pred Value : 0.26757
##              Prevalence : 0.74187
##          Detection Rate : 0.04458
##    Detection Prevalence : 0.04798
##       Balanced Accuracy : 0.52346
##
##        'Positive' Class : no
##
```

Same as before, resampling didn't help us to improve the score.

3.7.6 Naive Bayes. Additionally, we will apply Naive Bayes classifier for our task. This method implies very strong assumption about the conditional independence of predictors. Previously we have seen that this assumption doesn't work for our case. However, this method may produce good classification results.

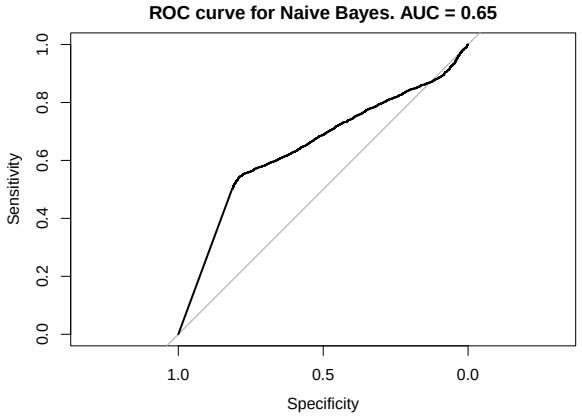
```r
# Naive Bayes
# modeling
pca_nb <- naiveBayes(y ~.-pdays-previous, data=pca_train)
pca_nb
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##       no       yes
## 0.9341671 0.0658329
##
## Conditional probabilities:
##      age
## Y         [,1]      [,2]
##   no  40.19483  9.279969
##   yes 39.77625 11.168696
##
##      job
## Y          admin. blue-collar entrepreneur   housemaid  management     retired
##   no  0.252026776 0.237039792  0.039010785 0.028151729 0.073298624 0.029750837
##   yes 0.273878628 0.191029024  0.039577836 0.020052770 0.075461741 0.055936675
##      job
## Y     self-employed    services     student  technician  unemployed
##   no    0.035589438 0.100223131 0.009074005 0.172666419 0.023168464
##   yes   0.040105541 0.085488127 0.031134565 0.165699208 0.021635884
##
##      marital
## Y        single  divorced   married
##   no  0.2485682 0.1166605 0.6347713
##   yes 0.3108179 0.1102902 0.5788918
##
##      education
## Y      illiterate    basic.4y    basic.6y    basic.9y high.school
##   no  0.000409074 0.105615470 0.058348829 0.152919301 0.226850130
##   yes 0.001055409 0.083377309 0.053825858 0.127704485 0.223218997
##      education
## Y     professional.course university.degree
##   no          0.130978059       0.324879137
##   yes         0.126121372       0.384696570
##
##      housing
## Y           no       yes
##   no  0.4635180 0.5364820
##   yes 0.4427441 0.5572559
##
##      loan
## Y           no       yes
##   no  0.8486426 0.1513574
##   yes 0.8448549 0.1551451
```

```
##
##      contact
## Y      cellular telephone
##   no  0.5424693 0.4575307
##   yes 0.7160950 0.2839050
##
##      day_of_week
## Y           mon       tue       wed       thu       fri
##   no  0.2114913 0.2022685 0.1947564 0.2046486 0.1868353
##   yes 0.1931398 0.1836412 0.2058047 0.2496042 0.1678100
##
##      campaign
## Y         [,1]     [,2]
##   no  2.093306 1.298719
##   yes 2.024802 1.266045
##
##      poutcome
## Y     nonexistent     failure     success
##   no  0.948642618 0.048568241 0.002789141
##   yes 0.919261214 0.052770449 0.027968338
##
##      cons.conf.idx
## Y          [,1]     [,2]
##   no  -40.12984 3.640253
##   yes -42.48332 4.382963
##
##      Dim.1
## Y           [,1]    [,2]
##   no   0.1005394 1.94552
##   yes -1.4266510 3.01692
##
##      Dim.2
## Y             [,1]      [,2]
##   no  -0.003502097 0.7449604
##   yes  0.049694674 0.6800925
##
##      Dim.3
## Y             [,1]      [,2]
##   no  -0.005303614 0.3351666
##   yes  0.075258154 0.3081599
##
##      Dim.4
## Y              [,1]      [,2]
##   no   0.0006943157 0.2458193
##   yes -0.0098523189 0.2183069
##
##      Dim.5
## Y             [,1]       [,2]
##   no  -0.001354378 0.03307866
##   yes  0.019218590 0.10366140
```

```r
# prediction
probabilities_pca_nb <- predict(pca_nb, pca_test, type = "raw")[,2]
preds_pca_nb <- predict(pca_nb, pca_test)
```

```r
# evaluation
roc_score_pca_nb <- roc(test$y, probabilities_pca_nb)
plot(roc_score_pca_nb ,main =sprintf("ROC curve for Naive Bayes. AUC = %s",
                                     round(roc_score_pca_nb$auc, 2)))
```



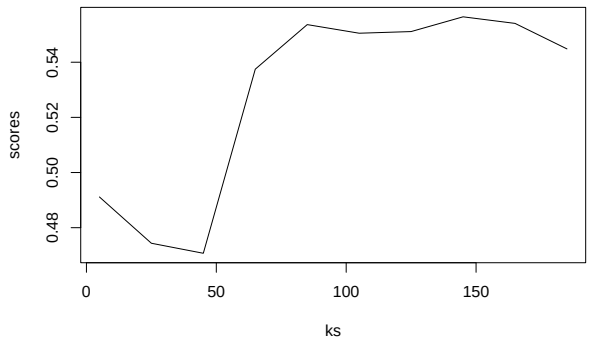

```r
confusionMatrix(data=preds_pca_nb, reference = test$y)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   no  yes
##        no    52   27
##        yes 7587 2631
##
##                Accuracy : 0.2606
##                  95% CI : (0.2521, 0.2692)
##     No Information Rate : 0.7419
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : -0.0017
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.006807
##             Specificity : 0.989842
##          Pos Pred Value : 0.658228
##          Neg Pred Value : 0.257487
##              Prevalence : 0.741867
##          Detection Rate : 0.005050
##    Detection Prevalence : 0.007672
##       Balanced Accuracy : 0.498325
##
##        'Positive' Class : no
##
```

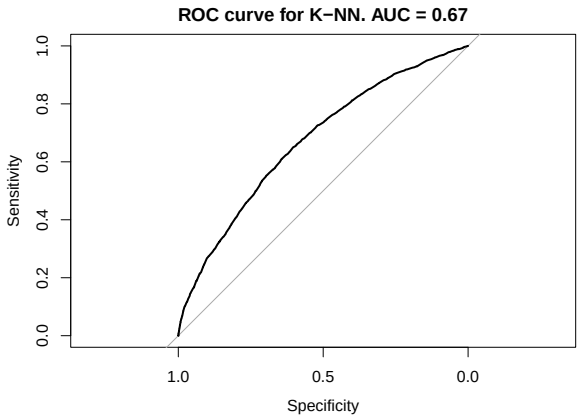
Here Naive Bayes has the worse performance than LDA and QDA.

3.7.7 K-NN. Finally, we will test K-NN performance for our task. Here we will use validation set for selecting number of neighbors k.

```
# KNN
# selecting k
scores <- c()
thresholds <- c()
ks <- seq(from = 5, to = 200, by = 20)
for (k in ks) {
  preds <- class::knn(train=X_val_train_mat,
                test=X_val_val_mat,
                cl = val_train_dummies_sc$y, k = k,
                prob = TRUE)
  probabilities <- attributes(preds)$prob
  score  <- roc(val_val_dummies_sc$y, probabilities)$auc
  scores <- c(scores, score)
}

k <- ks[scores == max(scores)][1]
plot(ks, scores, type='l')
```




```
# modeling
out.knn <- class::knn(train=X_train_mat,
                      test=X_test_mat,
                      cl = train_dummies_sc$y, k = k,
                      prob = TRUE)

# prediction
probabilities_knn <- attributes(out.knn)$prob
preds_knn <- ifelse(probabilities_knn > threshold, "yes", "no")
preds_knn <- factor(preds_knn, levels=c("no", "yes"))

# evaluation
roc_score_knn <- roc(test$y, probabilities_knn)
plot(roc_score_knn ,main =sprintf("ROC curve for K-NN. AUC = %s",
                                  round(roc_score_knn$auc, 2)))
```



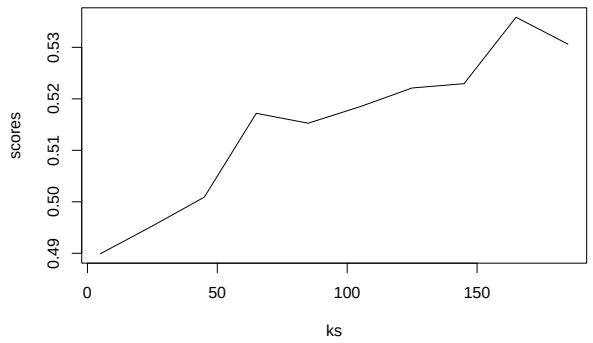

```
confusionMatrix(data=preds_knn, reference = test$y)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   no  yes
##        no     0    0
##        yes 7639 2658
##
##                Accuracy : 0.2581
##                  95% CI : (0.2497, 0.2667)
##     No Information Rate : 0.7419
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.0000
##             Specificity : 1.0000
##          Pos Pred Value :    NaN
##          Neg Pred Value : 0.2581
##              Prevalence : 0.7419
##          Detection Rate : 0.0000
##    Detection Prevalence : 0.0000
##       Balanced Accuracy : 0.5000
##
##        'Positive' Class : no
##
```

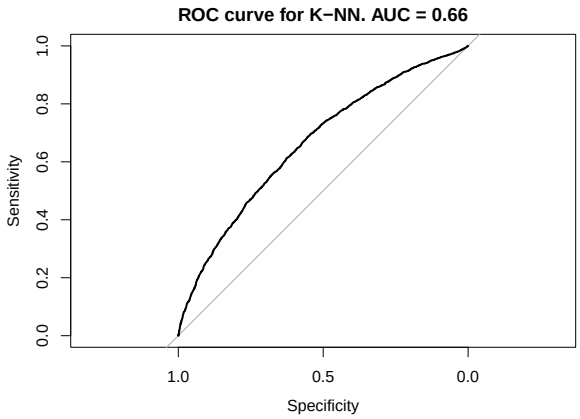
The score is pretty weak and the model doesn't give a postive labels for test samples. Next, we are going to repeat these steps with oversampling.

```
# oversampling

# selecting k
scores <- c()
thresholds <- c()
```

```
ks <- seq(from = 5, to = 200, by = 20)
for (k in ks) {
  preds <- class::knn(train=X_val_train_over_mat,
                      test=X_val_val_mat,
                      cl = val_train_dummies_sc_over$y, k = k,
                      prob = TRUE)
  probabilities <- attributes(preds)$prob
  score  <- roc(val_val_dummies_sc$y, probabilities)$auc
  scores <- c(scores, score)
}

k <- ks[scores == max(scores)][1]
plot(ks, scores, type='l')
```




```
# modeling
out.knn_over <- class::knn(train=X_train_over_mat,
                      test=X_test_mat,
                      cl = train_dummies_sc_over$y, k = k,
                      prob = TRUE)

# prediction
probabilities_knn_over <- attributes(out.knn_over)$prob
preds_knn_over <- ifelse(probabilities_knn_over > threshold, "yes", "no")
preds_knn_over <- factor(preds_knn_over, levels=c("no", "yes"))

# evaluation
roc_score_knn_over <- roc(test$y, probabilities_knn_over)
plot(roc_score_knn_over ,main =sprintf("ROC curve for K-NN. AUC = %s",
                              round(roc_score_knn_over$auc, 2)))
```




```
confusionMatrix(data=preds_knn_over, reference = test$y)
```

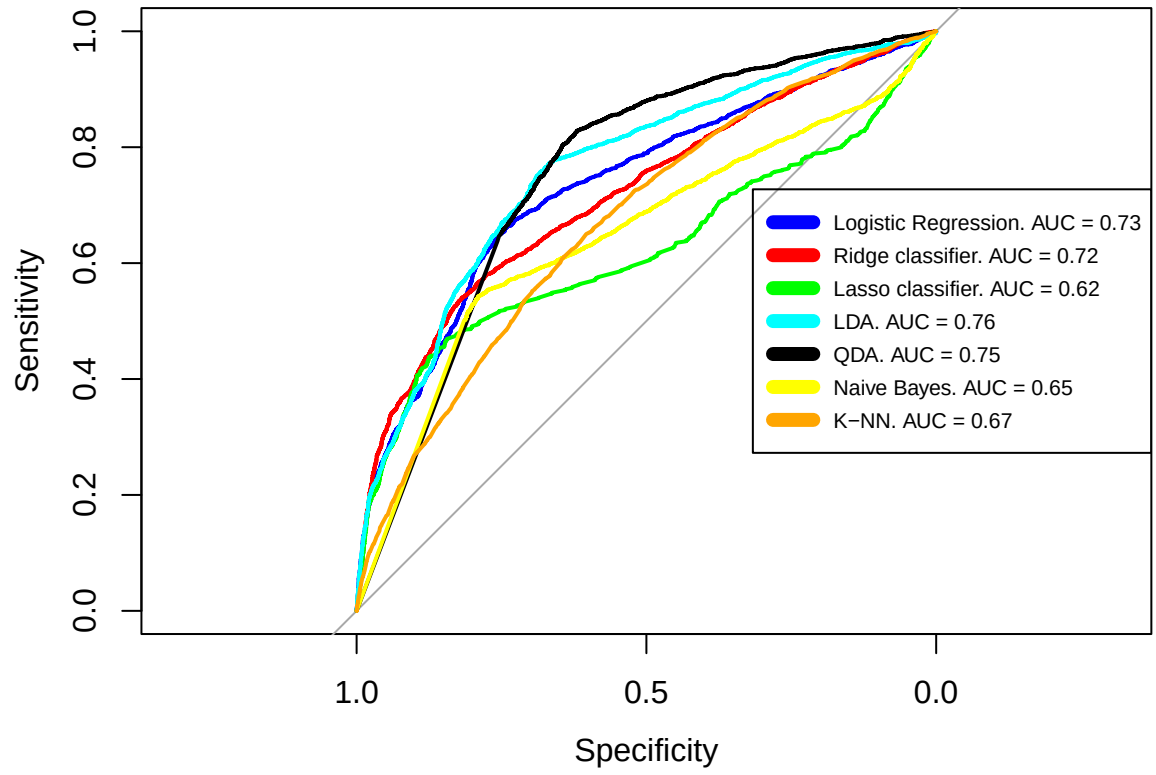
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   no  yes
##        no    15    3
##        yes 7624 2655
##
##                Accuracy : 0.2593
##                  95% CI : (0.2509, 0.2679)
##     No Information Rate : 0.7419
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 4e-04
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.001964
##             Specificity : 0.998871
##          Pos Pred Value : 0.833333
##          Neg Pred Value : 0.258294
##              Prevalence : 0.741867
##          Detection Rate : 0.001457
##    Detection Prevalence : 0.001748
##       Balanced Accuracy : 0.500417
##
##        'Positive' Class : no
##
```

We didn't reach higher score and we faced to opposite problem in terms of class preference for test prediction.

3.8 Model comparison

In this section we will plot ROC curves for all obtained results. We won't consider ones with resampling since their perfomance is worse.

```r
# Gather all results
roc_curve <- plot(roc_score_pca_fsel_lr, col = "blue")
roc_curve <- plot(roc_score_ridge, col = "red", add = TRUE)
roc_curve <- plot(roc_score_lasso, col = "green", add = TRUE)
roc_curve <- plot(roc_score_pca_lda, col = "cyan", add = TRUE)
roc_curve <- plot(roc_score_pca_qda, col = "black", add = TRUE)
roc_curve <- plot(roc_score_pca_nb, col = "yellow", add = TRUE)
roc_curve <- plot(roc_score_knn, col = "orange", add = TRUE)

col_legend <- c(sprintf("Logistic Regression. AUC = %s", round(roc_score_pca_fsel_lr$auc, 2)),
                sprintf("Ridge classifier. AUC = %s", round(roc_score_ridge$auc, 2)),
                sprintf("Lasso classifier. AUC = %s", round(roc_score_lasso$auc, 2)),
                sprintf("LDA. AUC = %s", round(roc_score_pca_lda$auc, 2)),
                sprintf("QDA. AUC = %s", round(roc_score_pca_qda$auc, 2)),
                sprintf("Naive Bayes. AUC = %s", round(roc_score_pca_nb$auc, 2)),
                sprintf("K-NN. AUC = %s", round(roc_score_knn$auc, 2)))

legend(x = "right",inset = 0,
       legend = col_legend,
       col=c("blue", "red", "green",
             "cyan", "black", "yellow",
             "orange"), lwd=7, cex=.7, horiz = FALSE)
```



By looking at ROC curves, we can conclude, that LDA and QDA turned out to have the best performances on the test set. By applying Ridge classifier we could achieve almost the same results as we got by using

Logistic Regression after feature selection and reducing multicollinearity. Unfortunately, K-NN, which is based on estimating similarity between points in a feature space, has failed predicting test labels. Apparently, Naive Bayes has too strong assumption for our case to have a good performance. Finally, the way Lasso Classifier penalizes coefficient values didn't show good results.

4 Conclusions

In this project, we faced with real world Data Science problem, which was related to searching for the clients during for the Bank marketing campaign. Despite the fact that project's authors don't have any background in bank and finance area, it was possible to analyse the data, reveal certain patterns, understand the way campaign was going and make an assumption about casual interaction between features. Then, according to business task, the proper metric and validation strategy has been chosen. The authors were aware of distinction between train and test sample (they belong to different policies), however this is the example of real world situation, and chosen strategy is suitable for searching for robust model. During the modeling part, seven different classification models have been tested. As result, we obtained two models with suitable for this kind of tasks quality. Obviously, those models could be used in production phase for improving the bank's business processes.