Group 7 - Groupwork on leding club loan data

L. Becker, A. Chebatarova, A. Kandel, A.Kusche, R. Mizrak

2019-05-27

Introduction

Our data comes from Lending Club (LC) which is a peer-to-peer online lending platform. The data contains information on the loans issued by LC including the details of the loans such as loan description, interest rate applied, if the debt was fully paid, date of last payment etc. Our goal is to find appropriate models for predicting interest rates for each request, as well as perform a classification model for setting reliable default status.

URL to source data: https://kaggle.com/wendykan/lending-club-loan-data

Required steps:

- Initialization of the environment.
- Data exploration through data visualization which helps to understand our dataset and its characteristics.
- Data preprocessing which includes omitting unnecessary columns, sorting and grouping, reformatting and other actions required for making our data adequate for performing further analysis and modeling.
- Part 1: Regression Analysis - we get into details with our data to define a meaningful amount of data, rational predictors, check correlations and determine models for prediction of interest rate and validate them.
- Part 2 - Classification Analysis - we take a step back in order to be sure that all necessary variables are included into our analysis, perform required transformations, define the models, check the errors and validate the results.
- Summary

Initializing the environment

At this step we clear the workspace and install necessary packages for data processing.

```r
# Clear objects from the workspace
rm(list=(ls()))

# load library to deal with packages
library(pacman)

# install and loading required packages
pacman::p_load(import, monomvn, party, dummies, ranger, data.table, rmarkdown, tidyverse,
caret, pls, corrplot, randomForest, foreach, plyr, tidyverse, magrittr, dplyr, tibble, doMC,
pROC, class,MLmetrics, tree, car, ridge, lmridge, xgboost)
```

Data Extraction

In order to avoid to have to work with the whole original dataset from kaggle dataset_7.Rds has been created as follows:

Read the subset of data from the previous step

```
# Read the RDS file
dataset <- readRDS(file = "dataset_7.Rds")
# Setting seed
set.seed(3452)
dataset <- dataset[sample(1:nrow(dataset),20000),]
```

Data Preprocessing

Preprocess the dataset, remove columns, check for na ...

```
  # sort dataset by column names, to facilite search
  dataset = dataset[ , order(names(dataset))]

  # Removing columns that have > 0.05 NAs
  dataset <- dataset[, -which(colMeans(is.na(dataset)) > 0.05)]

  # remove some columns because of the reasons below
  # to many levels: zip_code, emp_title
  # not_relevant: desc, id_2, addr_state, last_pymnt_d, next_pymnt_d, issue_d, title,
  #last_credit_pull_d, hardship_end_date, hardship_start_date, payment_plan_start_date,
  #debt_settlement_flag_date, settlement_date
  # same data in every colum: policy_code
  # covariance: grade (of sub_grade)

  dataset <- subset(dataset, select = -c(id_2, policy_code, desc, emp_title, issue_d, title,
zip_code, last_pymnt_d, next_pymnt_d, last_credit_pull_d, hardship_end_date, hardship_start_date,
payment_plan_start_date, debt_settlement_flag_date, settlement_date, addr_state, grade) )

  # sub_grade has more than 32 levels which is a hard limit for a Random Forest.
  # We will dummy code it to circument this
  levels_sub_grade <- levels(dataset$sub_grade)
  dataset$sub_grade<- as.numeric(mapvalues(dataset$sub_grade, levels_sub_grade,
  seq(from = 1, to = 35, by = 1)))


  # sorting emp_lenght, and dummy coding.
  levels_emp_length <- levels(dataset$emp_length)
  dataset$emp_length <- ordered(dataset$emp_length, levels = c("n/a", "< 1 year", "1 year",
"2 years", "3 years","4 years", "5 years","6 years", "7 years", "8 years", "9 years",
"10+ years"))
  levels(dataset$emp_length)
```

```
##  [1] "n/a"       "< 1 year"  "1 year"    "2 years"   "3 years"
##  [6] "4 years"   "5 years"   "6 years"   "7 years"   "8 years"
## [11] "9 years"   "10+ years"
```

```
  dataset$emp_length <- as.numeric(mapvalues(dataset$emp_length, levels_emp_length, c(-1, seq(from = 0,
```

```r
# grouping earliest_cr_line by year
dataset$earliest_cr_line <- as.integer(substring(dataset$earliest_cr_line, 5))

# grouping earliest_cr_line by year
dataset$sec_app_earliest_cr_line <-
as.integer(substring(as.character(dataset$sec_app_earliest_cr_line), 5))
dataset$sec_app_earliest_cr_line[is.na(dataset$sec_app_earliest_cr_line)] <- -1
typeof(dataset$sec_app_earliest_cr_line)
```

```
## [1] "double"
```

```r
na_count <-sapply(dataset, function(y) sum(length(which(is.na(y)))))

# changing the dataset as a tbl object.
dataset <- as.tbl(dataset)

# change NAs to -1 in integer columns
dataset <- mutate_if(dataset, is.integer, ~replace(., is.na(.), -1))

# change NAs to -1 in numeric columns
  dataset <- mutate_if(dataset, is.numeric, ~replace(., is.na(.), -1))

# change missing values to "NA" in strings columns
dataset <- mutate_if(dataset, is.character, ~replace(., is.na(.), "NA"))

# change missing Values to "NA" in strings columns
  dataset <- mutate_if(dataset, is.factor, ~replace(., is.na(.), "NA"))


# calculating the number of unique levels per column
sapply(dataset, function(col) length(unique(col)))
```

```
##             acc_now_delinq       acc_open_past_24mths
##                          5                         34
##                 annual_inc           application_type
##                       2482                          2
##                avg_cur_bal             bc_open_to_buy
##                      13493                      12312
##                    bc_util   chargeoff_within_12_mths
##                       1075                          4
##    collection_recovery_fee collections_12_mths_ex_med
##                       1500                          5
##       debt_settlement_flag                delinq_2yrs
##                          2                         18
##                delinq_amnt        disbursement_method
##                         62                          2
##                        dti           earliest_cr_line
##                       3862                         56
##                 emp_length                funded_amnt
##                         12                       1079
##            funded_amnt_inv              hardship_flag
##                       1207                          2
##       hardship_loan_status            hardship_reason
##                          5                          9
```

```
##          hardship_status               hardship_type
##                        4                           2
##           home_ownership         initial_list_status
##                        6                           2
##           inq_last_6mths                 installment
##                       12                       10601
##                 int_rate              last_pymnt_amnt
##                      475                       15422
##                loan_amnt                 loan_status
##                     1079                           8
##     mo_sin_old_rev_tl_op        mo_sin_rcnt_rev_tl_op
##                      571                         155
##           mo_sin_rcnt_tl                    mort_acc
##                      102                          18
##     mths_since_recent_bc        num_accts_ever_120_pd
##                      239                          20
##           num_actv_bc_tl               num_actv_rev_tl
##                       26                          37
##              num_bc_sats                   num_bc_tl
##                       33                          43
##                num_il_tl                num_op_rev_tl
##                       69                          44
##             num_rev_accts          num_rev_tl_bal_gt_0
##                       64                          33
##                 num_sats                 num_tl_30dpd
##                       51                           5
##       num_tl_90g_dpd_24m          num_tl_op_past_12m
##                       14                          20
##                 open_acc                   out_prncp
##                       54                        7727
##            out_prncp_inv                pct_tl_nvr_dlq
##                     7763                         347
##         percent_bc_gt_75                     pub_rec
##                      110                          11
##     pub_rec_bankruptcies                     purpose
##                        7                          14
##               pymnt_plan                  recoveries
##                        2                        1559
##                revol_bal                  revol_util
##                    14993                        1048
##  sec_app_earliest_cr_line           settlement_status
##                       48                           4
##                sub_grade                   tax_liens
##                       35                          11
##                     term                tot_coll_amt
##                        2                        1471
##              tot_cur_bal               tot_hi_cred_lim
##                    18536                       17432
##                total_acc             total_bal_ex_mort
##                       92                       17662
##           total_bc_limit total_il_high_credit_limit
##                     1495                       14477
##              total_pymnt               total_pymnt_inv
##                    19621                       19511
```

```
##              total_rec_int         total_rec_late_fee
##                      19125                        540
##            total_rec_prncp            total_rev_hi_lim
##                      10863                       1997
##        verification_status  verification_status_joint
##                          3                          4
```

```r
na_count <-sapply(dataset, function(y) sum(length(which(is.na(y)))))

#check data after processing
head(dataset)
```

```
## # A tibble: 6 x 82
##   acc_now_delinq acc_open_past_2~ annual_inc application_type avg_cur_bal
##            <dbl>            <dbl>      <dbl> <fct>                  <dbl>
## 1              0                0     100000 Individual              2774
## 2              0                9      95000 Individual              7160
## 3              0                7      70600 Individual             24916
## 4              0                5      75000 Individual              4112
## 5              0                9      85000 Individual             18856
## 6              0                3      88000 Individual             57576
## # ... with 77 more variables: bc_open_to_buy <dbl>, bc_util <dbl>,
## #   chargeoff_within_12_mths <dbl>, collection_recovery_fee <dbl>,
## #   collections_12_mths_ex_med <dbl>, debt_settlement_flag <fct>,
## #   delinq_2yrs <dbl>, delinq_amnt <dbl>, disbursement_method <fct>,
## #   dti <dbl>, earliest_cr_line <dbl>, emp_length <dbl>,
## #   funded_amnt <dbl>, funded_amnt_inv <dbl>, hardship_flag <fct>,
## #   hardship_loan_status <fct>, hardship_reason <fct>,
## #   hardship_status <fct>, hardship_type <fct>, home_ownership <fct>,
## #   initial_list_status <fct>, inq_last_6mths <dbl>, installment <dbl>,
## #   int_rate <dbl>, last_pymnt_amnt <dbl>, loan_amnt <dbl>,
## #   loan_status <fct>, mo_sin_old_rev_tl_op <dbl>,
## #   mo_sin_rcnt_rev_tl_op <dbl>, mo_sin_rcnt_tl <dbl>, mort_acc <dbl>,
## #   mths_since_recent_bc <dbl>, num_accts_ever_120_pd <dbl>,
## #   num_actv_bc_tl <dbl>, num_actv_rev_tl <dbl>, num_bc_sats <dbl>,
## #   num_bc_tl <dbl>, num_il_tl <dbl>, num_op_rev_tl <dbl>,
## #   num_rev_accts <dbl>, num_rev_tl_bal_gt_0 <dbl>, num_sats <dbl>,
## #   num_tl_30dpd <dbl>, num_tl_90g_dpd_24m <dbl>,
## #   num_tl_op_past_12m <dbl>, open_acc <dbl>, out_prncp <dbl>,
## #   out_prncp_inv <dbl>, pct_tl_nvr_dlq <dbl>, percent_bc_gt_75 <dbl>,
## #   pub_rec <dbl>, pub_rec_bankruptcies <dbl>, purpose <fct>,
## #   pymnt_plan <fct>, recoveries <dbl>, revol_bal <dbl>, revol_util <dbl>,
## #   sec_app_earliest_cr_line <dbl>, settlement_status <fct>,
## #   sub_grade <dbl>, tax_liens <dbl>, term <fct>, tot_coll_amt <dbl>,
## #   tot_cur_bal <dbl>, tot_hi_cred_lim <dbl>, total_acc <dbl>,
## #   total_bal_ex_mort <dbl>, total_bc_limit <dbl>,
## #   total_il_high_credit_limit <dbl>, total_pymnt <dbl>,
## #   total_pymnt_inv <dbl>, total_rec_int <dbl>, total_rec_late_fee <dbl>,
## #   total_rec_prncp <dbl>, total_rev_hi_lim <dbl>,
## #   verification_status <fct>, verification_status_joint <fct>
```

```r
dim(dataset)
```

```
## [1] 20000    82
```

```r
str(dataset)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    20000 obs. of  82 variables:
##  $ acc_now_delinq            : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ acc_open_past_24mths      : num  0 9 7 5 9 3 5 2 2 5 ...
##  $ annual_inc                : num  100000 95000 70600 75000 85000 88000 50000 60000 40000 83000 ...
##  $ application_type          : Factor w/ 2 levels "Individual","Joint App": 1 1 1 1 1 1 1 1 1 1 ...
##  $ avg_cur_bal               : num  2774 7160 24916 4112 18856 ...
##  $ bc_open_to_buy            : num  2867 29318 12535 2646 23194 ...
##  $ bc_util                   : num  58 18.1 33.7 39.9 30.1 34.8 83.4 87.6 31.7 87.9 ...
##  $ chargeoff_within_12_mths  : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ collection_recovery_fee   : num  0 0 243 0 0 0 0 0 0 0 ...
##  $ collections_12_mths_ex_med: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ debt_settlement_flag      : Factor w/ 2 levels "N","Y": 1 1 2 1 1 1 1 1 1 1 ...
##  $ delinq_2yrs               : num  0 0 2 3 0 0 0 0 0 0 ...
##  $ delinq_amnt               : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ disbursement_method       : Factor w/ 2 levels "Cash","DirectPay": 1 2 1 1 1 1 1 1 1 1 ...
##  $ dti                       : num  19.2 12.2 32.1 17.2 15.1 ...
##  $ earliest_cr_line          : num  2012 1995 1991 2006 1988 ...
##  $ emp_length                : num  12 12 12 6 1 12 12 12 12 10 ...
##  $ funded_amnt               : num  24000 4500 17600 5000 28000 12000 20800 3500 15500 21000 ...
##  $ funded_amnt_inv           : num  24000 4500 17600 5000 27750 ...
##  $ hardship_flag             : Factor w/ 2 levels "N","Y": 1 1 1 1 1 1 1 1 1 1 ...
##  $ hardship_loan_status      : Factor w/ 6 levels "","Current","In Grace Period",..: 1 1 1 1 1 1 1
##  $ hardship_reason           : Factor w/ 10 levels "","DISABILITY",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ hardship_status           : Factor w/ 4 levels "","ACTIVE","BROKEN",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ hardship_type             : Factor w/ 2 levels "","INTEREST ONLY-3 MONTHS DEFERRAL": 1 1 1 1 1 1
##  $ home_ownership            : Factor w/ 6 levels "ANY","MORTGAGE",..: 6 5 5 2 2 2 6 6 5 6 ...
##  $ initial_list_status       : Factor w/ 2 levels "f","w": 2 2 2 1 2 1 2 2 2 2 ...
##  $ inq_last_6mths            : num  0 0 1 0 0 0 0 0 0 3 ...
##  $ installment               : num  534 137 440 193 870 ...
##  $ int_rate                  : num  11.98 6.11 17.27 22.91 7.39 ...
##  $ last_pymnt_amnt           : num  533.63 7.43 106 193.32 101.68 ...
##  $ loan_amnt                 : num  24000 4500 17600 5000 28000 12000 20800 3500 15500 21000 ...
##  $ loan_status               : Factor w/ 9 levels "Charged Off",..: 2 2 1 2 6 6 2 2 6 2 ...
##  $ mo_sin_old_rev_tl_op      : num  70 275 295 113 331 122 110 271 56 332 ...
##  $ mo_sin_rcnt_rev_tl_op     : num  26 2 1 0 10 7 1 5 12 5 ...
##  $ mo_sin_rcnt_tl            : num  18 2 1 0 10 7 1 5 12 5 ...
##  $ mort_acc                  : num  0 5 5 0 2 6 0 0 0 0 ...
##  $ mths_since_recent_bc      : num  47 16 1 15 10 7 1 64 12 5 ...
##  $ num_accts_ever_120_pd     : num  0 1 2 0 6 0 0 0 0 0 ...
##  $ num_actv_bc_tl            : num  6 3 3 4 4 2 8 3 1 8 ...
##  $ num_actv_rev_tl           : num  13 4 4 4 7 3 14 8 1 11 ...
##  $ num_bc_sats               : num  6 6 7 4 7 2 8 3 3 9 ...
##  $ num_bc_tl                 : num  6 18 11 4 20 5 8 3 4 11 ...
##  $ num_il_tl                 : num  8 12 7 14 4 6 3 16 5 2 ...
##  $ num_op_rev_tl             : num  13 14 10 6 13 3 16 8 5 14 ...
##  $ num_rev_accts             : num  15 36 18 7 31 6 16 13 8 23 ...
##  $ num_rev_tl_bal_gt_0       : num  11 4 4 4 7 3 14 8 1 11 ...
##  $ num_sats                  : num  15 17 14 16 17 7 17 10 9 16 ...
##  $ num_tl_30dpd              : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ num_tl_90g_dpd_24m        : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ num_tl_op_past_12m        : num  0 4 3 3 3 1 3 2 1 3 ...
##  $ open_acc                  : num  15 20 14 16 17 7 17 10 9 16 ...
```

```
##  $ out_prncp                  : num  21246 0 0 3446 0 ...
##  $ out_prncp_inv              : num  21246 0 0 3446 0 ...
##  $ pct_tl_nvr_dlq             : num  100 92.3 88.9 85.7 84.2 100 100 100 100 100 ...
##  $ percent_bc_gt_75           : num  33.3 0 16.7 0 0 50 62.5 100 0 77.8 ...
##  $ pub_rec                    : num  1 1 0 0 0 0 0 0 0 1 ...
##  $ pub_rec_bankruptcies       : num  1 1 0 0 0 0 0 0 0 1 ...
##  $ purpose                    : Factor w/ 14 levels "car","credit_card",..: 3 3 3 3 3 3 3 10 3 3 ...
##  $ pymnt_plan                 : Factor w/ 2 levels "n","y": 1 1 1 1 1 1 1 1 1 1 ...
##  $ recoveries                 : num  0 0 1350 0 0 0 0 0 0 0 ...
##  $ revol_bal                  : num  18987 6787 10187 1754 14430 ...
##  $ revol_util                 : num  47 13.7 42.1 31.3 31.2 34.8 80 78.6 20.5 81.7 ...
##  $ sec_app_earliest_cr_line   : num  -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 ...
##  $ settlement_status          : Factor w/ 4 levels "","ACTIVE","BROKEN",..: 1 1 2 1 1 1 1 1 1 1 ...
##  $ sub_grade                  : num  10 1 17 21 4 2 20 10 6 15 ...
##  $ tax_liens                  : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ term                       : Factor w/ 2 levels " 36 months"," 60 months": 2 1 2 1 1 1 2 1 1 1 ...
##  $ tot_coll_amt               : num  0 0 0 66 0 0 0 0 0 0 ...
##  $ tot_cur_bal                : num  41618 114553 298993 65795 282837 ...
##  $ tot_hi_cred_lim            : num  80089 185528 347285 73847 368421 ...
##  $ total_acc                  : num  23 53 30 21 38 18 19 29 13 25 ...
##  $ total_bal_ex_mort          : num  41618 114553 91989 65795 42347 ...
##  $ total_bc_limit             : num  17200 35800 18900 4400 33200 10500 10700 18000 14500 31200 ...
##  $ total_il_high_credit_limit: num  39289 135828 93792 68247 48583 ...
##  $ total_pymnt                : num  4787 4568 11118 2700 30214 ...
##  $ total_pymnt_inv            : num  4787 4568 11118 2700 29944 ...
##  $ total_rec_int              : num  2032.3 67.8 4955.1 1146.3 2213.6 ...
##  $ total_rec_late_fee         : num  0 0 22 0 0 0 0 0 0 0 ...
##  $ total_rec_prncp            : num  2754 4500 4791 1554 28000 ...
##  $ total_rev_hi_lim           : num  40800 49700 24200 5600 46300 60500 22200 48000 22400 38300 ...
##  $ verification_status        : Factor w/ 3 levels "Not Verified",..: 3 1 1 2 3 1 2 1 1 2 ...
##  $ verification_status_joint : Factor w/ 4 levels "","Not Verified",..: 1 1 1 1 1 1 1 1 1 1 ...
```

Part 1 - Regression Analysis

Preparatory tasks:

Create a copy of your dataset, eliminating the entries that have an "na" in the interest rate variable int_rate. (Interest rate is used as output variable)

Initialization

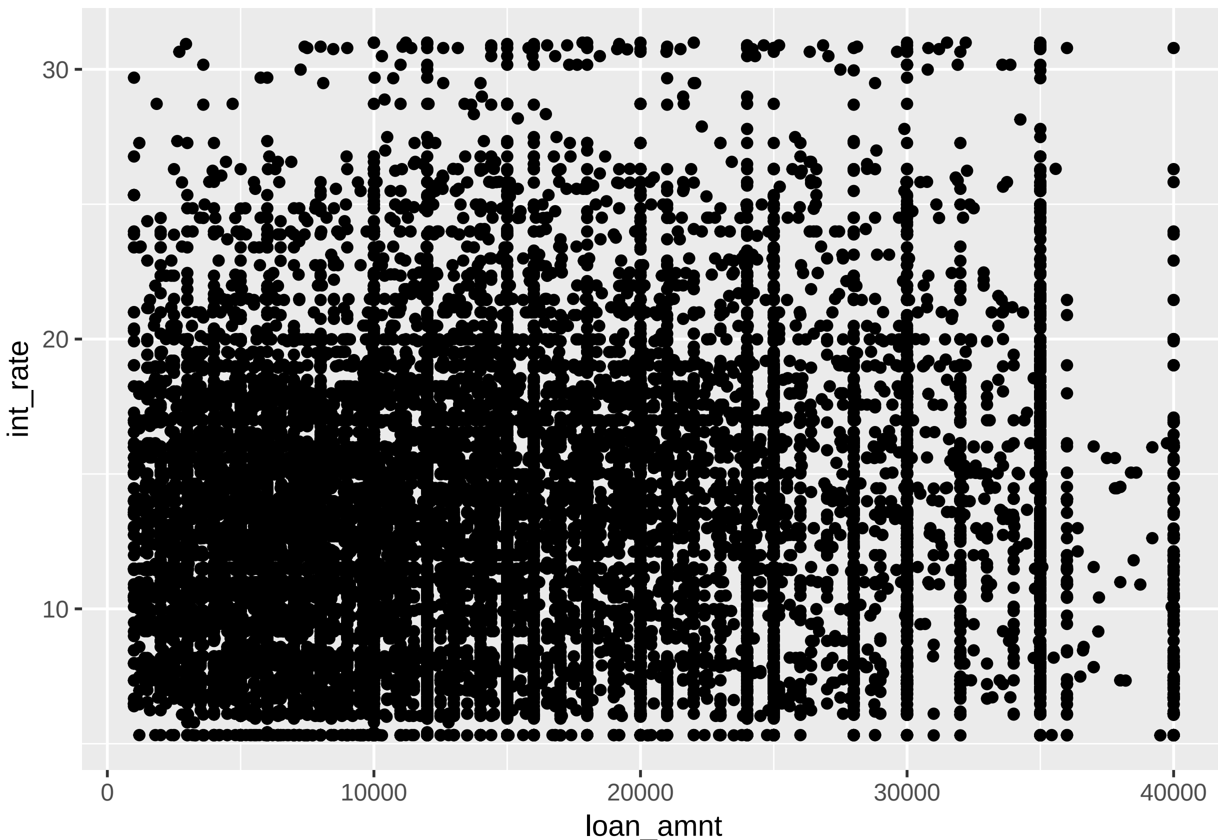
```r
# Creating a separate dataset_reg for regression
dataset_reg <- dataset
```

Using one of the approaches for model selection discussed in class, reduce the number of predictors. For interpretability reasons, start with approaches that conserve the original predictor space. If any useful significant subset is possible, use a base transformation.

Compute the correlation matrix for the selected set of predictors and the output variable, if useful, also using graphical representation.

Data Exploration

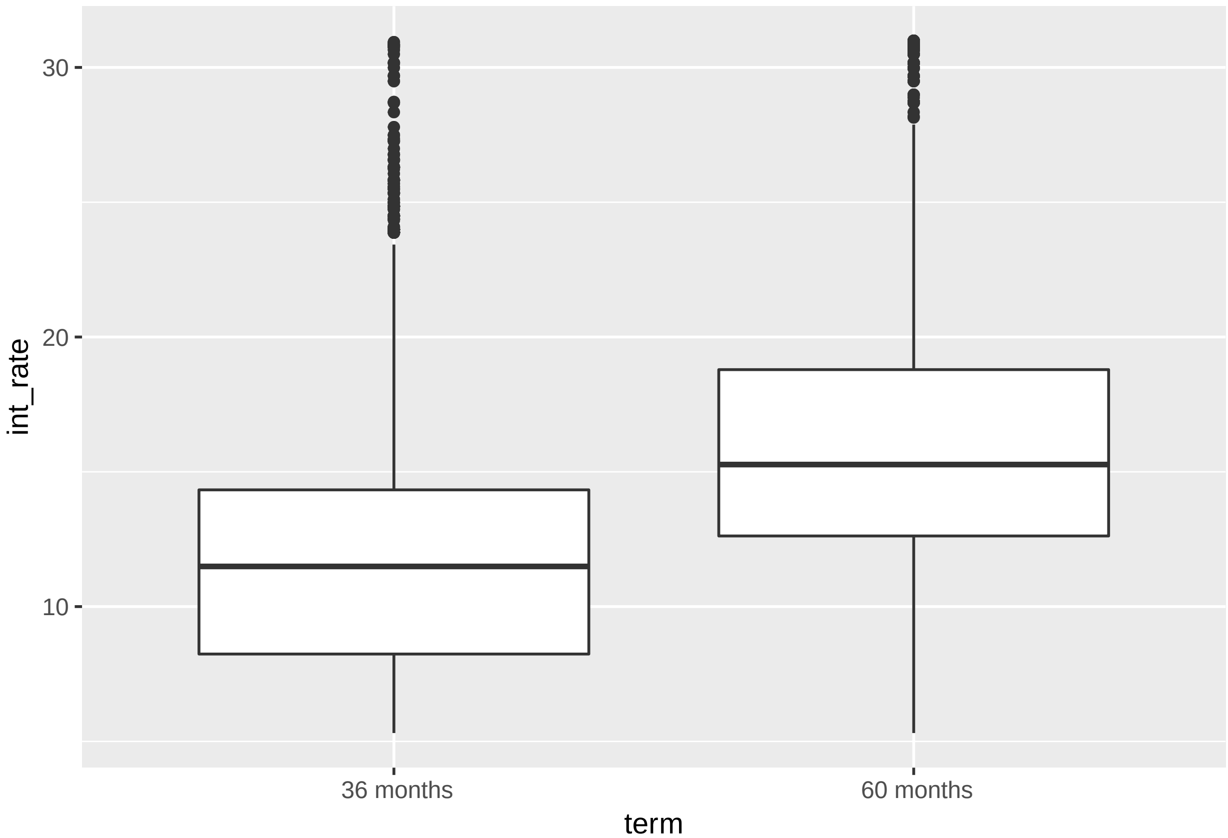
```r
# plotting the loan amount vs the interest rate
{ggplot(data = dataset_reg, mapping = aes(loan_amnt, int_rate)) +
  geom_point()}
```




```r
# from this visualisation we can't see any clear correlation.
```

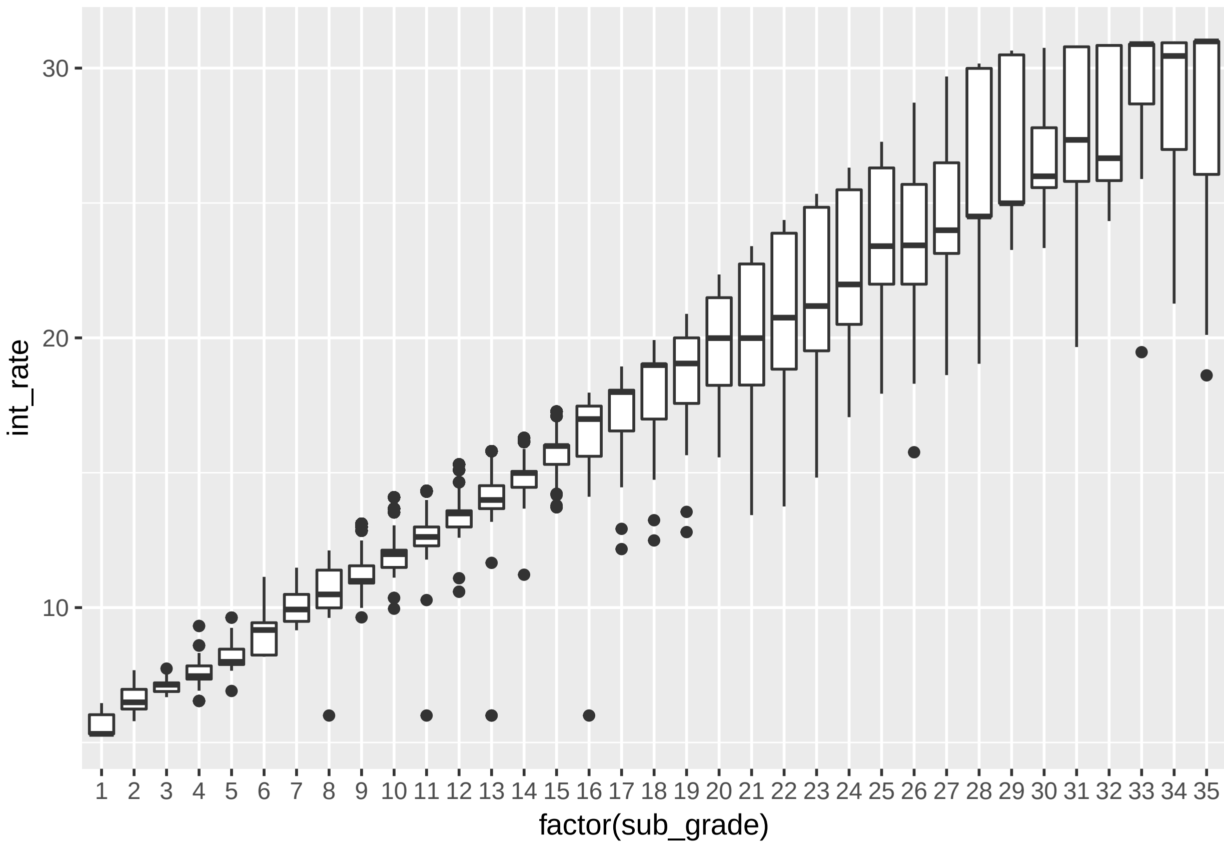
```r
# plotting the loan amount vs the term
{ggplot(data = dataset_reg, mapping = aes(term, int_rate)) +
```

```
  geom_boxplot()}
```



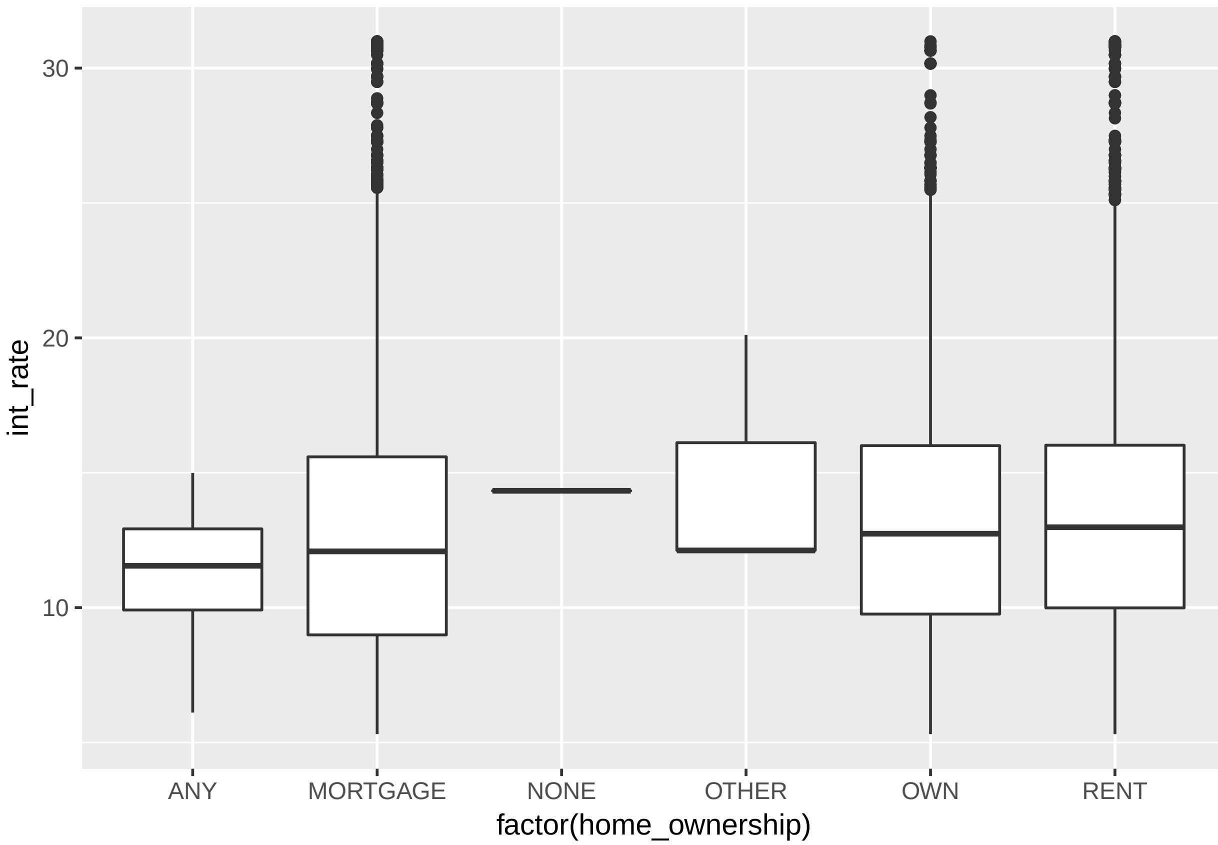

```
# Here we can see that the interest rate is generaly higher for 60 months loans and lower
#for 36 months loans.
```

```
# plotting the loan amount vs the grade
{ggplot(data = dataset_reg, mapping = aes(factor(sub_grade), int_rate), group = 2) +
  geom_boxplot()}
```



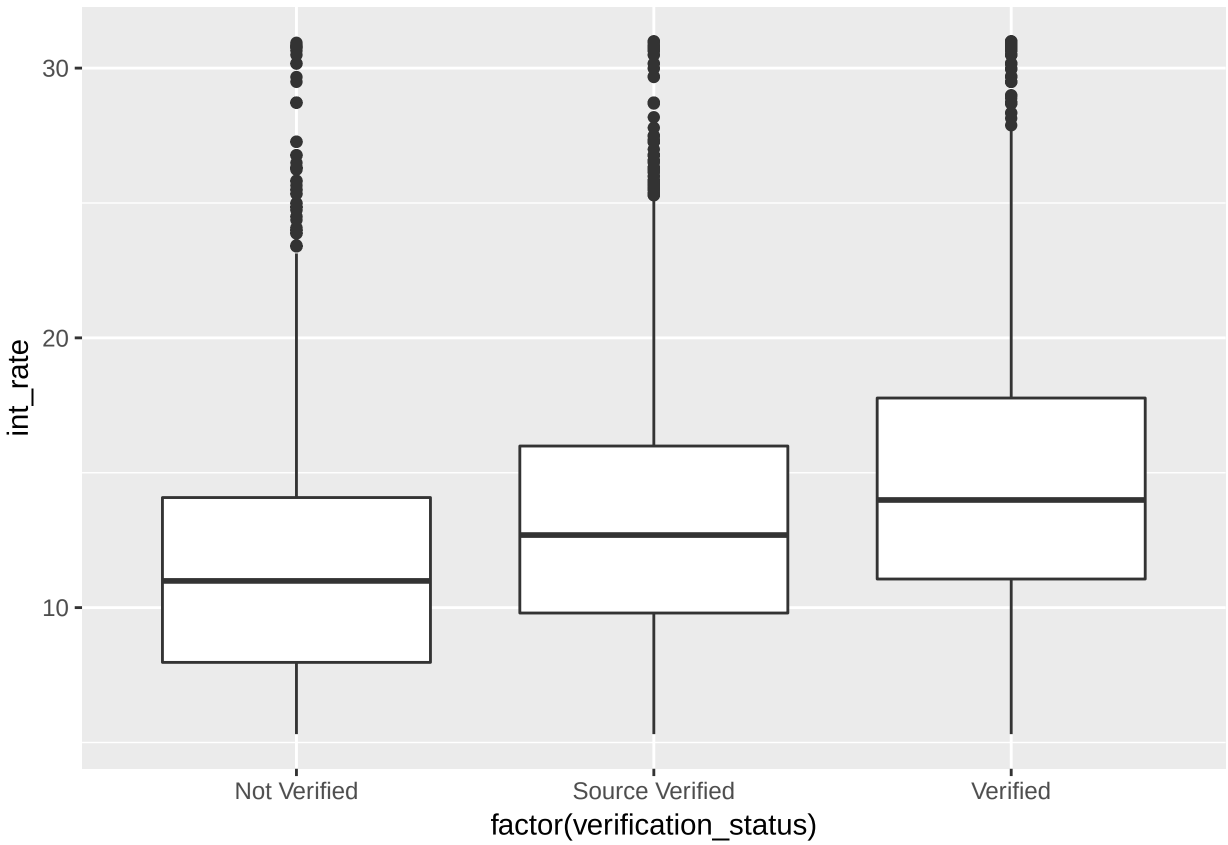

```
# The sub_grade seems to have a strong correlation with the interest rate
```

```
# plotting the loan amount vs home ownership
{ggplot(data = dataset_reg, mapping = aes(factor(home_ownership), int_rate), group = 2) +
  geom_boxplot()}
```




```
# Here we can see that the interest rate is quite close to each other between home_ownership.
```

```
# plotting the loan amount vs verification_status
{ggplot(data = dataset_reg, mapping = aes(factor(verification_status), int_rate), group = 2) +
  geom_boxplot()}
```




```
# The verification_status seems to also have correlation with the interest rate.
```

```
# lm on the whole dataset_reg
lmp.fit=lm(int_rate~.,data = dataset_reg)

summary(lmp.fit)
plot(lmp.fit)

#removing features that do not have at least a 0.001 P value
dataset_reg <- subset(dataset_reg, select = -c(annual_inc, dti, earliest_cr_line,total_pymnt,
total_rec_prncp, total_rec_int, total_rec_late_fee, recoveries, acc_open_past_24mths, avg_cur_bal,
bc_open_to_buy, delinq_amnt, mo_sin_old_rev_tl_op, mo_sin_rcnt_rev_tl_op, mort_acc, num_il_tl, pct_tl_n
tax_liens, total_bal_ex_mort, total_bc_limit, total_il_high_credit_limit, num_actv_bc_tl, mo_sin_rcnt_tl

# Doing a second round to see if further variables can be removed.
summary(lmp.fit)
lmp.fit=lm(int_rate~.,data = dataset_reg)

#removing features that do not have at least a 0.001 P value

dataset_reg <- subset(dataset_reg, select = -c(hardship_reason, inq_last_6mths, num_actv_rev_tl, percen

lmp.fit=lm(int_rate~.,data = dataset_reg)
summary(lmp.fit)

dataset_reg <- subset(dataset_reg, select = -c(bc_util))
```

Main task: ### Compare three different methods to perform regression, using the cross-validation method to compute the best parameters. Consider using some regularization for the parameters shrinkage. Test the train error rate, the CV error rate and the test error.

Stepwise Feature selection

```
# We run a linear model in order to check for which features are relevant in terms of P value.
## Read Model from file or compute it
if (file.exists("fit.glmStepAIC.reg.rds")) {
   fit.bridge.reg <- readRDS("fit.glmStepAIC.reg.rds")
} else {
    fit.glmStepAIC.reg <- train(int_rate~.,
                    data = dataset_reg,
                    method = "glmStepAIC",
                    na.action = na.pass)
}
fit.glmStepAIC.reg
fit.glmStepAIC.reg$finalModel
plot(fit.glmStepAIC.reg$finalModel)
# Here we can see the features that are significant in terms of p value.
# In addition these are sorted in terms of importance. Meaning that the most important
# feature explaining most of the variance will comme first.
```

Limit the dataset by stepwise feature selection

```
# Here we select the features that were deemed important either by the lm or the setpwise
#feature selection for estimating the interest rate.
dataset_reg <- subset(dataset_reg, select = c(int_rate, bc_util, delinq_2yrs, installment, initial_list_
```

Apply the "validation set approach" to reserve a meaningful amount of data for the test phase.

```
# setting a seed for reproducability
set.seed(7)
# We randomly split our dataframe in a train and test set
trainRows = sample(1:nrow(dataset_reg),0.8*nrow(dataset_reg))
testRows = nrow(dataset_reg) - trainRows
train.data.reg = dataset_reg[trainRows,]
test.data.reg = dataset_reg[-trainRows,]
```

```
# Here we setup k-Fold Cross Validation: we do 10 folds and repeat the proceedure 3 times.
control.reg <- trainControl(method = "repeatedcv", number = 10, repeats = 3)
```

```
# We will use a linear model again but this time for predicting the interest_rate.
## Read Model from file or compute it
if (file.exists("fit.lm.reg.rds")) {
   fit.lm.reg <- readRDS("fit.lm.reg.rds")
} else {
  fit.lm.reg <- train(int_rate~.,
                      data = train.data.reg,
                      trControl = control.reg,
                      method = "lm",
                      na.action = na.pass,
                      preProc = c("zv","center","scale"))
}
fit.lm.reg
```

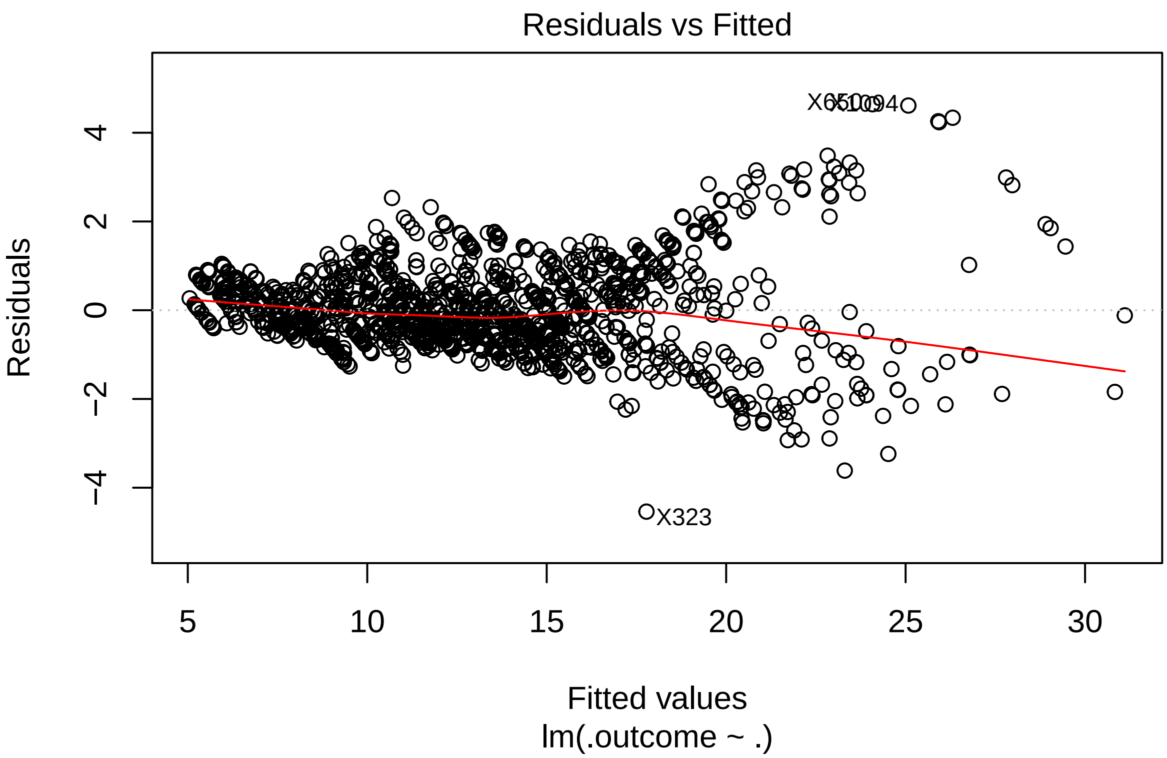
```
## Linear Regression
##
## 1600 samples
##    7 predictor
##
## Pre-processing: centered (13), scaled (13), remove (1)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 1440, 1440, 1439, 1440, 1440, 1440, ...
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.9786347  0.9562666  0.7197813
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

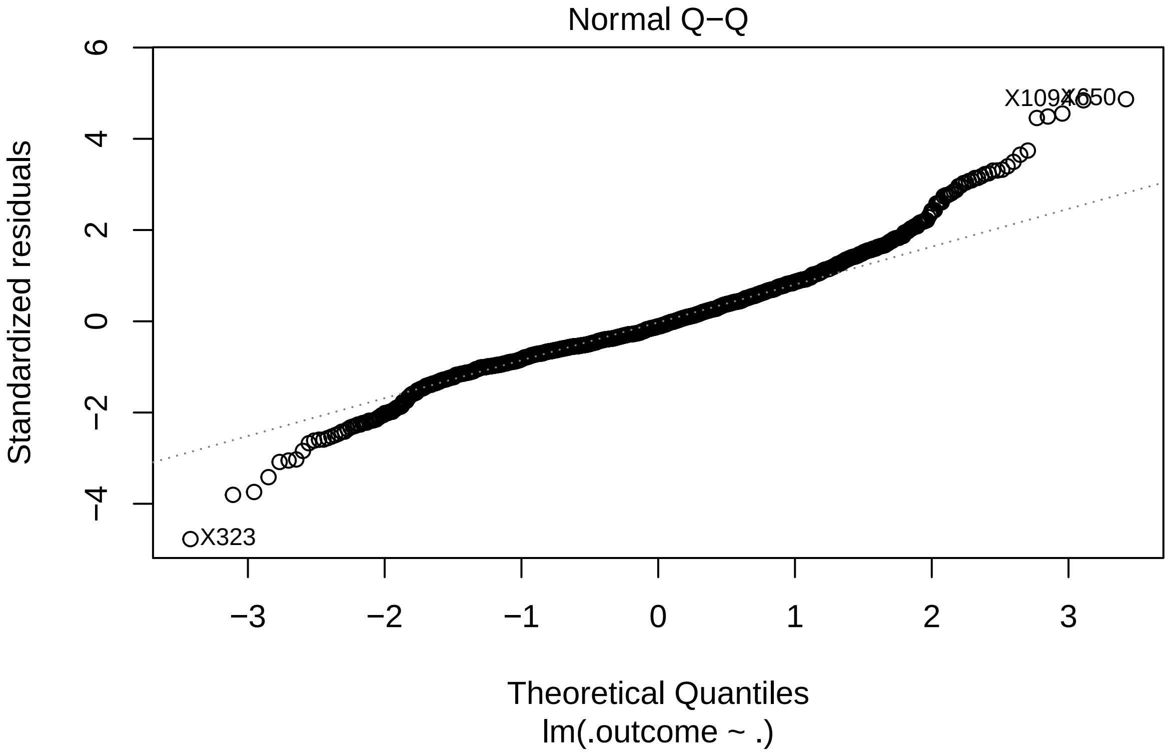
```r
fit.lm.reg$finalModel
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Coefficients:
##                                                         (Intercept)
##                                                          12.9413125
##                                                             bc_util
##                                                           0.0574986
##                                                         delinq_2yrs
##                                                          -0.0488102
##                                                         installment
##                                                           0.0003816
##                                                initial_list_statusw
##                                                          -0.0616466
##                                                  loan_statusCurrent
##                                                           0.2977986
## `loan_statusDoes not meet the credit policy. Status:Charged Off`
##                                                          -0.0908909
##  `loan_statusDoes not meet the credit policy. Status:Fully Paid`
##                                                          -0.1262483
##                                              `loan_statusFully Paid`
##                                                           0.1336417
##                                         `loan_statusIn Grace Period`
##                                                           0.0274737
##                                       `loan_statusLate (16-30 days)`
##                                                           0.0161922
##                                      `loan_statusLate (31-120 days)`
##                                                           0.0557831
##                                                          revol_util
##                                                          -0.0591781
##                                                           sub_grade
##                                                           4.5862552
```

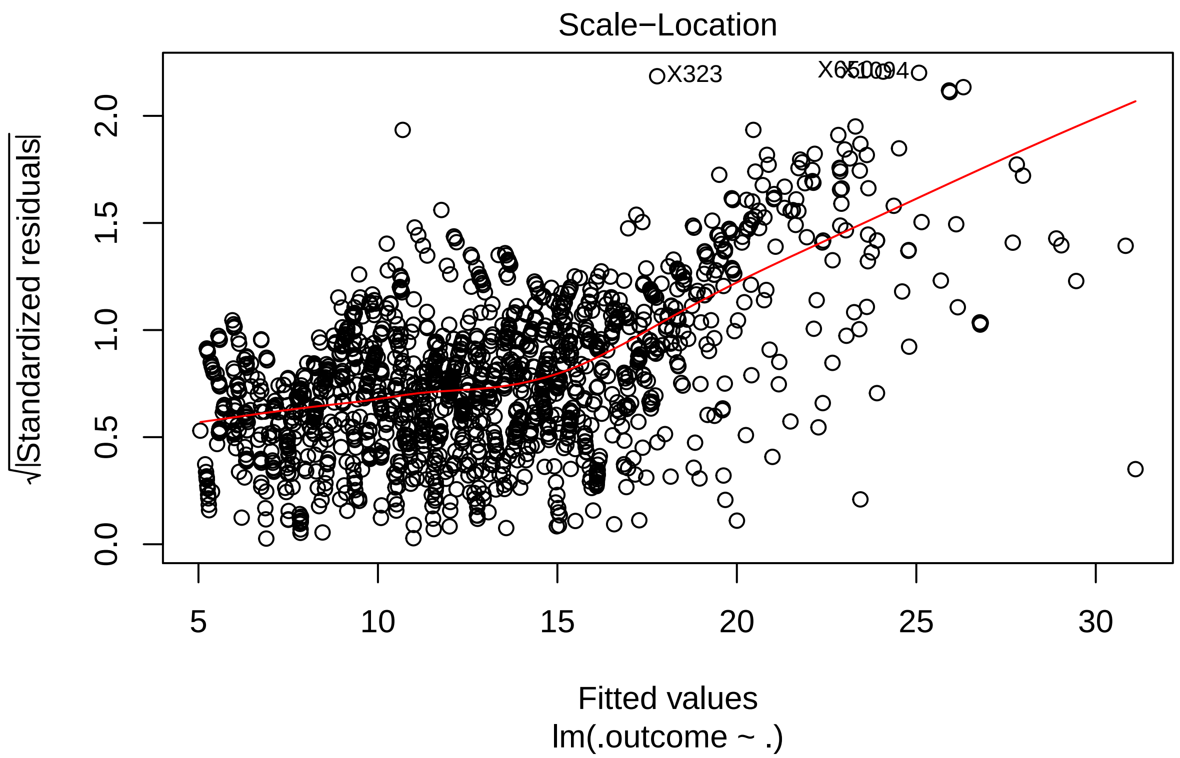
```r
plot(fit.lm.reg$finalModel)
```

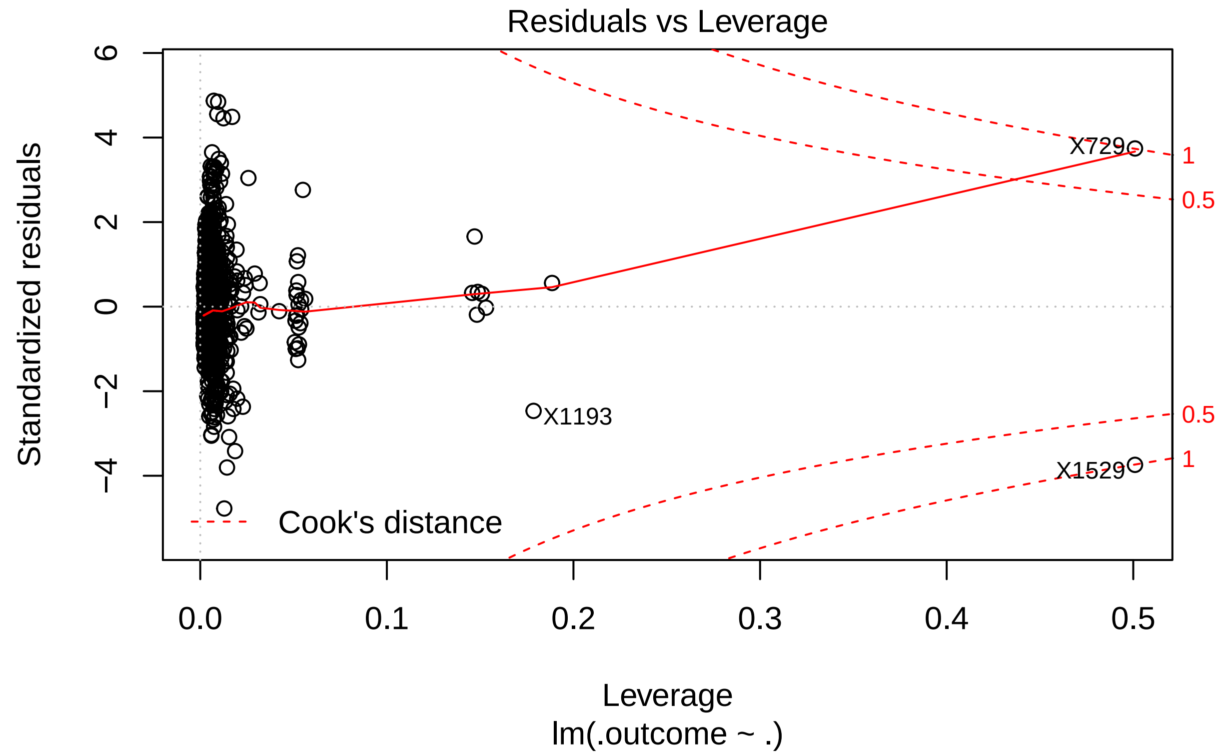
```
## Warning: not plotting observations with leverage one:
##   913, 1377
```






```
## Warning: not plotting observations with leverage one:
##   913, 1377
```





Fit the a extreme gradient boosting model.

```
# We will use an extreme gradient boosting for predicting the interest_rate.
## Read Model from file or compute it
if (file.exists("fit.egb.reg.rds")) {
   fit.egb.reg <- readRDS("fit.egb.reg.rds")
} else {
```

```r
    fit.egb.reg = train(int_rate~.,
                    data = train.data.reg,
                    trControl = control.reg,
                    method = "xgbTree"
    )
}
```

Fit the random forest model.

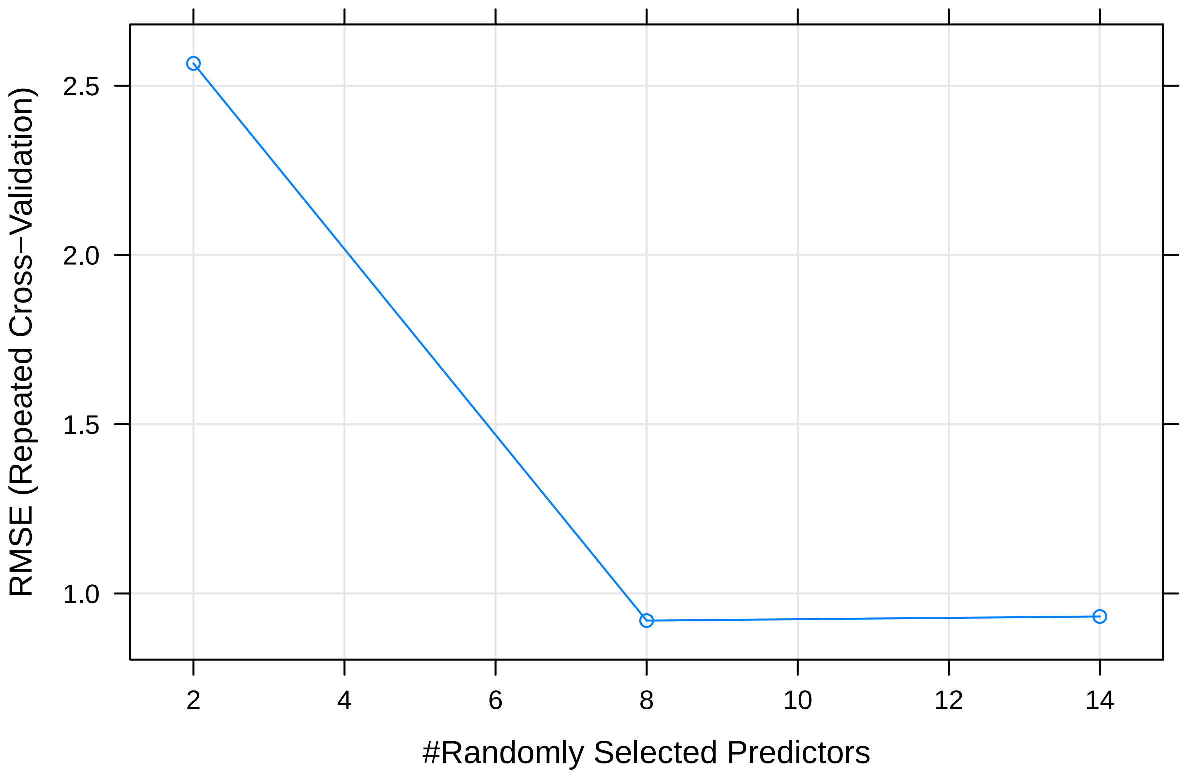
```r
# We will use a Random Forestg for predicting the interest_rate.
## Read Model from file or compute it
if (file.exists("fit.RF.reg.rds")) {
   fit.RF.reg <- readRDS("fit.RF.reg.rds")
} else {
   fit.RF.reg <- train(int_rate~.,
                    data = train.data.reg,
                    trControl = control.reg,
                    method = "parRF",
                    na.action = na.pass)
}

fit.RF.reg
```

```
## Parallel Random Forest
##
## 10000 samples
##     7 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 8999, 9000, 8999, 9001, 9000, 9001, ...
## Resampling results across tuning parameters:
##
##   mtry  RMSE       Rsquared   MAE
##    2    2.5657588  0.8720705  1.8120652
##    8    0.9199010  0.9640665  0.6358753
##   14    0.9321221  0.9630738  0.6342602
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was mtry = 8.
```

```r
fit.RF.reg$finalModel
```

```
##
## Call:
##  randomForest(x = "x", y = "y", ntree = 125, mtry = 8)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 8
```

```r
plot(fit.RF.reg)
```




```r
# MAPE LM TEST
mape_lm_test <- MAPE(predict(fit.lm.reg, test.data.reg), test.data.reg$int_rate)

# MAPE ExtremeGradiantBoosting TEST
mape_egb_test <- MAPE(predict(fit.egb.reg, test.data.reg), test.data.reg$int_rate)

# MAPE Random Forest TEST
mape_rf_test <- MAPE(predict(fit.RF.reg, test.data.reg), test.data.reg$int_rate)

test_mapes <- data.frame("LM" = mape_lm_test, "egb"= mape_egb_test, "Random Forest" = mape_rf_test)
```

```r
# MAPE LM train
mape_lm_train <- MAPE(predict(fit.lm.reg, train.data.reg), train.data.reg$int_rate)

# MAPE ExtremGradiantBoosting train
mape_egb_train <- MAPE(predict(fit.egb.reg, train.data.reg), train.data.reg$int_rate)

# MAPE Random Forest train
mape_rf_train <- MAPE(predict(fit.RF.reg, train.data.reg), train.data.reg$int_rate)

train_mapes <- data.frame("LM" = mape_lm_train, "egb"= mape_egb_train, "Random Forest" = mape_rf_train)
```

```r
# Grouping Test and Train MAPEs
mapes <- union(test_mapes, train_mapes)
row.names(mapes) <- c("test", "train")
print(mapes)
```

```
##                LM        egb Random.Forest
## test  0.05526587 0.05158806    0.03628801
## train 0.05299391 0.04419674    0.03419513
```

```
#As we can see from the table, Random Forest provides us with the lowest value of MAPE
#for both train and test sets, so we assume that Random Forest is the most accurate
#model from those being used here to predict interest rate.
```

Part 2 - Classification Analysis

Our goal in the second part of the assignment is to predict if a new customer will be able to fully pay back their loans using a classification method. Thus, we concentrate on the "concluded lends" in the data set, i.e., on all lends whose loan_status is not Current.

Preparatory tasks

We filter out all observations with loan_status == Current. For the remaining observations, we check if the loan_status is "Fully Paid". If not, change the value of loan_status to "DEFAULTED".

```
# set dataset as data.table::datatable
setDT(dataset)

# filter out all observations with loan_status == Current and storing it in a separate set
dataset_cla <- dataset[loan_status != 'Current']

# change all the loan status that are not "Fully Paid" to 1
dataset_cla$defaulted[dataset_cla$loan_status != "Fully Paid"] <- 1

# change level of defaulted to 1 and 0
levels(dataset_cla$defaulted) = c(1, 0)

# Change all the defaulted values that aren't "Default" to 0
dataset_cla$defaulted[is.na(dataset_cla$defaulted)] <- 0

# remove origin variable, because defaulted is relevant now
dataset_cla$loan_status <- NULL

# set defaulted as factor
dataset_cla$defaulted <- as.factor(dataset_cla$defaulted)

# confirm steps below, by checking results
table(dataset_cla$defaulted)
```

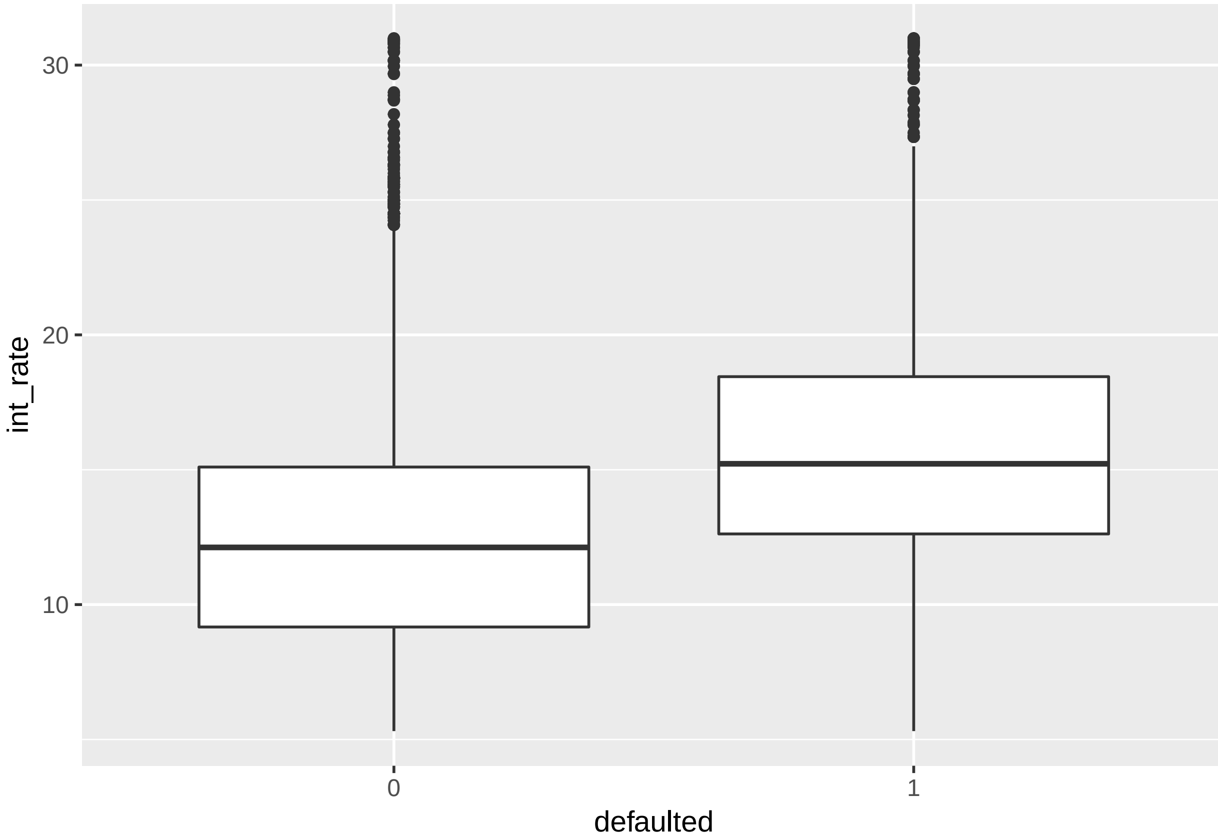
```
##
##    0    1
## 9205 2670
```

Create a validation set.

```
# setting a seed for reproducability
set.seed(7)

# random split into  train and test set, with a ratio of 20:80
trainIndex <- sample(1:nrow(dataset_cla),0.8*nrow(dataset_cla))

train.data.cla <- dataset_cla[trainIndex,]
test.data.cla  <- dataset_cla[-trainIndex,]
```

Data Exploration

```
{ggplot(data = dataset_cla, mapping = aes(defaulted, int_rate)) +
    geom_boxplot()}
```




```
# We can see that credits who defaulted generally have a higher int_rate
```

Main tasks:

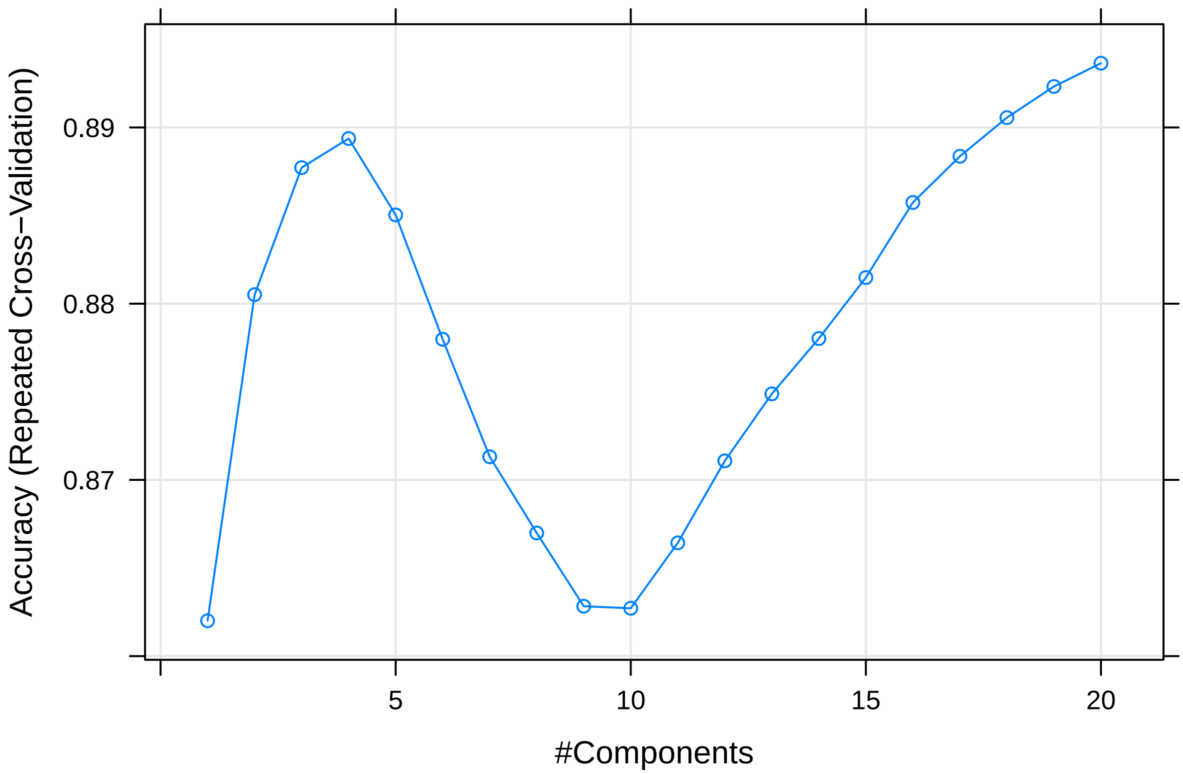
Now we can go over to do the analysis on the dataset. Therefore we use different approaches for feature selection (PLS and PCA). Based on the results, we choose the features and do the classification analysis.

Use Principal Component Analysis for base transformation and then compare it with the Partial Least Squares Regression result. Select the best base with cross validation, using the better of the two approaches.

```
# Compile cross-validation settings
set.seed(100)
myfolds.cla <- createMultiFolds(train.data.cla, k = 5, times = 10)
control.cla <- trainControl("repeatedcv", index = myfolds.cla, selectionFunction = "oneSE")
```

Perform Partial Least Squares Regression with caret package, to have a standardized handling.

```
# Train PLS model
fit.pls.cla <- train(defaulted ~ ., data = train.data.cla,
 method = "pls",
 metric = "Accuracy",
 tuneLength = 20,
 trControl = control.cla,
 preProc = c("zv","center","scale"))

plot(fit.pls.cla)
```

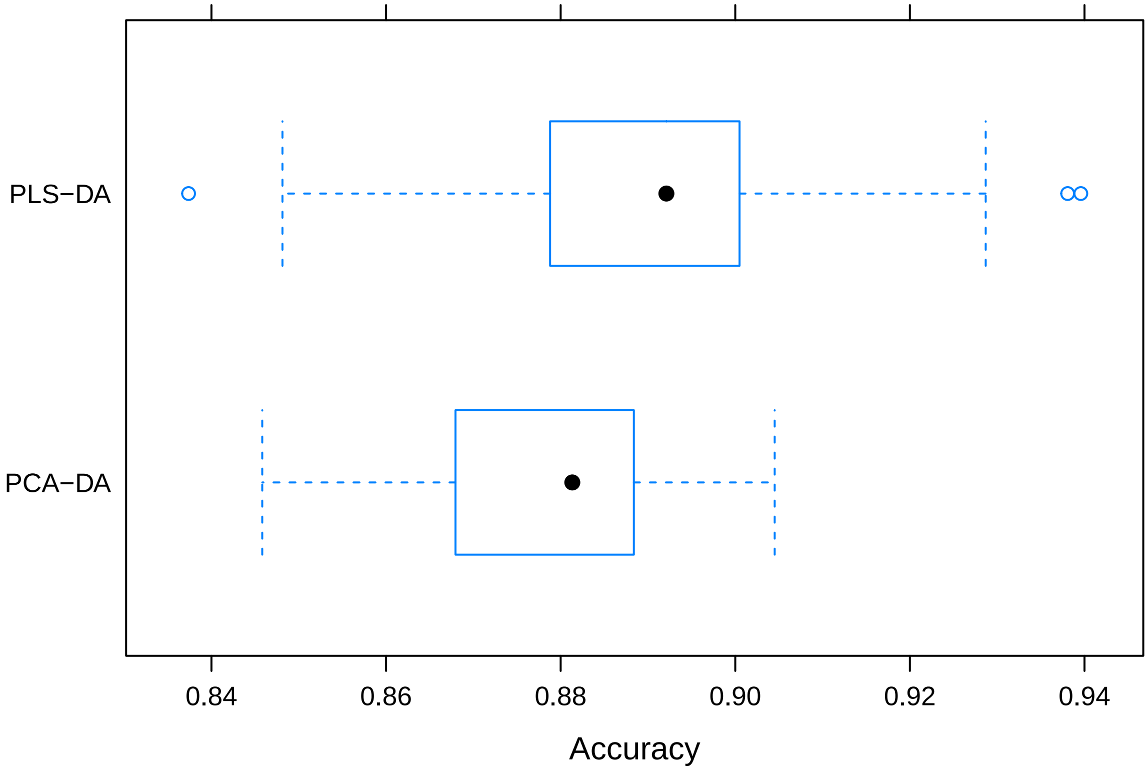


Perform Principal Component Analysis with caret package, to have a standardized handling.

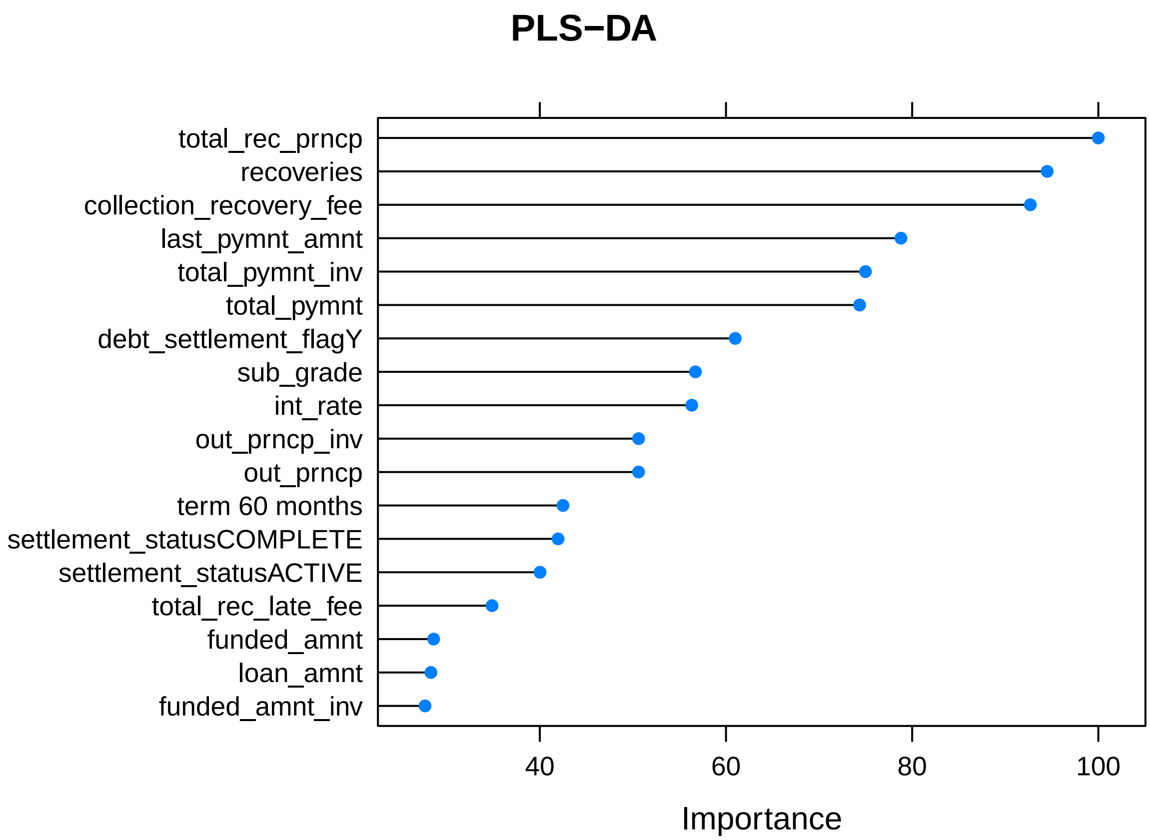
```
# Train PCA-DA
fit.lda.cla <- train(defaulted ~ ., data = train.data.cla,
method = "lda",
metric = "Accuracy",
trControl = control.cla,
preProc = c("zv","center","scale","pca"))
```

Compile models and compare performance

```
models <- resamples(list("PLS-DA" = fit.pls.cla, "PCA-DA" = fit.lda.cla))
bwplot(models, metric = "Accuracy")
```




```r
plot(varImp(fit.pls.cla), fit.pls.cla$bestTune$ncomp, main = "PLS-DA")
```



```r
# limiting the variables of the dataset based on the results from PLS analysis in the step before

train.data.cla <- subset(train.data.cla, select = c(defaulted, total_rec_prncp, recoveries,
collection_recovery_fee, last_pymnt_amnt, total_pymnt_inv, total_pymnt, debt_settlement_flag,
sub_grade, int_rate, out_prncp, out_prncp_inv, settlement_status, term))
test.data.cla  <- subset(test.data.cla, select = c(defaulted, total_rec_prncp, recoveries,
collection_recovery_fee, last_pymnt_amnt, total_pymnt_inv, total_pymnt, debt_settlement_flag,
sub_grade, int_rate, out_prncp, out_prncp_inv, settlement_status, term))
```

Perform the classification using KNN, Logistic Regression, Decision Tree and Random Forest.

Train a model with KNN

```r
## knn
fit.knn.cla  <- train(defaulted ~ .,
                data = train.data.cla,
                method = "knn",
                trControl = control.cla,
                preProc = c("zv","center","scale")
                )
```

Train a model with Logistic Regression

```r
## logistic regression
fit.lreg.cla <-train(defaulted ~ recoveries + collection_recovery_fee + total_rec_prncp
+ last_pymnt_amnt + total_pymnt,
                data = train.data.cla,
                method = "glm",
                trControl = control.cla,
                family=binomial(),
                preProc = c("zv","center","scale")
                )
```

Train a model with Decision Trees

```r
## decision tree
fit.dtree.cla <-train(defaulted ~ recoveries + collection_recovery_fee + total_rec_prncp
+ last_pymnt_amnt + total_pymnt,
               data=train.data.cla,
               method="ctree",
               trControl = control.cla,
               preProc = c("zv","center","scale"))
```

Train a model with Random Forest

```r
## random forest
fit.rf.cla <-train(defaulted ~ recoveries + collection_recovery_fee + total_rec_prncp
+ last_pymnt_amnt + total_pymnt,
               data=train.data.cla,
               method="rf",
               trControl = control.cla,
               preProc = c("zv","center","scale"))
```

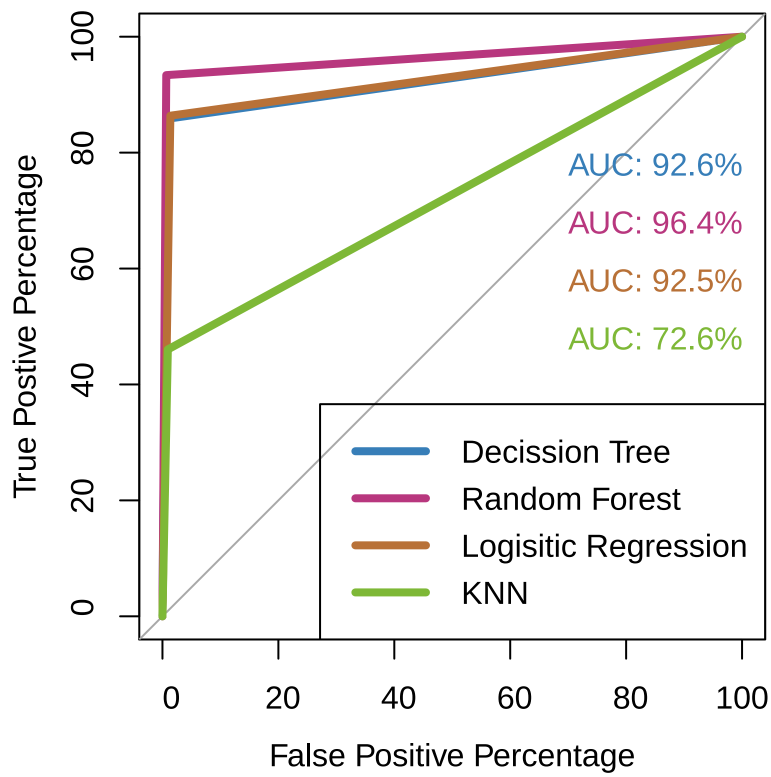
Compare the respective train and test error performances to select one of these approaches.

From the perspective of LC it is very important to identify high-risk individuals. The company might particularly wish to avoid incorrectly classifying an individual who will default. If we take this into account the model with the highest specificity, in this case Logistic Regression, might be the best model.

```
par(pty = "s")
roc(test.data.cla$defaulted, as.numeric(predict(fit.dtree.cla,
test.data.cla)), plot=TRUE, legacy.axes=TRUE,
    percent=TRUE, xlab="False Positive Percentage", ylab="True Postive Percentage", col="#377eb8",
    lwd=4, print.auc = TRUE, print.auc.y=80, print.auc.x=30)
```

```
##
## Call:
## roc.default(response = test.data.cla$defaulted, predictor = as.numeric(predict(fit.dtree.cla,     tes
##
## Data: as.numeric(predict(fit.dtree.cla, test.data.cla)) in 1832 controls (test.data.cla$defaulted 0)
## Area under the curve: 92.55%
```

```
plot.roc(test.data.cla$defaulted, as.numeric(predict(fit.rf.cla,
test.data.cla)), percent=TRUE, col="#b8377e",
         lwd=4, print.auc=TRUE, add=TRUE, print.auc.y=70, print.auc.x=30)

plot.roc(test.data.cla$defaulted, as.numeric(predict(fit.lreg.cla, test.data.cla)),
         percent=TRUE, col="#b87137", lwd=4, print.auc=TRUE, add=TRUE, print.auc.y=60, print.auc.x=30)

plot.roc(test.data.cla$defaulted, as.numeric(predict(fit.knn.cla, test.data.cla)),
         percent=TRUE, col="#7eb837", lwd=4, print.auc=TRUE, add=TRUE, print.auc.y=50, print.auc.x=30)

    legend("bottomright", legend=c("Decission Tree",
    "Random Forest", "Logisitic Regression", "KNN"),
    col=c("#377eb8", "#b8377e", "#b87137", "#7eb837"), lwd=4)
```



Perform the prediction on the validation set and compute the confusion matrix.

```r
#Confusion Matrix KNN
confusionMatrix( predict(fit.knn.cla, test.data.cla), test.data.cla$defaulted)    #Test
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1816  293
##          1   16  250
##
##                Accuracy : 0.8699
##                  95% CI : (0.8557, 0.8832)
##     No Information Rate : 0.7714
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5505
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9913
##             Specificity : 0.4604
##          Pos Pred Value : 0.8611
##          Neg Pred Value : 0.9398
##              Prevalence : 0.7714
##          Detection Rate : 0.7646
##    Detection Prevalence : 0.8880
##       Balanced Accuracy : 0.7258
##
##        'Positive' Class : 0
##
```

```r
#Confusion Matrix Logistic Regression
confusionMatrix( predict(fit.lreg.cla, test.data.cla), test.data.cla$defaulted)   #Test
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1807   74
##          1   25  469
##
##                Accuracy : 0.9583
##                  95% CI : (0.9495, 0.966)
##     No Information Rate : 0.7714
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8779
##
##  Mcnemar's Test P-Value : 1.406e-06
##
##             Sensitivity : 0.9864
##             Specificity : 0.8637
##          Pos Pred Value : 0.9607
```

```
##          Neg Pred Value : 0.9494
##              Prevalence : 0.7714
##          Detection Rate : 0.7608
##    Detection Prevalence : 0.7920
##       Balanced Accuracy : 0.9250
##
##        'Positive' Class : 0
##
```

```r
#Confusion Matrix Decision Trees
confusionMatrix( predict(fit.dtree.cla, test.data.cla), test.data.cla$defaulted)  #Test
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1819   77
##          1   13  466
##
##                Accuracy : 0.9621
##                  95% CI : (0.9536, 0.9694)
##     No Information Rate : 0.7714
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8879
##
##  Mcnemar's Test P-Value : 3.12e-11
##
##             Sensitivity : 0.9929
##             Specificity : 0.8582
##          Pos Pred Value : 0.9594
##          Neg Pred Value : 0.9729
##              Prevalence : 0.7714
##          Detection Rate : 0.7659
##    Detection Prevalence : 0.7983
##       Balanced Accuracy : 0.9255
##
##        'Positive' Class : 0
##
```

```r
#Confusion Matrix Random Forest
confusionMatrix( predict(fit.rf.cla, test.data.cla), test.data.cla$defaulted)     #Test
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1820   36
##          1   12  507
##
##                Accuracy : 0.9798
##                  95% CI : (0.9733, 0.9851)
##     No Information Rate : 0.7714
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9418
```

```
##
##  Mcnemar's Test P-Value : 0.0009009
##
##             Sensitivity : 0.9934
##             Specificity : 0.9337
##          Pos Pred Value : 0.9806
##          Neg Pred Value : 0.9769
##              Prevalence : 0.7714
##          Detection Rate : 0.7663
##    Detection Prevalence : 0.7815
##       Balanced Accuracy : 0.9636
##
##        'Positive' Class : 0
##
```

Conceptually comparison of our approach with a solution existing for this problem

We will compare our approach to the following one: Forecasting Credit Default Probability Author: Matthew Ludwig Date: 11 May 2017 URL: https://rstudio-pubs-static.s3.amazonaws.com/275340_24f229732ac04bf182ccae5fffdfb47a.html

Replacing missing data: In our case missing data for int_rate was simply removed, other variables that did have missing data, we treated as follows: - for numericals values we dummy coded them with a '-1' in the approach. - for factors we replaced missing values with 'unknown'

In the approach we are comparing the int_rate used a similar approach named "Missing Not At Random". Which basically replaced the missing int_rate value by 'Missing'

Outliers: We kept all the outliers and did not try to get these out of our data. In the approach we are comparing, outliers were removed.

Sampling:

We first started with a very small subset (1000 rows) to get ahold if the models we were running were doing so correctly. In our final approach we are using 20000 rows as using any more is too computationaly intensive for the hardware we have.

In the approach we are comparing to the whole dataset is of 29092 row. 20000 are used for training and 9092 are used from testing.

Parameter selection To find out which parameters were useful, we used a logistic regression and a stepwise regression.

This helped us to select a handful of predictors that were deemed relevant for our models based upon the p-value. We can see that this is the same approach that was taken by Matthew Ludwig.

Testing:

We used a crossvalidation set approach, and splitted the data in a ratio of 80/20(train/test).

Model:

We used various models to find which fit better for clasifying, namely: - knn - random forest - logistic regression - descision tree

The approach we are comparing to the used one model which is as follows:

"Final_Model <- glm(loan_status ~ loan_amnt + grade + annual_inc + int_bin + emp_bin, family ="binomial", data = training)" # Summary

We first generated a subset of the original lending-club-loan data and saved that file.

With the new subset we started with exploratory data analysis, based on correlation matrix and, boxplots and plots we could identify important variables and exclude correlating ones. Based on the forgoing data exploration and checking the DataDescription we limited the amount of features. For case of simplicity and available computation power we further had to limit the features once again.

The toplevel approach for the default prediction refers to the CRIPS for Data Mining https://en.wikipedia.org/wiki/Cross-industry_standard_process_for_data_mining and is similar to most of the approaches found on the internet. After a general Data Preparation that is identical for the regression and classifiction task, we did the specific classification data preparation, as changing dependent variable to a 2-factor (1,0) variable. Then the validation set approach was used to split the data into train & test data, to generally improve the model performance. With the Principal Component Analysis & Partial Least Squares we limited the amount of features to consider. Furthermore four different models were used (KNN, Logistic Regression, Decision Trees, Random Forest) and the performance compared with a ROC curve.

Nevertheless, the analysis shows that it's helpful with machine learning techniques to predict the interest rate and probability that a loan is defaulted. For us the Random Forest works the best for interest rate prediction while logistic regression has the best results for predicting default status. At the ROC curve, we get the best true positive vs false postive ratio.

Room for improvement

In future work, natural language processing (NLP) could be used to extract informatcion from Loan description or the job title. Feature engineering could be implemented, e. g. the zip_code could be used to determine the unemployment rate. Furthermore, it would be interesting to spend more time on the performance tuning of each model by using different parameters for the fitting.