Score Personal Loan Applicants using R

Guangming Lang

Score Personal Loan Applicants using R

A Step-by-Step Guide to Doing Predictive Analysis Using Logistic Regression and R

Guangming Lang

This book is for sale at http://leanpub.com/scorepersonalloanapplicantsusingr

This version was published on 2015-08-05

Tweet This Book!

Please help Guangming Lang by spreading the word about this book on Twitter!

The suggested hashtag for this book is #machinelearning.

Find out what other people are saying about the book by clicking on this link to search for this hashtag on Twitter:

https://twitter.com/search?q=#machinelearning

Contents

Preface

This book will teach you how to do data analysis and machine learning in R. It does that by showing you the entire process through a case study step by step. You will learn

- how to clean data
- how to do descriptive and exploratory analysis
- how to create nice looking statistical charts
- how to build and interpret logistic regression (or logit) models
- how to do automatic model selection (for logit models) using the best subsets method
- how to run K-fold Cross Validation
- how to backtest a logit model using different performance measures: accuracy, sensitivity, false positive rate, specificity, precision, and the F-measure
- how to draw the ROC curve and calculate the AUC

You will get most out of this book by typing and running the code given in the book. Do NOT just copy and paste. Type the code. This will help you become a better programmer. So let's get started!

Introduction

A bank has provided you with a dataset of Unsecured Personal Loans (UPLs) collected over the past two years. A description of all the variables is given in Table 1. A bad customer is someone who has missed three or more payments during the first year of the loan. Identifying bad customers is very important for the bank because the loss from each bad customer is on average five times larger than the profit from a good customer. So even if the number of bad customers in the dataset is relatively small they have a large impact on profits.

Table 1: Definitions of Variables

Variable Name	Definition
purpose	0: standard credit card; 1: balance transfer
age	applicant age (in years)
marital	1 if applicant is married; 0 otherwise
employment	1 employed full time 2 employed part time 3 self employed 4 temporary employment 5 homemaker 6 retired
annual_income	annual gross income
debt_to_income	ratio of existing liabilities to annual income
market_value	market value of property
own_property	1 if applicant is a property owner; 0 otherwise
late_repayments	1 if applicant delayed any debt repayment during the last year; 0 otherwise
repossess	1 if applicant has had private property repossessed in the past; 0 otherwise
conviction	1 if applicant has ever been convicted; 0 otherwise
bankruptcy	1 if applicant has declared bankruptcy; 0 otherwise
unspent_convictions	1 if applicant has unspent convictions; 0 otherwise
credit_applications	number of applications for credit in the past 12 months
credit_line_age	age of longest credit line (in months)
exist_customer	1 if applicant is an existing customer
bad	**1 bad customer** **0 good customer**

Your objective is to conduct a thorough analysis of the data and recommend a model to identify customers with a high risk of being '**bad**'. Here's a list of questions that may help guide your analysis:

- Which variables appear to be highly predictive of **bad** customers?
- Can a model be used to justify the decision to accept an applicant or not? If so does the model agree with common sense? (e.g. do the contributions of the variables in the model make sense; are the rules that arise from the model intuitive)?
- How do you propose to handle the specificity sensitivity trade-off? In particular,
  - What is the maximum proportion of good customers that can be granted loans while ensuring that x% of the bad customers are correctly identified.
  - What is the maximum proportion of the overall population that can be granted loans while ensuring that x% of the bad customers are correctly identified.

In subsequent chapters, I'll walk you through the complete analysis step by step, addressing these questions along the way. But first, let's set up the tools we'll be using for the analysis.

1. Set up

1. Install R[1] and Rstudio[2].
2. Install a set of development tools:
    - On Windows, download and install Rtools[3].
    - On Mac, install the Xcode command line tools[4].
    - On Linux, install the R development package, usually called **r-devel** or **r-base-dev**.
3. Install the following R packages.

```
install.packages("devtools")
devtools::install_github("gmlang/ezplot")
devtools::install_github("gmlang/loans")
install.packages("rJava")
install.packages("glmulti")
install.packages("dplyr")
install.packages("tidyr")
```

[1] http://www.r-project.org
[2] http://www.rstudio.com/products/rstudio/download/
[3] http://cran.r-project.org/bin/windows/Rtools/
[4] https://developer.apple.com/downloads

2. Preliminary Analysis

2.1 Clean Data

Create a directory called *score-loan-applicants* under your home directory. Use it as the project folder that will store all files related with our analysis, which include code, processed data, intermediate results, figures, and etc.

```
proj_path = "~/score-loan-applicants"
dir.create(proj_path, showWarnings=FALSE)
```

Load the `ezplot` and `loans` libraries. The former allows us to make nice looking ggplot2 plots easily. The latter contains the unsecured personal loans (upl) dataset that we'll analyze.

```
library(ezplot)
library(loans)
```

Examine the dataset. We see it contains 7250 observations and 17 variables.

```
str(upl, vec.len=3)
```

```
'data.frame':	7250 obs. of  17 variables:
 $ purpose           : num  0 0 0 1 NA 0 1 0 ...
 $ age               : num  38.3 40.3 21.7 37.5 ...
 $ marital           : num  0 1 0 1 0 0 0 1 ...
 $ employment        : num  1 1 1 3 2 1 5 1 ...
 $ annual_income     : num  225523 93072 66236 45626 ...
 $ debt_to_income    : num  0.393 0.357 0.868 1.574 ...
 $ market_value      : num  1540881 1159186 0 1069064 ...
 $ own_property      : num  1 1 0 1 1 0 0 1 ...
 $ late_repayments   : num  0 0 0 1 1 0 1 0 ...
```

```
$ repossess           : num  0 0 0 0 0 0 0 1 ...
$ conviction          : num  0 0 0 0 0 0 0 0 ...
$ bankruptcy          : num  0 0 0 0 0 0 0 0 ...
$ unspent_convictions: num  1 0 0 0 0 0 0 0 ...
$ credit_applications: num  2 2 3 7 3 4 4 2 ...
$ credit_line_age     : num  77.5 72.8 15.7 6.9 ...
$ exist_customer      : num  0 0 0 0 0 1 0 0 ...
$ bad                 : num  0 0 0 1 1 1 0 0 ...
- attr(*, "codepage")= int 65001
```

All variables are coded as numeric. Some shouldn't be. For example, we know the target variable is in fact binary. So we change it to factor.

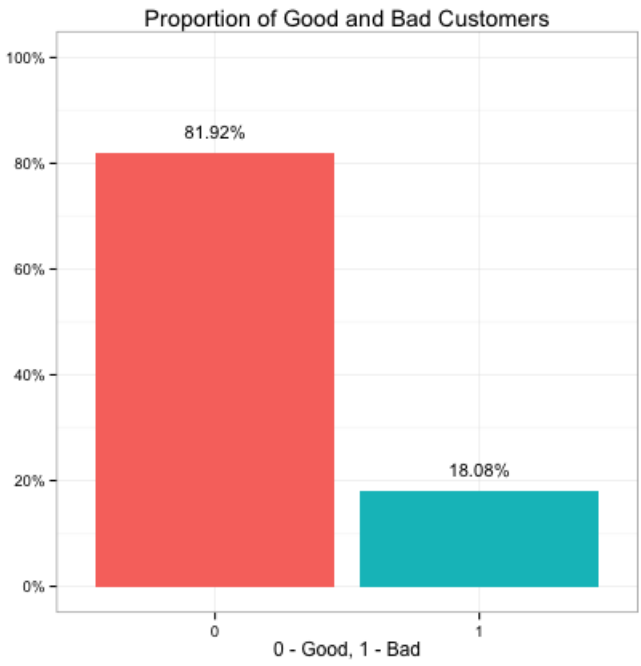
```
upl$bad = as.factor(upl$bad)
```

We can then look at the distribution of the target variable. First of all, we define a function that can be used to calculate the percent of good and bad customers.

```
pct_good_n_bad = function(dat, yvar, xvar = ""){
        # dat: a data frame
        # yvar, xvar: string, names of variables on dat
        if (xvar == "") tbl = data.frame(table(dat[[yvar]]))
        else tbl = data.frame(table(dat[[yvar]][is.na(dat[[xvar]])]))
        tbl$percent = tbl$Freq / sum(tbl$Freq)
        tbl$Freq = NULL
        names(tbl) = c(yvar, "percent")
        tbl
}
```

Next, we use it to calculate the percent of good and bad customers in the upl dataset, and we show the result in a bar chart.

```
tbl = pct_good_n_bad(upl, "bad")
# append a column of label positions to tbl
f = add_bar_label_pos(tbl)
tbl = f("bad", "percent", vpos=0.04)
# draw bar plot
plt = mk_barplot(tbl)
p = plt("bad", "percent", fillby="bad", xlab="0 - Good, 1 - Bad", legend=F,
        main = "Proportion of Good and Bad Customers", barlab="percent",
        barlab_at_top=T, barlab_use_pct=T, barlab_size=4)
p = scale_axis(p, "y", scale="pct", pct_jump=0.2)
print(p)
```



We see that ∼82% of the customers in the upl dataset are good while ∼18% are bad. This imbalanced distribution of the target variable implies that we can't merely use the overall classification accuracy to measure model performance. For example, suppose we build a model, and it gives us an accuracy of 82%. We wouldn't consider it good here because we can achieve the same 82% accuracy without fitting any model. Just simply guess every customer is good. Because we want to build models that can correctly identify the bad customers, we really need to use more granular measures such as sensitivity and specificity to measure model performance.

Having looked at the target variable, now let's turn our attention to the predictors. We need to change the binary predictors to factors. But first, we change them to characters.

```
iv_cat = c("bankruptcy", "purpose", "exist_customer", "unspent_convictions",
           "conviction", "repossess", "own_property", "late_repayments",
           "marital", "employment")
for (var in iv_cat) upl[[var]] = as.character(upl[[var]])
str(upl[, iv_cat], vec.len=3)
```

```
'data.frame':        7250 obs. of  10 variables:
 $ bankruptcy         : chr  "0" "0" "0" ...
 $ purpose            : chr  "0" "0" "0" ...
 $ exist_customer     : chr  "0" "0" "0" ...
 $ unspent_convictions: chr  "1" "0" "0" ...
 $ conviction         : chr  "0" "0" "0" ...
 $ repossess          : chr  "0" "0" "0" ...
 $ own_property       : chr  "1" "1" "0" ...
 $ late_repayments    : chr  "0" "0" "0" ...
 $ marital            : chr  "0" "1" "0" ...
 $ employment         : chr  "1" "1" "1" ...
```

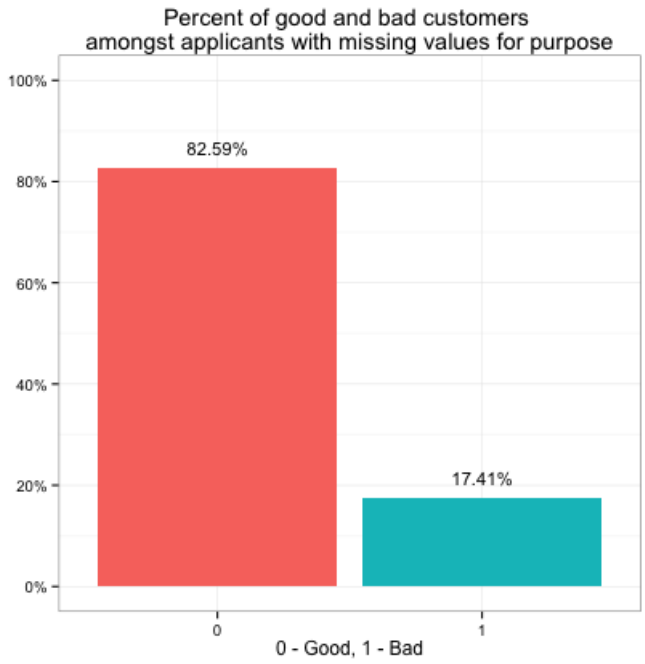
Next, we check which predictors have missing values.

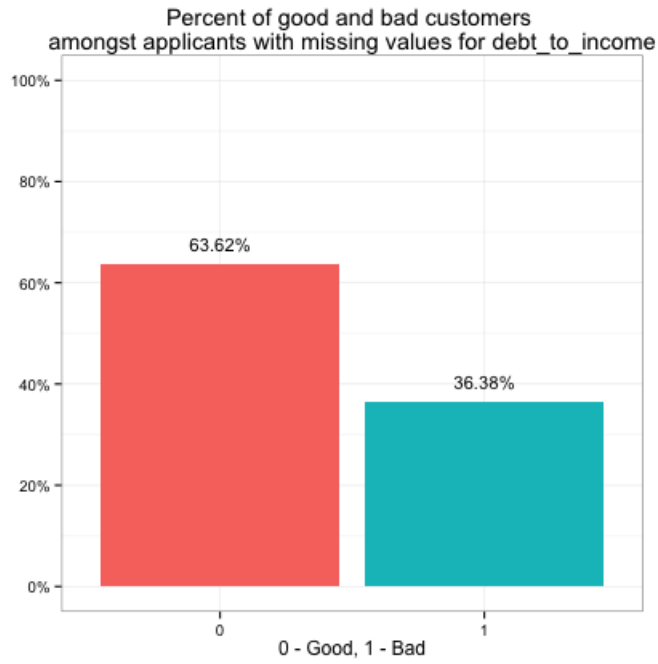
```
n = nrow(upl)
vars = names(upl)
varsNA = pctNA = c()
for (var in vars) {
        cntNA = sum(is.na(upl[[var]]))
        if (cntNA > 0) {
                varsNA = c(varsNA, var)
                pctNA = c(pctNA, cntNA/n)
        }
}
pctNA = paste0(round(pctNA*100, 2), "%")
pct_missing = data.frame(vars = varsNA, percent_missing = pctNA)
print(pct_missing)
```

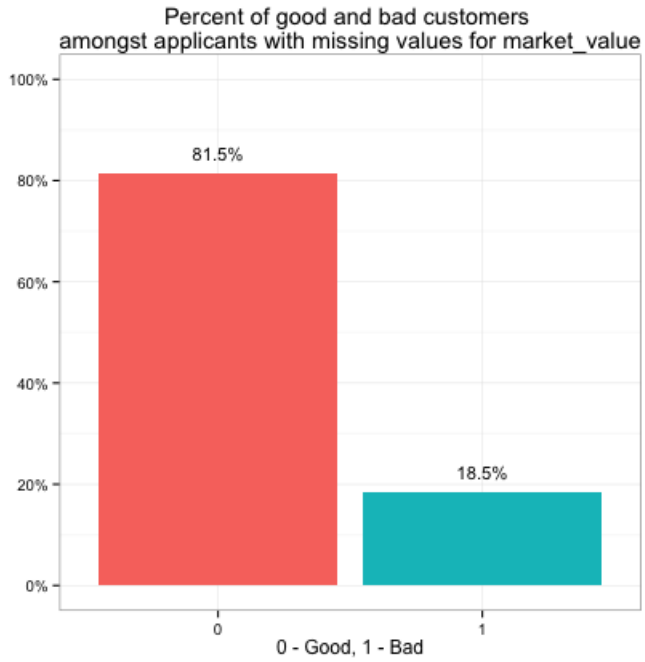
```
            vars percent_missing
1        purpose          15.13%
2 debt_to_income           6.03%
3   market_value           9.17%
4     bankruptcy           3.97%
```

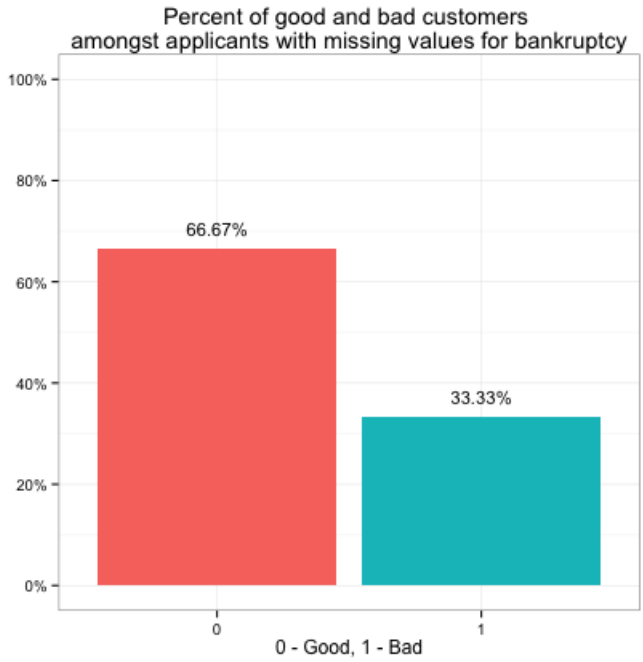
We see **purpose** has more than 15% of its values missing, and **market_value**, **debt_to_income** and **bankruptcy** all have mild missings. Let's explore the relationship between the target variable and the missing values.

```
for (var in varsNA) {
        # calculate the percent of good and bad customers amongst customers with\
 missing values in var
        tbl = pct_good_n_bad(upl, "bad", var)
        # append a column of label positions to tbl
        f = add_bar_label_pos(tbl)
        tbl = f("bad", "percent", vpos=0.04)
        # draw bar plot
        title = paste("Percent of good and bad customers \namongst applicants wi\
th missing values for", var)
        plt = mk_barplot(tbl)
        p = plt("bad", "percent", fillby="bad", xlab="0 - Good, 1 - Bad",
                main = title, legend=F, barlab="percent", barlab_at_top=T,
                barlab_use_pct=T, barlab_size=4)
        p = scale_axis(p, "y", scale="pct", pct_jump=0.2)
        print(p)
}
```









We see the target variable has the same distribution (82% good - 18% bad) amongst customers with missing purpose as amongst all customers. This is also true for market_value. This implies that we may choose to ignore the missing values when looking at the individual effect of purpose or market_value on the target variable. However, the target has a distribution of 67% good vs. 33% bad amongst customers with missing bankruptcy info, which is different from its overall distribution. The same is true for debt_to_income. This implies that we may not ignore the effect of missing

values in bankruptcy or debt_to_income on the target.

Let's perform the following missing value treatments. For purpose and bankruptcy, because they are categorical, we change their missing values to "unkown". For debt_to_income, because it's continuous, we fill its missing values with the median of its non-missing values. For market_value, because it is related with own_property, we first fill its missing values based on the values of own_property. In particular, for customers with own_property = 0, we fill their missing market values with zeros. For customers with own_property = 1, we fill their missing market values with the median of the non-missing values. Note that we use median instead of mean. This is because a few large values in market_value will result a big mean, while the median is more immune to outliers' influence, and hence is a better measure of average in this case.

```
upl$market_value[upl$own_property == 0 & is.na(upl$market_value)] = 0
for (var in varsNA) {
        if (class(upl[[var]]) == "character") {
                print(var)
                upl[[var]][is.na(upl[[var]])] = "unknown"
                print(table(upl[[var]]))
        } else {
                print(var)
                upl[[var]][is.na(upl[[var]])] = median(upl[[var]], na.rm=T)     \

                print(summary(upl[[var]]))
        }
}
```

```
[1] "purpose"
     0      1 unknown
  4290   1863    1097
[1] "debt_to_income"
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.0505  0.3000  0.4400  0.5620  0.6500 17.2000
[1] "market_value"
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      0       0  856000  730000 1290000 2680000
[1] "bankruptcy"
     0      1 unknown
  6924     38     288
```

Now there's no missing values in the data set. We can change the predictors from the character to factors

```
for (var in iv_cat) upl[[var]] = as.factor(upl[[var]])
```

Finally, let's create a *data* subfolder under proj_path and save the cleaned data there. We'll use this cleaned data for descriptive analysis in the next section.

```
data_path = file.path(proj_path, "data")
dir.create(data_path, showWarnings=F)
save(upl, iv_cat, file=file.path(data_path, "cleaned-01.rda"))
```

2.2 Descriptive Analysis

First, let's load the cleaned data.

```
proj_path = "~/score-loan-applicants"
data_path = file.path(proj_path, 'data')
file_path = file.path(data_path, 'cleaned-01.rda')
load(file_path)
```

Next, let's explore the relationships between the categorical predictors and the target. Because the target is binary (bad vs. good), we only need to look at how bad customers are distributed across the different levels of a given categorical predictor. We first create a helper function that takes a categorical predictor as an input parameter and calculates the percent of bad customers for each of its levels.

```
calc_pct_bad = function(dat, var) {
        # dat: a data frame
        # yvar, xvar: string, names of variables on dat
        tbl = table(dat$bad, dat[[var]])
        pct = tbl["1", ] / (tbl["0", ] + tbl["1", ])
        pct = data.frame(pct)
        pct = cbind(row.names(pct), pct)
```

```
        names(pct) = c(var, "pct_bad")
        row.names(pct) = NULL
        pct
}
```

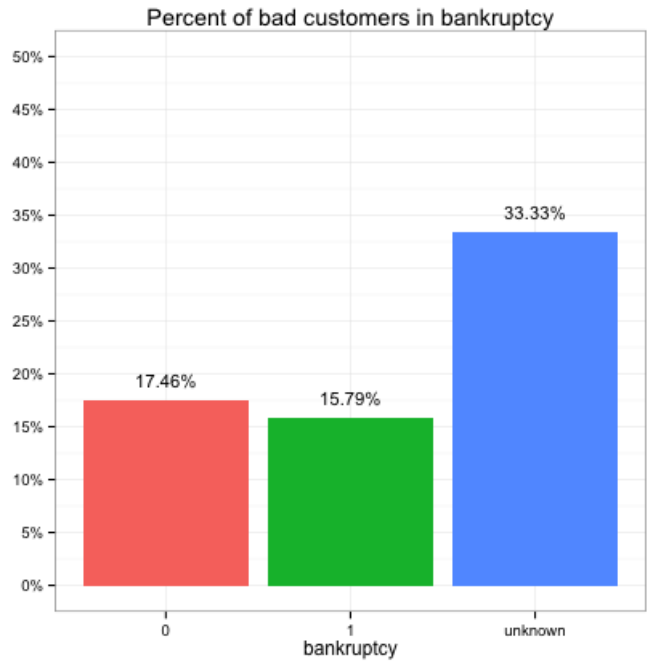
Here's an example of how to use it.

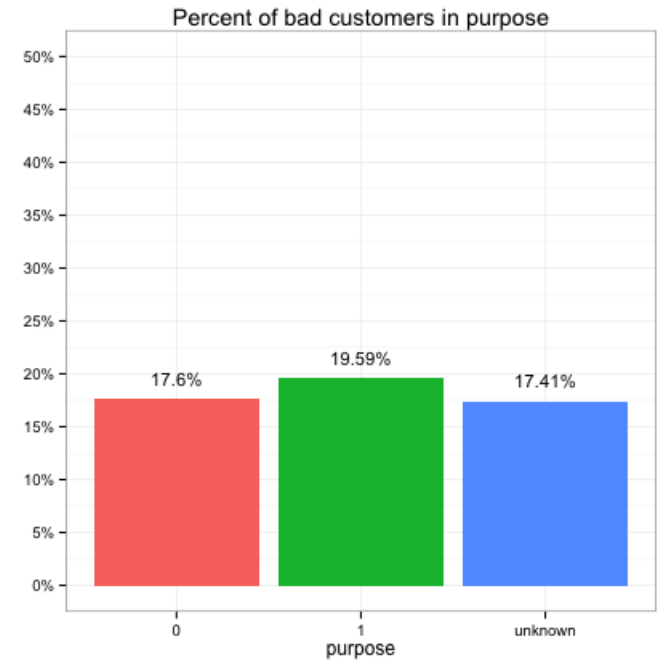
```
calc_pct_bad(upl, iv_cat[1])
```

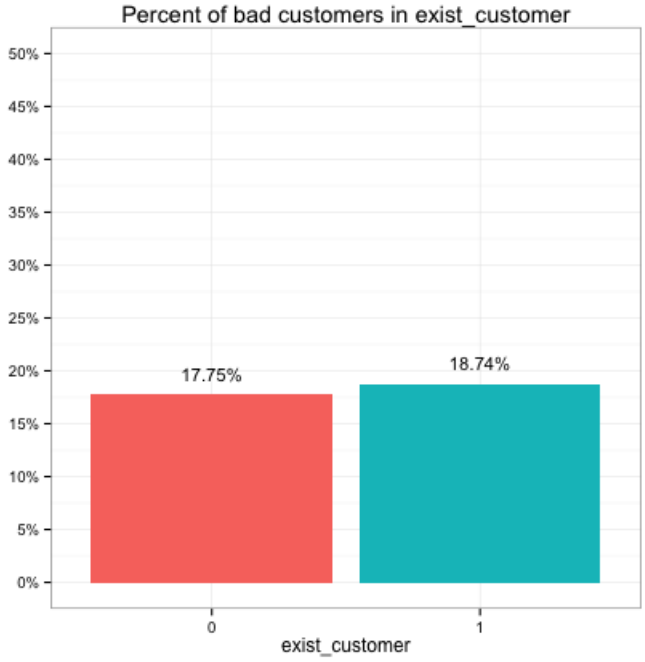
```
  bankruptcy pct_bad
1          0 0.17461
2          1 0.15789
3    unknown 0.33333
```

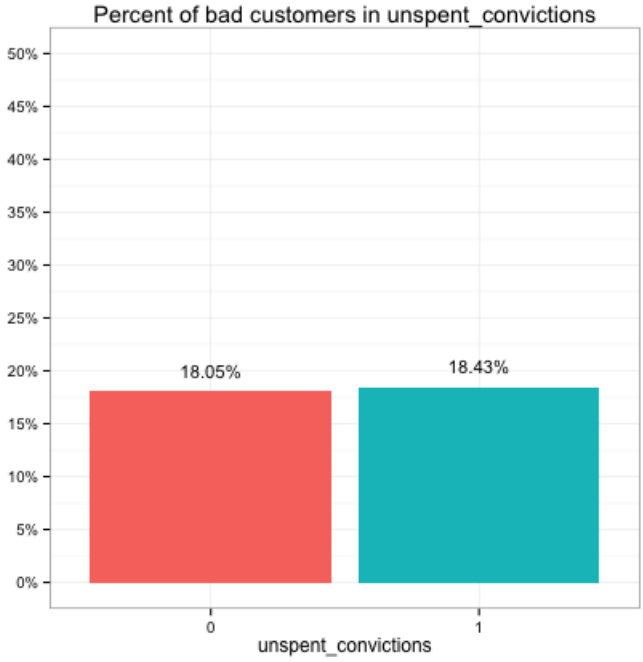
Now we can draw bar chart to display these percentages of bad customers.

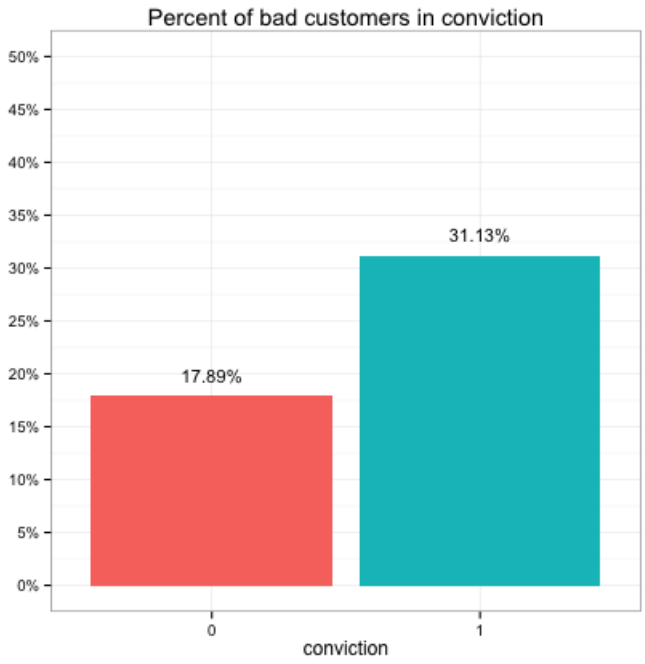
```
for (var in iv_cat) {
        tbl = calc_pct_bad(upl, var)
        # append a column of label positions to tbl
        f = add_bar_label_pos(tbl)
        tbl = f(var, "pct_bad", vpos=0.02)
        # draw bar plot
        plt = mk_barplot(tbl)
        p = plt(var, "pct_bad", fillby=var, xlab=var, legend=F,
                main=paste("Percent of bad customers in", var),
                barlab="pct_bad", barlab_at_top=T, barlab_use_pct=T,
                barlab_size=4)
        p = scale_axis(p, "y", scale="pct", pct_max=0.5, pct_jump=0.05)
        print(p)
}
```

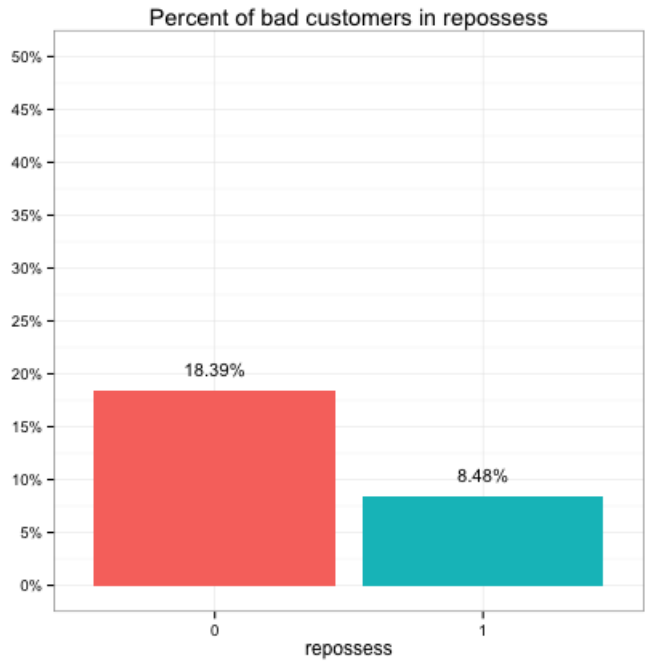

Percent of bad customers in bankruptcy
50%
45%
40%
35%
30%
25%
20%
15%
10%
5%
0%
17.46%
15.79%
33.33%
0
1
unknown
bankruptcy

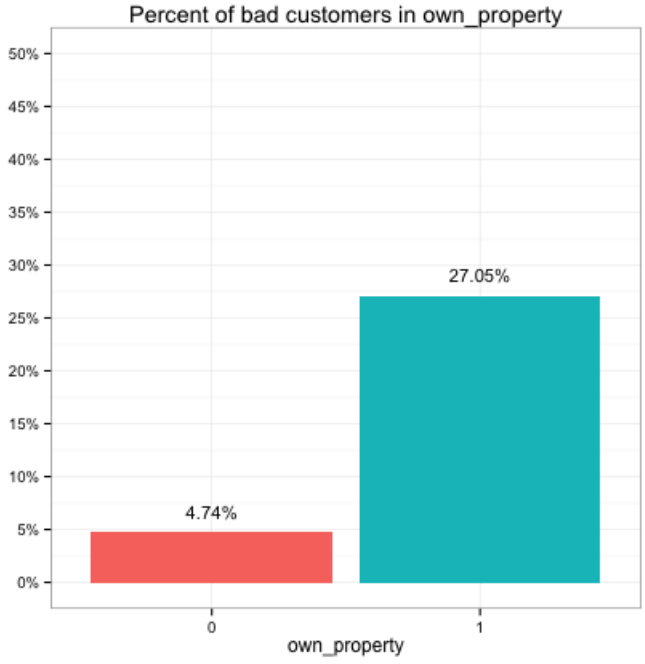

Percent of bad customers in purpose
50%
45%
40%
35%
30%
25%
20%
15%
10%
5%
0%
17.6%
19.59%
17.41%
0
1
unknown
purpose

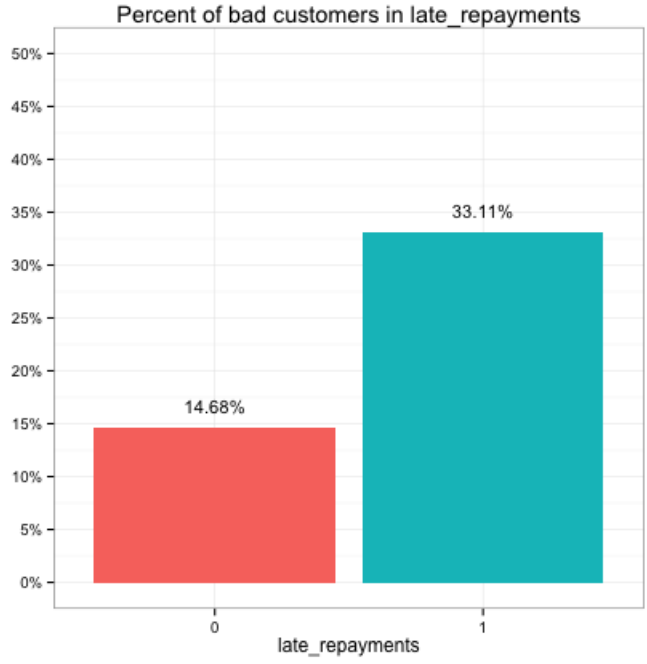

Percent of bad customers in exist_customer
50%
45%
40%
35%
30%
25%
20%
15%
10%
5%
0%
17.75%
18.74%
0
1
exist_customer

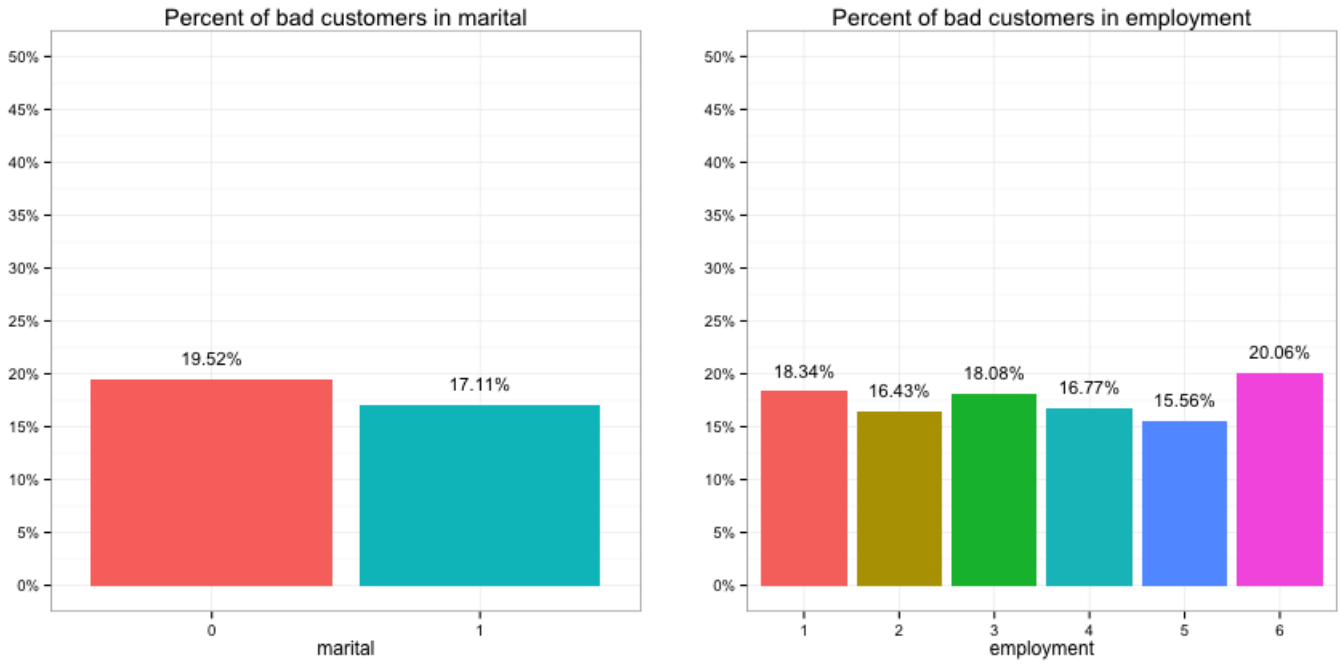

Percent of bad customers in unspent_convictions
50%
45%
40%
35%
30%
25%
20%
15%
10%
5%
0%
18.05%
18.43%
0
1
unspent_convictions


Percent of bad customers in conviction
50%
45%
40%
35%
30%
25%
20%
15%
10%
5%
0%
31.13%
17.89%
0
1
conviction


Percent of bad customers in repossess
50%
45%
40%
35%
30%
25%
20%
15%
10%
5%
0%
18.39%
8.48%
0
1
repossess


Percent of bad customers in own_property
50%
45%
40%
35%
30%
25%
20%
15%
10%
5%
0%
27.05%
4.74%
0
1
own_property


Percent of bad customers in late_repayments
50%
45%
40%
35%
30%
25%
20%
15%
10%
5%
0%
33.11%
14.68%
0
1
late_repayments

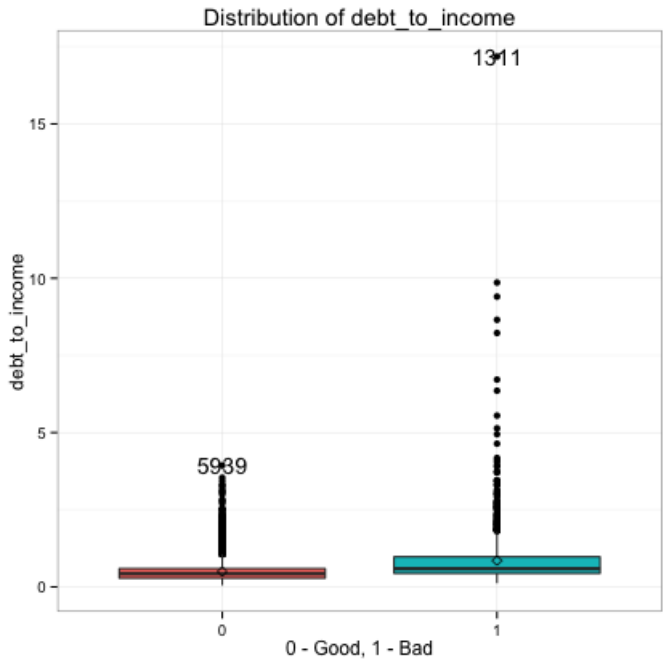


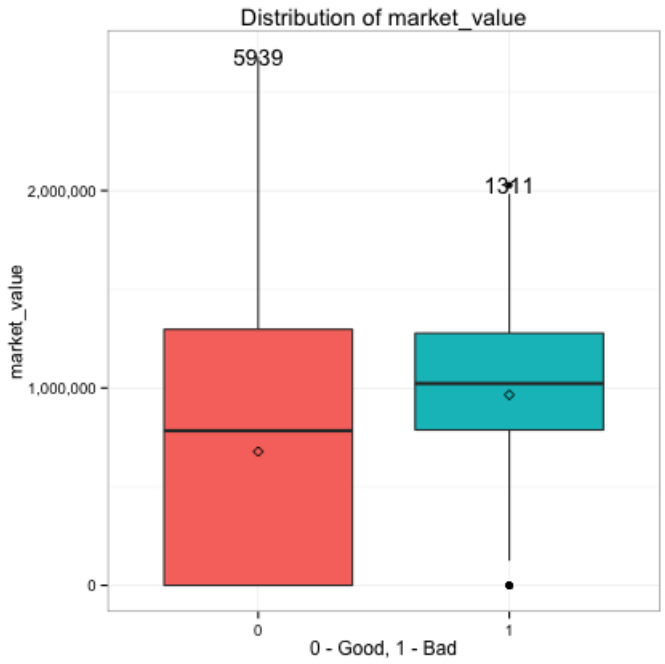
These plots suggest that the categorical predictors can be classified into three groups in terms of their potential predictive power:

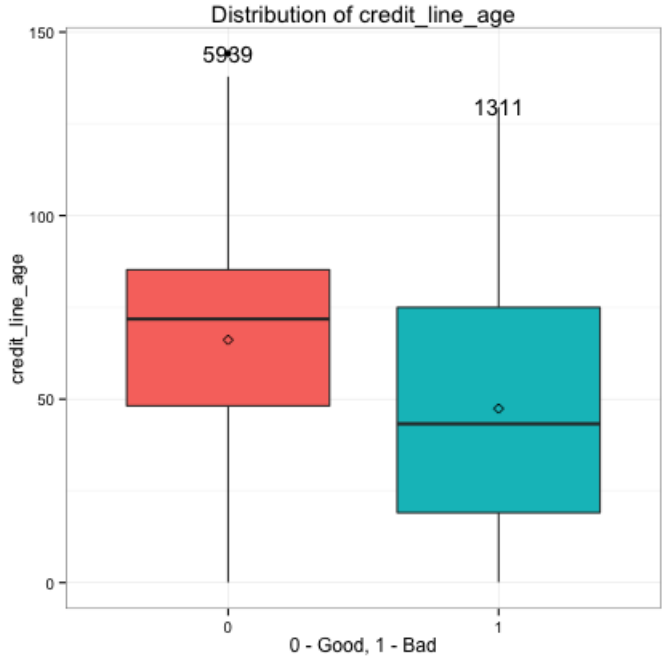
- Strong: bankruptcy, conviction, repossess, own_property, late_repayments
- Weak: purpose, marital, employment
- None: exist_customer, unspent_convictions

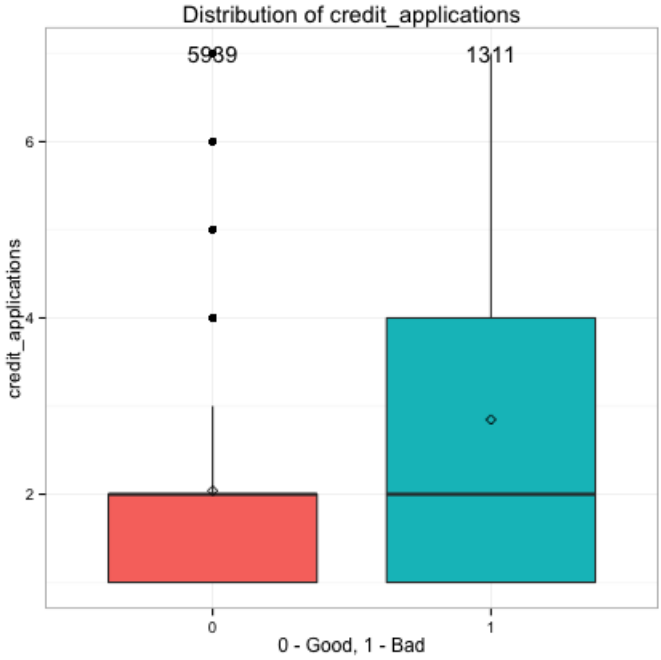
We also examine the relationships between the continuous predictors and the target.

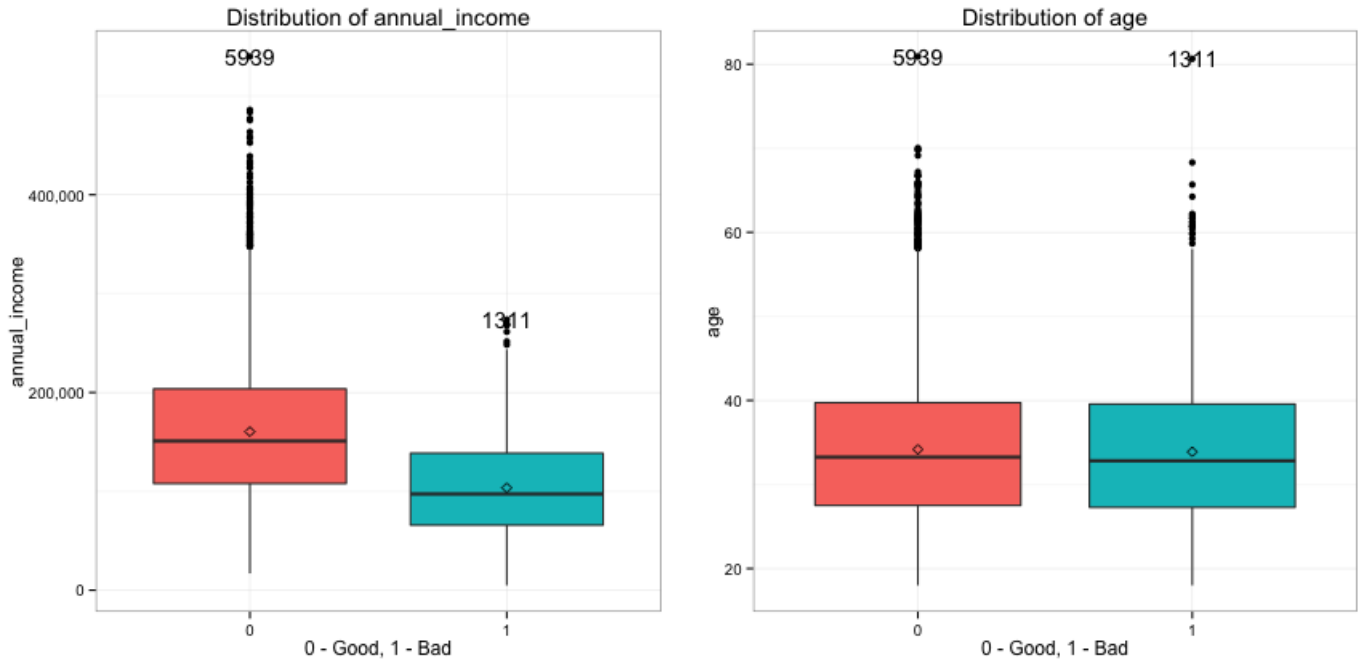
```
iv_con = c("debt_to_income", "market_value", "credit_line_age",
           "credit_applications", "annual_income", "age")
# make boxplots
plt = mk_boxplot(upl)
for (var in iv_con) {
        p = plt("bad", var, xlab="0 - Good, 1 - Bad", ylab=var,
                main = paste("Distribution of", var), legend=F)
        p = scale_axis(p, "y", scale="comma")
        print(p)
}
```


Distribution of debt_to_income
debt_to_income
15
10
5
0
5939
1311
0
1
0 - Good, 1 - Bad


Distribution of market_value
market_value
2,000,000
1,000,000
0
5939
1311
0
1
0 - Good, 1 - Bad

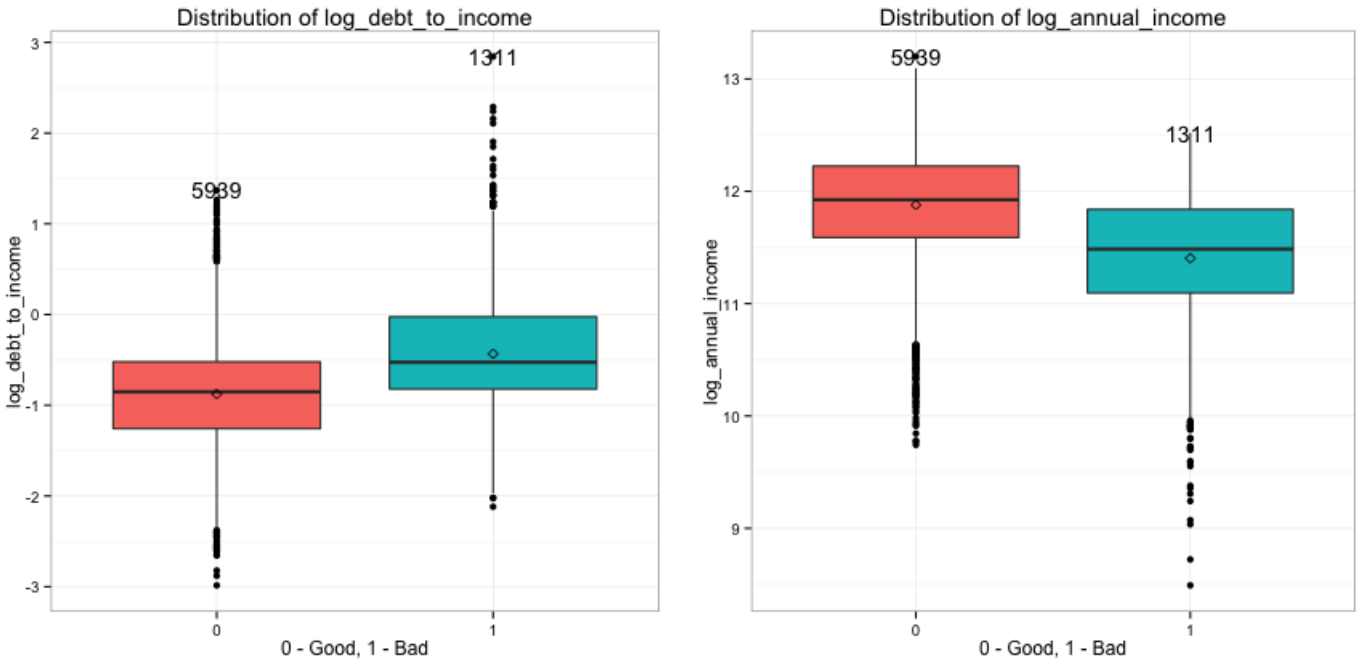

Distribution of credit_line_age
credit_line_age
150
100
50
0
5939
1311
0
1
0 - Good, 1 - Bad


Distribution of credit_applications
credit_applications
6
4
2
5939
1311
0
1
0 - Good, 1 - Bad



We see the distributions of debt_to_income and annual_income are heavily right skewed, so we take the log transform of debt_to_income and annual_income and replot.

```
upl = within(upl, {
             log_debt_to_income = log(debt_to_income)
             log_annual_income = log(annual_income)
             })
plt = mk_boxplot(upl)
for (var in c("log_debt_to_income", "log_annual_income")) {
        p = plt("bad", var, xlab="0 - Good, 1 - Bad", ylab=var,
                main = paste("Distribution of", var), legend=F)
        print(p)
}
```



These plots suggest that the continuous predictors can also be classified into three groups in terms of their potential predictive power:

- Strong: log_debt_to_income, log_annual_income, credit_line_age
- Weak: market_value, credit_applications
- None: age

We also observe that the bulk of zero market values belong to the good customers, while only a few bad customers have zero market values. This suggests owning property (market value > 0) is possibly a strong predictor of a bad customer, which we already discovered when looking at the distribution of own_property just a moment ago. Therefore, it's a good idea to create a categorical version of market_value by binning its values into different intervals based on its distribution.

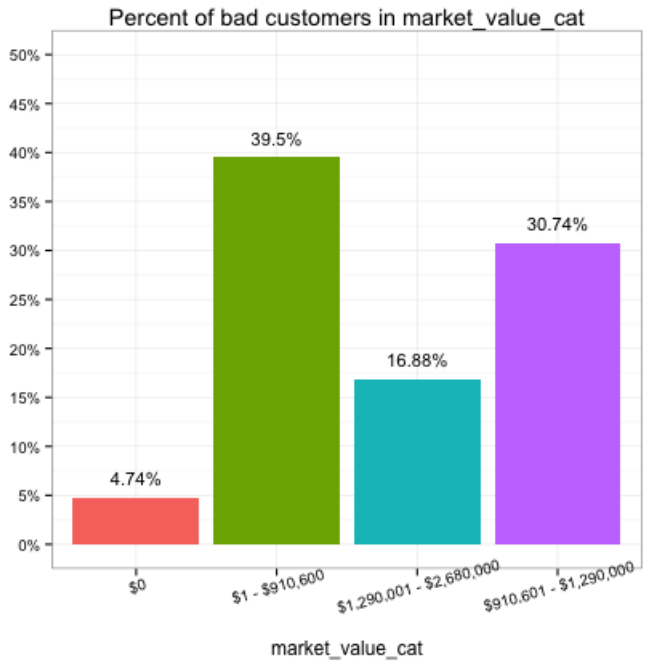
```
summary(upl$market_value)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      0       0  856000  730000 1290000 2680000
```

```
a = cut(upl$market_value, c(0, 1, 910600, 1290000, 2680000), right=F)
levels(a) = c("$0", "$1 - $910,600", "$910,601 - $1,290,000",
              "$1,290,001 - $2,680,000")
upl$market_value_cat = a
# update iv_cat
iv_cat = c(iv_cat, "market_value_cat")
```

We then plot the distribution of bad customers in market_value_cat.

```
# calculate the pct of bad customers
var = "market_value_cat"
tbl = calc_pct_bad(upl, var)
# append a column of label positions to tbl
f = add_bar_label_pos(tbl)
tbl = f(var, "pct_bad", vpos=0.02)
# draw bar plot
plt = mk_barplot(tbl)
p = plt(var, "pct_bad", fillby=var, xlab=var, legend=F,
        main=paste("Percent of bad customers in", var), barlab="pct_bad",
        barlab_at_top=T, barlab_use_pct=T, barlab_size=4)
p = scale_axis(p, "y", scale="pct", pct_max=0.5, pct_jump=0.05)
p = rotate_axis_text(p, 15)
print(p)
```



We see that market_value_cat is potentially a strong predictor. We'll use it instead of market_value.

Finally, we collect the predictors into strong, weak and none groups as above discussed so that we can easily access them for future analysis.

```
# categorical vars
iv_cat_strong = c("bankruptcy", "conviction", "repossess", "own_property",
                  "late_repayments", "market_value_cat")
iv_cat_weak = c("purpose", "marital", "employment")
iv_cat_none = c("exist_customer", "unspent_convictions")
# continuous vars
iv_con_strong = c("log_debt_to_income", "log_annual_income", "credit_line_age")
iv_con_weak = c("credit_applications") # exclude market_value
iv_con_none = c("age")
# save
save(upl, iv_cat_strong, iv_cat_weak, iv_cat_none, iv_con_strong,
     iv_con_weak, iv_con_none, file = file.path(data_path, "cleaned-02.rda"))
```