Credit Risk Modeling of Probability of Default - Logistic Regression (draft)

Marius, Sampid

14/12/2019

Introduction

Risk that a customer might default from his/her loan obligation is referred to as credit risk. Modeling the probability that a customer (debtor) will default from its loan obligations (be it personal or company loans) is of great importance for banks. Thus, credit risk modeling is all about loan default.

When a bank grants a loan to a customer, usually the entire amount is transferred to the customer's account. The customer will then reimburse the loan amount in smaller amounts including interest payments which might be monthly, quarterly or yearly depending on the terms of the loan contract.

There is always the risk that the customer will default, which will result to a loss for the bank referred to as the **Expected loss (EL)**. The **EL** is composed of three elements:

- The probability of default (**PD**): The probability that the customer will fail to pay back the loan in full.
- The loss given default (**LGD**): The amount of loss if there is a default, and
- The Exposure at default (**EAD**): The expected value of the loan at the time of default - that is, the amount of the loan that needs to be repaid at the time of default.

Therefore: **EL = PD * EAD * LGD**

Banks typically keeps information regarding default behavior of past customers which may subsequently be used to predict default behavior of new customers. This information are usually classified into two groups - application and behavioral information:

- **Application information**: income, marital status, level of education, age, etc,
- **Behavioral information**: current account balance, payment arrears in account history, credit card payment history, etc.

This report is focused on predicting the probability of default (PD).

Data information:

Dataset (reference: Datacamp) contains information on past loans with each row representing a customer and his/her information along with the loan status indicator. The loan status equals 1 if the customer defaulted, and 0 if the customer did not default.

Since I am interested in predicting **PD**, I will use the loan status as the **response** (target or dependent) variable and the rest of the variables as the **independent** or **explanatory** variables.

First, let's do some basic exploratory data analysis of the dataset

Take a look at the first 10 rows of the dataset:

```
loan_data <- readRDS("loan_data_ch1.rds") # read rds files - data stored as a
n RDS file.
# take a look at first 10 rows of dataset
head(loan_data, 10)
```

```
##    loan_status loan_amnt int_rate grade emp_length home_ownership
## 1            0      5000    10.65     B         10           RENT
## 2            0      2400       NA     C         25           RENT
## 3            0     10000    13.49     C         13           RENT
## 4            0      5000       NA     A          3           RENT
## 5            0      3000       NA     E          9           RENT
## 6            0     12000    12.69     B         11            OWN
## 7            1      9000    13.49     C          0           RENT
## 8            0      3000     9.91     B          3           RENT
## 9            1     10000    10.65     B          3           RENT
## 10           0      1000    16.29     D          0           RENT
##    annual_inc age
## 1       24000  33
## 2       12252  31
## 3       49200  24
## 4       36000  39
## 5       48000  24
## 6       75000  28
## 7       30000  22
## 8       15000  22
## 9      100000  28
## 10      28000  22
```

The credit history of each customer is reflected by the "**grade**" column known as the **bureau score of the customer** - the only behavioral information in the dataset, where "**A**" indicates the highest score of credit worthiness and "**G**" the lowest.

Look at the structure of the dataset:

```
str(loan_data)
```

```
## 'data.frame':    29092 obs. of  8 variables:
##  $ loan_status   : int  0 0 0 0 0 0 1 0 1 0 ...
##  $ loan_amnt     : int  5000 2400 10000 5000 3000 12000 9000 3000 10000 10
00 ...
##  $ int_rate      : num  10.6 NA 13.5 NA NA ...
##  $ grade         : Factor w/ 7 levels "A","B","C","D",..: 2 3 3 1 5 2 3 2
2 4 ...
##  $ emp_length    : int  10 25 13 3 9 11 0 3 3 0 ...
```

```
##  $ home_ownership: Factor w/ 4 levels "MORTGAGE","OTHER",..: 4 4 4 4 4 3 4
4 4 4 ...
##  $ annual_inc    : num  24000 12252 49200 36000 48000 ...
##  $ age           : int  33 31 24 39 24 28 22 22 28 22 ...
```

Dataset contains 29092 observations and 8 variables. Of the 8 variables, there are two factor or categorical variables. We also see a couple of missing values (NAs) on the variable containing interest rates.

See summary statistics of the dataset:

```
summary(loan_data)

##   loan_status       loan_amnt        int_rate     grade   
##  Min.   :0.0000   Min.   :  500   Min.   : 5.42   A:9649  
##  1st Qu.:0.0000   1st Qu.: 5000   1st Qu.: 7.90   B:9329  
##  Median :0.0000   Median : 8000   Median :10.99   C:5748  
##  Mean   :0.1109   Mean   : 9594   Mean   :11.00   D:3231  
##  3rd Qu.:0.0000   3rd Qu.:12250   3rd Qu.:13.47   E: 868  
##  Max.   :1.0000   Max.   :35000   Max.   :23.22   F: 211  
##                                   NA's   :2776    G:  56  
##    emp_length      home_ownership    annual_inc           age       
##  Min.   : 0.000   MORTGAGE:12002   Min.   :   4000   Min.   : 20.0  
##  1st Qu.: 2.000   OTHER   :   97   1st Qu.:  40000   1st Qu.: 23.0  
##  Median : 4.000   OWN     : 2301   Median :  56424   Median : 26.0  
##  Mean   : 6.145   RENT    :14692   Mean   :  67169   Mean   : 27.7  
##  3rd Qu.: 8.000                    3rd Qu.:  80000   3rd Qu.: 30.0  
##  Max.   :62.000                    Max.   :6000000   Max.   :144.0  
##  NA's   :809
```

Let’s take a closer look at the data structure of the categorical variables with the help of the ***CrossTable( )*** function in the ***{gmodels}*** package.

Cross table for each categorical variable gives an output table with each category in the variable with the number of cases and proportions:

```
library(gmodels)
CrossTable(loan_data$home_ownership)

## 
## 
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
## 
## 
## Total Observations in Table:  29092 
## 
## 
##           |  MORTGAGE |     OTHER |       OWN |      RENT |
```

```
##             |-----------|-----------|-----------|-----------|
##             |     12002 |        97 |      2301 |     14692 |
##             |     0.413 |     0.003 |     0.079 |     0.505 |
##             |-----------|-----------|-----------|-----------|
##
##
##
##
```

Adding loan status as a second argument gives us the possibility to look at the relationship between the response or target variable and other categorical variables.

```
CrossTable(loan_data$home_ownership, loan_data$loan_status, prop.r = T, prop.
c = F, prop.t = F, prop.chisq = F)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |-------------------------|
##
##
## Total Observations in Table:  29092
##
##
##                          | loan_data$loan_status
## loan_data$home_ownership |         0 |         1 | Row Total |
## -------------------------|-----------|-----------|-----------|
##                 MORTGAGE |     10821 |      1181 |     12002 |
##                          |     0.902 |     0.098 |     0.413 |
## -------------------------|-----------|-----------|-----------|
##                    OTHER |        80 |        17 |        97 |
##                          |     0.825 |     0.175 |     0.003 |
## -------------------------|-----------|-----------|-----------|
##                      OWN |      2049 |       252 |      2301 |
##                          |     0.890 |     0.110 |     0.079 |
## -------------------------|-----------|-----------|-----------|
##                     RENT |     12915 |      1777 |     14692 |
##                          |     0.879 |     0.121 |     0.505 |
## -------------------------|-----------|-----------|-----------|
##             Column Total |     25865 |      3227 |     29092 |
## -------------------------|-----------|-----------|-----------|
##
##
```

```r
# **NOTE**: Setting prop.r = T and the rest F gives us the row-wise proportions.

CrossTable(loan_data$grade, loan_data$loan_status, prop.r = T, prop.c = F, prop.t = F, prop.chisq = F)
```

```
## 
## 
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |-------------------------|
## 
## 
## Total Observations in Table:  29092 
## 
## 
##                 | loan_data$loan_status 
## loan_data$grade |         0 |         1 | Row Total | 
## ----------------|-----------|-----------|-----------|
##               A |      9084 |       565 |      9649 | 
##                 |     0.941 |     0.059 |     0.332 | 
## ----------------|-----------|-----------|-----------|
##               B |      8344 |       985 |      9329 | 
##                 |     0.894 |     0.106 |     0.321 | 
## ----------------|-----------|-----------|-----------|
##               C |      4904 |       844 |      5748 | 
##                 |     0.853 |     0.147 |     0.198 | 
## ----------------|-----------|-----------|-----------|
##               D |      2651 |       580 |      3231 | 
##                 |     0.820 |     0.180 |     0.111 | 
## ----------------|-----------|-----------|-----------|
##               E |       692 |       176 |       868 | 
##                 |     0.797 |     0.203 |     0.030 | 
## ----------------|-----------|-----------|-----------|
##               F |       155 |        56 |       211 | 
##                 |     0.735 |     0.265 |     0.007 | 
## ----------------|-----------|-----------|-----------|
##               G |        35 |        21 |        56 | 
##                 |     0.625 |     0.375 |     0.002 | 
## ----------------|-----------|-----------|-----------|
##    Column Total |     25865 |      3227 |     29092 | 
## ----------------|-----------|-----------|-----------|
## 
## 
```

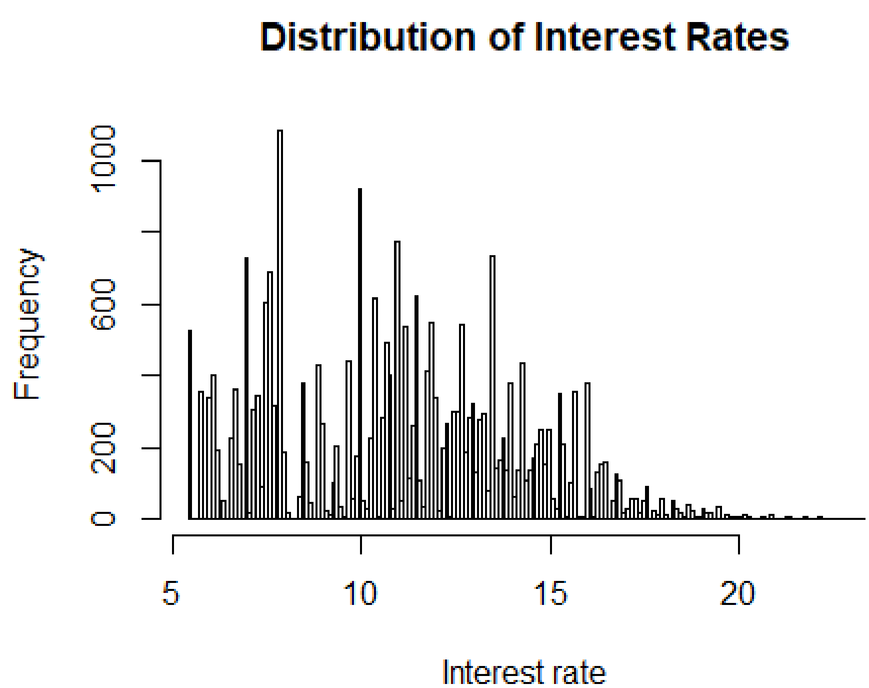
From the above tables we can deduce that customers with home ownership labelled as "**OTHER**" have the highest default rates of **17.5%** whereas customers with "**MORTGAGES**" have the lowest default rates of **9.8%**. Also, we see that the default rate increases as we move down the score of credit worthiness from "**A**" to "**G**".

Cleaning the dataset by getting rid of outliers and missing values (NAs)

Dealing with outliers:

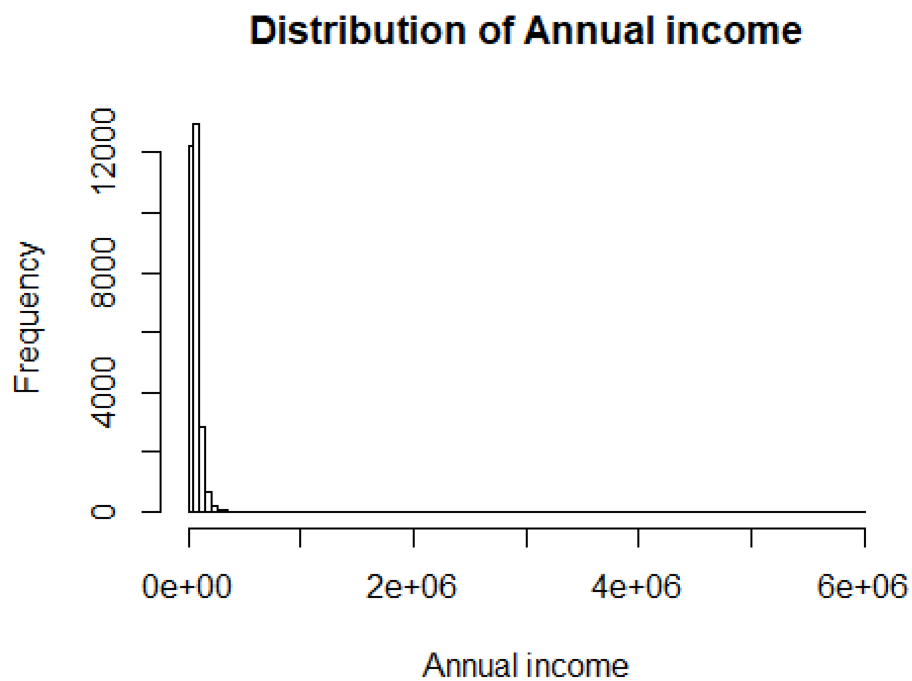
Let's check the continuous variables for outliers using plots: variables of interest being **interest rates**, **annual income** and **age**.

```
breaks <- sqrt(nrow(loan_data))
hist_loan <- hist(loan_data$int_rate, main = "Distribution of Interest Rates"
, xlab = "Interest rate", breaks = breaks)
```



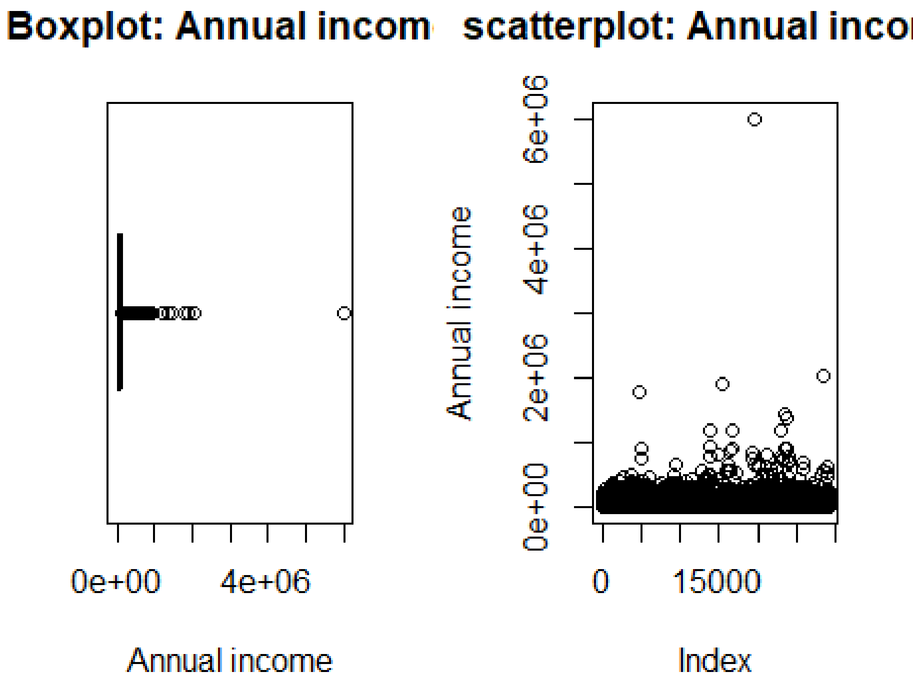
Notice that all loans had an interest rate over 5%, and very few loans had interest rates greater than 20%.

```
hist_income <- hist(loan_data$annual_inc, main = "Distribution of Annual inco
me", xlab = "Annual income", breaks = breaks)
```



It is hard to deduce any peculiar information on the distribution of annual income. A scatterplot and a boxplot might give us more reasonable information:

```
par(mfcol=c(1,2))
boxplot(loan_data$annual_inc, horizontal = T, xlab = "Annual income",
        main = "Boxplot: Annual income")
plot(loan_data$annual_inc, ylab ="Annual income",
     main = "scatterplot: Annual income")
```



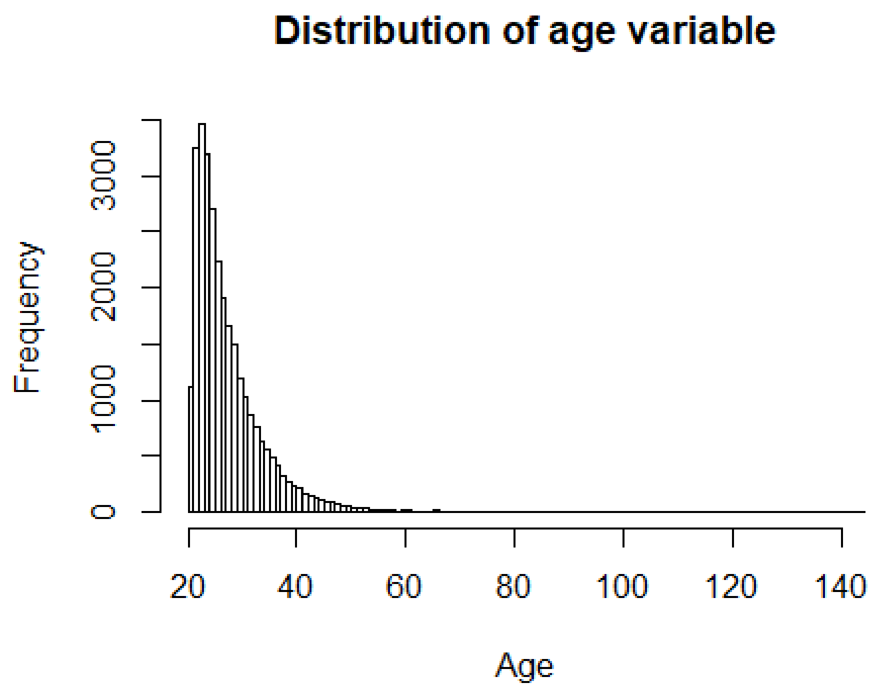
From the scatterplot, notice that there is someone with an annual income of about 6 million, whereas, the rest have an annual income of approximately 2 million and less. This is considered an outlier, also confirmed with the boxplot.

How do we deal with the outliers? We can either:

1) delete all outliers, that is, based on expert judgement, or
2) as a ***"rule of thumb"***, any point in the dataset that is greater than **Q3 + 1.5*IQR** or less than **Q1-1.5*IQR** is considered an outlier, where **Q1** is the ***First Quartile***, **Q3** the ***Third Quartile*** and **IQR** is the ***Inter Quartile Range***.

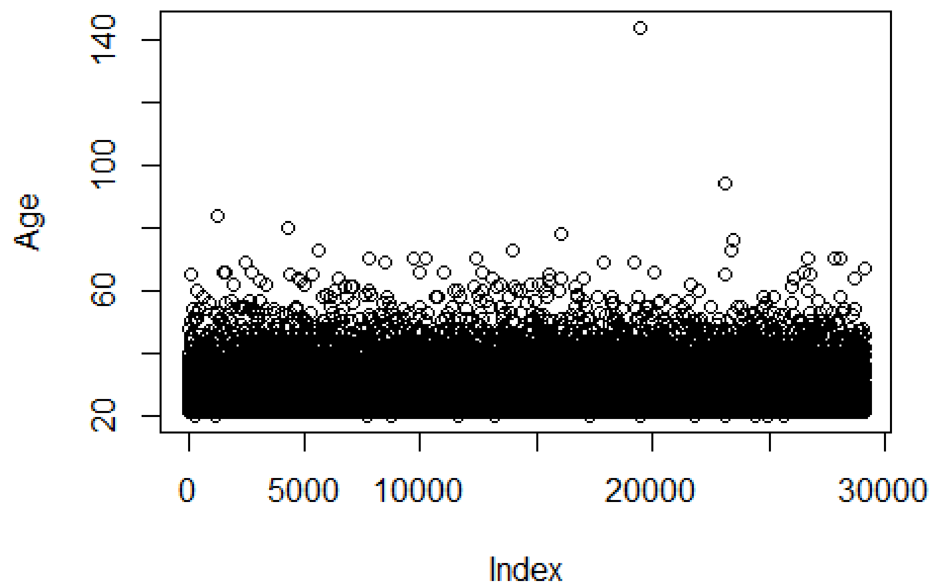
Lets see a histtogram plot for the age variable:

```
hist(loan_data$age, breaks = breaks, xlab = "Age",
     main = "Distribution of age variable")
```



There is one customer who is above or about 140 years old. This must be an outlier. We can get more insights using a scatter plot of the age variable, but first let's see the summary statistics:

```
summary(loan_data$age)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    20.0    23.0    26.0    27.7    30.0   144.0

plot(loan_data$age, ylab = "Age")
```



It is now clear that the age of about 144 years (as seen in summary statistics) is an outlier. We will proceed to remove these outliers using the **rule of thumb**.

There are no outliers below the 1st quartile, so we set a cutoff point for the outliers in the 3rd quartile based on the "age" variable:

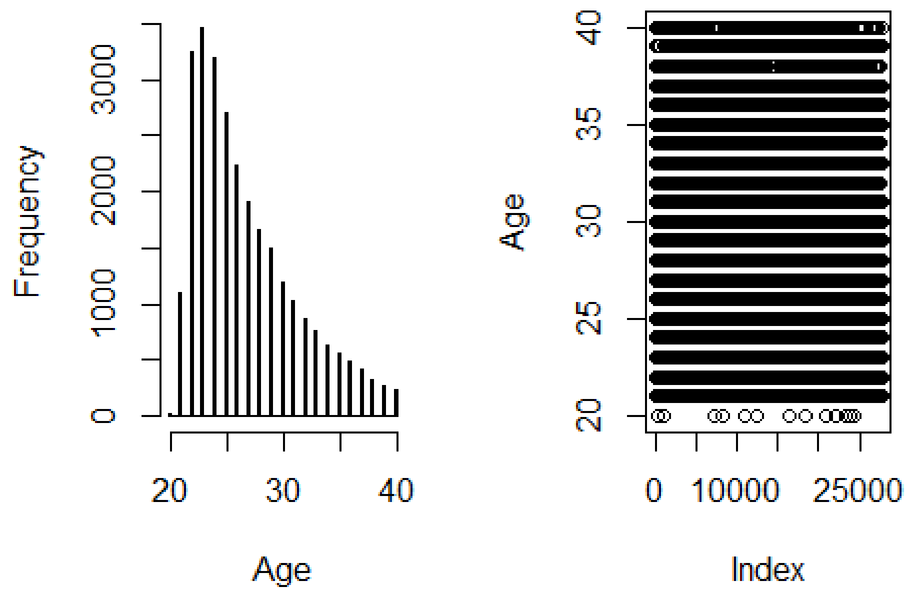
```
# calculates the cutoff point
cutoff_point_age <- quantile(loan_data$age, 0.75) + 1.5*IQR(loan_data$age)
```

The next step is to index all data points that are above the cutoff value and then delete the corresponding rows in the dataset:

```
# index cutoff point.
index_high_age <- which(loan_data$age > cutoff_point_age)
# **NOTE** : That subsetting is done in the original dataset.

# delete corresponding rows in the dataset
loan_data_new <- loan_data[-index_high_age,]
```

Check the new or subsetted dataset with no outliers using plots:

```
par(mfcol=c(1,2))
hist(loan_data_new$age, breaks = sqrt(nrow(loan_data_new)), xlab = "Age", mai
n = "")
plot(loan_data_new$age, ylab = "Age")
```




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

Notice that a quite number of observations have been deleted from the dataset.

Dealing with NAs (missing values)

We have now removed the outliers, but we still need to take care of the NAs.

From the summary statistics above, we noticed that there are NAs in the interest rate and employment length variables, that is, about 2.8% and 9.5% of the entire columns, respectively.

We can either do one of the following:

- **Likewise deletion**:
  - Delete all observations from participants with missing data if the sample is large enough. However, it is wise to make sure that the data is missing at random so we are not deleting an entire cross section of the dataset which might lead to missing valuable information.
  - Delete the entire column if more than about 65% of the values are missing.
- **Median imputation**:
  - Replace the missing values with the median of the observed values in the entire column.
- **Keep the NAs**:
  - The fact that the value is missing is important information. However, keeping NAs as such is not always be possible as some methods will automatically delete them because they cannot deal with them. So how can we keep NAs? By **coarse**

classification. That is, put continuous variables into **bins.** Employment length, for example, ranges between 0 and 62 years. We can make bins of +/- 15 years for the following categories: "0-15", "15-30", "30-45", "45+", "missing", where missing represents the NAs. Try bins of different ranges but with same frequencies to get more balance bins. It should be noted that outliers can also be treated as missing values.

- **Regression substitution**:
  - Multi regression analysis can be used to estimate the missing values.

In this report we employ **Median imputation** and **coarse classification**.

Median imputation

```
# Make copy of loan_data_new
loan_data_replace <- loan_data_new

# calculare median for interest rate
median_ir <- median(loan_data_new$int_rate, na.rm =TRUE)

# calculare median for emp_length
median_emp_length <- median(loan_data_new$emp_length, na.rm =TRUE)

# Replace NAs on int_rate with median
loan_data_replace$int_rate[is.na(loan_data_replace$int_rate)] <- median_ir

# Replace NAs on emp_length with median
loan_data_replace$emp_length[is.na(loan_data_replace$emp_length)] <- median_e
mp_length
```

Let's check if all NAs have been replaced with the medians of the interest rates and employment length variables:

```
# Check if NAs have been replaced in the interest rate variable
summary(loan_data_replace$int_rate)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    5.42    8.49   10.99   11.00   13.11   23.22

any(is.na(loan_data_replace$int_rate))

## [1] FALSE

# Check if the NAs have been replaced in the employment length variable:
summary(loan_data_replace$emp_length)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   2.000   4.000   6.076   8.000  62.000

any(is.na(loan_data_replace$emp_length))

## [1] FALSE
```

See summary statistics of the entire dataset stored in object **loan_data_replace** to be sure it is free of missing values:

```r
# see summary statistics of loan_data_replace:
summary(loan_data_replace)
```

```
##   loan_status       loan_amnt        int_rate     grade   
##  Min.   :0.0000   Min.   :  500   Min.   : 5.42   A:9211  
##  1st Qu.:0.0000   1st Qu.: 5000   1st Qu.: 8.49   B:8922  
##  Median :0.0000   Median : 8000   Median :10.99   C:5477  
##  Mean   :0.1112   Mean   : 9573   Mean   :11.00   D:3071  
##  3rd Qu.:0.0000   3rd Qu.:12150   3rd Qu.:13.11   E: 833  
##  Max.   :1.0000   Max.   :35000   Max.   :23.22   F: 199  
##                                                   G:  54  
##    emp_length      home_ownership    annual_inc           age      
##  Min.   : 0.000   MORTGAGE:11427   Min.   :   4080   Min.   :20.0  
##  1st Qu.: 2.000   OTHER   :   91   1st Qu.:  40000   1st Qu.:23.0  
##  Median : 4.000   OWN     : 2205   Median :  56000   Median :26.0  
##  Mean   : 6.076   RENT    :14044   Mean   :  66126   Mean   :26.8  
##  3rd Qu.: 8.000                    3rd Qu.:  80000   3rd Qu.:29.0  
##  Max.   :62.000                    Max.   :1200000   Max.   :40.0  
## 
```

The data frame **loan_data_replace** is now clean and ready for analysis - free of outliers and missing values.

Coarse classification

We will perform coarse classification on the interest rate variable. The values in this variable ranges between 5.4 and 23.2. Thus, we create a new variable labelled **ir_cat** and the values are binned in categories of “0-8”, “8-11”, “11-13.5”, “13.5+” and “missing”.

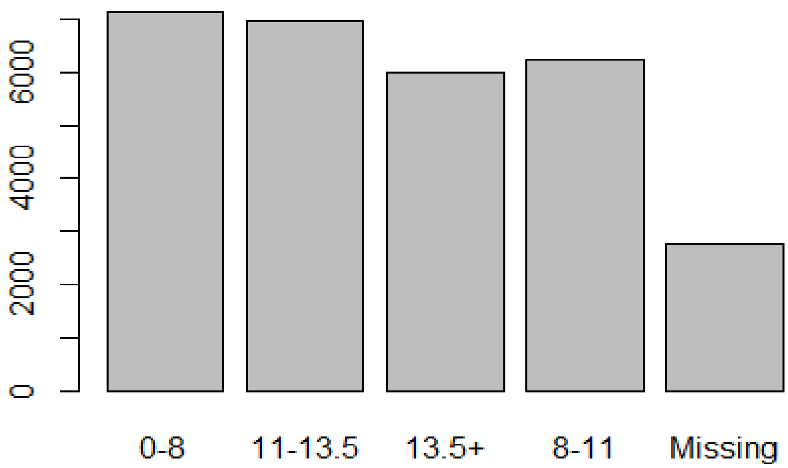
```r
loan_data_cat <- loan_data # make copy of dataset

loan_data_cat$ir_cat <- rep(NA, length(loan_data_cat$int_rate))

loan_data_cat$ir_cat[which(loan_data_cat$int_rate <= 8)] <- "0-8"
loan_data_cat$ir_cat[which(loan_data_cat$int_rate > 8 & loan_data$int_rate <=
11)] <- "8-11"
loan_data_cat$ir_cat[which(loan_data_cat$int_rate > 11 & loan_data$int_rate <
= 13.5)] <- "11-13.5"
loan_data_cat$ir_cat[which(loan_data_cat$int_rate > 13.5)] <- "13.5+"
loan_data_cat$ir_cat[which(is.na(loan_data_cat$int_rate))] <- "Missing"
```

Make sure the new categorical variable is stored as a factor of categorical variable:

```r
loan_data_cat$ir_cat <- as.factor(loan_data_cat$ir_cat)
```

Let’s check that we have more balanced bins by plotting the new variable **ir_cat**:

```r
# Look at your new variable using plot( )
plot(loan_data_cat$ir_cat)
```




```
# check dimension and summary of dataset with interest rates put in bins
dim(loan_data_cat)

## [1] 29092     9

summary(loan_data_cat)

##   loan_status        loan_amnt        int_rate      grade
##  Min.   :0.0000   Min.   :  500   Min.   : 5.42   A:9649
##  1st Qu.:0.0000   1st Qu.: 5000   1st Qu.: 7.90   B:9329
##  Median :0.0000   Median : 8000   Median :10.99   C:5748
##  Mean   :0.1109   Mean   : 9594   Mean   :11.00   D:3231
##  3rd Qu.:0.0000   3rd Qu.:12250   3rd Qu.:13.47   E: 868
##  Max.   :1.0000   Max.   :35000   Max.   :23.22   F: 211
##                                   NA's   :2776    G:  56
##    emp_length      home_ownership    annual_inc           age
##  Min.   : 0.000   MORTGAGE:12002   Min.   :   4000   Min.   : 20.0
##  1st Qu.: 2.000   OTHER   :   97   1st Qu.:  40000   1st Qu.: 23.0
##  Median : 4.000   OWN     : 2301   Median :  56424   Median : 26.0
##  Mean   : 6.145   RENT    :14692   Mean   :  67169   Mean   : 27.7
##  3rd Qu.: 8.000                    3rd Qu.:  80000   3rd Qu.: 30.0
##  Max.   :62.000                    Max.   :6000000   Max.   :144.0
##  NA's   :809
##      ir_cat
##  0-8    :7130
##  11-13.5:6954
##  13.5+  :6002
##  8-11   :6230
##  Missing:2776
##
```

The data frame **loan_data_cat** is now ready for analysis - free of outliers but we keep the NAs using coarse classification.

Analysis

Now that the data is fully preprocessed and clean, we can now go ahead and start with analysis.

Splitting data into test and training sets

We can run the model on the entire dataset and use same data in evaluating the model. However, this is not the best option because it will lead to a model that is too realistic.

The best way is to split the data into two groups - a **test set** for evaluating the model and a **training set** for building the model.

To split the dataset into training and test sets, we first set a seed using the **set.seed( )** function. Seeds allow us to create a starting point for randomly generated numbers, so that each time the code is run the same answers are generated. The advantage of doing this in the sampling is that the exact same training and test sets data will be reproduced by anyone using the same seed and dataset, which is good for testing and learning purposes.

We randomly assign observations to the training and test sets data using the **sample( )** function. Assigning 2/3 of the dataset for training the model and 1/2 for evaluating or testing the model:

```
# rename dataset
loan_data_clean <- loan_data_replace

set.seed(15)
# Store row numbers for training set in object: index_train
index_train <- sample(1:nrow(loan_data_clean), 2 / 3 * nrow(loan_data_clean))

# Subset training set and stored in object: training_set
training_set <- loan_data_clean[index_train, ]

# Subset test set and store in object: test_set
test_set <- loan_data_clean[-index_train, ]
```

Let's also split the **loan_data_cat** dataset into training and test data.

```
set.seed(16)
# Store row numbers for training set in object: index_train_2
index_train_2 <- sample(1:nrow(loan_data_cat), 2 / 3 * nrow(loan_data_cat))

# Subset training set and stored in object: training_set_2
training_set_2 <- loan_data_cat[index_train_2, ]

# Subset test set and stored in object: test_set_2
test_set_2 <- loan_data_cat[-index_train_2, ]
```

Before we proceed, let's look at the dimensions of the training and test sets data, and the structure of the training data:

```
dim(training_set)

## [1] 18511     8

dim(test_set)

## [1] 9256    8

dim(training_set_2)

## [1] 19394     9

dim(test_set_2)

## [1] 9698    9

str(training_set)

## 'data.frame':    18511 obs. of  8 variables:
##  $ loan_status   : int  0 1 0 0 0 1 0 0 1 0 ...
##  $ loan_amnt     : int  8250 13000 14400 10000 19000 20000 14000 10000 550
0 24000 ...
##  $ int_rate      : num  7.49 7.66 13.49 10.95 16.82 ...
##  $ grade         : Factor w/ 7 levels "A","B","C","D",..: 1 1 3 2 5 3 2 2
1 1 ...
##  $ emp_length    : num  1 3 4 2 3 3 13 7 4 4 ...
##  $ home_ownership: Factor w/ 4 levels "MORTGAGE","OTHER",..: 4 4 4 4 4 4 4
1 1 4 ...
##  $ annual_inc    : num  25200 66000 79820 50000 50000 ...
##  $ age           : int  21 23 30 25 28 26 37 24 22 26 ...

str(training_set_2)

## 'data.frame':    19394 obs. of  9 variables:
##  $ loan_status   : int  0 1 0 1 1 0 0 0 1 0 ...
##  $ loan_amnt     : int  9600 4800 7800 15000 7750 6800 13000 11500 11500 1
3000 ...
##  $ int_rate      : num  10 14.84 8.88 14.96 13.85 ...
##  $ grade         : Factor w/ 7 levels "A","B","C","D",..: 2 4 2 4 3 1 2 1
2 4 ...
##  $ emp_length    : int  5 2 6 21 NA 15 0 2 3 2 ...
##  $ home_ownership: Factor w/ 4 levels "MORTGAGE","OTHER",..: 1 4 4 1 4 1 1
1 1 4 ...
##  $ annual_inc    : num  82000 30000 40320 88800 24000 ...
##  $ age           : int  23 31 29 25 29 25 31 25 32 43 ...
##  $ ir_cat        : Factor w/ 5 levels "0-8","11-13.5",..: 4 3 4 3 3 1 4 4
4 3 ...
```

From the results of the above outputs - structure of the training and test sets, we can happily proceed with modeling the probability of default (**PD**) using **logistic regression** as data is now very clean and organized.

Logistic Regression

Logistic regression is same as linear regression in many ways except that the output of the model is a value between 0 and 1. This is important as we are interested in predicting the probability of default, which is by definition, a value between 0 and 1. Thus,

$$PD = Pr(loan\_status = 1|x_1, x_2, \dots, x_m)$$

$$= log\left(\frac{p}{1-p)}\right) = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \cdots + \hat{\beta}_m x_m,$$

where $x_1, x_2 \dots, x_m$ represents the explanatory or independent variables, $\beta_0, \beta_1, \dots, \beta_m$ are parameters to be estimated, and $\beta_0 x_0 + \beta_1 x_1 + \cdots + \beta_m x_m$ is the linear predictor.

Logistic regression is fitted in R using the **glm( )** function - the Generalized Linear Model function and setting the family argument to "binomial".

Let's construct a logistic regression model using the **training_set_2** dataset with **loan_status** as the target variable and the categorical variable **ir_cat** as the independent variable:

```
log_model_cat <- glm(formula = loan_status ~ ir_cat, family = "binomial",
                     data = training_set_2)

# print the model
log_model_cat

##
## Call:  glm(formula = loan_status ~ ir_cat, family = "binomial", data = tra
ining_set_2)
##
## Coefficients:
##   (Intercept)  ir_cat11-13.5    ir_cat13.5+     ir_cat8-11  ir_catMissing
##       -2.8761         0.9570         1.3132         0.6233         0.6260
##
## Degrees of Freedom: 19393 Total (i.e. Null);  19389 Residual
## Null Deviance:      13380
## Residual Deviance: 13020     AIC: 13030
```

We can also look at the distribution of the **ir_cat** variable:

```
table(training_set_2$ir_cat)

##
##     0-8 11-13.5   13.5+    8-11 Missing
##    4705    4603    4087    4132    1867
```

Notice that parameter estimate for the category **0-8** under coefficients in the **log_model_cat** printed model is not reported. This is because logistic regression models in R for categorical variables reports parameter estimates for all but one of the categories. The category for which no parameter estimate is reported is referred to as the reference category. The parameter for each of the other categories represents the odds ratio in favor of a loan default between the category of interest and the reference category.

Adding more variables in the model

Once more let's see what variables we have in the **training_set_2** dataset:

```
colnames(training_set_2)
```

```
## [1] "loan_status"    "loan_amnt"      "int_rate"       "grade"
## [5] "emp_length"     "home_ownership" "annual_inc"     "age"
## [9] "ir_cat"
```

Let's modify the **log_model_cat** by including variables age, grade, loan_amnt and annual_inc. We call this model **log_model_multi**:

```
log_model_multi <- glm(loan_status ~ ir_cat + age + grade + loan_amnt + annua
l_inc,
                       family = "binomial", data = training_set_2)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
# see the structure of the model using the summary () function:
summary(log_model_multi)
```

```
##
## Call:
## glm(formula = loan_status ~ ir_cat + age + grade + loan_amnt +
##     annual_inc, family = "binomial", data = training_set_2)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.0794  -0.5294  -0.4437  -0.3368   3.2821
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -2.447e+00  1.284e-01 -19.064  < 2e-16 ***
## ir_cat11-13.5  4.680e-01  1.349e-01   3.470 0.000521 ***
## ir_cat13.5+    4.520e-01  1.498e-01   3.017 0.002550 **
## ir_cat8-11     3.438e-01  1.201e-01   2.862 0.004204 **
## ir_catMissing  2.030e-01  1.332e-01   1.524 0.127528
## age           -3.111e-03  3.888e-03  -0.800 0.423672
## gradeB         3.597e-01  1.087e-01   3.310 0.000933 ***
## gradeC         6.289e-01  1.245e-01   5.052 4.36e-07 ***
## gradeD         9.679e-01  1.407e-01   6.878 6.07e-12 ***
## gradeE         1.007e+00  1.693e-01   5.946 2.74e-09 ***
## gradeF         1.495e+00  2.322e-01   6.440 1.19e-10 ***
## gradeG         2.019e+00  3.774e-01   5.349 8.84e-08 ***
## loan_amnt     -3.218e-06  4.168e-06  -0.772 0.440054
## annual_inc    -5.092e-06  7.344e-07  -6.934 4.08e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 13381  on 19393  degrees of freedom
## Residual deviance: 12870  on 19380  degrees of freedom
## AIC: 12898
##
## Number of Fisher Scoring iterations: 5
```

Important to us from the model output is the parameter estimates and the statistical significance of the parameter estimates denoted as **Pr(>|t|)**. Significance is denoted by **"."** - very weak, to very strong significance denoted by "***".

Predicting probabilities of default

We now proceed with the **log_model_multi** model to predict probabilities of default. Remember that we will now have to use the test datasets for predictions and later to evaluate the model. In this case, we use the **test_set_2** dataset. Prediction is done using the predict( ) function in R:

```
predictions_all_small <- predict(log_model_multi,newdata = test_set_2,
                                        type = "response")
```

Get an initial idea of how well the model can discriminate using range( )

```
# Look at the range of the object "predictions_all_small"
range(predictions_all_small)

## [1] 5.495437e-06 4.586539e-01
```

Let's construct another model (named **log_model_full**) but this time including all available variable predictors in the **training_set** dataset and use the model to predict **PD**:

```
# Create a logistic regression model using all available predictors in the tr
aining_set dataset
log_model_full <- glm(loan_status ~., family = "binomial", data = training_se
t)

#predict PD:
predictions_all_full <- predict(log_model_full,newdata = test_set,
                                 type = "response")
```

Check out the first five rows of the predicted **PD**:

```
# first five rows of predicted **PD**:
head(predictions_all_full,5)

##          2          16          17          20          29
## 0.18601755 0.12673057 0.11799722 0.15110112 0.06517805

# Make predictions using the log_model_multi model:
predictions_log_model_multi <- predict(log_model_multi,newdata = test_set_2,
                                 type = "response")
```

```r
# Make predictions using the log_model_cat model:
predictions_log_model_cat <- predict(log_model_cat,newdata = test_set_2,
                                     type = "response")
```

Binary predictions

To evaluate the performance of the models, we compare the loan status in the test sets with the model predictions. But first, the predicted probabilities of the loan status lies between 0 and 1, thus we need to specify a cutoff or threshold value between 0 and 1 where if the predicted probability is above the cutoff value, the prediction is set to 1 for a default otherwise 0 for a non-default.

Let's assume a cutoff or threshold value of 0.15:

```r
predict_Cutoff_15 <- ifelse(predictions_all_full > 0.15,1,0)
```

Check out the first five rows of binary predictions:

```r
# Binary predictions:
head(predict_Cutoff_15,5)

##  2 16 17 20 29
##  1  0  0  1  0

# Make binary predictions using the log_model_multi model
predict_Cutoff_15_multi <- ifelse(predictions_log_model_multi > 0.15,1,0)

# Make binary predictions using the log_model_cat model:
predict_Cutoff_15_cat <- ifelse(predictions_log_model_cat > 0.15,1,0)
```

Confusion matrix

A popular method for summarizing credit risk models especially when dealing with large set of predictions is with the help of confusion matrices.

A confusion matrix is a contingency table of correct and incorrect classifications. The measures derived from the confusion matrix are:

- Classification accuracy: percentage of correctly classified instances,
- Sensitivity: percentage of bad customers that are classified correctly,
- Specificity: percentage of good customers that are classified correctly.

```r
# Calculates the confusion matrix
conf_matrix <- table(test_set$loan_status, predict_Cutoff_15)
conf_matrix

##    predict_Cutoff_15
##        0    1
##   0 6455 1781
##   1  607  413

Classification_accracy <- sum(diag(conf_matrix))/nrow(test_set)
round(Classification_accracy*100,1)
```

```
## [1] 74.2
```

The classification accuracy of the model is 74.2%.

```
sensitivity <- 413/(607+413)
round(sensitivity*100,1)

## [1] 40.5

Specificity <- 6455/(6455+1781)
round(Specificity*100,1)

## [1] 78.4
```

General remarks

- As cutoff value increases accuracy also increases up to a certain point and then stays constant (typical with credit risk modeling).
- Higher accuracy simply means that majority of cases are classified as non-default.
- Specificity increases as cutoff value increases.
- Sensitivity decreases as cutoff value increases.

Note

The logistic regression models we have constructed so far are also known as logistic regression models with the **logit** link function. Other alternatives include the **probit** and the **cloglog** link functions. However, the differences between these models are generally insignificant. The results will depend on the chosen cut-off value.

Comparing link functions for a given cut-off

Let's fit three logistic regression models using link functions logit, probit and cloglog, respectively, using the ***training_set** data, make predictions using the **test_set** data, and compare results while assuming a cutoff value of 14%.

```
# Fit the logit model:
log_model_logit <- glm(loan_status ~., family = binomial(link = logit),
                       data = training_set)

# Fit the probit model:
log_model_probit <- glm(loan_status ~., family = binomial(link = probit),
                        data = training_set)

# Fit the cloglog model:
log_model_cloglog <- glm(loan_status ~., family = binomial(link = cloglog),
                       data = training_set)

# Make predictions for all models using the test_set dataset:
predictions_logit <- predict(log_model_logit, newdata = test_set,
                             type = "response")
```

```r
predictions_probit <- predict(log_model_probit, newdata = test_set,
                              type = "response")

predictions_cloglog <-predict(log_model_cloglog, newdata = test_set,
                              type = "response")

# Assume cutoff of value 14% to make binary predictions-vectors:
cutoff <- 0.14
class_pred_logit <- ifelse(predictions_logit > cutoff, 1, 0)
class_pred_probit <- ifelse(predictions_probit > cutoff, 1, 0)
class_pred_cloglog <- ifelse(predictions_cloglog > cutoff, 1, 0)

# Calculate confusion matrix for the three models:
true_val <- test_set$loan_status
tab_class_logit <- table(true_val,class_pred_logit)
tab_class_probit <- table(true_val,class_pred_probit)
tab_class_cloglog <- table(true_val,class_pred_cloglog)

# Compute the classification accuracy for all three models:
acc_logit <- sum(diag(tab_class_logit)) / nrow(test_set)
acc_probit <- sum(diag(tab_class_probit)) / nrow(test_set)
acc_cloglog <- sum(diag(tab_class_cloglog)) / nrow(test_set)

# print results in percentages
round(acc_logit*100,2)

## [1] 71.23

round(acc_probit*100,2)

## [1] 71.04

round(acc_cloglog*100,2)

## [1] 71.38
```

Evaluating the credit risk model

How do we select a proper cutoff value? The choice of a cutoff value is important as it changes the validation matrix. Also, using the credit risk model on future applicants, the cutoff could be a way of deciding who gets a loan and who doesn't.

NOTE: It should be noted that no single model is perfect, and no matter how many applicants the bank rejects to loan money to, there will always be customers that default on their loan obligations.

The credit risk model simply helps the bank to decide on how many loans they should approve if they don't want to exceed a certain percentage of default in their portfolio of customers.

For example, assuming the test set contains new customers and the bank decides to reject 20% of the new applicants based on their fitted probability of default. This will mean that 20% of customers with the highest probability of default will be rejected.

The cutoff value is obtained by looking at the 80 quantile of the predictions vector. For example:

```
# Fit the logit, probit and cloglog-link logistic regression models
model_logistic <- glm(loan_status ~., family = "binomial", data = training_se
t)

# Make predictions:
predictions_logistic <- predict(model_logistic, newdata = test_set,
                             type = "response")

# Calculate cutoff point based on 20% customers with the highest probability
of default:
cutoff_20 <- quantile(predictions_logistic, 0.8)
cutoff_20

##      80%
## 0.158326
```

We now make binary predictions based on this cutoff point:

```
# make binary predictions:
binary_pred_logistic <- ifelse(predictions_logistic > cutoff_20, 1, 0)
```

Let's now compare the actual loan status and the predictions to see what loans would have been accepted using this cutoff value, and what percentage of the accepted loans actually defaulted - the bad rate.

```
actual_vs_predictions <- as.data.frame(cbind(test_set$loan_status, binary_pre
d_logistic))
# **NOTE**: as.data.frame( ) function is necessary here for us to easily

subset data.
colnames(actual_vs_predictions) <- c("Actual", "Predictions")

# see the first 10 rows:
head(actual_vs_predictions,10)

##    Actual Predictions
## 2       0           1
## 16      0           0
## 17      0           0
## 20      1           0
## 29      0           0
## 30      0           0
## 31      0           0
## 33      0           1
## 34      0           1
## 35      0           0
```

```
# Take a look at the confusion martix:
conf_matrix_binary_pred_logistic <- table(actual_vs_predictions)
conf_matrix_binary_pred_logistic

##       Predictions
## Actual    0    1
##      0 6746 1490
##      1  659  361

# Calculate the accuracy of the model:
Accuracy_binary_pred_logistic_model <-
                        sum(diag(conf_matrix_binary_pred_logistic)/nrow(test_
set))
round(Accuracy_binary_pred_logistic_model*100,2)

## [1] 76.78
```

Thus, the model has an accuracy of 76.78%.

```
# Calculate sensitivity (percentage of bad customers classified:
sensitivity_binary_pred_logistic_model <- 361/(361+659)
round(sensitivity_binary_pred_logistic_model*100,2)

## [1] 35.39

# Calculate specifisity (percentage of good customers classified correctly):
Specificity_binary_pred_logistic_model <- 6746/(6746+1490)
round(Specificity_binary_pred_logistic_model*100,2)

## [1] 81.91

# Calculate the bad rate or the percentage of default:
AcceptedLoans_binary_model <- actual_vs_predictions[binary_pred_logistic == 0
,1]

bad_rate_binary_pred_model <- sum(AcceptedLoans_binary_model/length(AcceptedL
oans_binary_model))
bad_rate_binary_pred_model

## [1] 0.08899392
```

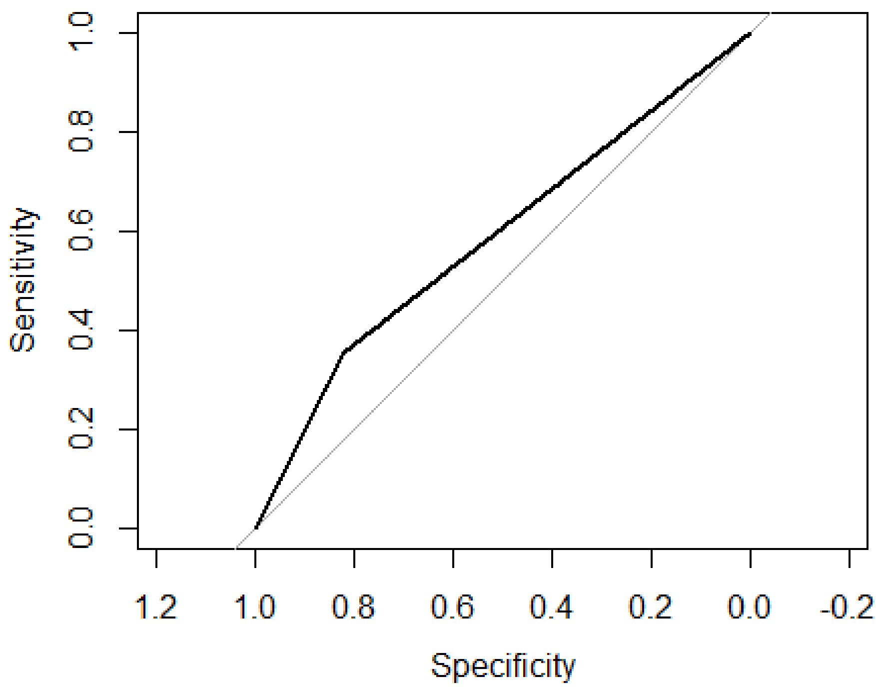
What model is the best?

Most often, banks just want to know which model is the best without having to make any assumptions of the minimum bad rate they can have or cutoff value. We have seen previously from the decision matrix (confusion matrix) how we could evaluate the model based on **accuracy**, **sensitivity** and **specificity**. **Accuracy,** however, is usually maximized when a high cutoff point is selected or when all test set arguments are classified as non-defaults, which is problematic.

One of the most popular methods of evaluating credit risk models is based on **sensitivity** and **specificity** referred to as the **Receiver Operating Characteristics curve (ROC-curve)**. That is, plotting the sensitivity against 1-specificity for each possible cutoff.

The closer a ROC-curve is to the top left corner the better the model. The curve will have higher specificities associated with higher sensitivities:

```
library(pROC)

roc_binary_pred_logistic_model <- roc(actual_vs_predictions$Actual, actual_vs
_predictions$Predictions)

# plot ROC-curve for the binary_pred_logistic_model
plot(roc_binary_pred_logistic_model)
```



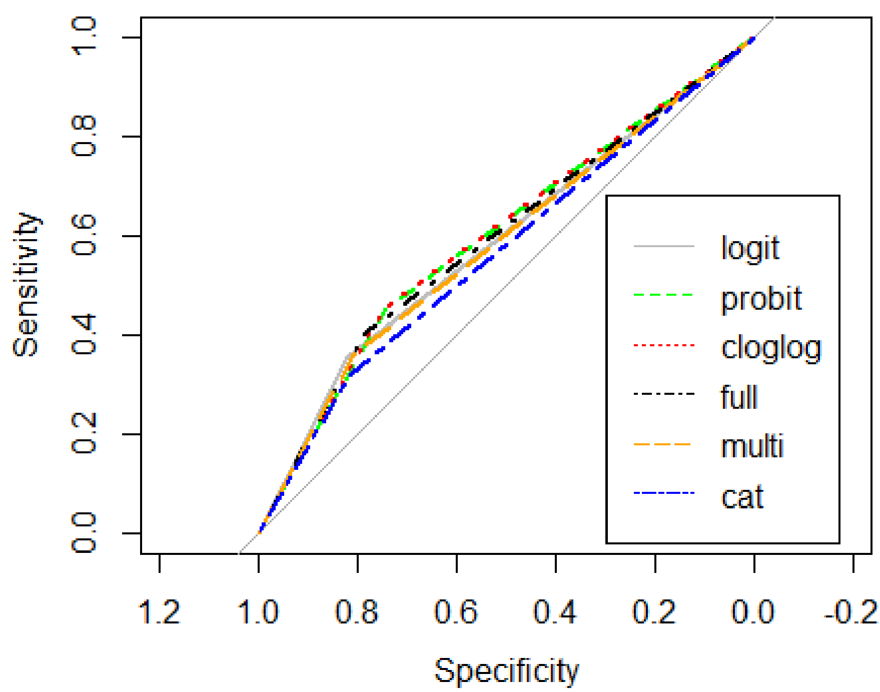
Sometimes it becomes very difficult or not clear to distinguish between models using ROC-curves. In this situation, we calculate the **area under the curve (AUC)** for each model. The **AUC** of a model is between 0.5 and 1, and the larger the **AUC**, the better the model.

```
# calculate AUC for the binary_pred_logistic_model
auc(roc_binary_pred_logistic_model)

## Area under the curve: 0.5865
```

Let's plot the ROC-curves and calculate the AUCs for all the credit risk models we have constructed so far and compare to see which model is the best:

```
true_val <- test_set$loan_status
true_val_2 <- test_set_2$loan_status
roc_class_logit <- roc(true_val,class_pred_logit)

roc_class_probit <- roc(true_val,class_pred_probit)

roc_class_cloglog <- roc(true_val,class_pred_cloglog)
```

```r
roc_predictions_all_full <- roc(true_val, predict_Cutoff_15)

roc_predict_Cutoff_15_multi <- roc(true_val_2, predict_Cutoff_15_multi)

roc_predict_Cutoff_15_cat <- roc(true_val_2, predict_Cutoff_15_cat)


# plot ROC-curves for the binary models:
plot(roc_binary_pred_logistic_model, col = "grey", lty = 1)
lines(roc_class_probit, col = "green", lty = 2)
lines(roc_class_cloglog, col = "red", lty = 3)
lines(roc_predictions_all_full, col = "black", lty = 4)
lines(roc_predict_Cutoff_15_multi, col = "orange", lty = 5)
lines(roc_predict_Cutoff_15_cat, col = "blue", lty = 6)
legend(0.3, 0.68, legend = c("logit", "probit", "cloglog", "full", "multi", "
cat"),
       col = c("grey", "green", "red", "black", "orange", "blue"), lty = 1:6)
```



As we can see from the above plots, it is hard to tell which model is better as there are cross overs between models and very close. Thus, we must calculate **AUC** for the models to select the best model:

```r
# Calculates AUCs:

AUC_class_logit <- auc(true_val,class_pred_logit)

AUC_class_logit

## Area under the curve: 0.5974
```

```
AUC_class_probit <- auc(true_val,class_pred_probit)

AUC_class_probit

## Area under the curve: 0.5972

AUC_class_cloglog <- auc(true_val,class_pred_cloglog)

AUC_class_cloglog

## Area under the curve: 0.5991

AUC_predictions_all_full <- auc(true_val, predict_Cutoff_15)

AUC_predictions_all_full

## Area under the curve: 0.5943

AUC_predict_Cutoff_15_multi <- auc(true_val_2, predict_Cutoff_15_multi)

AUC_predict_Cutoff_15_multi

## Area under the curve: 0.5841

AUC_predict_Cutoff_15_cat <- auc(true_val_2, predict_Cutoff_15_cat)

AUC_predict_Cutoff_15_cat

## Area under the curve: 0.5674
```

As we can see, the **class_pred_cloglog** model has the highest AUC and is thus selected as the best model.

One final note is that banks are indeed interested in knowing what variables are best for predicting defaults. The ROC-curve or AUC could be used for variable selection. Variables that improves the model will improve the ROC-curve and AUC while variables that are not useful to the model will have no influence on the ROC-curve or AUC.