The Likelihood of Clients Getting Rejected for Loan Applications

Jezen Alexander

Seattle Pacific University

April 2019

Introduction

Mathematics and advances in technology have certainly had a huge impact to the world. The use of mathematical and programming techniques has become a norm in almost every part of the world. This is especially true for businesses. One of the main purposes of using these techniques is to help companies in their decision-making processes. There are a lot of things in this world that require decision-making processes. One of them that we will see here is regarding loans; whether to accept or reject clients' applications to borrow money.

We will look at the data posted by Home Credit Group; a global loan provider which operates in many different countries including the US [1], which is made available for download through Kaggle [6]. It has a little over 300,000 data about Home Credit's clients and each of the data has more than 100 variables. These variables tell things such as clients' income, family members, occupations, whether clients own a house, whether clients own a car and many more. The binary variable called "TARGET" is the response variable, where 1 means clients have payment difficulties and 0 means clients don't have payment difficulties. Therefore, using this dataset, we can create a model that predicts the likelihood that clients would have difficulties in paying off their loans, which in turn could be use by the company to make decisions on whether to approve or reject loan applications. The mathematical technique that will be used for this project is logistic regression and this will be done through R, a programming language.

Regression

First, regression is a statistical technique used to understand the relationship between independent or explanatory variables and dependent or response variables, which are normally denoted by x and y, respectively.

Linear Regression

The type of regression that we all might be familiar with is linear regression which produces the equation:

$$y = mx + c.$$

This is the equation of a line. Let us look at a very simple dataset where linear regression can be applied.

x	y
1	3
2	5
3	7
4	10
5	12

Table 1: Data showing the values for x and y

Assume x is the number of hours spent and y is the number of homework questions completed by a certain person. Now, we can use R to create a model that would predict the approximate number of homework questions that would be completed given the number of hours that he will spend doing these (see *Appendix A*).

```
Call:
lm(formula = y ~ x, data = HW)

Coefficients:
(Intercept)            x
        0.5          2.3
```

Figure 1: Output produced in R indicating the coefficients of x and the intercept

Based on the output shown in Figure 1, the model would be as follow,

$$y = 2.3x - 0.5.$$

According to the model above, if this person decides to spend 8 hours of his time (or x = 8) to do his homework, he will be able to approximately complete 18 questions. If we plot the original dataset and the regression line together, it will look as follow:

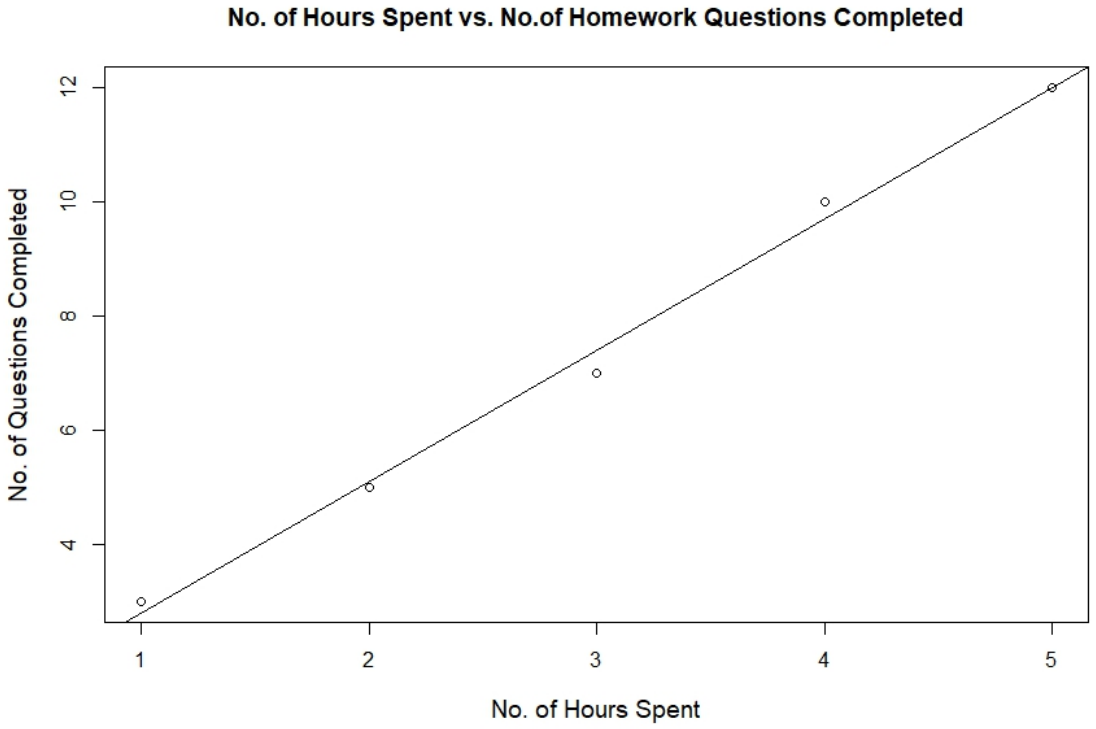

Figure 2: Fitting a linear model to the data from Table 1

As can be seen, the regression line, which is produced from the model we just created, tries to fit the points in the dataset. That is the idea of regression models; they try to fit all the points of the dataset then used to make predictions.

Logistic Regression

Linear regression, though, is only appropriate for predicting response variables that are quantitative, such as number of homework questions completed (like the example above), and linear. Looking back at the Home Credit's dataset, its response variable, "TARGET," is binary, meaning it only takes exactly two values, 0 and 1. Again, 1 means clients have payment

difficulties and 0 means clients don't have payment difficulties. Figure 3 below shows what happen when a dependent variable that is binary is modeled using linear regression (see *Appendix A*).

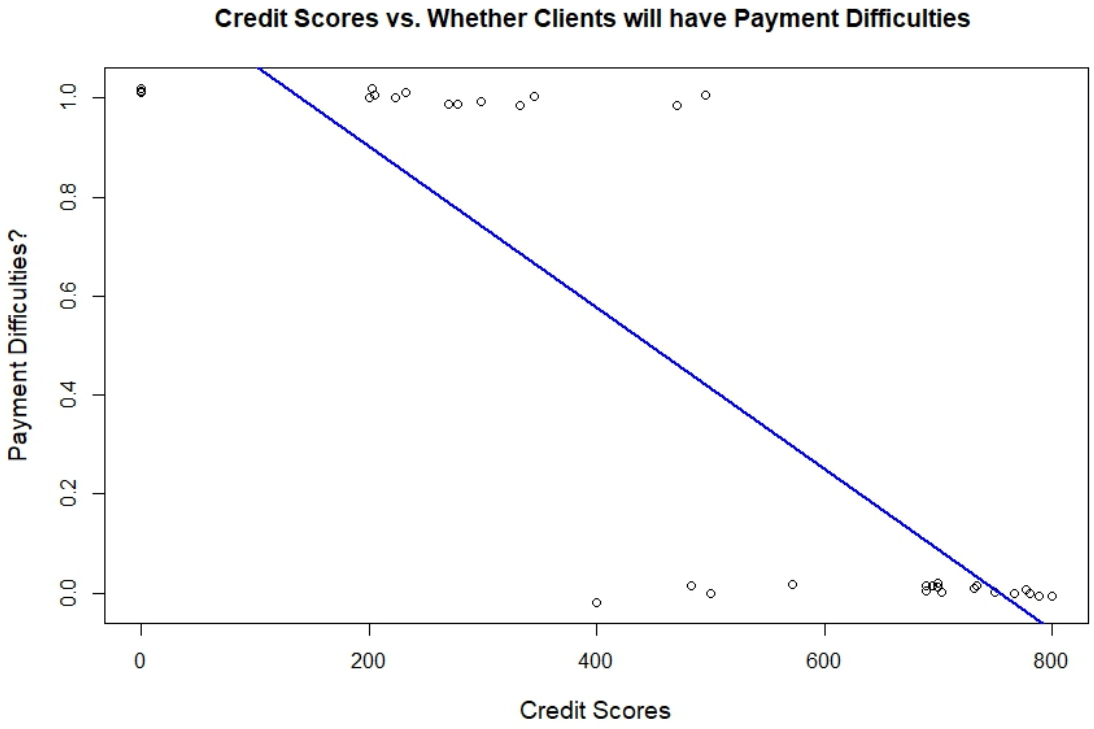

Figure 3: Fitting a linear model to the data where the response variable is binary

On the x axis we have clients' credit scores and on the y axis we have the response on whether the clients have payment difficulties. Note that the values used to create the above plot are arbitrary for illustration purposes, where most clients with high credit scores do not have payment difficulties (response = 0) and most clients with low credit scores experience payment difficulties (response = 1). In the actual dataset from Home Credit Group, there are many clients with low to no credit scores but do not have payment difficulties. Similarly, there are lots of clients with high credit scores but experience payment difficulties. Credit scores alone are not sufficient (or might not be important at all) for predicting whether clients will

have payment difficulties. It is part of our analysis to test which variables are important to predict the likelihood that clients will have payment difficulties.

Nonetheless, for now, we will use the plot in Figure 3 to discuss why linear regression is not appropriate to predict response variables that are binary. Clearly, the plot in Figure 3 is different from the plot in Figure 2. That is, in Figure 2, the data points are continuous. Moreover, there is a strong positive correlation between the response variable and the predictor. Therefore, fitting the points with a straight line works well. However, in the case of predicting whether clients will have payment difficulties, the response variable is discrete and thus a straight line is no longer appropriate to fit the data points as can be seen in Figure 3. "There is no gradual transition; the y value abruptly jumps from one binary outcome to the other [8]."

In this case, a curve would be more suitable to fit the data points and it can be done using logistic regression. Logistic regression models always give fitted values between 0 and 1 through some transformations [2]. Below is an example of how logistic curve would fit the points.

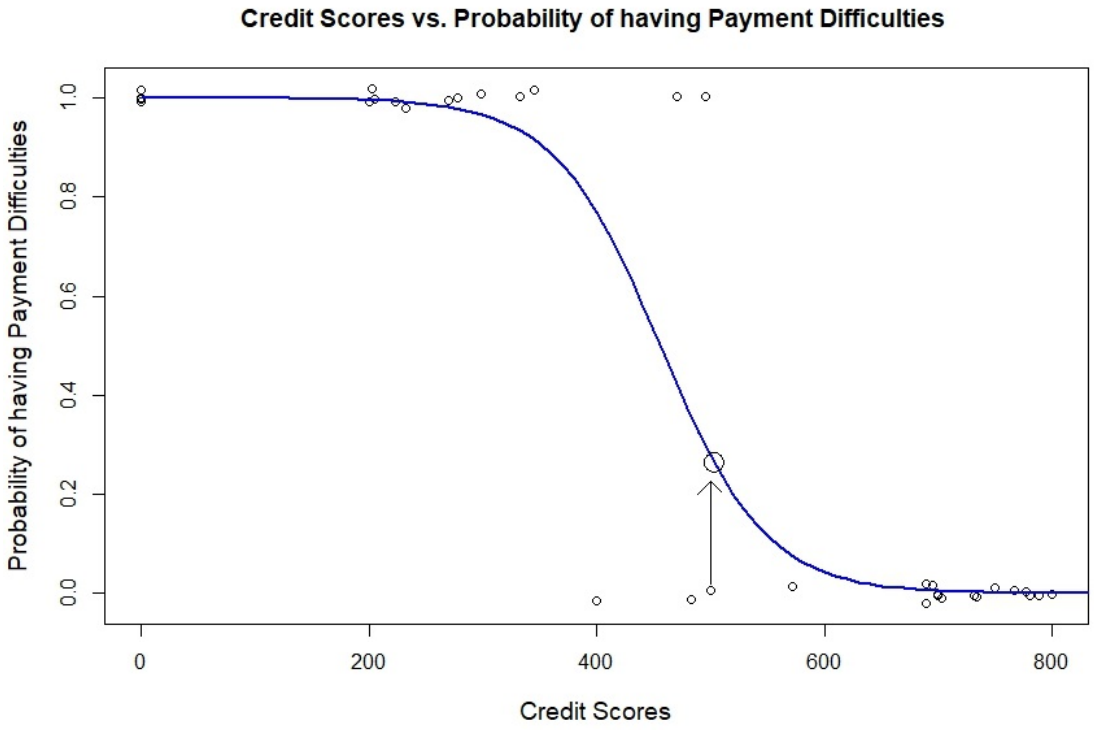

Figure 4: Fitting a logistic model to discrete data

Notice that the y axis is now a probability. The interpretation of logistic model is not as straightforward as linear model. Logistic curves have the form

$$f(x) = \frac{e^{g(x)}}{1 + e^{g(x)}}$$

Both the numerator and denominator are positive since exponential functions are always positive. Also, the numerator is always smaller than the denominator. This makes $0 < f(x) < 1$ and thus it is an ideal function for modeling proportions or probabilities which are also constrained between 0 and 1. Hence, in this case, we are going to predict the probability or the likelihood that clients will have payment difficulties.

Once we have the model, $g(x)$, we will plug it into the logistic form to compute the probability. We can also refer to Figure 4 to see how we can interpret logistic model graphically. The indicated point shows a client with a credit score of about 500 which corresponds to a probability of approximately 0.3. This means that the client has about 30% chance of having payment difficulties. Again, this is arbitrary. We will start to work with the real dataset in the next part.

Data Preparation

Considering the size of the data (307,511 rows and 122 variables), the first step that we need to do is to clean up the data in R (see *Appendix B*). Large datasets tend to have lots of missing data and there is no different with this dataset.

Variables with too many missing data would not be useful at all and thus we want to avoid using these variables in our model. For instance, a variable called "OWN_CAR_AGE" (age of the car that clients owned), over 60% of the data are missing. For other variables where missing values occur completely randomly and the frequency of occurrence is quite rare, we can replace these by the average values [7]. Although there are disadvantages of filling missing data with the average values, it is a basic and simple method that is not time-consuming. Considering we have lots of variable, we would try to stick with this method first. Then, there are variables called "EXT_SOURCE_1," "EXT_SOURCE_2," and "EXT_SOURCE_3," and each of these variables tells about clients' credit scores but from different sources. We can create a new variable called "Scores" that will take the average value from these three variables. We also need to create a new variable for age. The variable "DAYS_BIRTH" in this dataset tells the client's age in days and

the values are negative. To make this easier to understand and interpret, it is best that we convert the age into years by creating a new variable called “Age.”

Other things that need to be taken care of are the nominal variables such as gender (male or female), family status (single, married, widow, etc.), education (academic degree, higher, incomplete, etc.) and more. Before we can create models using these variables, we first need to convert them into numeric. For instance, for gender, 1 indicates male and 0 indicates woman. For variables that have more than one category like family status, it is necessary to create indicator variables. If there are 3 categories (i.e. single, married, widow), we would need two indicator variables and each take values of 1 and 0 for true and false respectively. The reason we only need two indicator variables is because, if both of them are zero, which means two out of three categories are false, then the third category must be true. Thus, if there is a third indicator variable, it will be redundant as its response can be determined without including it in the model.

Proportion Tables

Since we have a lot of variables, not to mention those with many categories, we can create proportion tables as a way to reduce the amounts of categorical variable. Consider the following proportion tables computed from the dataset,

	Academic Degree	Lower Education
No Payment Difficulties	98.2%	89.1%
Payment Difficulties	1.8%	10.9%

Table 2: Proportions of client that have payment difficulties based on their education level

	Own House=Yes	Own House=No
No Payment Difficulties	91.7%	92%
Payment Difficulties	8.3%	8%

Table 3: Proportions of client that have payment difficulties based on whether they own a house or not

From Table 2, it seems that clients with lower education tend to have more difficulties paying off the loans as compare to clients that have academic degree (10.9% vs. 1.8%). Remember that our objective is to predict the likelihood that clients would have payment difficulties. Thus, the proportions regarding client's education could be useful and we would consider keeping this variable. On the other hand, Table 3 seems to indicate that the information regarding whether clients own a house or not does not seem to be a good indicator on whether they would have payment difficulties. So, it would be reasonable to exclude this variable. Also, there is a variable called "FLAG_MOBIL" which tells whether clients provided mobile phone numbers (1 if they provide and 0 otherwise). This variable shows no zeros. In other words, all clients provided phone numbers. This variable will not be useful at all and thus should be dropped.

After going through the above processes, we now have 33 variables. Now that we have a clean dataset, we first need to split this into training and testing dataset. "A standard rule of thumb is for two-thirds of the data to go to training and one-third to go to the test data set [7]." Therefore, we split the dataset into 70:30 ratio. We will only work with the training dataset for creating models. As for the testing dataset, we will treat it as unobserved data so we can use it to check the validity of the models.

Correlations

Although we managed to significantly reduce the amounts of variable, 33 variables are still too much and there is no way we would have a model using all these variables. Thus, we should pick variables that are significant for predicting the target. One standard way to choose which variables are important and which are not is by looking at the correlations between the predictors and the response variable. However, this method is only appropriate for quantitative variables. We want to look for variables that have strong correlations with our response variable, "TARGET." Among the variables that are strongly correlated with "TARGET," we also need to ensure that they are not strongly correlated with each other. Otherwise, we need to eliminate and only pick those that have the strongest correlation with the response variable. This is because, "In cases where the correlation between predictors is high, but not 1, the coefficients can be estimated, but interpretation of individual terms can be problematic [3]."

Out of the 33 variables, there are 23 quantitative variables in the dataset. The correlations between these variables and the response variable are fairly low, with the highest value of only 0.1737. Taking a cutoff value of 0.02, we have the following 10 variables ordered from the most correlated to the least.

Variable	**Description**
Scores	Clients' credit scores from external sources
Age	Clients' age in years
DAYS_LAST_PHONE_CHANGED	How many days before applications did clients change phone
DAYS_ID_PUBLISHED	How many days before the application did client change the identity document with which he applied for the loan

DAYS_REGISTRATION	How many days before the application did client change his registration
AMT_GOODS_PRICE	For consumer loans, it is the price of the goods for which the loan is given
REGION_POPULATION_RELATIVE	Normalized population of region where client lives (higher number means the client lives in more populated region)
DEF_30_CNT_SOCIAL_CIRCLE	How many observations of client's social surroundings defaulted on 30 DPD (days past due)
CNT_CHILDREN	No. of children clients have
AMT_REQ_CREDIT_BUREAU_YEAR	Number of enquiries to Credit Bureau about the client one year before application

Table 4: Quantitative variables that have the strongest correlation with the response variable ordered from the most correlated to the least

Variable Selection and Modeling

Before we go into modeling, it is a good idea to rename some of our variables to make the models simple and easy to understand. The table below shows the list of all the variables that we have left.

Original Variable	**Renamed Variables**	**Description**
TARGET	Target	Response variable. 1=payment difficulties, 0=no payment difficulties
Scores	-	Clients' credit scores from external sources
Age	-	Clients' age in years
DAYS_LAST_PHONE_CHANGED	DaysPhone	How many days before applications did clients change his/her phone
DAYS_ID_PUBLISHED	DaysId	How many days before the application did client change the identity document with which he/she applied for the loan
DAYS_REGISTRATION	DaysRegist	How many days before the application did client change his/her registration

AMT_GOODS_PRICE	GoodsPrice	The price of the goods for which the loan is given
REGION_POPULATION_RELATIVE	RegPopulation	Normalized population of region where client lives (higher number means the client lives in more populated region)
DEF_30_CNT_SOCIAL_CIRCLE	Def	How many observations of client's social surroundings defaulted on 30 DPD (days past due)
CNT_CHILDREN	Children	No. of children clients have
AMT_REQ_CREDIT_BUREAU_YEAR	EnquiriesYr	Number of enquiries to Credit Bureau about the client, one year before application
NAME_CONTRACT_TYPE	LoanType	The type of loan: cash or revolving
CODE_GENDER	Gender	Client's gender
FLAG_EMP_PHONE	EmpPhone	Did client provide work phone?
REGION_RATING_CLIENT	RegRating	Home Credit's rating of the region where clients live (1,2,3)
REG_CITY_NOT_LIVE_CITY	LiveMatch	Flag if client's permanent address does not match contact address (1=different, 0=same, at city level)
REG_CITY_NOT_WORK_CITY	WorkMatch	Flag if client's permanent address does not match work address (1=different, 0=same, at city level)
Secondary	-	Indicator variable. Did client finish secondary education?
FamCivil	-	Indicator variable. Marital status: Civil Marriage
FamSingle	-	Indicator variable. Marital status: Single
HouseParents	-	Indicator variable. Does client lives in parent's house?

Table 5: All the available variables left

Now, we will start the modeling process using R (see *Appendix C*). The first thing we can do is to create a model using all the variables or predictors. Then, we will start eliminating the predictors one at a time, based on the p-value. Generally, a predictor with p-value less than or equal to 0.05 is considered significant (the smaller it is the more significant). Thus, once we eliminate the predictors with p-values greater than 0.05, we will stop and use the model [5]. The figure below shows the summary of a model using all the predictors available which also shows the p-values on the most right.

```
Coefficients:
               Estimate Std. Error z value Pr(>|z|)
(Intercept)   -1.837e+00  8.521e-02 -21.563  < 2e-16 ***
LoanType       5.168e-01  3.375e-02  15.312  < 2e-16 ***
Gender         2.052e-01  1.731e-02  11.856  < 2e-16 ***
Children       3.785e-02  1.190e-02   3.180 0.001471 **
GoodsPrice    -1.170e-07  2.595e-08  -4.510 6.49e-06 ***
RegPopulation  1.828e-01  8.191e-01   0.223 0.823434
EmpPhone       2.526e-01  3.208e-02   7.876 3.39e-15 ***
RegRating2     2.800e-01  3.840e-02   7.291 3.07e-13 ***
RegRating3     5.137e-01  4.417e-02  11.631  < 2e-16 ***
LiveMatch      1.796e-01  3.024e-02   5.939 2.87e-09 ***
WorkMatch      4.269e-02  2.146e-02   1.990 0.046634 *
Def30          1.766e-01  1.646e-02  10.730  < 2e-16 ***
EnquiriesYr    2.907e-02  4.870e-03   5.969 2.39e-09 ***
Scores        -3.814e+00  6.143e-02 -62.094  < 2e-16 ***
Secondary      3.519e-01  2.072e-02  16.988  < 2e-16 ***
FamCivil       1.222e-01  2.676e-02   4.565 4.99e-06 ***
FamSingle      8.773e-02  2.385e-02   3.679 0.000234 ***
HouseParents   4.477e-02  3.469e-02   1.291 0.196809
Age           -1.038e-02  9.958e-04 -10.428  < 2e-16 ***
DaysRegist    -1.778e-05  2.653e-06  -6.701 2.06e-11 ***
DaysId        -4.271e-05  5.860e-06  -7.288 3.16e-13 ***
DaysPhone     -9.274e-05  1.123e-05  -8.258  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Figure 5: summary of the model with all the predictors included

We see from Figure 5 that the predictor with the largest p-value and greater than 0.05 is “HouseParents.” Thus, there is no reason in keeping this predictor as it is statistically insignificant. When we drop one predictor in a model, the p-values for the other predictors will change. Once we dropped the second predictor, the remaining predictors turn out to be significant which left us with 18 predictors. This still seems to be too many variables to be

included in a final model. Even so, let us look at the performance of this model using the testing dataset.

	Predicted: No	Predicted: Yes
Actual: No	94,279	3
Actual: Yes	8,214	3

Table 6: Confusion matrix showing the performance of the 18-predictors model

As we can see, the values highlighted in green are the number of predictions that our model correctly predicted. If we compute the overall performance, this model has an accuracy of about 92%, which is very high. Certainly, we want a model with high accuracy. Nevertheless, if we look again at Table 6, there are only 3 people that are correctly predicted to have payment difficulties. So, what is going on here? It turns out that our dataset is very imbalance. From about 300,000 clients in our dataset, only about 8% of them have payment difficulties. This is a very common problem which tend to make models to be biased [4]. At this point, reducing or adding variables does not affect the performance of the model too much.

Before we move on to the next part, let us further reduce the number of predictors to make our model less complex. We can do this using the same method as before, but this time, we will not use any cutoff value. We will just eliminate any predictor with the largest p-value, once at a time. This results in a 10-predictors model with the following summary.

```
Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.720e+00  7.256e-02 -23.699   <2e-16 ***
LoanType     4.902e-01  3.319e-02  14.769   <2e-16 ***
Gender       1.984e-01  1.712e-02  11.589   <2e-16 ***
EmpPhone     2.767e-01  3.111e-02   8.895   <2e-16 ***
RegRating2   3.082e-01  3.282e-02   9.392   <2e-16 ***
RegRating3   5.538e-01  3.680e-02  15.048   <2e-16 ***
Def30        1.811e-01  1.643e-02  11.023   <2e-16 ***
Scores      -3.927e+00  6.055e-02 -64.844   <2e-16 ***
Secondary    3.632e-01  2.051e-02  17.706   <2e-16 ***
Age         -1.595e-02  8.852e-04 -18.021   <2e-16 ***
DaysPhone   -1.003e-04  1.105e-05  -9.076   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Figure 6: summary of the 10-predictors model

All the predictors in this model have extremely small p-values that R could not determine the exact values. Thus, we stopped removing more predictors and we are going to use the 10-predictors model for the next part.

Classification

As have already mentioned previously, because our dataset is very imbalance, there is not much that we can do to improve the performance of the model. If we stop here, all that we have done becomes meaningless because the loan provider could just approve every application and would still get 92% accuracy. However, there is one last thing that we can do, and it is something that is also common in practice when using logistic regression. That is, classification.

Recall that logistic regression has the form,

$$f(x) = \frac{e^{g(x)}}{1 + e^{g(x)}}.$$

The models that we produced in the previous part are the $g(x)$s. Remember that $f(x)$ or the prediction, can be considered as probability since it is constrained between 0 and 1. Now, how does the confusion matrix like Table 6 is produced? In other words, how does the model decide

which clients fall under “payment difficulties” and which clients fall under “no payment difficulties?” It is done by setting the threshold at a certain level. The confusion matrix in Table 6 was produced using the default threshold level which is 0.5. That means, if a client gets a probability that is greater than 0.5, he/she will be classified as a client with payment difficulties. Changing the threshold level would allow us to classify more accurately clients with payment difficulties. Nonetheless, overall accuracy is not going to improve. In fact, overall accuracy will decrease. This is because there will be more errors in predicting clients that don’t have payment difficulties.

The question now would be whether it is worth to reject some clients who are capable of paying the loans in order to reduce the approvals of clients who are risky. This brings us to the next questions, how much could a loan provider possibly loss from a client who default? And how much could they possibly loss from rejecting a capable client? Some assumptions need to be made to answer these questions. According to some experts, banks or loan providers can become insolvent when too many of their borrowers default [9]. It makes sense why banks and loan providers choose their clients very carefully. Some banks even require clients to have excellent credit scores to get approved for loans or credit cards. Hence, it is reasonable to assume that the amount that could be loss from approving clients who are risky far outweighs the amount that could be loss from rejecting capable clients. We will need to make assumptions on the possible amounts that could be loss for both cases so that we can simulate and see the best threshold level to take to minimize the loss.

Final Model

We will need to decide on the final model first before moving on to the simulations. As mentioned before, reducing the number of predictors in the model does not seems to affect the performance too much. Thus, we might think that the 10-predictors model should be the final model since it is much simpler than the 18-predictors model. It also has a similar performance, that is, the values in the confusion matrix are only slightly different. However, would the values still be similar if we change the threshold level? Let us compare the performance at the default threshold level with the performance at, for instance, 0.2 threshold level. At 0.2, if a client gets a probability greater than 0.2, he/she will be considered a risky borrower.

	Predicted: No	Predicted: Yes
Actual: No	94,279	3
Actual: Yes	8,214	3

	Predicted: No	Predicted: Yes
Actual: No	94,280	2
Actual: Yes	8,215	2

Tables 7 & 8: Confusion matrix showing the performance of the 18-predictors model (left) and the performance of the 10-predictors model (right) at 0.5 threshold level

	Predicted: No	Predicted: Yes
Actual: No	90,264	4,018
Actual: Yes	6,869	1,348

	Predicted: No	Predicted: Yes
Actual: No	90,364	3,918
Actual: Yes	6,931	1,286

Tables 9 & 10: Confusion matrix showing the performance of the 18-predictors model (left) and the performance of the 10-predictors model (right) at 0.2 threshold level

This time, we want to focus on the errors, which are the highlighted values. The values highlighted in orange are the number of capable clients who are predicted to be risky by our models (Type I errors). While the values highlighted in yellow are clients with payment difficulties but are predicted as capable by our model (Type II errors). If we compare the

performance of the 18-predictors model and 10-predictors model at the default threshold level, we can see that the 10-predictors model produces one extra type II error but it also produces one less of type I error. Now, let us refer to the performance at the threshold of 0.2. Although the 10-predictors model has lesser type I errors than the 18-predictors model by 100, it produces 62 more of type II errors. Remember that our assumption is that the amount that could be loss from approving clients who are risky far outweigh the amount that could be loss from rejecting capable clients. In other words, the loss from having equal amount of type II errors is much greater than the loss from having equal amount of type I errors.

Let now assumes that the loss from rejecting a capable client is $1,000 and the loss from approving risky clients is $5,000. This means that if we use the 10-predictors model, we would lose $210,000 more as compare to using the 18-predictors model. It is computed by subtracting the total amount that could be loss from the 10-predictos model by the total amount that could be loss from the 18-predictors model. For some banks or loan providers, a loss of $210,000 might not be a big issue if it means they can spend less time collecting data about their clients and use a simpler model. However, we can improve the 10-predictors model by adding two variables (which were removed previously) that turn out to be quite important, which are "LiveMatch" and "DaysId." With these variables added, we now have a 12-predictors model with the following performance at threshold level of 0.2.

	Predicted: No	Predicted: Yes
Actual: No	90,346	3,936
Actual: Yes	6,902	1,315

Table 11: Confusion matrix showing the performance of the 12-predictors model at 0.2 threshold level

If we were to use this model, we would still lose more compare to using the 18-predictors model. But the loss is much lesser compare to the 10-predictors model where the difference with the 18-predictors model is $83,000 as oppose to $210,000. At this point, it is difficult to find the perfect model. The model would either be too complex or would give more costly errors. Moreover, the best model would also be subject to the ratio of the possible amount loss between the two errors. Nevertheless, we will choose the 12-predictors model as our final model. This is because the 12-predictors model provides significant improvement over the 10-predictors model in terms of the amount loss and it also has lesser predictors than the 18-predictors model. Although, as we have discussed, the 18-predictors still has an advantage in terms of the amount loss compare to the 12-predictors model, the difference is not very significant. Thus, it is reasonable to choose the 12-predictors model. We will use this model to do some simulations on the next part. Our final model has the following equation,

$$g(x) = 0.5 \cdot LoanType + 0.205 \cdot Gender + 0.253 \cdot EmpPhone + 0.303 \cdot RegRating2 + 0.546 \cdot RegRating3 + 0.212 \cdot LiveMatch + 0.179 \cdot Def - 3.88 \cdot Scores + 0.365 \cdot Secondary - 0.0139 \cdot Age - 0.00004 \cdot DaysId - 0.000092 \cdot DaysPhone$$

We will be able to compute the probability of whether a client would have payment difficulties by plugging in $g(x)$ into the logistic equation, $f(x)$.

Simulations to Minimize Loss

Because we don't know what the actual amount (on average) is that could be loss from rejecting a capable client as well as accepting a risky client, we will simulate different scenarios using our final model. The objective is to find a threshold level at each case that would

minimize the possible amount that could be loss. We will simulate the following three scenarios.

Loss from rejecting a capable client ($)	Loss from accepting a risky client ($)
1,000	3,000
1,000	5,000
1,000	10,000

Table 12: Assumptions on the possible amount that could be loss from rejecting a capable client and loss from accepting a risky client, which will be used for simulations

The way we are going to simulate this is by creating a program in R (see *Appendix D*) that will compute the number of capable clients who are considered risky by our model (type I errors) and the number of risky clients who are considered capable by our model (type II errors) at each threshold level from 0.01 to 0.5. Once these have been computed, at each threshold level, the total amount that could be loss will be calculated. Then, a new dataset will be created that will store each threshold value and its corresponding total amount that could be loss. Finally, we will plot these values and find the global minimum, which is the point where the total amount that could be loss is the smallest.

At 1000:3000 ratio:

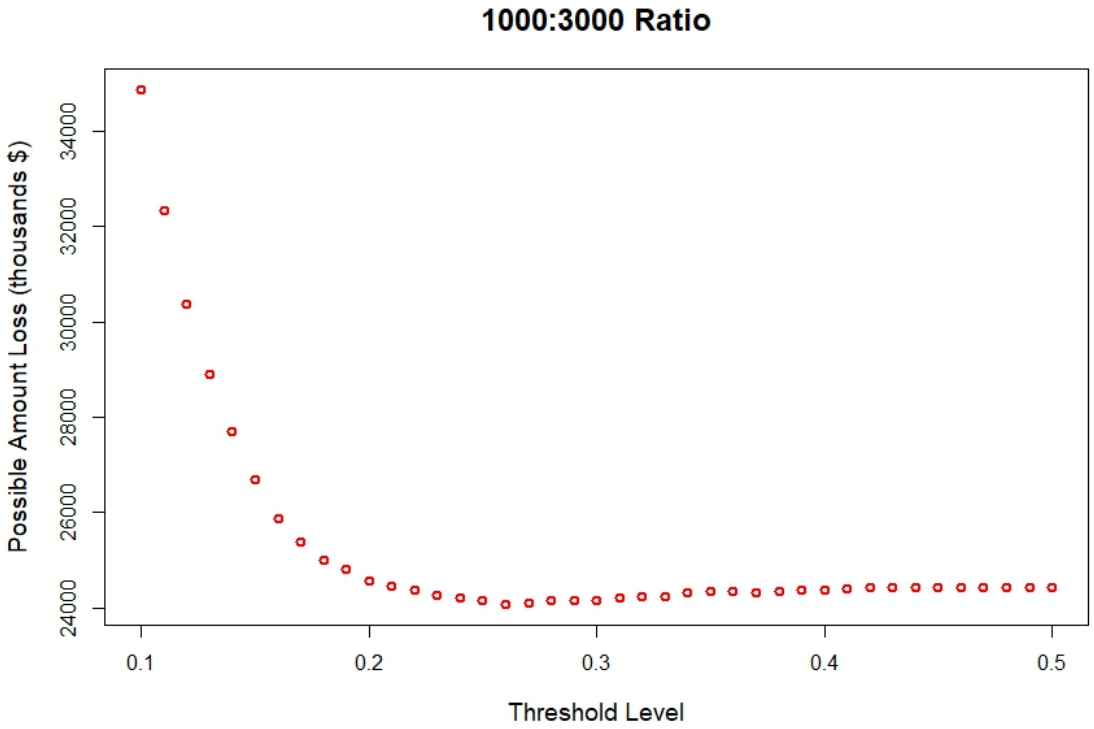

Figure 7: Plot of the possible amount loss vs. threshold level when the ratio is 1000:3000

From the above plot (Figure 7), we can see that the lowest point is at threshold level of 0.26. It is the threshold level that our model suggests us to use to minimize the loss when the ratio is one thousand to three thousand. However, as we can see, even if we use higher threshold levels, the amount that could be loss only changes slightly.

At 1000:5000 ratio:

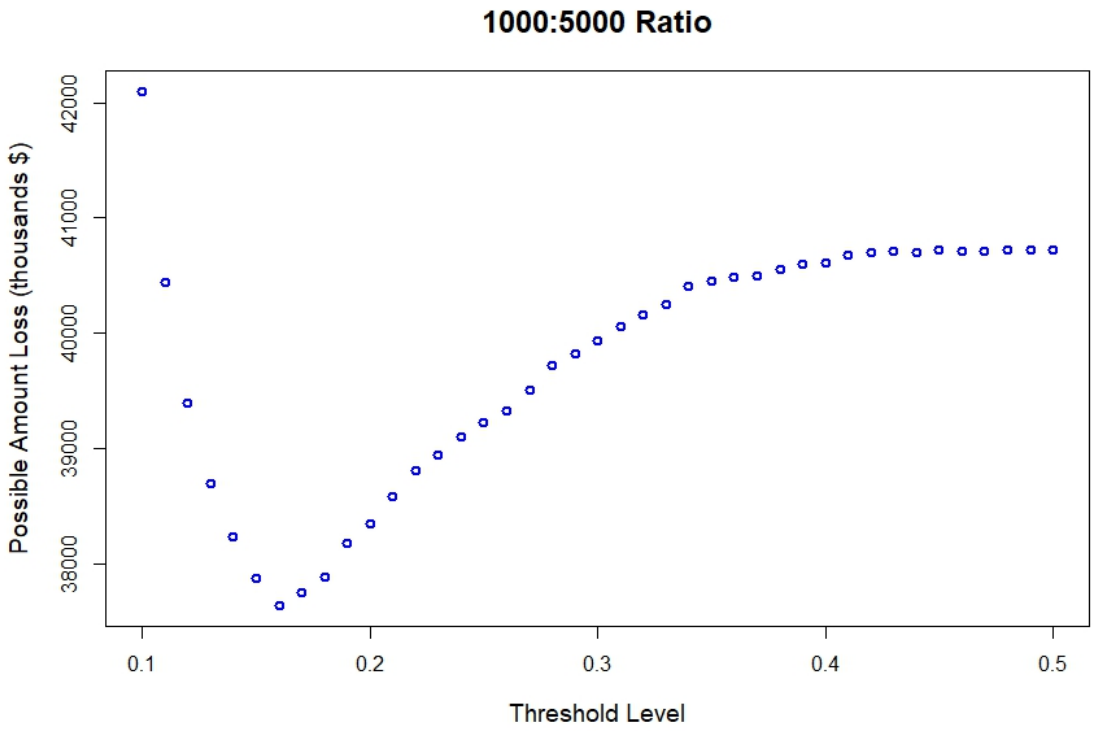

Figure 8: Plot of the possible amount loss vs. threshold level when the ratio is 1000:5000

When the amount that could be loss from accepting a risky client is five times greater than the amount that could be loss from rejecting a capable client, a very different plot is produced as can be seen in Figure 8, compare to the previous one (Figure 7). Clearly from this plot, our model suggests us to use a threshold level of 0.16 to minimize the amount that could be loss if this ratio is true. Using threshold level other than 0.16 would give a much greater loss as can be seen from the plot.

At 1000:10000 ratio:

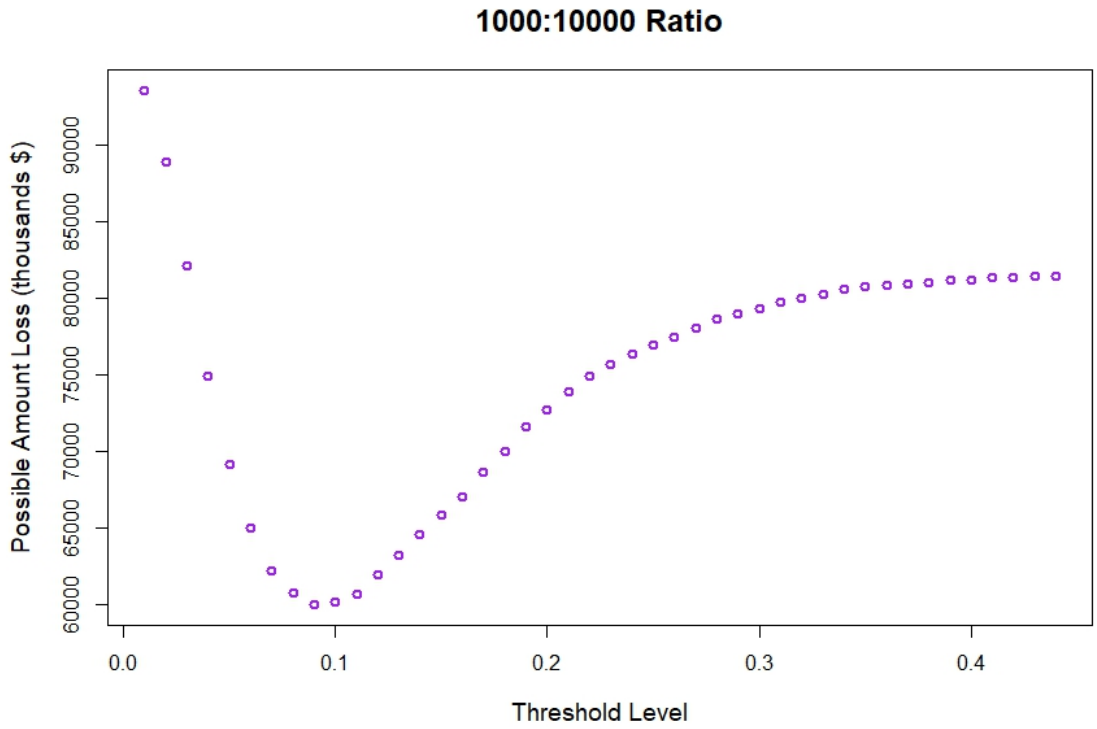

Figure 9: Plot of the possible amount loss vs. threshold level when the ratio is 1000:5000

Now, if the amount that could be loss from accepting a risky client is ten times greater than the amount that could be loss from rejecting a capable client, we need an even lower threshold level to minimize the loss as can be seen in Figure 9. The threshold level that should be used to minimize the loss would be 0.09.

Summary of the Simulations:

From the simulations, we have determined, at each case, the threshold level that should be used to minimize the amount that could be loss using our final model. The table below shows the summary of our simulations.

Loss from rejecting a capable client ($)	Loss from accepting a risky client ($)	Threshold level that should be used
1,000	3,000	0.26
1,000	5,000	0.16
1,000	10,000	0.09

Table 13: Summary of the simulations, indicating the threshold level that should be used in each case so that the possible amount that could be loss is minimized

We have already seen that as the amount that could be loss from accepting a risky client gets much higher than the amount that could be loss from rejecting a capable client, the threshold level that minimizes the loss becomes lower. However, we have also observed from the first simulation, when the amount that could be loss from accepting a risky client is only three times greater than the amount that could be loss from rejecting a capable client, using higher threshold levels do not increase the possible amount that could be loss by much. If this is the case, using the default threshold level might be a better option. As we have already known, at the default threshold level, our model would predict that almost everyone does not have payment difficulties. If this is implemented in a real situation, loan providers would basically just approve almost every applicant. This might be better because they do not have to do more works analyzing and rejecting more clients just to reduce the loss that is not worth that much. However, again, if the ratio is like one thousand to five thousand or one thousand to ten thousand, it is better to stick with the threshold level shown in Table 13. Otherwise, the possible amount that could be loss would be much greater.

Conclusions

Through our analysis, we have seen that threshold level plays an important role in the process of approving and rejecting loan applications. So, who should get the loan and who shouldn't? It is not so simple answering this question as we have already seen. It depends on the companies or loan providers. Some loan providers focus more on clients with low to no credit history like Home Credit while others will give the loans only to clients with high income and excellent credit history. Deciding which clients should be given the loan would also depends on the model that is used and the total amount that could be loss from both the errors (type I errors and type II errors). Some would prefer the more complex model if it means the loss can be fully minimized. But others might prefer simpler model with less predictors even though they might experience slightly higher loss.

However, we will answer the question with the analysis that we have already done. Since our model was created using the dataset from Home Credit Group, our answers would only be relevant to Home Credit Group or other loan providers that are similar to them. To answer the question of who should be given the loan, we would compute every client's probability of being risky using our model. Through our simulation, when the possible amount that could be loss from approving risky clients is five times greater than the possible amount that could be loss from rejecting capable clients, clients with probabilities greater than 0.16 would be considered risky and thus would highly be rejected. Similarly, when the possible amount that could be loss from approving risky clients is ten times greater than the possible amount that could be loss from rejecting capable clients, clients with probabilities greater than 0.09 would be considered risky and would highly be rejected. On the other hand, if the possible

amount that could be loss from approving risky clients is not so much greater than the possible amount that could be loss from rejecting capable clients, say three times greater or even less, then it might be reasonable to use the default threshold level as we have already discussed before. This means that if clients have probabilities greater than 0.5, they would be considered risky and would highly be rejected. Although, as we have already known, almost 100% of the clients would probably be approved at the default threshold level.

Strengths and Weaknesses

One of the strengths is the dataset itself which provides a lot of information. This helps us to gain greater insights. Although the dataset is part of the strength, it is also one of the weaknesses because of how imbalance it is which makes our model to be biased. Even so, we managed to find a way to classify clients by using the amount that could be loss as a criteria and modifying the threshold level. The greatest strength would probably be the program that is used for the simulations. Once the actual amounts loss (on average) from both case of rejecting capable clients and approving risky clients are known, it is just a matter of plugging the values into the program. Within seconds, a nice plot will be produced which allow loan providers to decide on a threshold level that will minimize loss.

Nevertheless, classifying clients the way we did is also another weakness because of the assumption that all risky clients would default on their loans. Also, as mentioned previously, this analysis is only relevant to Home Credit Group or other loan providers similar to them. Other banks or loan providers might have different clients (i.e. all their clients have exceptional credit scores) which make their data to be different than that of Home Credit Group.

References

[1] About us. (n.d.). Retrieved April 19, 2019, from http://www.homecredit.net/about-us.aspx

[2] Cannon, A. R., Cobb, G. W., Hartlaub, B. A., Legler, J. M., Lock, R. H., Moore, T. L., Witmer, J. A. (2013). Logistic Regression. In *STAT2: Building Models for a World of Data* (pp. 449-580). New York: W.H. Freeman.

[3] Cannon, A. R., Cobb, G. W., Hartlaub, B. A., Legler, J. M., Lock, R. H., Moore, T. L., Witmer, J. A. (2013). Multiple Regression. In *STAT2: Building Models for a World of Data* (pp. 95-149). New York: W.H. Freeman.

[4] Class Imbalance Problem. (2013, August 30). Retrieved from http://www.chioka.in/class-imbalance-problem/

[5] Faraway, J. J. (2016). Binary Response. In *Extending the linear model with R: Generalized linear, mixed effects and nonparametric regression models* (pp. 25-44). Boca Raton: CRC Press.

[6] Home Credit Default Risk. (2018, May 17). Retrieved from https://www.kaggle.com/c/home-credit-default-risk/data

[7] Kotu, V., & Deshpande, B. (2015). Data Mining Process. In *Predictive Analytics and Data Mining: Concepts and Practice with RapidMiner* (pp. 17-34). Amsterdam: Elsevier/Morgan Kaufmann.

[8] Kotu, V., & Deshpande, B. (2015). Regression Methods. In *Predictive Analytics and Data Mining: Concepts and Practice with RapidMiner* (pp. 165-180). Amsterdam: Elsevier/Morgan Kaufmann.

[9] Ryan-Collins, J., Greenham, T., Werner, R., & Jackson, A. (n.d.). How do Banks Become Insolvent? Retrieved from https://positivemoney.org/how-money-works/advanced/how-do-banks-become-insolvent/

Appendix A – Simple illustrations of Linear and Logistic Regression

```
## Example of Linear Regression ##

#Simple dataset for illustration purposes
x=c(1,2,3,4,5)
y=c(3,5,7,10,12)
HW <- data.frame(x,y)

#Linear regression model
(linear_model <- lm(y ~ x, data=HW))
plot(HW$x, HW$y, xlab="No. of Hours Spent", ylab="No. of Questions
Completed", main="No. of Hours Spent vs. No.of Homework Questions Completed"
     , cex.lab=1.2)
abline(linear_model)

## Example of Logistic Regression ##

#Simple dataset for illustration purposes
w <- c(1,0,0,0,1,1,0,1,0,1,0,1,0,0,0,1,1,1,0,0,0,1,1,1,0,1,0,1,1,0,1,0,0,0)
z <- c(470,781,734,483,345,200,689,205,700,278,750,223,689,703,767,
       0,298,270,700,788,777,0,232,333,732,0,800,0,203,695,495,572,500,400)
loan_payment <- data.frame(w,z)

#Discrete data modeled using linear regression and its plot
(linear_model_2 <- lm(w ~ z, data=loan_payment))
plot(jitter(loan_payment$z,0.1), jitter(loan_payment$w,0.1),ylab="Payment
Difficulties?", xlab="Credit Scores",
     main="Credit Scores vs. Whether Clients will have Payment Difficulties
", cex.lab=1.2)
abline(linear_model_2,lwd=2, col="blue")  #Regression line

#Discrete data modeled using logistic regression and its plot
(logistic_model <- glm(w ~ z , data = loan_payment, family = "binomial"))
ww <- seq(0,850,10)
zz <- predict(logistic_model, list(z = ww),type="response")
plot(jitter(loan_payment$z,0.1), jitter(loan_payment$w,0.1),ylab="Probability
of having Payment Difficulties",
     xlab="Credit Scores", main="Credit Scores vs. Probability of having
Payment Difficulties ", cex.lab=1.2)
lines(ww,zz,lwd=2,col="blue")  #Logistic curve
```

Appendix B **– Data Preparation**

```
#Import Home Credit Group data into R
Loan <- read.csv(file="c:/Users/jezen/OneDrive/Desktop/SPU/Fall'18/MAT
4899/LoanDifficulty.csv", header=TRUE, sep=",")
nrow(Loan)
table(Loan$TARGET)

#Check missing values
colSums(is.na(Loan))
names(Loan)

#Remove variables with too many mising values
loan2 <- Loan[c(1:21, 23:44, 87:88, 90:122)]
colSums(is.na(loan2))

#Summary of variables
summary(loan2$AMT_ANNUITY)
summary(loan2$AMT_GOODS_PRICE)
summary(loan2$CNT_FAM_MEMBERS)
summary(loan2$OBS_30_CNT_SOCIAL_CIRCLE)
summary(loan2$OBS_60_CNT_SOCIAL_CIRCLE)
summary(loan2$DEF_30_CNT_SOCIAL_CIRCLE)
summary(loan2$DEF_60_CNT_SOCIAL_CIRCLE)
summary(loan2$AMT_REQ_CREDIT_BUREAU_HOUR)
summary(loan2$AMT_REQ_CREDIT_BUREAU_DAY)
summary(loan2$AMT_REQ_CREDIT_BUREAU_WEEK)
summary(loan2$AMT_REQ_CREDIT_BUREAU_MON)
summary(loan2$AMT_REQ_CREDIT_BUREAU_QRT)
summary(loan2$AMT_REQ_CREDIT_BUREAU_YEAR)

#Replace missing values
loan2$CNT_FAM_MEMBERS[which(is.na(loan2$CNT_FAM_MEMBERS))] <- 0

loan2$AMT_GOODS_PRICE[which(is.na(loan2$AMT_GOODS_PRICE))] <-
mean(loan2$AMT_GOODS_PRICE, na.rm = TRUE)

loan2$OBS_30_CNT_SOCIAL_CIRCLE[which(is.na(loan2$OBS_30_CNT_SOCIAL_CIRCLE))]
<- mean(loan2$OBS_30_CNT_SOCIAL_CIRCLE, na.rm = TRUE)
loan2$OBS_60_CNT_SOCIAL_CIRCLE[which(is.na(loan2$OBS_60_CNT_SOCIAL_CIRCLE))]
<- mean(loan2$OBS_60_CNT_SOCIAL_CIRCLE, na.rm = TRUE)
loan2$DEF_30_CNT_SOCIAL_CIRCLE[which(is.na(loan2$DEF_30_CNT_SOCIAL_CIRCLE))]
<- mean(loan2$DEF_30_CNT_SOCIAL_CIRCLE, na.rm = TRUE)
loan2$DEF_60_CNT_SOCIAL_CIRCLE[which(is.na(loan2$DEF_60_CNT_SOCIAL_CIRCLE))]
<- mean(loan2$DEF_60_CNT_SOCIAL_CIRCLE, na.rm = TRUE)

loan2$AMT_REQ_CREDIT_BUREAU_HOUR[which(is.na(loan2$AMT_REQ_CREDIT_BUREAU_HOUR
))] <- mean(loan2$AMT_REQ_CREDIT_BUREAU_HOUR, na.rm = TRUE)
loan2$AMT_REQ_CREDIT_BUREAU_DAY[which(is.na(loan2$AMT_REQ_CREDIT_BUREAU_DAY))
] <- mean(loan2$AMT_REQ_CREDIT_BUREAU_DAY, na.rm = TRUE)
loan2$AMT_REQ_CREDIT_BUREAU_WEEK[which(is.na(loan2$AMT_REQ_CREDIT_BUREAU_WEEK
))] <- mean(loan2$AMT_REQ_CREDIT_BUREAU_WEEK, na.rm = TRUE)
loan2$AMT_REQ_CREDIT_BUREAU_MON[which(is.na(loan2$AMT_REQ_CREDIT_BUREAU_MON))
] <- mean(loan2$AMT_REQ_CREDIT_BUREAU_MON, na.rm = TRUE)
loan2$AMT_REQ_CREDIT_BUREAU_QRT[which(is.na(loan2$AMT_REQ_CREDIT_BUREAU_QRT))
] <- mean(loan2$AMT_REQ_CREDIT_BUREAU_QRT, na.rm = TRUE)
```

```
loan2$AMT_REQ_CREDIT_BUREAU_YEAR[which(is.na(loan2$AMT_REQ_CREDIT_BUREAU_YEAR
))] <- mean(loan2$AMT_REQ_CREDIT_BUREAU_YEAR, na.rm = TRUE)

loan2$EXT_SOURCE_1[which(is.na(loan2$EXT_SOURCE_1))] <- 0
loan2$EXT_SOURCE_2[which(is.na(loan2$EXT_SOURCE_2))] <- 0
loan2$EXT_SOURCE_3[which(is.na(loan2$EXT_SOURCE_3))] <- 0

colSums(is.na(loan2))
sum(is.na(loan2))
names(loan2)

#Omit remaining missing values (AMT_ANNUITY = 12 missing)
loan3 <- na.omit(loan2)

#Take the average of the three credit scores from different sources and store
in a new variable called 'Scores'
library(dplyr)
loan4 <- mutate(loan3, Scores = (EXT_SOURCE_1 + EXT_SOURCE_2 +
EXT_SOURCE_3)/3)
names(loan4)

#Remove EXT_SOURCE 1-3 since already have average score
loan5 <- loan4[c(1:40, 44:79)]
names(loan5)
colSums(is.na(loan5))

#Check frequencies#
#Loan Type (Cash loans, Revolving loan)
t = table(loan5$NAME_CONTRACT_TYPE)
as.data.frame(t)

#Gender (M, F, XNA), Omit the 4 rows with gender XNA
g = table(loan5$CODE_GENDER)
as.data.frame(g)
which(loan5$CODE_GENDER == "XNA")
loan5 <- loan5[-35657, ]
loan5 <- loan5[-38565, ]
loan5 <- loan5[-83374, ]
loan5 <- loan5[-189629, ]

#OwnCar (Y, N)
c = table(loan5$FLAG_OWN_CAR)
as.data.frame(c)

#OwnHouse (Y, N)
h = table(loan5$FLAG_OWN_REALTY)
as.data.frame(h)

#AccompaniedBy (Children,Family,Group of people, Other_A, Other_B, Spouse,
partner, Unaccompanied)
a = table(loan5$NAME_TYPE_SUITE)
as.data.frame(a)

#IncomeType (Businessman, Commercial associate, Maternity leave, Pensioner,
State Servant
# Student, Unemployed, Working)
i = table(loan5$NAME_INCOME_TYPE)
```

```
as.data.frame(i)

#Education (Academic degree, Higher education, Incomplete higher, Lower
secondary,
# Secondary / secondary special)
e = table(loan5$NAME_EDUCATION_TYPE)
as.data.frame(e)

#FamilyStatus (Civil marriage, Married, Separated, Single / not married,
Unknown, Widow)
fs = table(loan5$NAME_FAMILY_STATUS)
as.data.frame(fs)

#HouseType (Co-op apartment, House / apartment, Municipal apartment, Office
apartment,
# Rented apartment, With parents)
ht = table(loan5$NAME_HOUSING_TYPE)
as.data.frame(ht)

#Organization (too much)
o = table(loan5$ORGANIZATION_TYPE)
as.data.frame(o)

#Occupation (19)
oc = table(loan5$OCCUPATION_TYPE)
as.data.frame(oc)

#Housetype_mode
hm = table(loan5$HOUSETYPE_MODE)
as.data.frame(hm)

#Proportion tables for binary and categorical variables
prop.table(table(loan5$TARGET, loan5$NAME_CONTRACT_TYPE),margin=2)
prop.table(table(loan5$TARGET, loan5$CODE_GENDER),margin=2)
prop.table(table(loan5$TARGET, loan5$FLAG_OWN_CAR),margin=2)
prop.table(table(loan5$TARGET, loan5$FLAG_OWN_REALTY),margin=2)
prop.table(table(loan5$TARGET, loan5$NAME_TYPE_SUITE),margin=2)
prop.table(table(loan5$TARGET, loan5$NAME_INCOME_TYPE),margin=2)
prop.table(table(loan5$TARGET, loan5$NAME_EDUCATION_TYPE),margin=2)
prop.table(table(loan5$TARGET, loan5$NAME_FAMILY_STATUS),margin=2)
prop.table(table(loan5$TARGET, loan5$NAME_HOUSING_TYPE),margin=2)
prop.table(table(loan5$TARGET, loan5$FLAG_MOBIL),margin=2)
prop.table(table(loan5$TARGET, loan5$FLAG_EMP_PHONE),margin=2)
prop.table(table(loan5$TARGET, loan5$FLAG_WORK_PHONE),margin=2)
prop.table(table(loan5$TARGET, loan5$FLAG_CONT_MOBILE),margin=2)
prop.table(table(loan5$TARGET, loan5$FLAG_PHONE),margin=2)
prop.table(table(loan5$TARGET, loan5$FLAG_EMAIL),margin=2)
prop.table(table(loan5$TARGET, loan5$OCCUPATION_TYPE),margin=2)
prop.table(table(loan5$TARGET, loan5$REG_REGION_NOT_LIVE_REGION),margin=2)
prop.table(table(loan5$TARGET, loan5$REG_REGION_NOT_WORK_REGION),margin=2)
prop.table(table(loan5$TARGET, loan5$LIVE_REGION_NOT_WORK_REGION),margin=2)
prop.table(table(loan5$TARGET, loan5$REG_CITY_NOT_LIVE_CITY),margin=2)
prop.table(table(loan5$TARGET, loan5$REG_CITY_NOT_WORK_CITY),margin=2)
prop.table(table(loan5$TARGET, loan5$LIVE_CITY_NOT_WORK_CITY),margin=2)
prop.table(table(loan5$TARGET, loan5$ORGANIZATION_TYPE),margin=2)
prop.table(table(loan5$TARGET, loan5$FLAG_DOCUMENT_2),margin=2)
prop.table(table(loan5$TARGET, loan5$FLAG_DOCUMENT_3),margin=2)
```

```
prop.table(table(loan5$TARGET, loan5$FLAG_DOCUMENT_4),margin=2)
prop.table(table(loan5$TARGET, loan5$FLAG_DOCUMENT_5),margin=2)
prop.table(table(loan5$TARGET, loan5$FLAG_DOCUMENT_6),margin=2)
prop.table(table(loan5$TARGET, loan5$FLAG_DOCUMENT_7),margin=2)
prop.table(table(loan5$TARGET, loan5$FLAG_DOCUMENT_8),margin=2)
prop.table(table(loan5$TARGET, loan5$FLAG_DOCUMENT_9),margin=2)
prop.table(table(loan5$TARGET, loan5$FLAG_DOCUMENT_10),margin=2)
prop.table(table(loan5$TARGET, loan5$FLAG_DOCUMENT_11),margin=2)
prop.table(table(loan5$TARGET, loan5$FLAG_DOCUMENT_12),margin=2)
prop.table(table(loan5$TARGET, loan5$FLAG_DOCUMENT_13),margin=2)
prop.table(table(loan5$TARGET, loan5$FLAG_DOCUMENT_14),margin=2)
prop.table(table(loan5$TARGET, loan5$FLAG_DOCUMENT_16),margin=2)
prop.table(table(loan5$TARGET, loan5$FLAG_DOCUMENT_17),margin=2)
prop.table(table(loan5$TARGET, loan5$FLAG_DOCUMENT_18),margin=2)
prop.table(table(loan5$TARGET, loan5$FLAG_DOCUMENT_19),margin=2)
prop.table(table(loan5$TARGET, loan5$FLAG_DOCUMENT_20),margin=2)
prop.table(table(loan5$TARGET, loan5$FLAG_DOCUMENT_21),margin=2)

loan6 <- loan5[c(2:31, 34:39, 41, 45:49, 70:76)]
colSums(is.na(loan6))
loan6 <- na.omit(loan6)
sum(is.na(loan6))

#Convert nominal variables into numeric
loan6$CODE_GENDER <- as.numeric(loan6$CODE_GENDER == "M")
loan6$NAME_CONTRACT_TYPE <- as.numeric(loan6$NAME_CONTRACT_TYPE == "Cash
loans")
loan6$FLAG_OWN_CAR <- as.numeric(loan6$FLAG_OWN_CAR == "Y")
loan6$FLAG_OWN_REALTY <- as.numeric(loan6$FLAG_OWN_REALTY == "Y")

loan6$AccByChildren <- as.numeric(loan6$NAME_TYPE_SUITE == "Children")
loan6$AccByFamily <- as.numeric(loan6$NAME_TYPE_SUITE == "Family")
loan6$AccByGroup <- as.numeric(loan6$NAME_TYPE_SUITE == "Group of people")
loan6$AccBySpouse <- as.numeric(loan6$NAME_TYPE_SUITE == "Spouse, partner")
loan6$AccByNone <- as.numeric(loan6$NAME_TYPE_SUITE == "Unaccompanied")
loan6$AccByUnknown <- as.numeric(loan6$NAME_TYPE_SUITE == "")

loan6$IncomeBusiness <- as.numeric(loan6$NAME_INCOME_TYPE == "Businessman")
loan6$IncomeCommercial <- as.numeric(loan6$NAME_INCOME_TYPE == "Commercial
associate")
loan6$IncomeMaternity <- as.numeric(loan6$NAME_INCOME_TYPE == "Maternity
leave")
loan6$IncomePensioner <- as.numeric(loan6$NAME_INCOME_TYPE == "Pensioner")
loan6$IncomeState <- as.numeric(loan6$NAME_INCOME_TYPE == "State servant")
loan6$IncomeStudent <- as.numeric(loan6$NAME_INCOME_TYPE == "Student")
loan6$IncomeUnemployed <- as.numeric(loan6$NAME_INCOME_TYPE == "Unemployed")
loan6$IncomeWorking <- as.numeric(loan6$NAME_INCOME_TYPE == "Working")

loan6$AcademicDegree <- as.numeric(loan6$NAME_EDUCATION_TYPE == "Academic
degree")
loan6$HigherEdu <- as.numeric(loan6$NAME_EDUCATION_TYPE == "Higher
education")
loan6$IncompleteEdu <- as.numeric(loan6$NAME_EDUCATION_TYPE == "Incomplete
higher")
loan6$LowerSec <- as.numeric(loan6$NAME_EDUCATION_TYPE == "Lower seconadry")
loan6$Secondary <- as.numeric(loan6$NAME_EDUCATION_TYPE == "Secondary /
secondary special")
```

```
loan6$FamCivil <- as.numeric(loan6$NAME_FAMILY_STATUS == "Civil marriage")
loan6$FamMarried <- as.numeric(loan6$NAME_FAMILY_STATUS == "Married")
loan6$FamSeparated <- as.numeric(loan6$NAME_FAMILY_STATUS == "Separated")
loan6$FamSingle <- as.numeric(loan6$NAME_FAMILY_STATUS == "Single / not
married")
loan6$FamWidow <- as.numeric(loan6$NAME_FAMILY_STATUS == "Widow")

loan6$HouseCoop <- as.numeric(loan6$NAME_HOUSING_TYPE == "Co-op apartment")
loan6$HouseOrApt <- as.numeric(loan6$NAME_HOUSING_TYPE == "House /
apartment")
loan6$HouseMunicipal <- as.numeric(loan6$NAME_HOUSING_TYPE == "Municipal
apartment")
loan6$HouseOffice <- as.numeric(loan6$NAME_HOUSING_TYPE == "Office
apartment")
loan6$HouseRent <- as.numeric(loan6$NAME_HOUSING_TYPE == "Rented apartment")
loan6$HouseParents <- as.numeric(loan6$NAME_HOUSING_TYPE == "With parents")

loan6$Drivers <- as.numeric(loan6$OCCUPATION_TYPE == "Drivers")
loan6$Laborers <- as.numeric(loan6$OCCUPATION_TYPE == "Laborers")
names(loan6)

#Convert age in days into age in years. Also convert other days' variables
that are negative into positive values
loan7 <- mutate(loan6, Age = (DAYS_BIRTH/-365))
loan7 <- mutate(loan7, DaysEmp = (DAYS_EMPLOYED/-1))
loan7 <- mutate(loan7, DaysRegist = (DAYS_REGISTRATION/-1))
loan7 <- mutate(loan7, DaysId = (DAYS_ID_PUBLISH/-1))
loan7 <- mutate(loan7, DaysPhone = (DAYS_LAST_PHONE_CHANGE/-1))
names(loan7)

#Take all quantitative variables only and check correlations with TARGET
loan.quant <- loan7[c(1,6:10, 16,28,38:41,43:49,82:86)]
names(loan.quant)
corr <- cor(loan.quant)
round(corr,4)

#Remove all insignificant variables based on proportion tables and
correlations
loan8 <- loan7[c(1:3,6, 10, 16, 22, 30, 34:35, 39, 48:49, 68:69, 72,
79,82,84:86)]

#Rename variables for simplicity
names(loan8)
names(loan8)[1] <- "Target"
names(loan8)[2] <- "LoanType"
names(loan8)[3] <- "Gender"
names(loan8)[4] <- "Children"
names(loan8)[5] <- "GoodsPrice"
names(loan8)[6] <- "RegPopulation"
names(loan8)[7] <- "EmpPhone"
names(loan8)[8] <- "RegRating"
names(loan8)[9] <- "LiveMatch"
names(loan8) [10] <- "WorkMatch"
names(loan8)[13] <- "Scores"
names(loan8)[11] <- "Def30"
names(loan8)[12] <- "EnquiriesYr"
```

```
names(loan8)
loan8$RegRating <- as.factor(loan8$RegRating)
str(loan8)

#Split dataset into training and testing sets
library(caTools)
split <- sample.split(loan8, SplitRatio = 0.7)
training <- subset(loan8, split == "TRUE")
testing <- subset(loan8, split == "FALSE")

names(training)
```

Appendix C **– Modeling**

```
#Start with all variables then remove one at a time based on p-value
m.all <- glm(Target ~ ., data = training, family = "binomial")
summary(m.all)

model_1 <- glm(Target ~ .-HouseParents, data = training, family = "binomial")
summary(model_1)

model_2 <- glm(Target ~ .-HouseParents-RegPopulation, data = training, family
= "binomial")
summary(model_2)

#18-predictors model (all p-values < 0.05)
model_3 <- glm(Target ~ .-HouseParents-RegPopulation-WorkMatch, data =
training, family = "binomial")
summary(model_3)

model_4 <- glm(Target ~ .-HouseParents-RegPopulation-WorkMatch-Children, data
= training, family = "binomial")
summary(model_4)

model_5 <- glm(Target ~ .-HouseParents-RegPopulation-WorkMatch-Children
               -FamSingle, data = training, family = "binomial")
summary(model_5)

model_6 <- glm(Target ~ .-HouseParents-RegPopulation-WorkMatch-Children-
FamSingle
               -FamCivil, data = training, family = "binomial")
summary(model_6)

model_7 <- glm(Target ~ .-HouseParents-RegPopulation-WorkMatch-Children-
FamSingle
               -FamCivil-GoodsPrice, data = training, family = "binomial")
summary(model_7)

model_8 <- glm(Target ~ .-HouseParents-RegPopulation-WorkMatch-Children-
FamSingle
               -FamCivil-GoodsPrice-EnquiriesYr, data = training, family =
"binomial")
summary(model_8)

model_9 <- glm(Target ~ .-HouseParents-RegPopulation-WorkMatch-Children-
FamSingle
               -FamCivil-GoodsPrice-EnquiriesYr-DaysRegist, data = training,
family = "binomial")
summary(model_9)

model_10 <- glm(Target ~ .-HouseParents-RegPopulation-WorkMatch-Children-
FamSingle
                -FamCivil-GoodsPrice-EnquiriesYr-DaysRegist-DaysId, data =
training, family = "binomial")
summary(model_10)


#10-predictors model (all predictors have extremely small p-values)
```

```
model_11 <- glm(Target ~ .-HouseParents-RegPopulation-WorkMatch-Children-
FamSingle
                -FamCivil-GoodsPrice-EnquiriesYr-DaysRegist-DaysId
                -LiveMatch, data = training, family = "binomial")
summary(model_11)

#12-predictors model (Final Model)
model_12 <- glm(Target ~ .-HouseParents-RegPopulation-WorkMatch-Children-
FamSingle
                -FamCivil-EnquiriesYr-DaysRegist-GoodsPrice
                , data = training, family = "binomial")
summary(model_12)

#Check performance of models using confusion matrix
resp1 <- predict(model_3, training, type="response")
table(ActualValue=training$Target, PredictiveValue=resp>0.5)
table(ActualValue=training$Target, PredictiveValue=resp>0.2)

resp2 <- predict(model_4, testing, type="response")
table(ActualValue=training$Target, PredictiveValue=resp>0.5)

resp3 <- predict(model_5, testing, type="response")
table(ActualValue=training$Target, PredictiveValue=resp>0.5)

resp4 <- predict(model_6, testing, type="response")
table(ActualValue=training$Target, PredictiveValue=resp>0.5)

resp5 <- predict(model_7, testing, type="response")
table(ActualValue=training$Target, PredictiveValue=resp>0.5)

resp6 <- predict(model_8, testing, type="response")
table(ActualValue=training$Target, PredictiveValue=resp>0.5)

resp7 <- predict(model_9, testing, type="response")
table(ActualValue=training$Target, PredictiveValue=resp>0.5)

resp8 <- predict(model_10, testing, type="response")
table(ActualValue=training$Target, PredictiveValue=resp>0.5)

resp9 <- predict(model_11, testing, type="response")
table(ActualValue=testing$Target, PredictiveValue=resp9>0.5)
table(ActualValue=training$Target, PredictiveValue=resp>0.2)

resp10 <- predict(model_12, testing, type="response")
s=table(ActualValue=testing$Target, PredictiveValue=resp10>0.5)
table(ActualValue=training$Target, PredictiveValue=resp>0.2)

#ROC plot
library(ROCR)
ROCRPred <- prediction(resp, training$Target)
ROCRPerf <- performance(ROCRPred, "tpr", "fpr")
plot(ROCRPerf,colorize=TRUE,print.cutoffs.at=seq(0.1,by=0.1))
```

Appendix D **– Simulations**

```
#Create new datasets
data_1 = numeric(41)
data_2 = numeric(41)
threshold = numeric(41)

#Loop to compute total amount that could be loss at each threshold level
for (i in 10:50){
  t=i/100                                  #Threshold from 0.10 to 0.50
  cm=table(ActualValue=testing$Target, PredictiveValue=resp10>t) #C.matrix
  fp=cm[1,2]                               #Generate type I errors
  fn=cm[2,1]                               #Generate type II errors
  loss_1=(1*fp)+(3*fn)                     #1000:3000 ratio
  loss_2=(1*fp)+(5*fn)                     #1000:5000 ratio
  data_1[i-9] <- loss_1                    #Store the amount loss at each
                                           # threshold levels from 0.10 to 0.50
                                           # for the 1000:3000 ratio into data_1
  data_2[i-9] <- loss_2                    #Store the amount loss at each
                                           #threshold levels from 0.10 to 0.50
                                           # assuming 1000:5000 ratio into data_2
  threshold[i-9] <- t                      #Store each threshold levels
}

#Plot of threshold level vs. possible amount loss for the 1000:3000 ratio
plot(data_1~threshold, xlab="Threshold Level", ylab="Possible Amount Loss
(thousands $)",
     type="p", lwd=2, col="red", main="1000:3000 Ratio", cex.main=1.5,
cex.lab=1.2)

#Plot of threhold level vs. possible amount loss for the 1000:5000 ratio
plot(data_2~threshold, xlab="Threshold Level", ylab="Possible Amount Loss
(thousands $)",
     type="p", lwd=2, col="blue", main="1000:5000 Ratio", cex.main=1.5,
cex.lab=1.2)

#Same process as above for 1000:10000 ratio.
#Need new loop because minimum point is at threshold level of 0.09. So, need
to start the threshold at 0.01 instead of 0.10 to show this.

data_3 = numeric(44)
threshold_2 = numeric(44)

for (j in 1:44){
  th=j/100
  M=table(ActualValue=testing$Target, PredictiveValue=resp10>th)
  fpos=M[1,2]
  fneg=M[2,1]
  loss_3=(1*fpos)+(10*fneg)
  data_3[j] <- loss_3
  threshold_2[j] <- th
}

#Plot of threhold level vs. possible amount loss for the 1000:10000 ratio
plot(data_3~threshold_2, xlab="Threshold Level", ylab="Possible Amount Loss
(thousands $)",type="p", lwd=2, col="purple", main="1000:10000 Ratio",
cex.main=1.5, cex.lab=1.2)
```