# Lending Club -

In the lending industry, investors provide loans to borrowers in exchange for the promise of repayment with interest. If the borrower repays the loan, then the lender profits from the interest. However, if the borrower is unable to repay the loan (default), then the lender loses money. Therefore, lenders face the problem of predicting the risk of a borrower being unable to repay a loan.

This dataset represents a sample of loans that were funded through the LendingClub.com platform in 2007 – 2015.

## Let upload our dataset

```
library(readr)

loan_status <- read_csv("C:/Users/Amara Diallo/Desktop/SPRING 2019/Predictive 353/dataset/LendingClub(1
#View(loan_status)
#summary(loan_status)
```

Here, the dependent variable is loan_status.
Based on this data, we would like to build a model that predicts which borrowers will pay back the full loan, that is loan_status = 1. We use logistic regression that computes for each loan the probability that it will be paid in full.

```
#str(loan_status)
```

## How many loans were paid in full? What proportion is that?

In the following section we will look at the proportion of loans that were paid on this loan dataset

```
sum(loan_status$loan_status=="1")
```

```
## [1] 9436
```

```
prop.table(table(loan_status$loan_status))
```

```
##
##         0         1
## 0.1838076 0.8161924
```

## What proportion of loans were paid full in train? What about in test?

Here we will create a training data and the test data as follows. we will Load the package caTools, set the seed to 20. Use sample.split to randomly allocate 80 % of the rows to data frame, train, and the remaining 20% to data frame, test. Recall using sample.split function with lending$loan_status will ensure that the distribution of loan_status is similar between the training set and the test set. (Use the set of commands we went over in class.)

```
require(caTools)
```

```
## Loading required package: caTools
```

```
set.seed(20)

splitt<-sample.split(loan_status$loan_status, SplitRatio = 0.80)
splitt[1:80]
```

```
##  [1] FALSE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE FALSE  TRUE  TRUE
## [13]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE
## [25]  TRUE  TRUE FALSE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE FALSE
## [37]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE
## [49] FALSE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [61]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE
## [73]  TRUE FALSE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE
```

```
summary(splitt)
```

```
##    Mode   FALSE    TRUE
## logical    2312    9249
```

```
trainSet<-subset(loan_status, splitt==1)
testSet<-subset(loan_status, splitt==0)

prop.table(table(trainSet$loan_status))
```

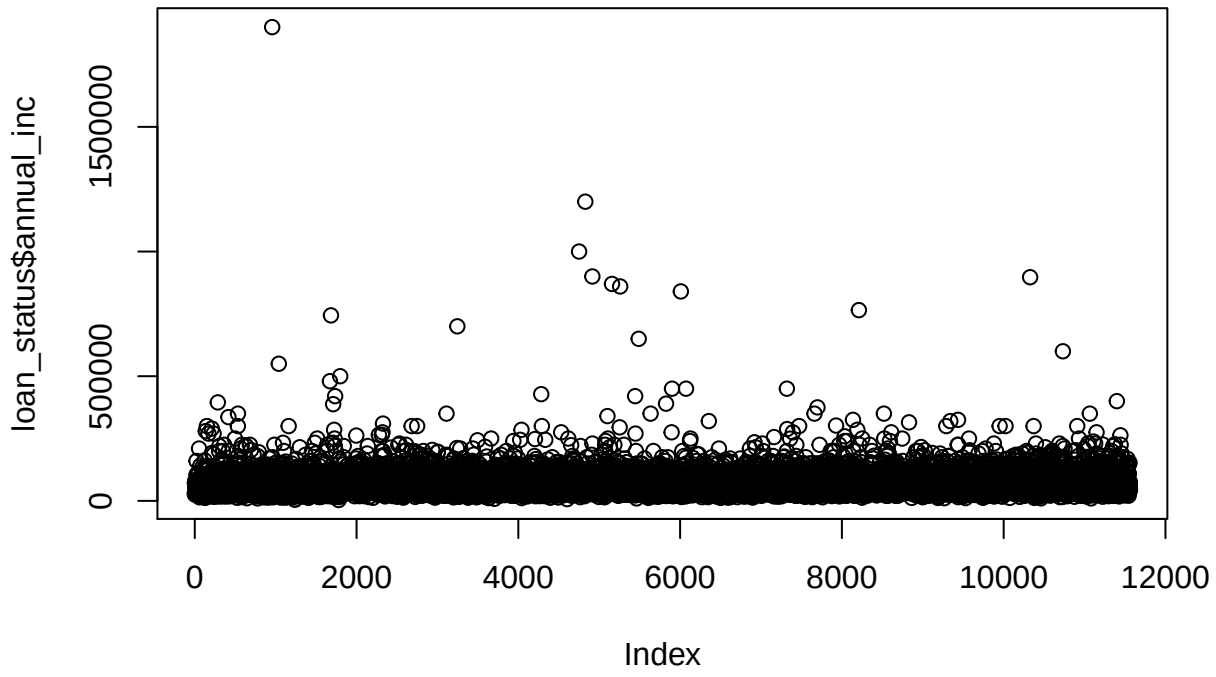
```
##
##         0         1
## 0.1838037 0.8161963
```

```
prop.table(table(testSet$loan_status))
```

```
##
##         0         1
## 0.1838235 0.8161765
```

Let look at the distribution of our data before building our model

```
plot(loan_status$annual_inc)
```




Let's work with the train data first. We will Build a logistic regression with the following as the predictors:log(annual_inc), delinq_2yrs, dti, int_rate, loan_amnt. Here, log is the natural log of the annual income.
We are using the log because annual income is highly right-skewed.

```
#
fitlog<-glm(loan_status~log(annual_inc) + delinq_2yrs + dti + int_rate + loan_amnt, data=trainSet,family

options(scipen = 999)
summary(fitlog)
```

```
##
## Call:
## glm(formula = loan_status ~ log(annual_inc) + delinq_2yrs + dti +
##     int_rate + loan_amnt, family = binomial, data = trainSet)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.5651   0.3539   0.5113   0.6669   1.5311
##
## Coefficients:
##                      Estimate   Std. Error z value             Pr(>|z|)
## (Intercept)     -1.756848961  0.745397262  -2.357              0.0184 *
```

```
## log(annual_inc)  0.529189398  0.068375415   7.739  0.00000000000000998 ***
## delinq_2yrs     -0.040348425  0.037239064  -1.083               0.2786
## dti             -0.018266643  0.003641622  -5.016  0.00000052738212919 ***
## int_rate        -0.137118477  0.006625382 -20.696 < 0.0000000000000002 ***
## loan_amnt       -0.000017068  0.000004148  -4.115  0.00003878346237547 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 8825.6  on 9248  degrees of freedom
## Residual deviance: 8135.6  on 9243  degrees of freedom
## AIC: 8147.6
##
## Number of Fisher Scoring iterations: 4
```

It is really weird to see that all these variables have negative, it does not make sense to me.

loan_amnt int_rate dti log(annual_inc)

## Here is the expression for logit, z. (This kind format: z = 2.567 + 0.435 x - 1.345 y)

$z = -1.756848961 + log(annual_inc)0.529189398 - delinq_2yrs(0.040348425) - dti(0.018266643) - int_rate(0.137118477) - loan_amt$

$$exp(z)$$

Suppose we have the following data about borrower A. Annual income is $73,500, debt-to-income ratio is 13.0, there was one delinquency in the past two years, the loan amount he wants is $20,000 with the interest rate 8%. What is the probability borrower A will pay back the loan in full? (Compute by plugging in variable values in the regression to get z then plugging this z value into the expression 1/(1+exp(-z)). What are the odds of him paying back in full? Would we lend to this borrower?

```
#
borrower_Z<-data.frame(annual_inc=73500,dti =13.0, loan_amnt=20000, int_rate=8, delinq_2yrs=1)

(z<-predict(fitlog, newdata = borrower_Z))
```

```
##        1
## 2.456616
```

```
prob<-1/(1+exp(-z))
```

Now we will compute the probability directly with the predict function. we should get the same probability we computed above.

```
predict(fitlog,type = "response",newdata = borrower_Z)
```

```
##         1
## 0.9210439
```

For all the observations in the train data set, we use predict function to compute the probabilities of paying back in full. We will Name the result predTrain. Since we are using the train data that the model came from, we don't need to specify newdata in the predict function.

```
predTrain<-predict(fitlog, type="response",trainSet)

length(predTrain)
```

```
## [1] 9249
```

Now for all the observations in the test data set,we use predict function to compute the probabilities of paying back in full.

```
predTest<-predict(fitlog, type="response", testSet)
```