Probability of Default Predictor

Agustin Daniel Ross

8/3/2022

Introduction section: Project's goal and database used.

This is a project based on the creation of a code that intends to estimate the probability of default for a financial entity's client (the probability of not paying a credit card debt). To do this, I borrowed a small data sample from an Argentinian financial entity I work in. In this paper will be shown the different methods and models that have been tested and select the best of them in the mention case of study.

Exploratory analysis: understanding the data.

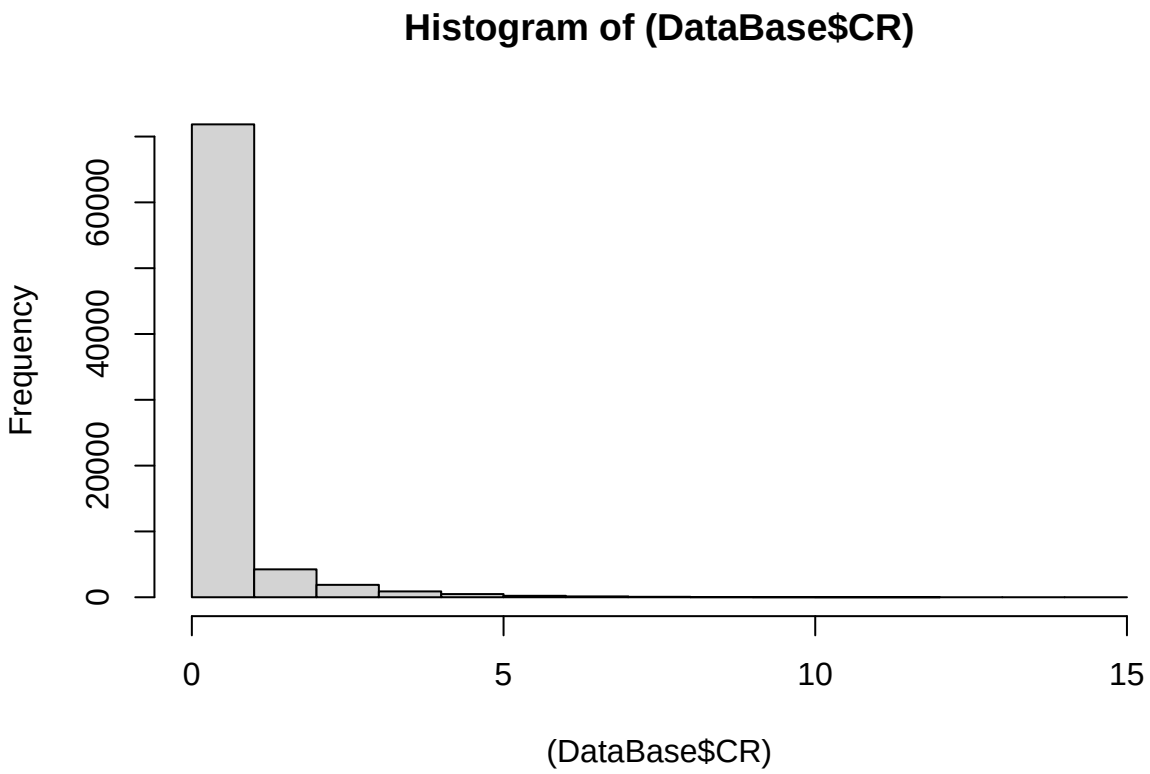
As it was told, the data is a sample taken from an Argentinian financial entity. It has 12 columns, eleven independent variables (X) and one dependent (Y). The dependent variable is a flag that shows if the person has paid the credit card debt or not, it is a binary variable (0 = Debt paid, 1 = Debt not paid, default). Also, it contains 79800 observations, this means it has information from 79800 people. The data from the independent variables is a picture taken on January 2021, and the Default / No Default flag shows the payment behavior of this people during all the 2021. If a person has a delay in his payment for more than 90 days, the flag is activated (Default = 1), this situation is an absorbing state, once you are flagged (in this specific database) with Default = 1, you can not take off the mark.

Variables Analysis.

Univariate Analysis.

Let's start with the analysis of the independent variables so we can understand them better. To study the distribution of the different variables I have used some histograms.:

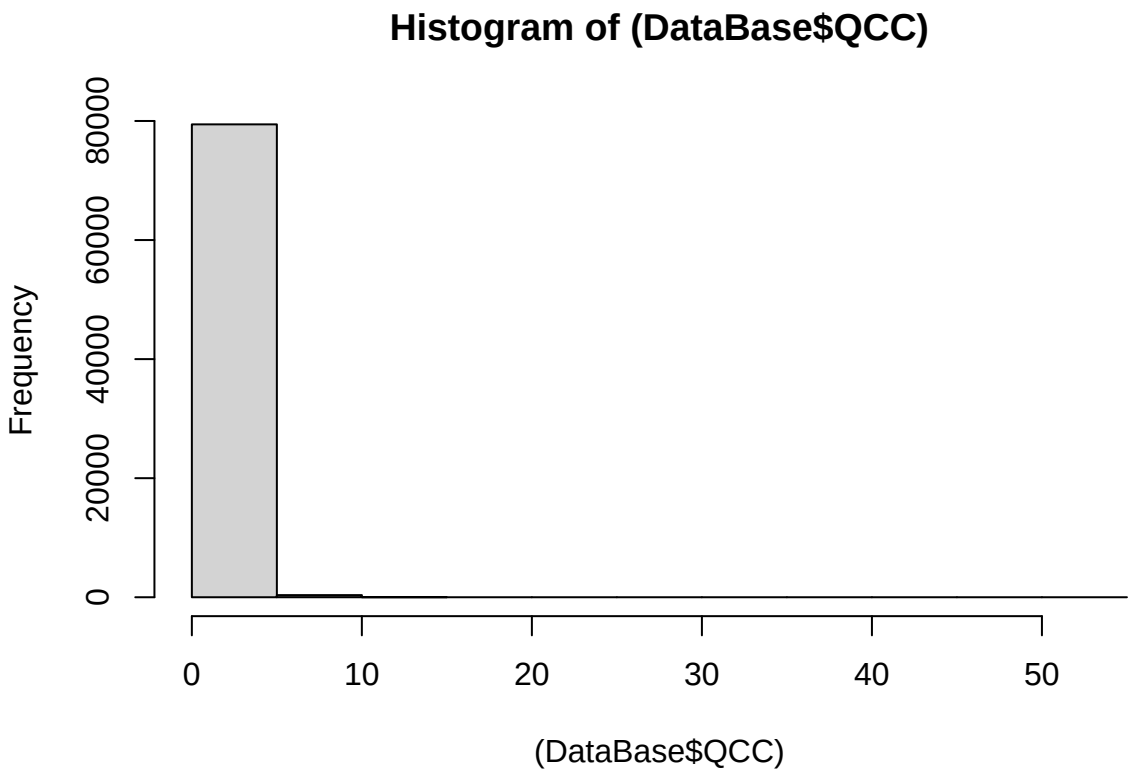
The first variable is "Commercial References", it shows the amount of times the person under analysis did not pay some expenses like electricity and cellphone bills (here in Argentina you can buy this type of information). This is its histogram:



The majority of the observations are distributed around the "0", as expected.

-

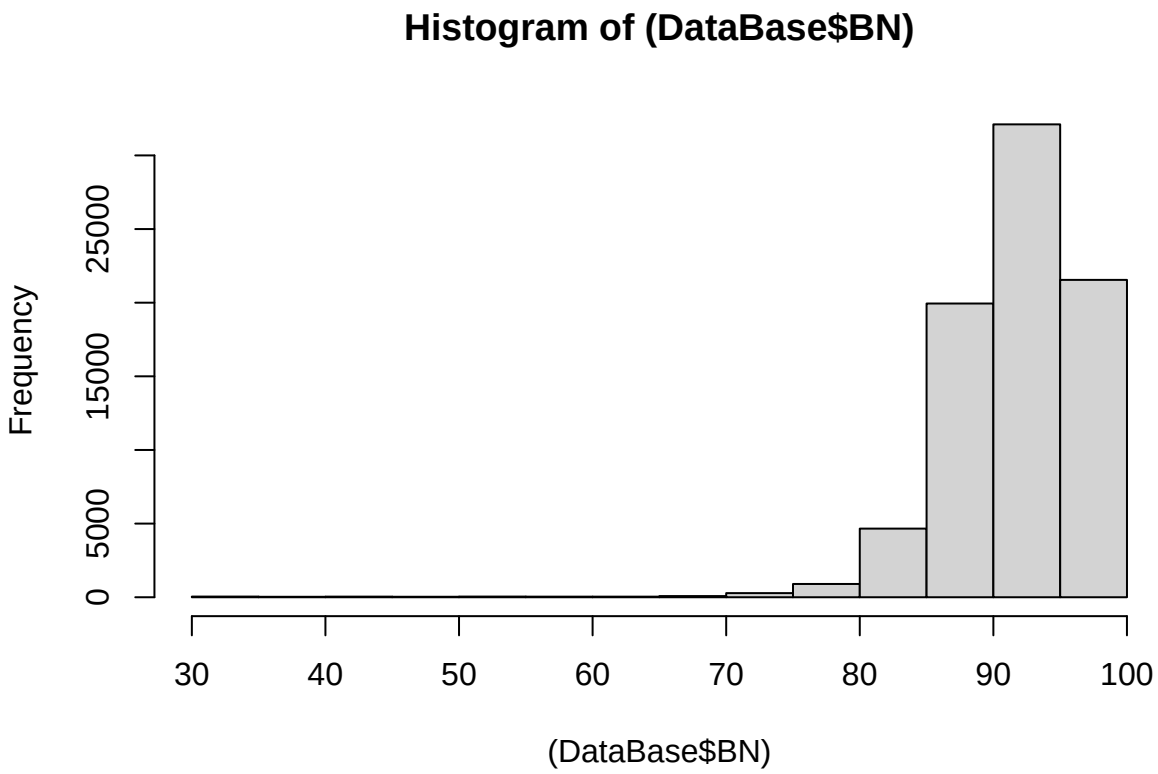
The second variable is "Quantity of credit cards the person owns". This is its histogram:



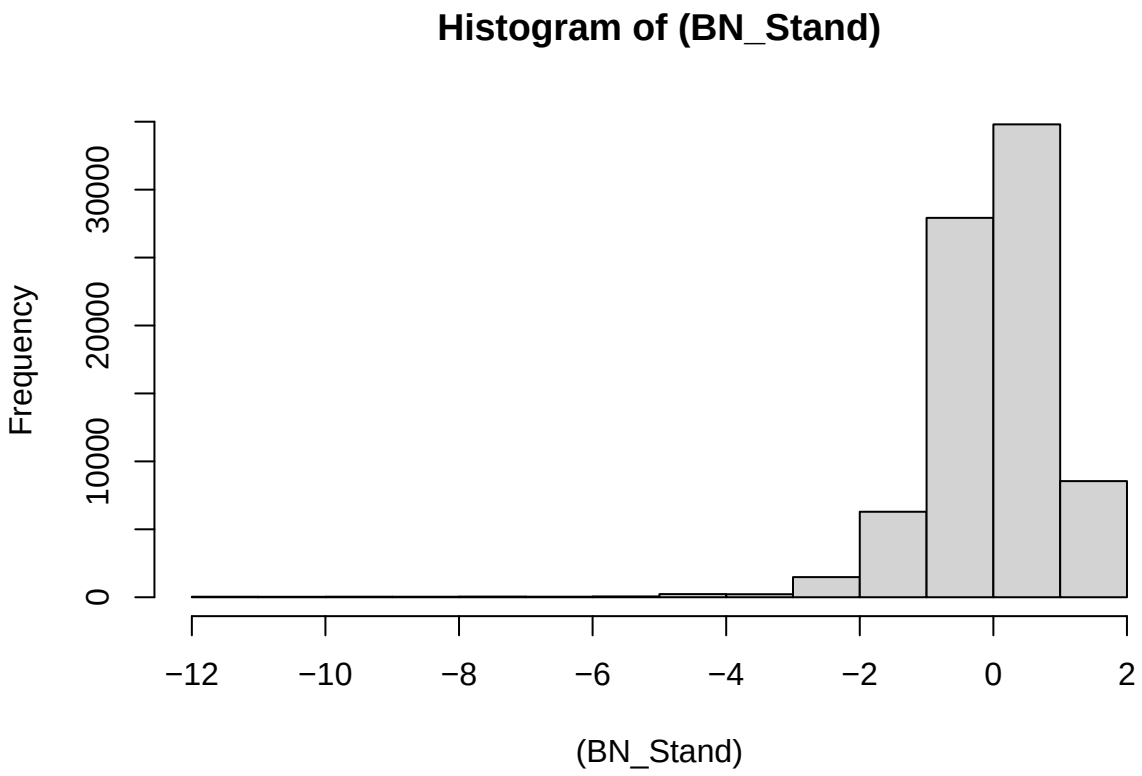
As expected, the distribution of the variable is also around "0".

-

The third variable is a very interesting one, it is "Basic Needs". It shows the percentage of basic needs that are covered in the region the person lives (it can be a huge indicator for the good or bad behavior in payments). This is its histogram:



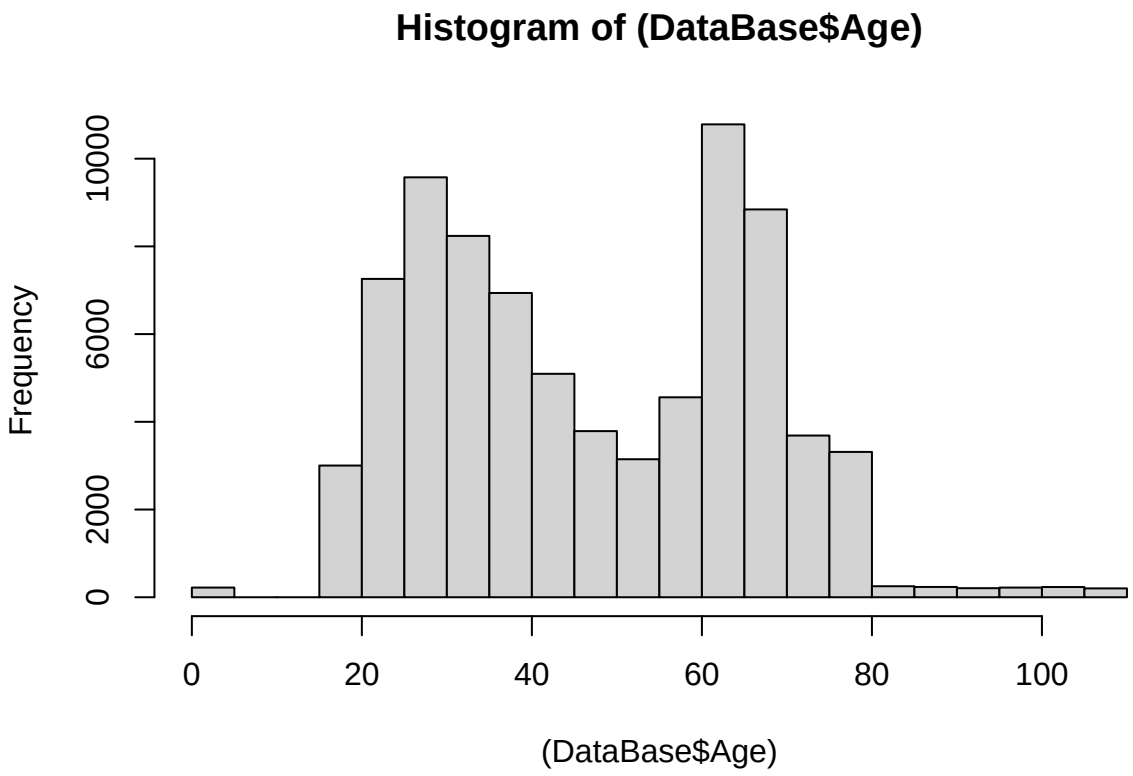
It is distributed around 91.560644 (its mean). We can also see a low variance, and it is almost impossible to find an observation below the 60 points. In this case, it could be suitable to standardize the variable, so I subtracted the mean and divided by the SD. Here is the new histogram:



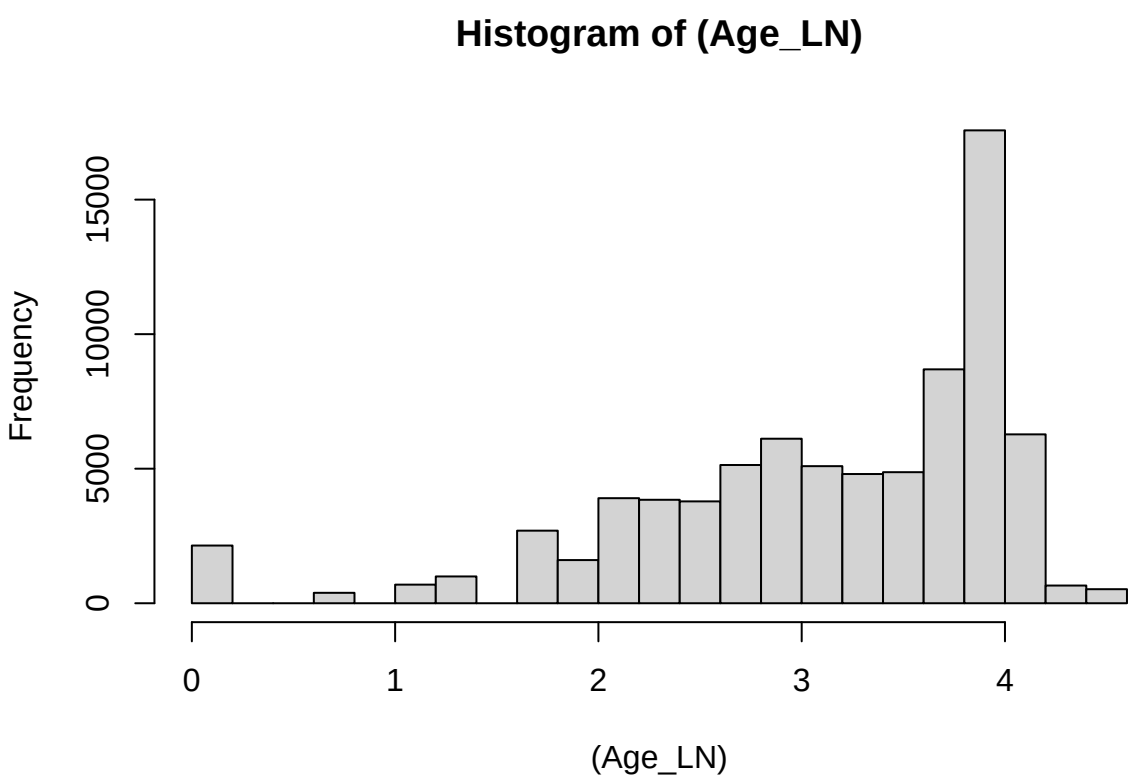
Now it is distributed around “0”.

-

The fourth variable is “Age”. This is its histogram:

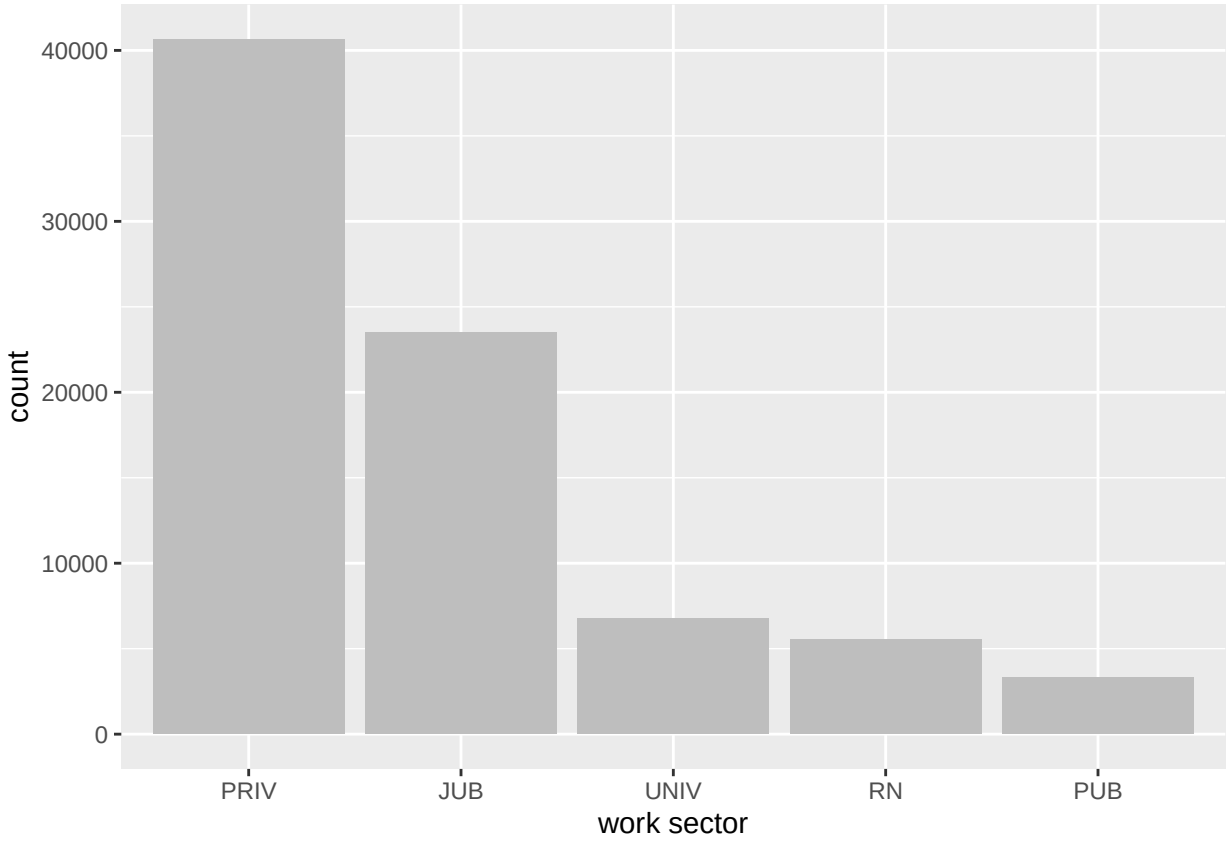


Since this is supposed to apply to people that are not underage, I truncated the data to 18 years old. It also can be seen how we have just a few observations over the 80 years old, so it could be appropriated to truncated it to 80. We can also see that the data has high variance and it doesn't seem to be normally distributed. So, I proceeded to apply the LN function to try to catch better the prediction power of the variable. Here is the new histogram:



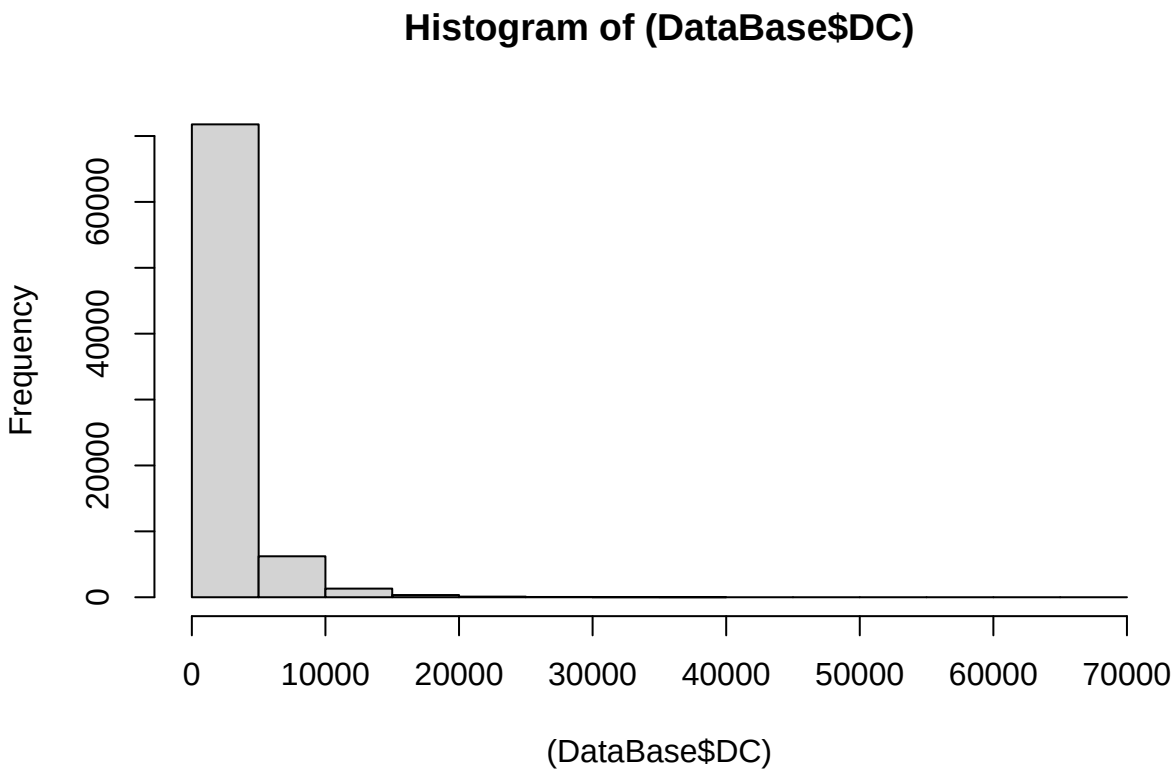
-

The fifth variable is the sector where the person work. PRIV = Private sector, PUB = Public sector , JUB = Retired person, UNIV = University teacher , RN = Worker for the state “Rio Negro” in Argentina. This is its plot:

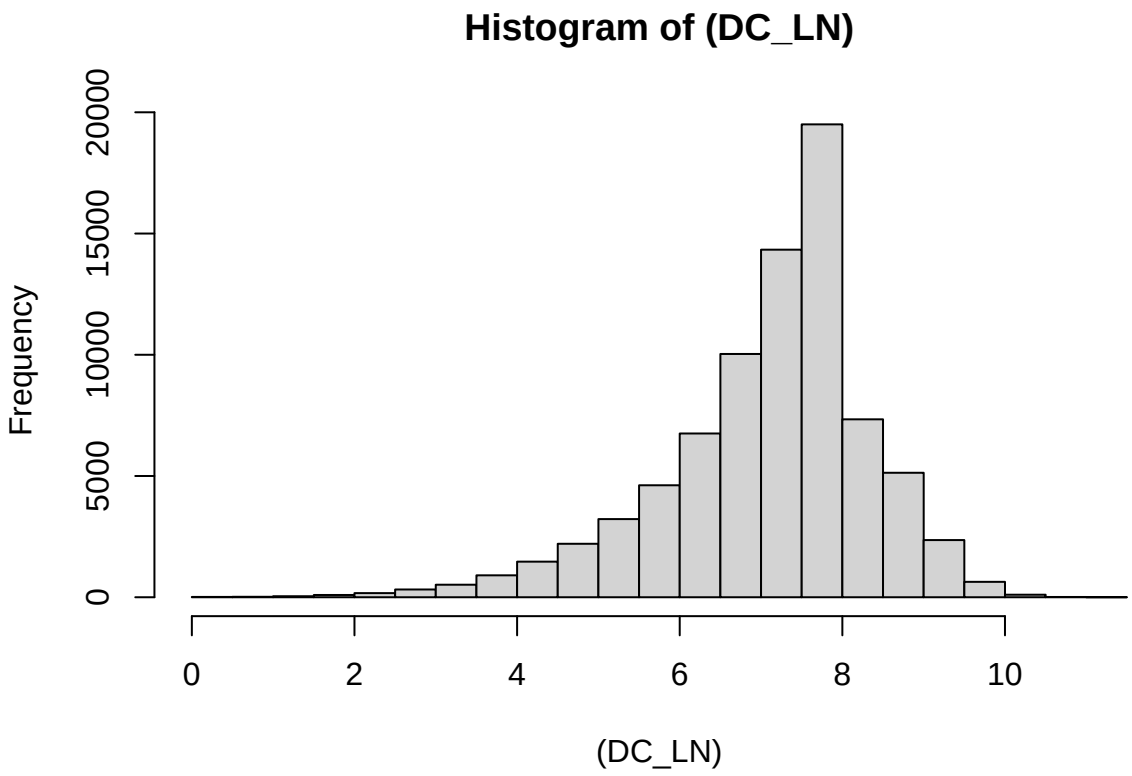


-

The sixth variable is the amount of money spent in Debit Card. This is its histogram:

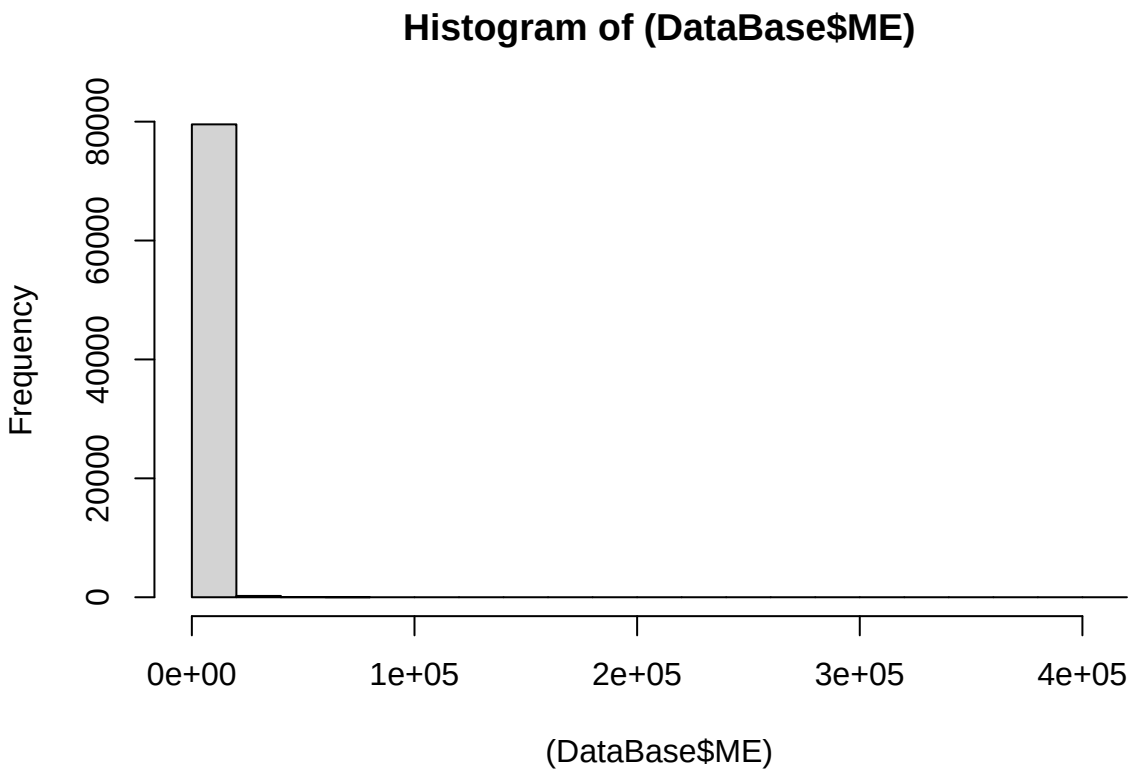


Since this is a variable in money units, it could be appropriated to apply LN function here, it should make the distribution of the variable more normal. This is the new histogram:

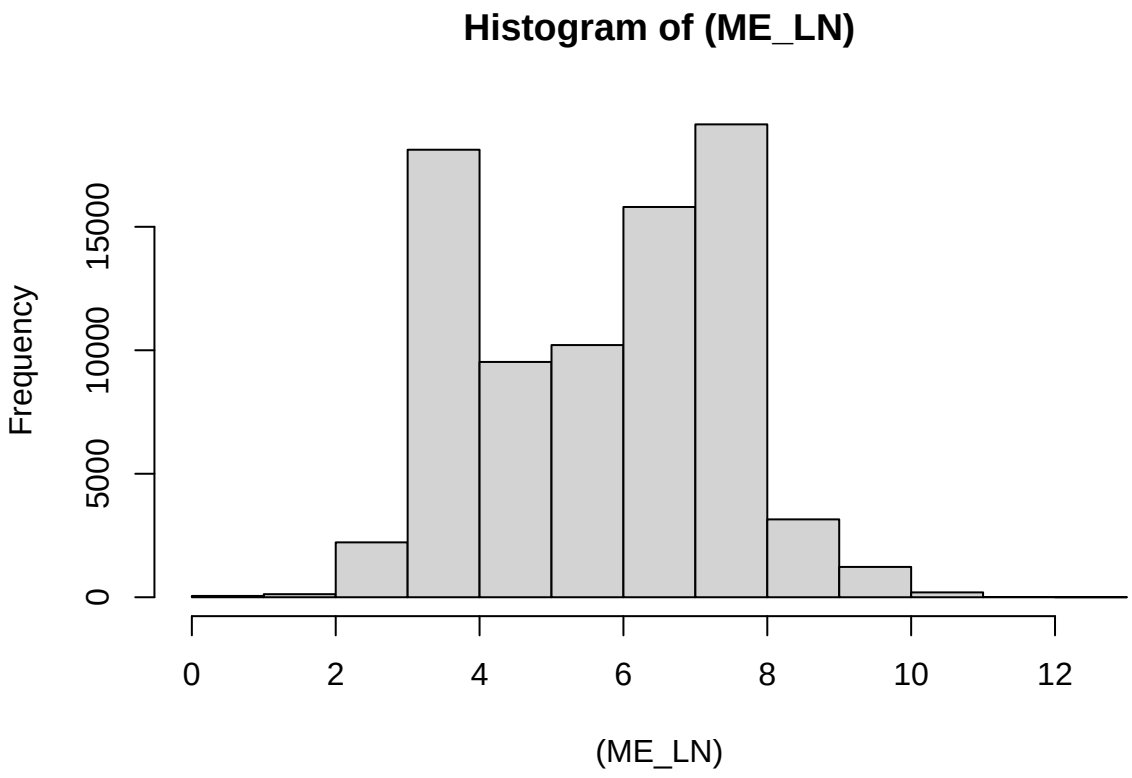


-

The seventh variable is Monthly Expenses. This is its histogram:

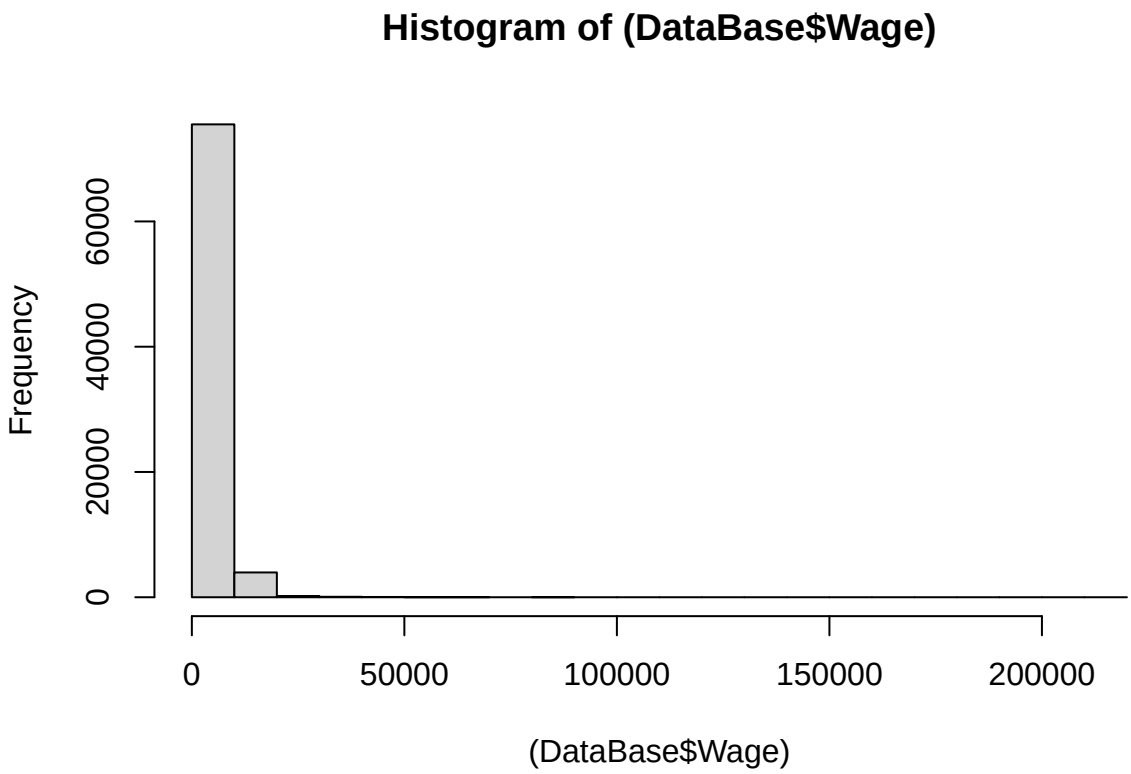


Same as DebitCard, I applied LN function. This is the new histogram:

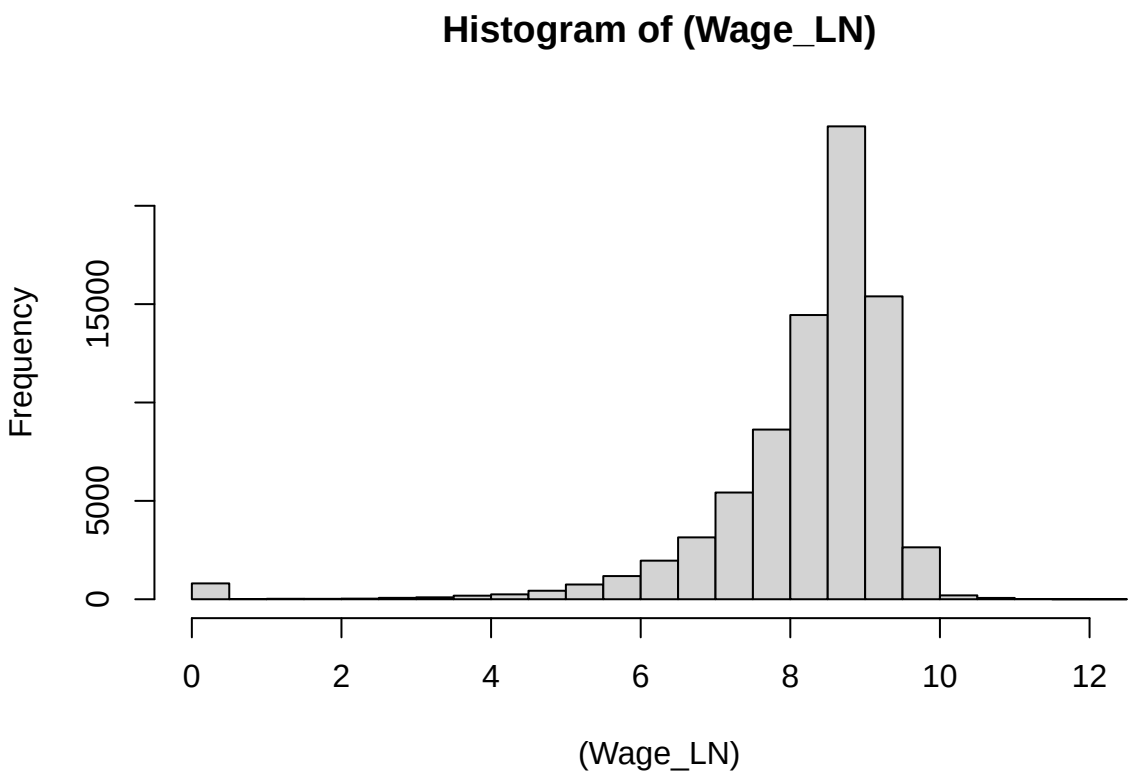


-

The eighth variable is Wage, this is a very important variable to estimate Default. This is its histogram:

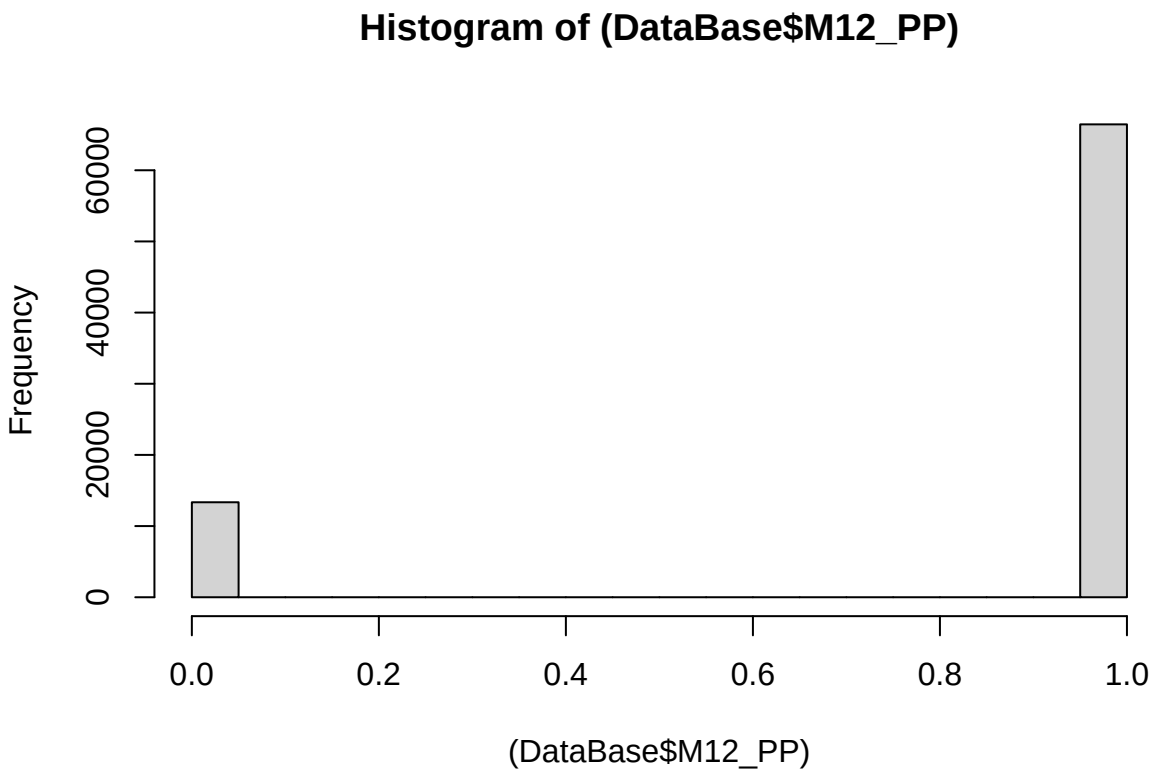


With the same logic as before, I applied the LN function. This is the new histogram:



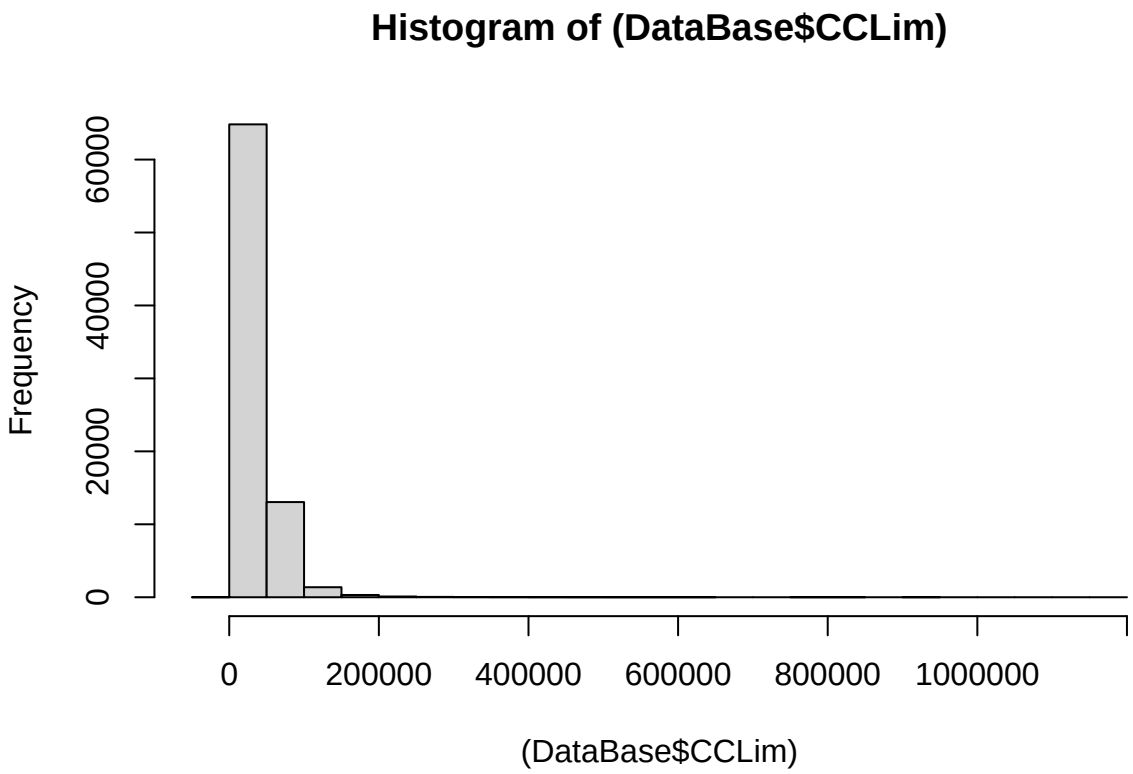
-

The ninth variable is a flag that shows if the person has used a passive product (like debit card) 12 months in a row. (It could be a good indicator of how much use the person gives to the bank's products). This is its histogram:

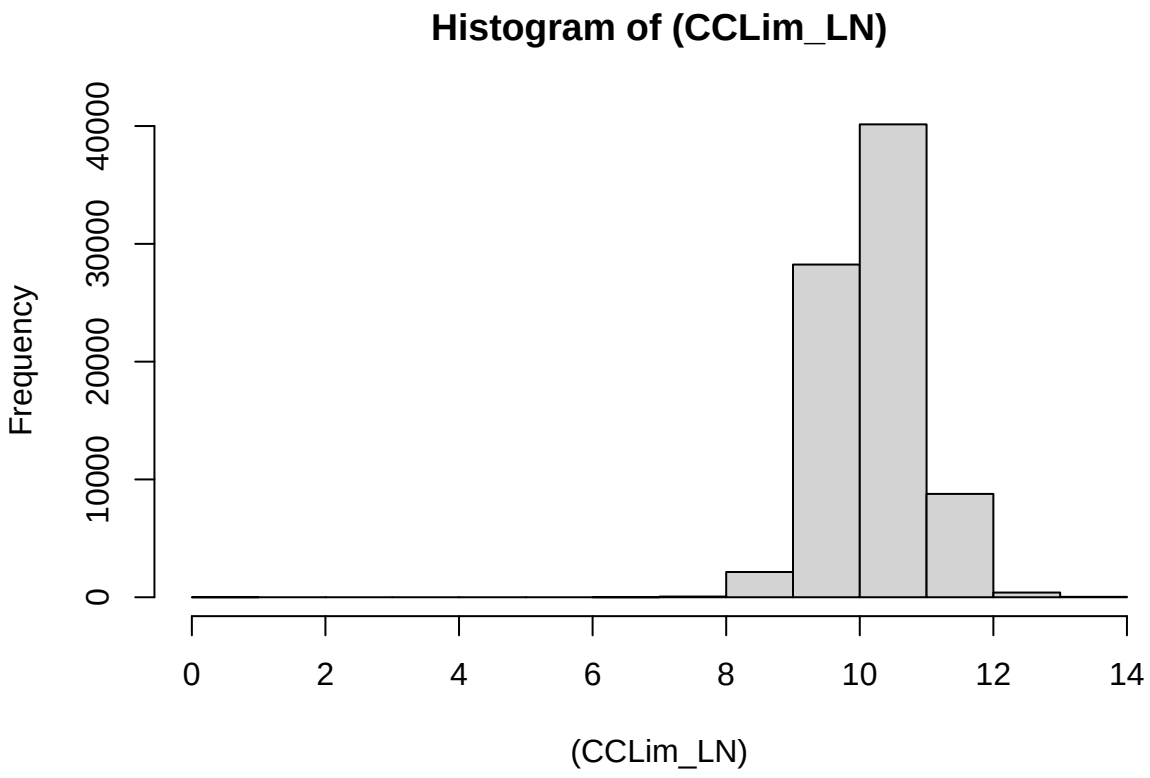


-

The tenth variable is the limit the person has in his/her credit card in the financial sector in Argentina. This is its histogram:

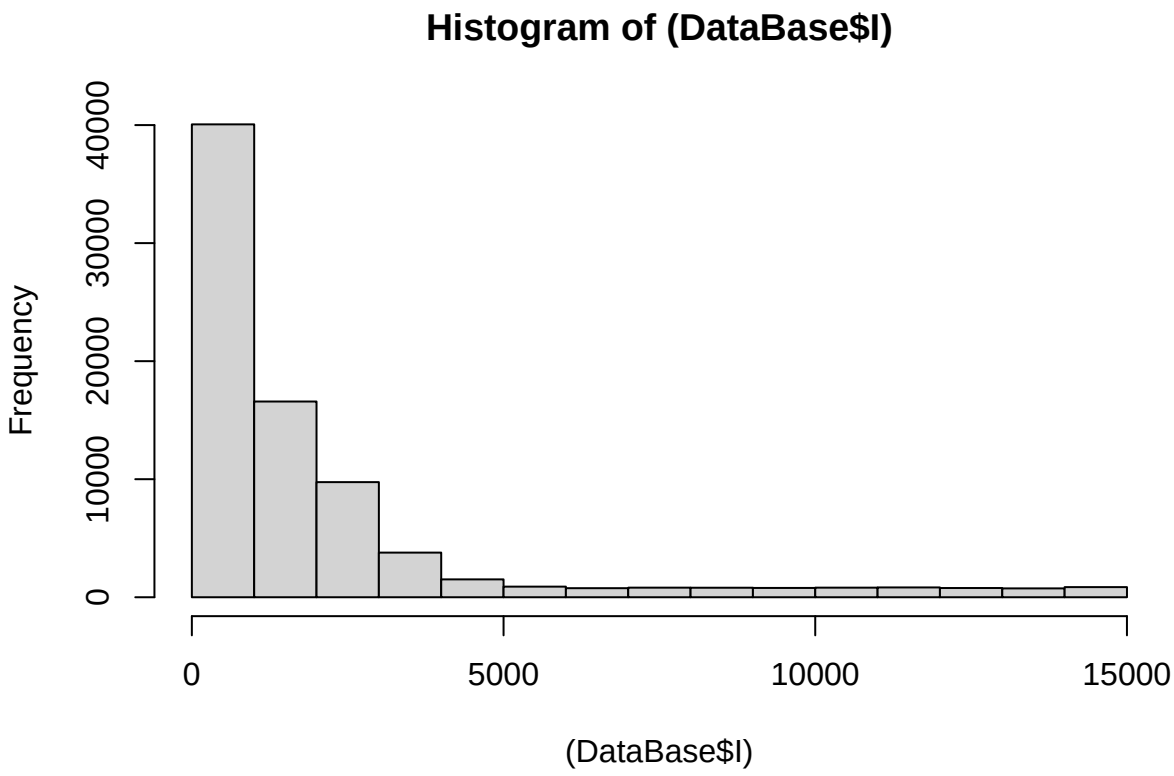


I applied the LN function. This is the new histogram:

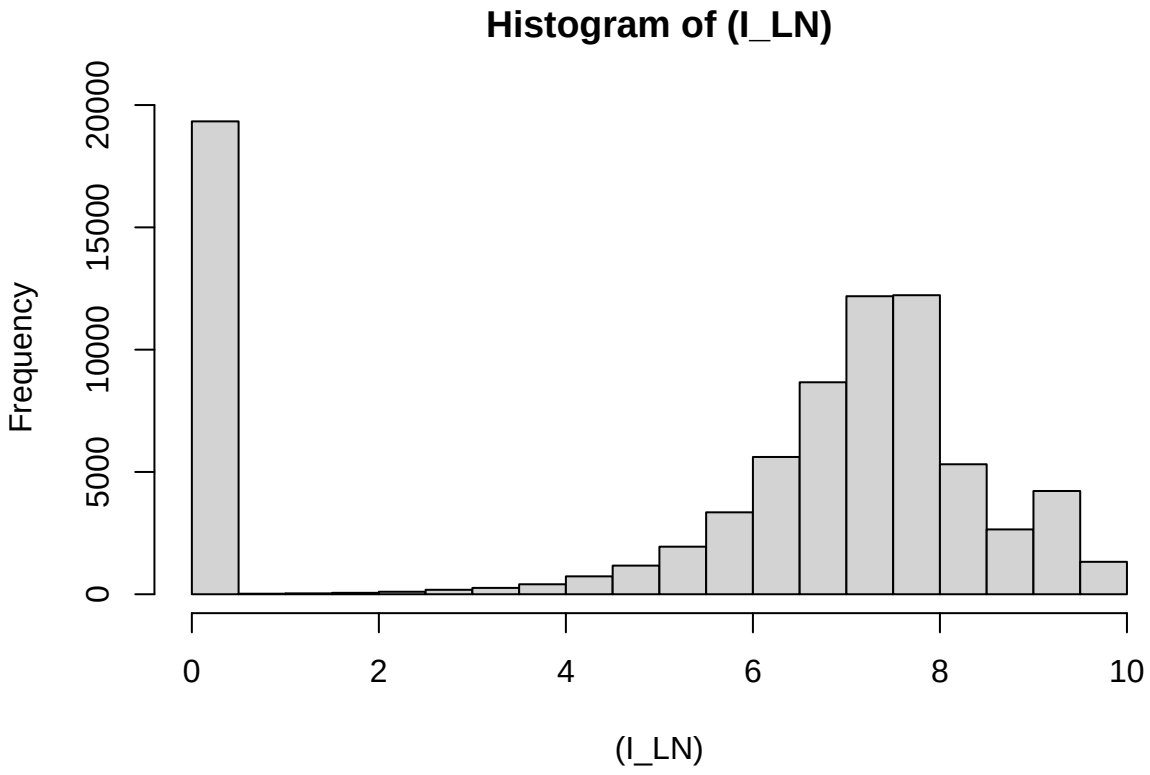


•

The last variable is Interest, and it shows the amount of interest (it could be in fix rent) the person won the last year. This is its histogram:



With the same logic as before, I applied LN function. This is the new histogram:



We have seen the distribution of all the variables. It was also shown how, on some particular cases, a transformation was applied, this was done trying to capture better the prediction power of the variables.

Bivariate Analysis

Let's start now with the bivariate analysis. Here I compared the independent variables with each other and with the dependent variable.

First, it would be interesting to see how well the independent variables explains the dependent one individually. To do this let's introduce the performance metrics that are commonly used to test this types of models: KS and AUC-ROC.

- KS: The Kolmogorov–Smirnov statistic quantifies a distance between the empirical distribution function of the sample and the cumulative distribution function of the reference distribution. The rule is: bigger the number of KS, the better the model. Commonly, a KS of 50 points or more can be reference of a good model.
- AUC-ROC: The ROC curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings. The true-positive rate is also known as sensitivity, recall or probability of detection. The false-positive rate is also known as probability of false alarm. The area under the curve (AUC) is a performance measurement for the classification problems at various threshold settings. Like the KS statistic, the rule is: the bigger the number of KS, the better the model. Commonly, a AUC-ROC of 80 points or more can be reference of a good model.

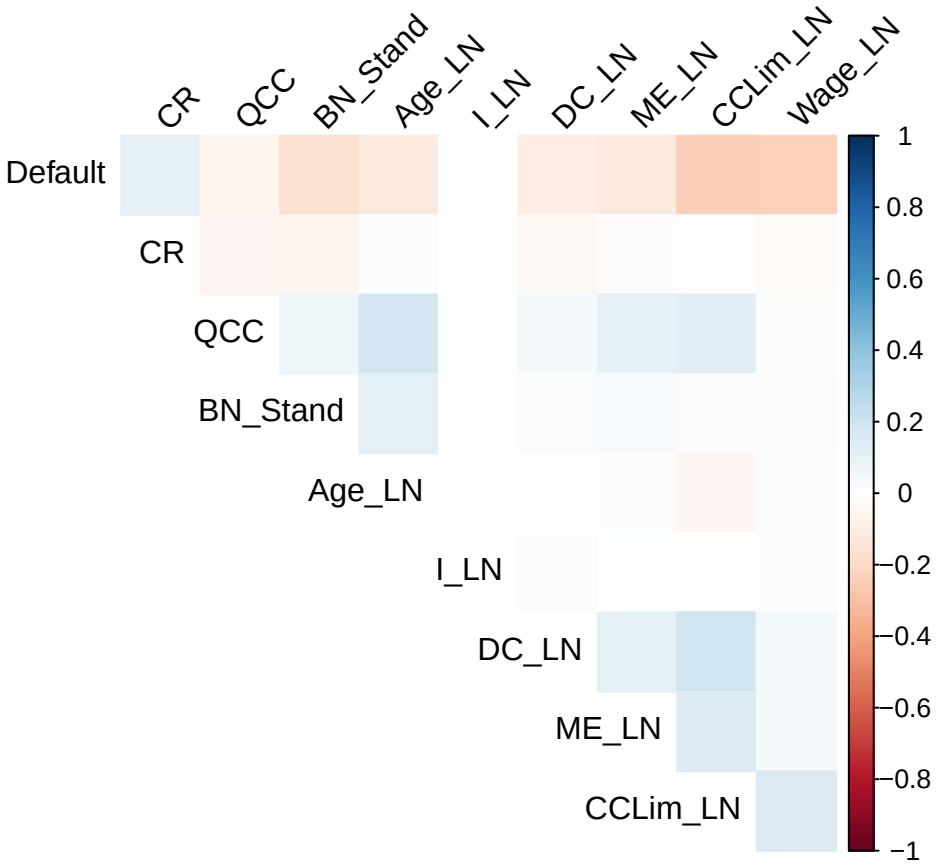
Let's proceed now to create a univariate logistic regression model for each independent variable vs the dependent variable. The KS's and AUC-ROC's results of each model are shown below:

```
##       varaible    KS
## 1   KS_Wage_LN 23.00
## 2        KS_CR 21.46
## 3       KS_QCC 16.66
## 4     KS_DC_LN 36.33
## 5    KS_Age_LN 29.29
## 6  KS_BN_Stand 16.72
## 7     KS_ME_LN 45.35
## 8      KS_Work 33.61
## 9      KS_I_LN  1.18
## 10   KS_M12_PP  0.30
## 11 KS_CCLim_LN 45.93
```

```
##        variable AUC.ROC
## 1   ROC_Wage_LN   49.57
## 2        ROC_CR   61.43
## 3       ROC_QCC   59.65
## 4     ROC_DC_LN   67.06
## 5    ROC_Age_LN   67.96
## 6  ROC_BN_Stand   62.23
## 7     ROC_ME_LN   67.36
## 8      ROC_Work   67.89
## 9      ROC_I_LN   50.27
## 10   ROC_M12_PP   50.15
## 11 ROC_CCLim_LN   77.71
```

It can be seen how "M12_PP" and "I" have a nearly 0 KS and nearly 50 ROC. It means they do not explain anything of the dependent variable, so I proceed to eliminate them from the database.

Let's now proceed to make a correlation plot to study correlations between the variables. To do this, I excluded the "Work" variable because it is a "factor" variable.



There is not a strong correlation between independent variables.

Multivariate models

Let's now start with the different models that will be tested. I have tested 6 different models:

1_ Linear Regression with the intercept only

2_ Multivariate Linear Regression

3_ Logistic Regression with the intercept only

4_ Multivariate Logistic Regression

5_ Decision tree

6_ Neural Network

First of all, it is necessary to transform the factor variable into dummies and to divide the database between train and test

```
# First, I have to convert to dummies the factor variable
factor_variable <- "Work"
DataBase_wDummies <- as.data.frame(get_dummies.(DataBase))
DataBase_wDummies <- DataBase_wDummies[,-c(which(colnames(DataBase_wDummies) == "Work"))]


# Then, I set a seed
seed = 1
```

```
set.seed(seed)

# I divided the database between train and test.
Muestra = 0.8
Coord = sample(nrow(DataBase_wDummies),nrow(DataBase_wDummies)*Muestra)
Test = DataBase_wDummies[-Coord,]
Train = DataBase_wDummies[Coord,]
```

1ST MODEL: LINEAR MODEL - Intercept Only

```
Model_Linear_null <- glm(Default ~ 1, data=Train)

# With the created model, I estimated the estimated probability of default
Prob_est_null <- predict(Model_Linear_null,
                              Test,type = 'response')

# Then I calculated the KS and the ROC
m1_pred_null <- prediction(Prob_est_null , Test$Default )
m1_perf_null <- performance(m1_pred_null,"tpr","fpr")
KS_lm_null <- round(max(attr(m1_perf_null,'y.values')[[1]]-
                          attr(m1_perf_null,'x.values')[[1]])*100, 2)
ROC_lm_null <- round(performance(m1_pred_null, measure =
                                   "auc")@y.values[[1]]*100, 2)

KS_lm_null
```

```
## [1] 0
```

```
ROC_lm_null
```

```
## [1] 50
```

This model does not explain the default. A KS = 0 and a ROC = 50 are similar to an aleatory decision between default and no default.

2ND MODEL: MULTIVARIATE LINEAR MODEL

```
Model_Linear <- lm(Default ~ 1 +  . ,data = Train)

Prob_est <- predict(Model_Linear,
                         Test,type = 'response')
m1_pred <- prediction(Prob_est , Test$Default )
m1_perf <- performance(m1_pred,"tpr","fpr")
KS_lm <- round(max(attr(m1_perf,'y.values')[[1]] -
                     attr(m1_perf,'x.values')[[1]])*100, 2)
ROC_lm <- round(performance(m1_pred, measure =
                              "auc")@y.values[[1]]*100, 2)

KS_lm
```

```
## [1] 52.54
```

```
ROC_lm
```

```
## [1] 81.07
```

This model is much better than the one with only the intercept. Let's try to improve it with a logistic.

3RD MODEL: LOGISTIC REGRESSION - Intercept Only

```
Model_Logistic_null <- glm(Default ~ 1, data=Train, family = "binomial")

Prob_est_null <- predict(Model_Logistic_null,
                              Test,type = 'response')
m1_pred_null <- prediction(Prob_est_null , Test$Default )
m1_perf_null <- performance(m1_pred_null,"tpr","fpr")

KS_glmL_null <- round(max(attr(m1_perf_null,'y.values')[[1]]-
                            attr(m1_perf_null,'x.values')[[1]])*100, 2)
ROC_glmL_null <- round(performance(m1_pred_null, measure =
                                     "auc")@y.values[[1]]*100, 2)

KS_glmL_null
```

```
## [1] 0
```

```
ROC_glmL_null
```

```
## [1] 50
```

The same as the first one, a model with only an intercept does not explain the default.

4TH MODEL: MULTIVARIATE LOGISTIC MODEL

```
Model_Logistic <- glm(Default ~ 1 +  . ,data = Train , family = "binomial")

Prob_est <- predict(Model_Logistic,
                         Test,type = 'response')
m1_pred <- prediction(Prob_est , Test$Default )
m1_perf <- performance(m1_pred,"tpr","fpr")
KS_glmL_log <- round(max(attr(m1_perf,'y.values')[[1]]-
                           attr(m1_perf,'x.values')[[1]])*100, 2)
ROC_glmL_log <- round(performance(m1_pred, measure =
                                    "auc")@y.values[[1]]*100, 2)

KS_glmL_log
```

```
## [1] 58.99
```

```
ROC_glmL_log
```

```
## [1] 86.53
```

Here we can see how the logistic model improves the linear one. This is mainly because the dependent variable is binary and the logistic model applies much better in these cases.

An extra analysis could be to take a look at the p-values of the variables:

```
pvalue <- as.data.frame(summary(Model_Logistic)$coefficients)
pvalue <- pvalue[order(pvalue$`Pr(>|z|)`),]
pvalue
```

```
##                 Estimate  Std. Error    z value      Pr(>|z|)
## (Intercept) 18.757476960 0.403066861  46.536887  0.000000e+00
## CCLim_LN    -1.743571179 0.039713282 -43.903981  0.000000e+00
## BN_Stand    -0.408667362 0.017269669 -23.663879 8.494816e-124
## ME_LN       -0.347794151 0.015393522 -22.593539 5.016378e-113
## Wage_LN     -0.196511346 0.010918377 -17.998219  2.011857e-72
## CR           0.253821166 0.016092309  15.772824  4.786085e-56
## DC_LN       -0.207300184 0.017033064 -12.170458  4.465715e-34
## Work_PRIV    1.042151657 0.107250648   9.716973  2.552568e-22
## Age_LN      -0.213122523 0.024723822  -8.620129  6.687901e-18
## Work_JUB    -1.034634189 0.134960026  -7.666227  1.771296e-14
## Work_PUB     0.459151738 0.167540626   2.740540  6.133837e-03
## QCC         -0.054121302 0.024021171  -2.253067  2.425494e-02
## Work_RN      0.294135797 0.147878098   1.989042  4.669653e-02
## I_LN         0.008349601 0.006979928   1.196230  2.316068e-01
```

Here we can see that all the p-values are quite small, so all the variables are significant to the model.

5TH MODEL: DECISION TREES

```
# Creation of the model
tree<-rpart(Default~., Train)
```

Let's now see the performance of this model:

```
Prob_estimada <- (predict(tree,Test,type = 'vector'))
m1_pred <- prediction(as.numeric(Prob_estimada), as.numeric(Test$Default))
m1_perf <- performance(m1_pred,"tpr","fpr")

KS_tree <- round(max(attr(m1_perf,'y.values')[[1]]-attr(m1_perf,'x.values')[[1]])*100, 2)
ROC_tree <- round(performance(m1_pred, measure ="auc")@y.values[[1]]*100, 2)

KS_tree
```

```
## [1] 77.51
```

```
ROC_tree
```

```
## [1] 91.57
```

The KS and the ROC are much higher with this model. It can discriminate very well the defaults.

6TH MODEL: NEURAL NETWORK

```
# Set the formula to use
formula <- as.formula(paste("Default ~ ."))


# This model takes a while to run (2 or 3 minutes), so I added a timer
t <- proc.time()  ; Modelo_Redes_prueba <- neuralnet(formula , hidden = c(3,3) ,data =
               Train ,linear.output = FALSE, algorithm =
               'rprop+',likelihood = TRUE,threshold = 0.5, stepmax = 100000) ;proc.time()-t
```

```
##    user  system elapsed
##   79.28    8.63   89.78
```

Now, let's see the performance:

```
Variables <- Modelo_Redes_prueba$model.list$variables
model_results3 <- neuralnet::compute(Modelo_Redes_prueba, Test[,c(which(colnames(Test)
                                                                    %in% Variables))])
Prob_estimada <- as.vector(model_results3$net.result)
detach(package:neuralnet,unload = T)
m1_pred <- prediction(Prob_estimada , Test$Default)
m1_perf <- performance(m1_pred,"tpr","fpr")
KS_nn <- round(max(attr(m1_perf,'y.values')[[1]]-
                     attr(m1_perf,'x.values')[[1]])*100, 2)
ROC_nn <- round(performance(m1_pred, measure =
                              "auc")@y.values[[1]]*100, 2)
KS_nn
```

```
## [1] 58.21
```

```
ROC_nn
```

```
## [1] 86.32
```

The neural network is a good model, as good as the logistic regression in this case.

Resume and finals results:

Let's analyze all models results at once:

```
##                         model    KS   ROC
## 1   Linear - only Intercept  0.00 50.00
## 2       Multivariate Linear 52.54 81.07
## 3 Logistic - only Intercept  0.00 50.00
## 4     Multivariate Logistic 58.99 86.53
## 5             Decision Tree 77.51 91.57
## 6            Neural Network 58.21 86.32
```

It could be appropriated to remind that a model with KS above 60 and ROC above 80 can be considered as a good one. A model with KS above 70 and AUC-ROC above 90 can be considered as a very good one.

The model with the best KS is:

```
## [1] "Decision Tree"
```

The model with the best AUC-ROC is:

```
## [1] "Decision Tree"
```

Conclusion

It can be concluded that the linear regression model is not suitable for this kind of project. In the other hand, the logistic regression, the neural networks and the decision trees are all good choices for the probability of default estimation. In this particular case, the decision tree was the model which best estimated this probability.