Finding the Best Model for Credit Approval

By Daniel Kim, Arman Spinola,
Josiah Marsh, and Ethan Boos

Data Science Competition
University of Georgia
April 15th, 2021

Abstract

Approving lines of credit is one of the core functions of a bank, and one of the requirements of providing a line of credit is to be able to intelligently assess the risk being taken by loaning money to an individual. Given previous financial history and information of an individual, it is possible to construct a mathematical model to predict the likelihood that they will default on the loan. This paper explores logistic regression and random forest models as options to construct such a model. In the early stages it focuses on the data which the model is to be trained on, and how the data should be formatted and cleaned so as to optimize the effectiveness of the final model. Models are then constructed and analyzed to determine which features of an applicant's financial history are most relevant in determining whether they will default on a loan. Four factors emerged as most important to a client not defaulting on their credit: non_mtg_acc_past_due_12_months_num, inq_12_month_num, uti_card, and credit_age. Two other factors appear influential: rep_income and non_mtg_acc_past_due_6_months_num. Aside from these most important factors in predicting the outcome of a loan, it was also determined that a random forest model is more suitable for this application than a logistic regression model. Although random forest models are less transparent, it is worth the more in depth analysis of interdependence between variables in an applicant's financial history.

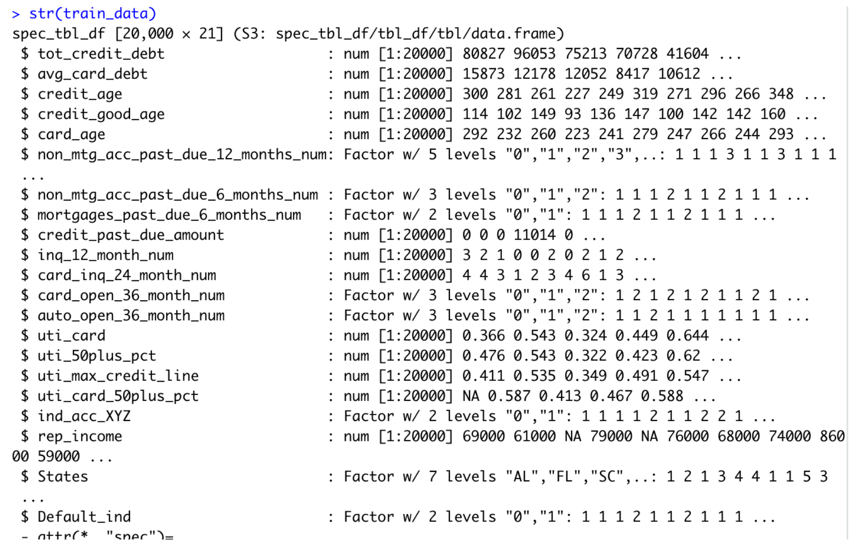
Intro:

For this competition, we were tasked with building a model from a data set of credit applications, some of which defaulted. The model should be able to accurately predict whether an applicant for a line of credit will default, based on information from their credit history.
Our model was built through a process of examining and filtering the provided data, deciding on the best structure to use in our implementation, implementing the model, examining discrepancies in our results, making adjustments to make our model more effective, then finalizing and reviewing our model.

Data Exploratory/Data Processing/ Cleaning Section

Part 1: Changing data type

Courtesy by the Wells Fargo Team, we were given three datasets. One is the training dataset which has 20,000 users. The second is the validation dataset, which has 3,000 users. And third is the testing dataset, which has 5,000 user. All of the dataset has 21 columns, one response variable which is the default credit column and 20 predictors variables to use to figure out the credit approval. When we explored all of the datasets, we noticed that they all have the same problems, the only thing that was different between them was the sum of each of their problems (ex. Number of NA). To start this off, we explored the datasets within our programming systems to see if there are any strange variables/distribution/typos in the datasets. Using Rstudio, we used str() to immediately see the data type and columns on our training dataset. At first glance we found that 20 of the column data types were all set to numerical (num in R) and one column is character (chr in R). It seemed odd because some of the descriptions from the data **book** suggested that some were categorical. It's important to match the description from the **book** with the corresponding data type because this little mistake could rob us of valuable insights. To verify this, we turned the following variables into a factor in R: Default_ind, non_mtg_acc_past_due_12_months_num, non_mtg_acc_past_due_6_months_num, mortgages_past_due_6_months_num, `auto_open_ 36_month_num`, card_open_36_month_num, States, and ind_acc_XYZ. After changing these columns, we checked its str() function and counted the amount of levels they have. The highest level belonged to State, which has 7 levels, while the lowest levels belonged to Default_ind and ind_acc_XYZ, which only had 2 levels: 0 and 1.



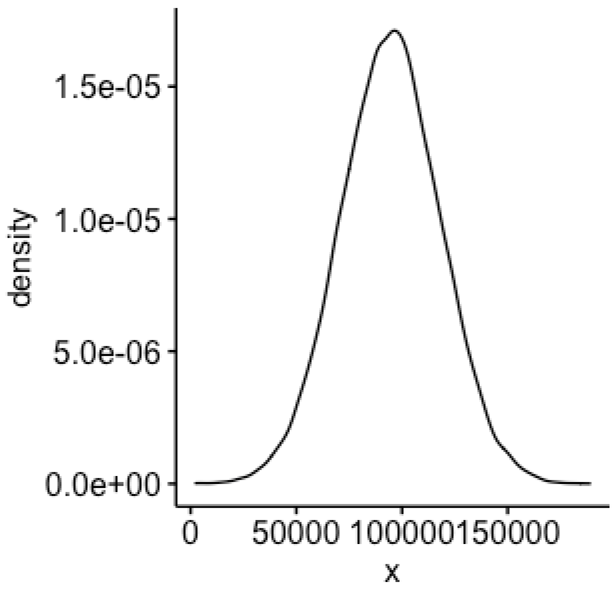
```
> str(train_data)
spec_tbl_df [20,000 × 21] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ tot_credit_debt                  : num [1:20000] 80827 96053 75213 70728 41604 ...
 $ avg_card_debt                    : num [1:20000] 15873 12178 12052 8417 10612 ...
 $ credit_age                       : num [1:20000] 300 281 261 227 249 319 271 296 266 348 ...
 $ credit_good_age                  : num [1:20000] 114 102 149 93 136 147 100 142 142 160 ...
 $ card_age                         : num [1:20000] 292 232 260 223 241 279 247 266 244 293 ...
 $ non_mtg_acc_past_due_12_months_num: Factor w/ 5 levels "0","1","2","3",..: 1 1 1 3 1 1 3 1 1 1
 ...
 $ non_mtg_acc_past_due_6_months_num : Factor w/ 3 levels "0","1","2": 1 1 1 2 1 1 2 1 1 1 ...
 $ mortgages_past_due_6_months_num   : Factor w/ 2 levels "0","1": 1 1 1 2 1 1 2 1 1 1 ...
 $ credit_past_due_amount           : num [1:20000] 0 0 0 11014 0 ...
 $ inq_12_month_num                 : num [1:20000] 3 2 1 0 0 2 0 2 1 2 ...
 $ card_inq_24_month_num            : num [1:20000] 4 4 3 1 2 3 4 6 1 3 ...
 $ card_open_36_month_num           : Factor w/ 3 levels "0","1","2": 1 2 1 2 1 2 1 1 2 1 ...
 $ auto_open_36_month_num           : Factor w/ 3 levels "0","1","2": 1 1 2 1 1 1 1 1 1 1 ...
 $ uti_card                         : num [1:20000] 0.366 0.543 0.324 0.449 0.644 ...
 $ uti_50plus_pct                   : num [1:20000] 0.476 0.543 0.322 0.423 0.62 ...
 $ uti_max_credit_line              : num [1:20000] 0.411 0.535 0.349 0.491 0.547 ...
 $ uti_card_50plus_pct              : num [1:20000] NA 0.587 0.413 0.467 0.588 ...
 $ ind_acc_XYZ                      : Factor w/ 2 levels "0","1": 1 1 1 1 2 1 1 2 2 1 ...
 $ rep_income                       : num [1:20000] 69000 61000 NA 79000 NA 76000 68000 74000 860
00 59000 ...
 $ States                           : Factor w/ 7 levels "AL","FL","SC",..: 1 2 1 3 4 4 1 1 5 3
 ...
 $ Default_ind                      : Factor w/ 2 levels "0","1": 1 1 1 2 1 1 2 1 1 1 ...
 - attr(*, "spec")=
```

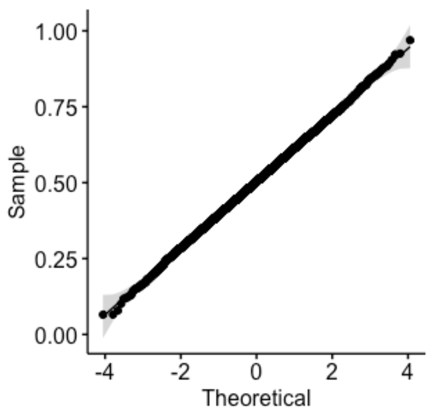
Part 2: Fixing the name

Once all the data types were fixed in the columns, the next thing we noticed in the str() function is that the variable `auto_open_ 36_month_num` has a space in its name. To fix it we used the names function from R and change the name to auto_open_36_month_num. Its a small fix, and almost identical to the original, but it's needed because a typo in a column's name can interfere with your testing when running it (example: regression imputation, shown later in here).

Part 3: Finding the distribution

Once that's taken care of, next we checked to see if all of the predictor variables are normally distributed. By using a mixture of boxplots, histograms, density test, and a Q-Q plot, we found that all of the columns were normally distributed except for the following variables: avg_card_debt, and credit_past_due_amount.

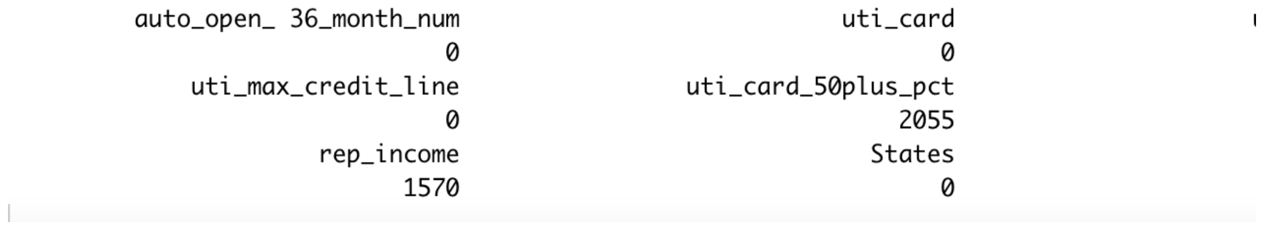




For some the reason the first variable has an almost bimodal distribution (instead of it having one peak hill like a normal distribution, it had two peak hills) while the second distribution has a very left-skewed distribution (when the peak hill is all the way toward the left side instead of it being in the middle). Upon further inspection, we found out that the avg_card_debt has around hundreds of 99999 variables stashed in the rows. It's far higher than the highest card debt in the column, so we speculated that it could be the result of someone computing the system to write 99999 instead of NA (Not applicable) as a replacement, which led to the bimodal distribution. As for the credit_past_due_amount column, we saw that a few of the cells (elements or individuals) had outliers that were very, around 10000's or higher. The rest of the cells were 0, hence the left skewness. To fix the credit_past_due_amount column, we decided to use a median to circumvent those numbers and use its real value. To get rid of those outliers could interfere with our model, so it's better just to keep it where it is. As for the avg_card_debt column, we used the replace function in R to change all of the cells that has 99999 into NA, so that it can make our process easier for the next step in our data processing.

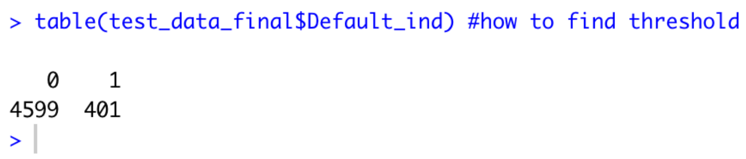
Part 4: Fixing the NA's

After finding the distributions in all of the columns, the next step was finding empty/NA in each column. With the usage of package tidyverse, it was able to fill the empty spaces with the characters NA, so finding them in the dataset was much easier. Using the sapply function alongside the sum and is.na function, we were able to see all of the missing values in the dataset. It turns out the missing values were only in two columns: uti_card_50plus_pct and rep_income. The avg_card_debt also has missing values as well, but that's only because we replaced the 99999 variables into NA. Removing the empty values would not be effective because it could change the model we plan to run. On the other hand, leaving it alone would interfere with our test, and might give us wrong information. To fix this, we used regression imputation to replace the missing values with generated values to help with the flow of our model testing. The imputation works by running a regression model to all of the available predictors/columns to find its estimates of the observed data, then using the regressions weights to replace the missing values. The one we used, Stochastic regression imputation, adds a random error term using the standard errors from each of the predictors/columns as a parameter and uses it to help reproduce the correlation more appropriately. However it should be noted that the stochastic has the potential to reduce the quality of the dataset if done incorrectly, but we decided to go with this route because the randomness is something we'll see in real-life data, so simulating that could give our dataset the boost it needs. To perform the Stochastic regression, we used the R package called mice to run a **random forest** simulation on our dataset and runs the regression on all of the predictors/columns in the dataset to replace the missing values. Once that was done, we created another data frame to hold in all of our changes to the original dataset and used the finished product for our model testing.

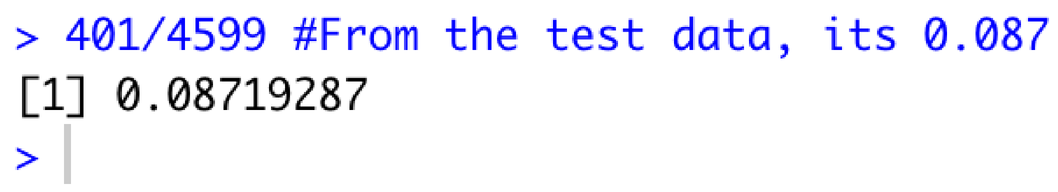


```
        auto_open_ 36_month_num                       uti_card
                              0                              0
             uti_max_credit_line             uti_card_50plus_pct
                              0                           2055
                     rep_income                         States
                           1570                              0
```

Logistic Regression Section:
Meeting the Assumptions
Since the main objective of our model is to figure out whether a person's credit should be defaulted or not, we used logistic regression as one of our main models to use. It fits the five assumptions needed to use the logistic regression. First the response variable Default_ind is binary, its entire column is filled with either yes (1) or no (0). Second the predictor variables like rep_income and mortgages_past_due_6_months_num are all independent from one another. Third there has to be an assurance that the predictors do not have too much correlation with one another, which we made sure of later on in the testing. Fourth the regression assumes linearity of the predictor variables, and last but not least, the fifth assumption states we have to use a large sample size to test this out. With the data that we have, it meets all of the assumptions. Before we started, we had to establish what the classification threshold would be for our logistic regression model. To find it, we used the table function in R to find the number of defaults made and not made in the response variable. Once we found the answer, we divided default made (401) by the default not made (4599) to get the answer 8.7% as our threshold. Once its been established, we moved on to the logistic regression testing.



```
> table(test_data_final$Default_ind) #how to find threshold

   0    1 
4599  401 
> 
```

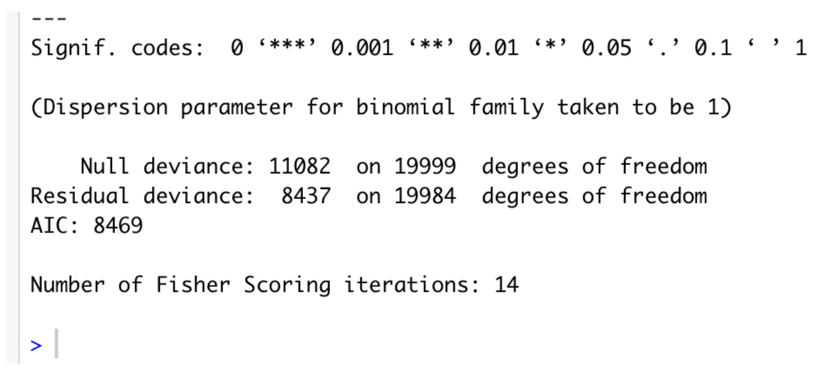


```
> 401/4599 #From the test data, its 0.087
[1] 0.08719287
> 
```

Performing Logistic Regression

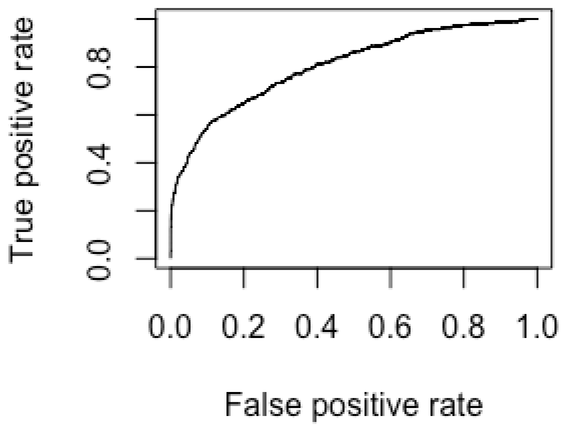
Using the glm function to perform the logistic regression on all of our predictors again. However this time we want to improve the fitting of our model, so we used a technique called the backward elimination procedure. We started with the initial model, all of the predictors, then we observe the p-value and see which has the highest out of everyone. Then we eliminate that variable and continue with this until all of our p-values are statistically significant with the response variable. The remaining variables we have in the model that are statistically significant are: avg_card_debt (p-value:0.000), credit_age (p-value:0.004), card_age (p-value:0.066), non_mtg_acc_past_due_12_months_num (p-value:0.000), non_mtg_acc_past_due_6_months_num (p-value:0.000), inq_12_month_num (p-value:0.000), card_open_36_month_num (p-value:0.092), uti_card (p-value:0.000), ind_acc_XYZ (p-value:0.004), and rep_income (p-value:0.075). To further test the model before we run it through the predictive analysis, we use McFadden's pseudo *R*2 to check the fit of the model. Created by McFadden, D. (1974), the pseudo *R*2, or P2, acts as an alternative for R2 when determining how well models like the logistic regression fits for machine learning analysis. According to McFadden, anywhere between 0.2 - 0.4 means that the model is a very good fit for machine learning predictive analysis. Using the pR2 test obtained from the pscl package, we found out that our P2 is 0.238, well within the recommended range and decided to proceed forward into the predictive analysis test.

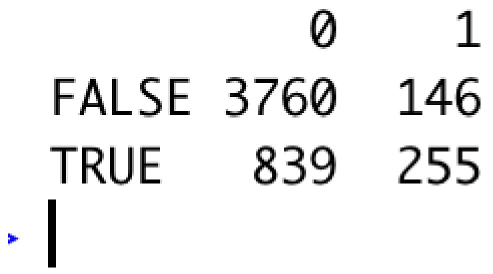
```
Call:
glm(formula = Default_ind ~ avg_card_debt + credit_age + card_age +
    non_mtg_acc_past_due_12_months_num + non_mtg_acc_past_due_6_months_num +
    inq_12_month_num + card_open_36_month_num + uti_card + ind_acc_XYZ +
    rep_income, family = binomial(link = "logit"), data = train_data_final)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.2660  -0.3734  -0.2709  -0.1922   3.3067

Coefficients:
                                     Estimate Std. Error z value Pr(>|z|)
(Intercept)                        -3.679e+00  2.871e-01 -12.816  < 2e-16 ***
avg_card_debt                      -4.955e-05  1.069e-05  -4.636 3.55e-06 ***
credit_age                         -4.029e-03  1.404e-03  -2.868  0.00413 **
card_age                           -2.697e-03  1.467e-03  -1.838  0.06601 .
non_mtg_acc_past_due_12_months_num1  1.155e+00  1.014e-01  11.395  < 2e-16 ***
non_mtg_acc_past_due_12_months_num2  3.357e+00  1.768e-01  18.983  < 2e-16 ***
non_mtg_acc_past_due_12_months_num3  3.482e+00  3.139e-01  11.092  < 2e-16 ***
non_mtg_acc_past_due_12_months_num4  1.573e+01  2.962e+02   0.053  0.95764
non_mtg_acc_past_due_6_months_num1   8.887e-01  1.914e-01   4.642 3.45e-06 ***
non_mtg_acc_past_due_6_months_num2   1.470e+01  2.354e+02   0.062  0.95019
inq_12_month_num                     1.790e-01  1.586e-02  11.291  < 2e-16 ***
card_open_36_month_num1              1.345e-01  8.008e-02   1.680  0.09293 .
card_open_36_month_num2              4.787e-01  3.049e-01   1.570  0.11640
uti_card                             6.024e+00  2.834e-01  21.259  < 2e-16 ***
ind_acc_XYZ1                       -2.077e-01  7.300e-02  -2.845  0.00444 **
rep_income                         -3.250e-06  1.827e-06  -1.779  0.07528 .
```



```
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 11082  on 19999  degrees of freedom
Residual deviance:  8437  on 19984  degrees of freedom
AIC: 8469

Number of Fisher Scoring iterations: 14

> 
```

Predictive Test
To start this off we used the predict function in R and added in the test dataset to see how accurate our model is. Once we made it, then we set up our model to where if it isn’t less than the 8.7% threshold we created, then it will count as 1 (default), otherwise it would be 0 (not default). Once the model is primed for testing, we use 4 different tests to find the accuracy (how often it's correct), precision (percentage it was correct), and sensitivity (percentage of actual positives) of our model. The tests we used are accuracy test, confusion matrix, ROC plot, and AUC. For the accuracy test, we set it up to where the model shows only prediction values from the model that weren’t the same as the actual values from the testing data set, then find the mean from those results. After that we subtracted it to one and received the result 80.58%. Next we ran the confusion matrix to test out its precision and recall numbers. To clarify the confusion matrix, the false and 0 are positive, while the true and 1 are negative. This is made in this way because the customer’s ideal goal would be to not have a default in their account, so a false would be positive. To measure the precision, we take the true positive (3760) and divide it by the sum of true positive (3760) and false positive (146). The result we receive is 96.26% for precision. Next we measured sensitivity by taking the true positive (3760) and dividing it by the sum of true positive (3760) and false negative (839). The result we receive for sensitivity is 81.76%. Considering how both the sensitivity and precision seems to be both high, we used an f1 formula to ensure that false negatives and positives are not high in this model. By using the formula (2*(precision*sensitivity))/(precision + sensitivity), we found the f1 score to be 88.41%, so it seems that the false negatives and positives are low. Next we used the ROC plot to visualize true positive rate against false positive rates, and the results seem to support the accuracy test. With our threshold being 8.7%, it seems that our true positive rates outweigh the false positive rate a good amount until it reaches to the 0.8 in the scale where it evens out. To show a deeper look into the ROC curve, we used the AUC (Area Under the Curve) test and found that the percentage was 80.62%. Since its closer to one, it seems that our model is a good fit to use for analysis.





```
          0    1
FALSE  3760  146
TRUE    839  255
>
```

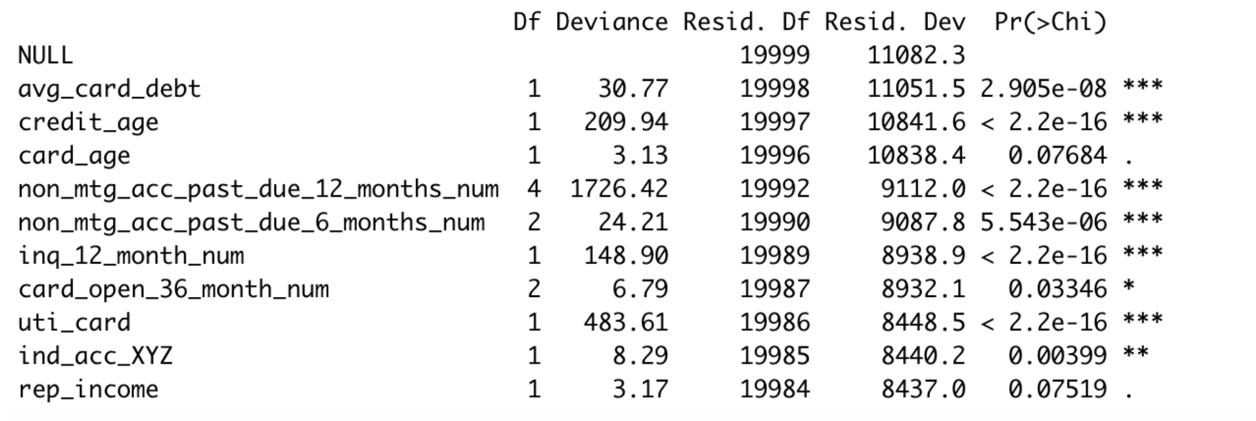
Interpretation for the Logistic Regression
According to the model, the null deviance is 11082 with degrees of freedom 19999. With a high null deviance it seems that the model doesn't seem to fit well, but when the predictors are added in, the deviance decreases to 8437, improving the model by a considerable amount. Now we want to see the which of these predictors caused the most deviances, so we used anova test with chi-square in the mix to give us more detail. Right off the bat the predictors that has the highest deviance are: non_mtg_acc_past_due_12_months_num, inq_12_month_num, uti_card , and credit_age. These predictors significantly reduce the residual deviance from the null deviance, which shows that they affect the response variable the most. The avg_card_debt and non_mtg_acc_past_due_6_months_num variable does have a little deviance, but not to the levels that the other predictors have.

For non_mtg_acc_past_due_12_months_num, for every increase in delinquency, there seems to be a range of 1 to 4 increase chance that the account will receive a default depending on the number of delinquencies the account has. Once it reaches level 5 it doesn't have a significant impact on the default.

For credit_age it seems that with every increase in credit age, there is a -0.006250 decrease chance that the default will occur on the account.

For uti_card, it seems that with every increase in the balance/debt ratio, there is a 5.3388 increase chance that the default will occur on the account.

For inq_12_month_num it seems that every credit request the bank lenders have made, there has been an increasing 0.31723 chance that the default will occur on the account. The estimate increases each levels of the credit request until it reaches level 10, showing that the higher levels may not have a significance on getting a default.



```
                                 Df Deviance Resid. Df Resid. Dev  Pr(>Chi)    
NULL                                             19999    11082.3              
avg_card_debt                     1    30.77     19998    11051.5 2.905e-08 ***
credit_age                        1   209.94     19997    10841.6 < 2.2e-16 ***
card_age                          1     3.13     19996    10838.4   0.07684 .  
non_mtg_acc_past_due_12_months_num 4 1726.42     19992     9112.0 < 2.2e-16 ***
non_mtg_acc_past_due_6_months_num 2    24.21     19990     9087.8 5.543e-06 ***
inq_12_month_num                  1   148.90     19989     8938.9 < 2.2e-16 ***
card_open_36_month_num            2     6.79     19987     8932.1   0.03346 *  
uti_card                          1   483.61     19986     8448.5 < 2.2e-16 ***
ind_acc_XYZ                       1     8.29     19985     8440.2   0.00399 ** 
rep_income                        1     3.17     19984     8437.0   0.07519 .  
```

Random Forest Building Section:

To give direction to our data, we decided we had to understand the meaning of credit a bit more deeply. Our determination was that credit, from a bank's point of view, was simply an investment in a consumer, with an expectation to get the investment back in addition to the interest paid. We also created groups based on similarities in the data, which allowed for easier application of specific tests. The groups were determined to be Card Utilization (uti_card, uti_50plus_pct, uti_max_credit_line, uti_card_50plus_pct), Inquiries and Openings (inq_12_month_num, card_inq_24_month_num, card_open_36_month_num, auto_open_36_month_num), Past Due Balances (non_mtg_acc_past_due_12_months_num, non_mtg_acc_past_due_6_months_num, mortgages_past_due_6_months_num, credit_past_due_amount), and User account statistics(avg_card_debt, credit_age, credit_good_age, card_age, ind_acc_XYZ, rep_income, States).

We thought that this grouping was helpful to see trends in data that might otherwise be covered by noise in the data set that might not reveal the true measure of certain variables.

Past Due Balances

The analysis of our past due balances began by creating an additional column, called past_due_total_12_months, which included both the mortgage and non-mortgage payments that had been missed in the last 12 months. The thinking behind this was to whittle down the number of variables in the final model while maintaining the same amount of information about each of the accounts.

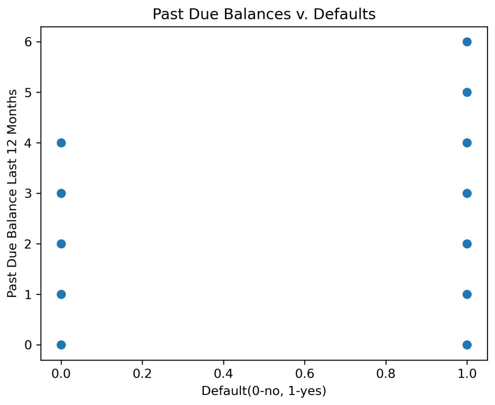
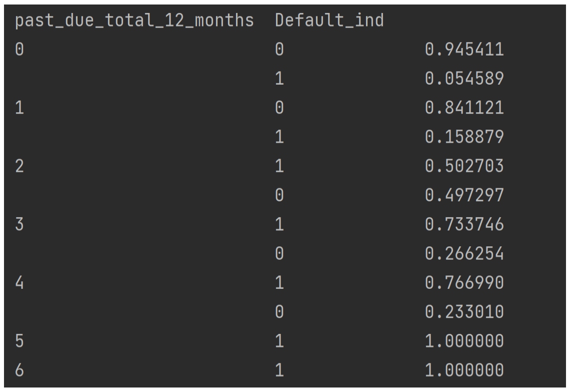


Figure1



```
past_due_total_12_months  Default_ind
0                         0              0.945411
                          1              0.054589
1                         0              0.841121
                          1              0.158879
2                         1              0.502703
                          0              0.497297
3                         1              0.733746
                          0              0.266254
4                         1              0.766990
                          0              0.233010
5                         1              1.000000
6                         1              1.000000
```

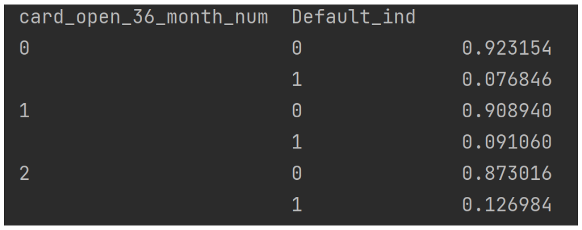
Figure2

The combination of all of the past due balances revealed to us that there seemed to be a strong connection between the number of past due to balances an account had and the default rate, as the percentage of defaults steadily increased, reaching the level where 100% of all accounts have defaulted with 5 or more past due balances. The evidence provided by the ratios of defaults to past due balances was sufficient to keep our total past due balances in our model.

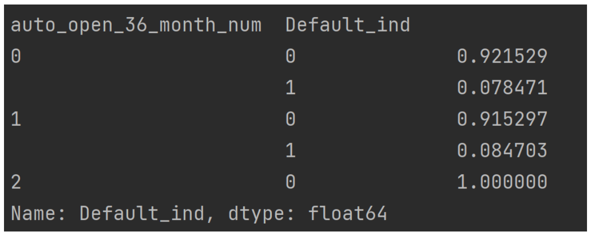
We next looked at how the past due amount related to our model. Figures 3,4 all initially suggested that the model was extremely skewed, however we noticed that there was a significant number of users that had 0 due in past due balances. Since the median of the users was 0 for both defaulted and non-defaulted accounts, we determined that the skewness was based off of a small sample of the overall dataset. In Figure4 we see that there seems to be a slight correlation between past due amount and defaulting, but nothing that suggested it was a direct measure of whether the account was defaulted or not. We created a small model of the past due balances category to which variables had the most weight out of the few selected and saw that mortgages past due over 6 months seemed to have the greatest affect on our outcome, with an odds ratio of 3.15. When interpreted in context this suggests that for every additional past due mortgage in the past 6 months, the likelihood of default increase by a factor of 3.15. credit past due amount did not increase.

Credit inquiries and openings

The next category analyzed was our inquiries and openings. The most interesting information gathered from this was that auto and credit card openings over the past 36 months almost directly correlated to the overall defaulting rate of our sample.



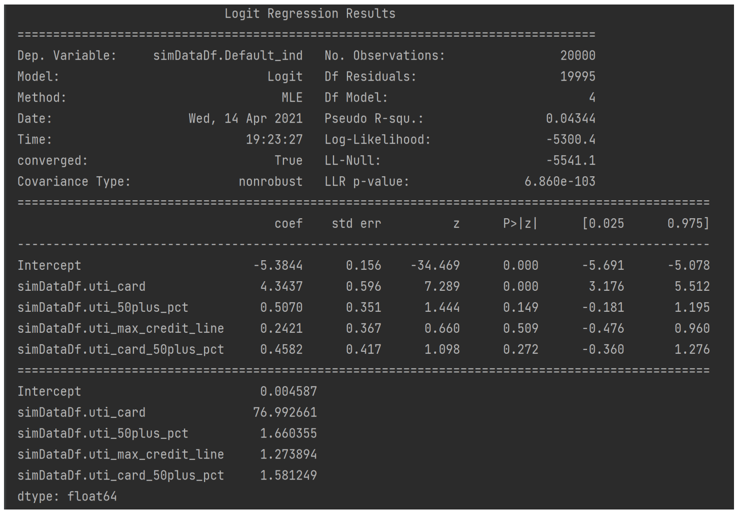
```
card_open_36_month_num  Default_ind
0                       0              0.923154
                        1              0.076846
1                       0              0.908940
                        1              0.091060
2                       0              0.873016
                        1              0.126984
```



```
auto_open_36_month_num  Default_ind
0                       0              0.921529
                        1              0.078471
1                       0              0.915297
                        1              0.084703
2                       0              1.000000
Name: Default_ind, dtype: float64
```

This can be attributed to the fact that the customers that have a card or auto loan opened within the last 36 months means they have been approved by a credit lender in the past 36 months, so the default rate matches the average of the entire sample. This suggests that there may not be independence between defaulting rates and new lines of credit being opened, which could lead to multicollinearity in our initial model.

Card Utilization

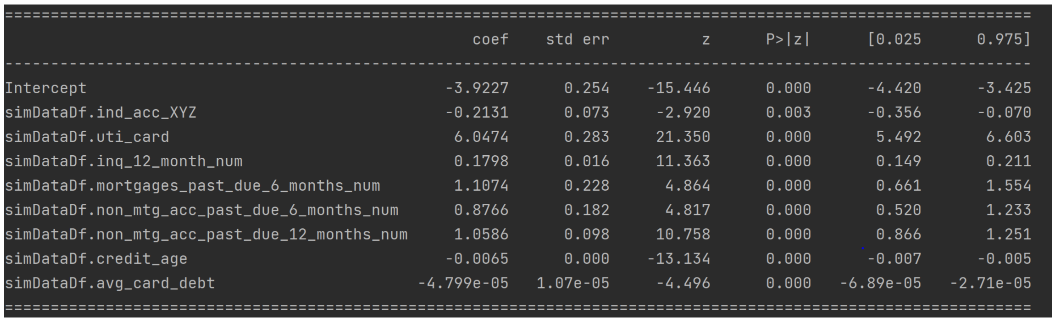


```
                           Logit Regression Results
==============================================================================
Dep. Variable:     simDataDf.Default_ind   No. Observations:                20000
Model:                             Logit   Df Residuals:                    19995
Method:                              MLE   Df Model:                            4
Date:                   Wed, 14 Apr 2021   Pseudo R-squ.:                 0.04344
Time:                           19:23:27   Log-Likelihood:                -5300.4
converged:                          True   LL-Null:                       -5541.1
Covariance Type:               nonrobust   LLR p-value:                6.860e-103
================================================================================================
                                   coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------------------------
Intercept                       -5.3844      0.156    -34.469      0.000      -5.691      -5.078
simDataDf.uti_card               4.3437      0.596      7.289      0.000       3.176       5.512
simDataDf.uti_50plus_pct         0.5070      0.351      1.444      0.149      -0.181       1.195
simDataDf.uti_max_credit_line    0.2421      0.367      0.660      0.509      -0.476       0.960
simDataDf.uti_card_50plus_pct    0.4582      0.417      1.098      0.272      -0.360       1.276
================================================================================================
Intercept                         0.004587
simDataDf.uti_card               76.992661
simDataDf.uti_50plus_pct          1.660355
simDataDf.uti_max_credit_line     1.273894
simDataDf.uti_card_50plus_pct     1.581249
dtype: float64
```

Analyzing our card utilization, we saw that card utilization seemed to be the only measure that significantly increased the odds of defaulting on your account.

Variable elimination

Now we have reached a point where we can eliminate variables that are not very significant to our model in a stepwise manner and reduce our VIF to improve the overall model. We first remove our highest p-valued variable which in this case is 'past_due_total_12_months'. This improved our accuracy to 85.5% but our VIF remained unchanged. The next largest p-value in our model was 'tot_credit_debt', upon removal we see that VIF and accuracy remain almost the same. The next value we remove is 'state_MS' which is one of our dummy variables created for states. 'state_SC' , 'uti_max_credit_line', and 'card_inq_24_months' follow. The final remaining variables are tabulated below.

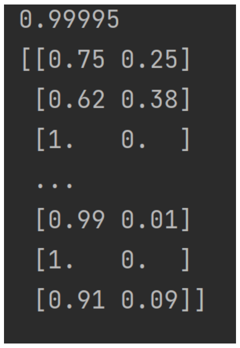


```
==============================================================================================================
                                                 coef    std err          z      P>|z|      [0.025      0.975]
--------------------------------------------------------------------------------------------------------------
Intercept                                     -3.9227      0.254    -15.446      0.000      -4.420      -3.425
simDataDf.ind_acc_XYZ                         -0.2131      0.073     -2.920      0.003      -0.356      -0.070
simDataDf.uti_card                             6.0474      0.283     21.350      0.000       5.492       6.603
simDataDf.inq_12_month_num                     0.1798      0.016     11.363      0.000       0.149       0.211
simDataDf.mortgages_past_due_6_months_num      1.1074      0.228      4.864      0.000       0.661       1.554
simDataDf.non_mtg_acc_past_due_6_months_num    0.8766      0.182      4.817      0.000       0.520       1.233
simDataDf.non_mtg_acc_past_due_12_months_num   1.0586      0.098     10.758      0.000       0.866       1.251
simDataDf.credit_age                          -0.0065      0.000    -13.134      0.000      -0.007      -0.005
simDataDf.avg_card_debt                    -4.799e-05   1.07e-05     -4.496      0.000   -6.89e-05   -2.71e-05
==============================================================================================================
```

We now can redo our model and check our accuracy on the test data. We have run into a problem here, which may be a result of the linear regression or a result of mistaken calculation. We see that our accuracy for the model is 93%, but our VIF is above 10 suggesting that a large amount multicollinearity still exists between the data, and there are no more variables that we can remove.

Random Forest

Our random forest model gives us a 99% accuracy on our test data, which also likely means that the data is overfit. The issue here is that there are no more variables that we can safely remove from the data set as they all have p-values that imply importance to the model.



```
0.99995
[[0.75 0.25]
 [0.62 0.38]
 [1.   0.  ]
 ...
 [0.99 0.01]
 [1.   0.  ]
 [0.91 0.09]]
```

Conclusion

Based on the model that has been produced, we do not think that a logistic regression model would be favorable over a random forest model as there is too much dependence between variables for it to be a useful measure. This was noted early on when we noticed that many of the variables were able to be grouped into categories that had common elements (missed payments over 12 months and missed payments over 6 months). In our model, account holders actually had a slightly lower probability of receiving a new account based on the odds ratio of about .8. This suggests that having an account with the bank lowers your chance of card approval by a factor of .8. With the errors in our models, we see that there is essentially a 0% chance that a customer is rejected, which is clearly a problematic interpretation.

When we looked at the previous deviance from the model and from the anova test, it seems that having an account in the past does not significantly impact on receiving a default. From a glance it does seem like it has an effect on getting a default due to the estimate being -0.31 and the p-value being very significant. But when we observe the deviance, it was only a mere 8 compared to other predictors that have deviances that are around 400. However, we did find that previous account holders who had a delinquency in their record do have a significant impact on the increased chance of receiving a default, particularly those who have those delinquencies for a period of 12 months in non-mortgages credit.

We recommend that the bank use credit age, number of non-mortgages credit accounts by applicants considered as delinquents in the past 12 months, number of credit inquiry, and your uti card balance as their main predictors, because there seems to be a high correlation between these variables and the account being defaulted. We would also recommend that they take a look at average credit card debt and non-mortgages credit accounts by applicants considered as delinquents in the past 6 months in the case they aren't really sure and need a few more data points to solidify the decision.

If we worked at Wells Fargo and a client asks me why there is a default on their account, we would explain to them that the number of delinquencies they had in the past and the uneven ratio between credit available and debt is the main reason why they received a default on their account. If they do have a long credit history with the bank, it is something to consider, but it wouldn’t be enough to change the decision if the numbers from past delinquencies were too high. Even with all of these interpretations from our analysis, it's important to take into account the human element when determining the decision about their account because by the end of the day, we don’t know everything, we’re just human.

Appendix

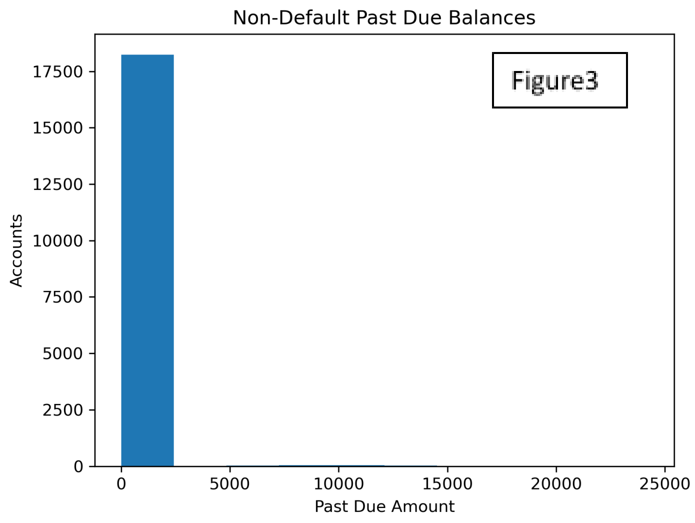


Figure3

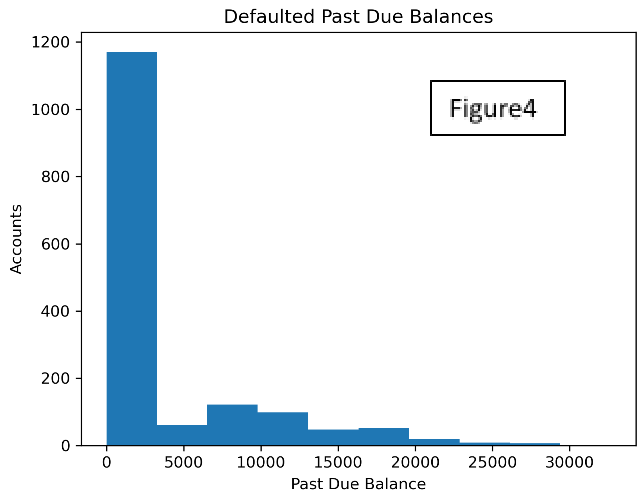


Figure4

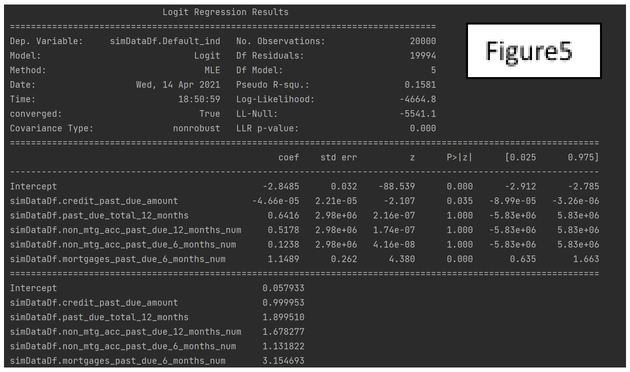
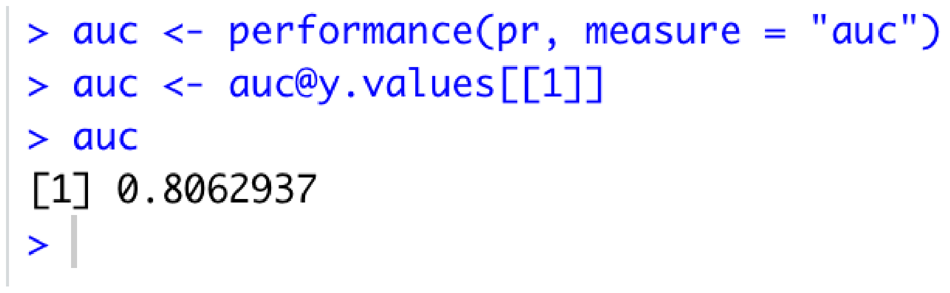
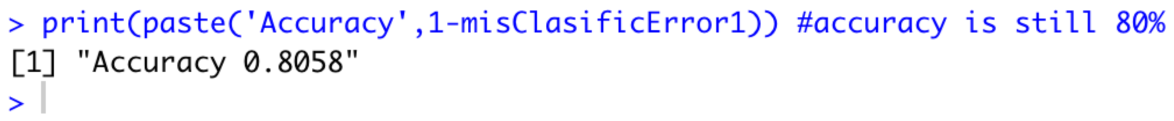


Figure5

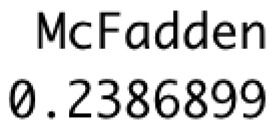
```
                           Logit Regression Results
==============================================================================
Dep. Variable:     simDataDf.Default_ind   No. Observations:                20000
Model:                             Logit   Df Residuals:                    19994
Method:                              MLE   Df Model:                            5
Date:                   Wed, 14 Apr 2021   Pseudo R-squ.:                  0.1581
Time:                           18:50:59   Log-Likelihood:                -4664.8
converged:                          True   LL-Null:                       -5541.1
Covariance Type:               nonrobust   LLR p-value:                     0.000
=============================================================================================================
                                               coef    std err          z      P>|z|      [0.025      0.975]
-------------------------------------------------------------------------------------------------------------
Intercept                                   -2.8485      0.032    -88.539      0.000      -2.912      -2.785
simDataDf.credit_past_due_amount          -4.66e-05   2.21e-05     -2.107      0.035   -8.99e-05   -3.26e-06
simDataDf.past_due_total_12_months           0.6416   2.98e+06   2.16e-07      1.000   -5.83e+06    5.83e+06
simDataDf.non_mtg_acc_past_due_12_months_num    0.5178   2.98e+06   1.74e-07      1.000   -5.83e+06    5.83e+06
simDataDf.non_mtg_acc_past_due_6_months_num     0.1238   2.98e+06   4.16e-08      1.000   -5.83e+06    5.83e+06
simDataDf.mortgages_past_due_6_months_num       1.1489      0.262      4.380      0.000       0.635       1.663
=============================================================================================================
Intercept                                   0.057933
simDataDf.credit_past_due_amount            0.999953
simDataDf.past_due_total_12_months          1.899510
simDataDf.non_mtg_acc_past_due_12_months_num    1.678277
simDataDf.non_mtg_acc_past_due_6_months_num     1.131822
simDataDf.mortgages_past_due_6_months_num       3.154693
```



```
> auc <- performance(pr, measure = "auc")
> auc <- auc@y.values[[1]]
> auc
[1] 0.8062937
> 
```



```
> print(paste('Accuracy',1-misClasificError1)) #accuracy is still 80%
[1] "Accuracy 0.8058"
> 
```



```
  McFadden 
 0.2386899 
```