GiveMeSomeCredit

a competition by Kaggle

Overview of Training Data

Feature	Min	Max	Mean	Std
RevolvingUtilizationOfUnsecuredLines	0	50708	6.048	249.755
age	0	109	52.295	14.772
NumberOfTime30-59DaysPastDueNotWorse	0	98	0.421	4.193
DebtRatio	0	329664	353.005	2037.819
MonthlyIncome	0	3008750	6670.221	14384.674
NumberOfOpenCreditLinesAndLoans	0	58	8.453	5.146
NumberOfTimes90DaysLate	0	98	0.266	4.169
NumberRealEstateLoansOrLines	0	54	1.018	1.130
NumberOfTime60-89DaysPastDueNotWorse	0	98	0.240	4.155
NumberOfDependents	0	20	0.757	1.115

Overview of Training Data

- Imbalanced, #samples(150k) >> #features(10)
  - Number of y=1 : 6,68% total size

Value	Count	Percent
0	139974	93.32%
1	10026	6.68%

- Missing data
  - 29731 missing in MonthlyIncome
  - 3924 missing in NumberOfDependents

Summary of the problem : binary classification with imbalanced classes and missing data

Proposed methods

- Missing data
  - Filing missing data with mean values
- Imbalance
  - Do nothing
  - Upsample the minorties or undersample the mijorities.
  - Anomaly detection
  - Othe methods such as using ensembles and boosting

Proposed methods

- The following are chosen to run
  - **Logistic regression** on full sample size
  - **Neural network** with 1 hidden layer (#nodes = 5), all y=1 samples are selected, 20k of y=0 samples are randomly seleted.
  - Anomaly detection using **Isolation Forest** which was reported to have good results with skewed data and fast to run (by Fei Tony Liu et al., 2008)
  - **RUSBoost**, a hybrid approach to alleviate imbalance class (by Christ Seiffert et al., 2010)
- Features are standardised
- We will proceed without any data cleaning except dealing with missing data, then those four methods are run again after data is cleaned to llok into the effect of data cleaning on results.
- Machine used: Intel Core i5-6200U 2.3Ghz, 8GB of RAM
- Scripts are in MATLAB

Results

Logistic Regression

lambda	0.01	1	100
Training AUC	0.698524	0.698559	0.699888
Testing AUC	0.696193	0.696241	0.698956

Expectedly, the simplest method yields the worst results.

Neural Network

lambda	0.01	1	100
Training AUC	0.832331	0.831945	0.811223
Testing AUC	0.835876	0.835416	0.813816

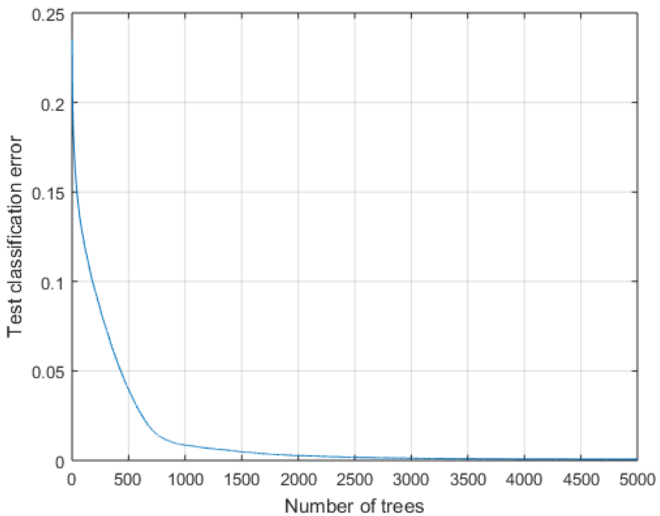
Isolation Forest

#subsample	128	256	512	1024
Training AUC	0.809209	0.802698	0.791381	0.786107
Testing AUC	0.812078	0.805104	0.793656	0.788564

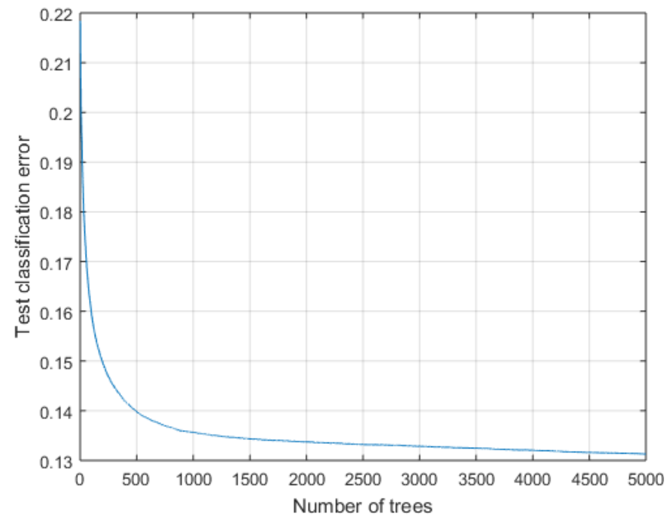
Not better than a simple Neural Network

RUSBoost

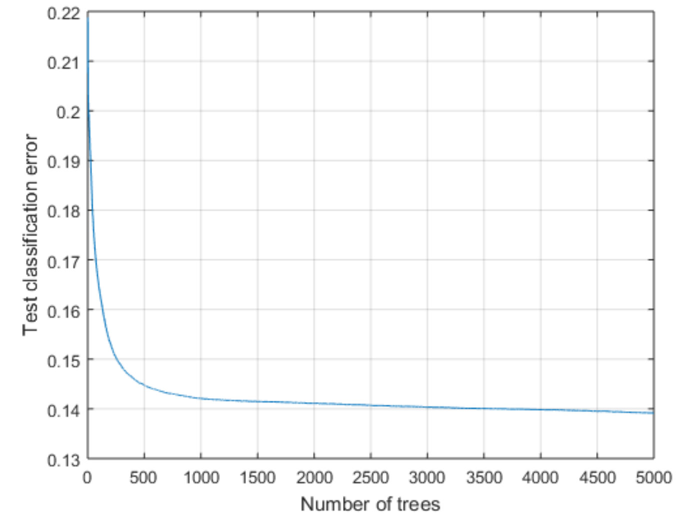
minleaf_#ensemble	**2_5000**	**50_5000**	**100_5000**	**300_5000**	**5000_5000**
Training AUC	0.999889	0.895696	0.879818	0.868595	0.822152
Testing AUC	0.827538	0.867186	**0.867329** Rank 95	0.865336	0.823692



Min leaf size=2
#Ensembles = 5000
Overfitting

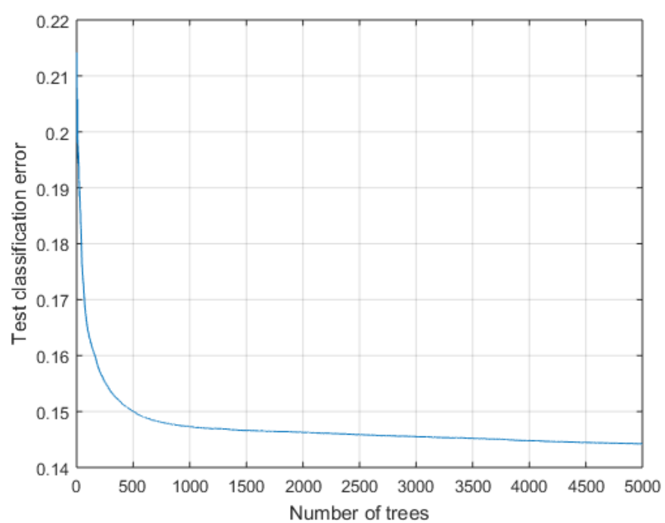


Min leaf size=50
#Ensembles = 5000

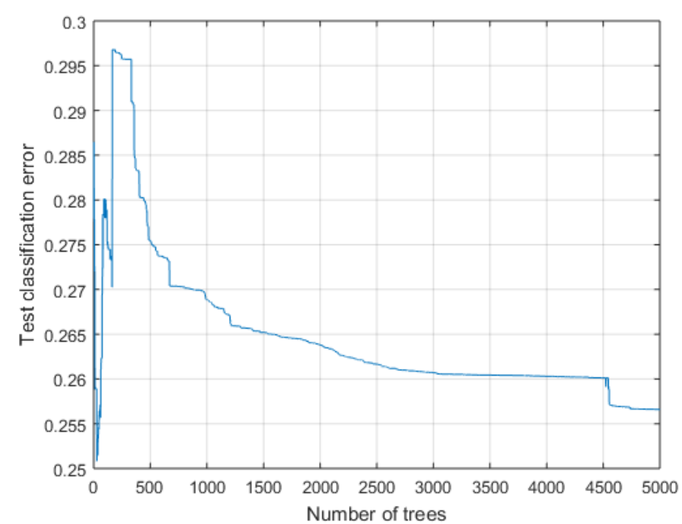


Min leaf size=100
#Ensembles = 5000

Loss function of RUSBoost



Min leaf size=300
#Ensembles = 5000



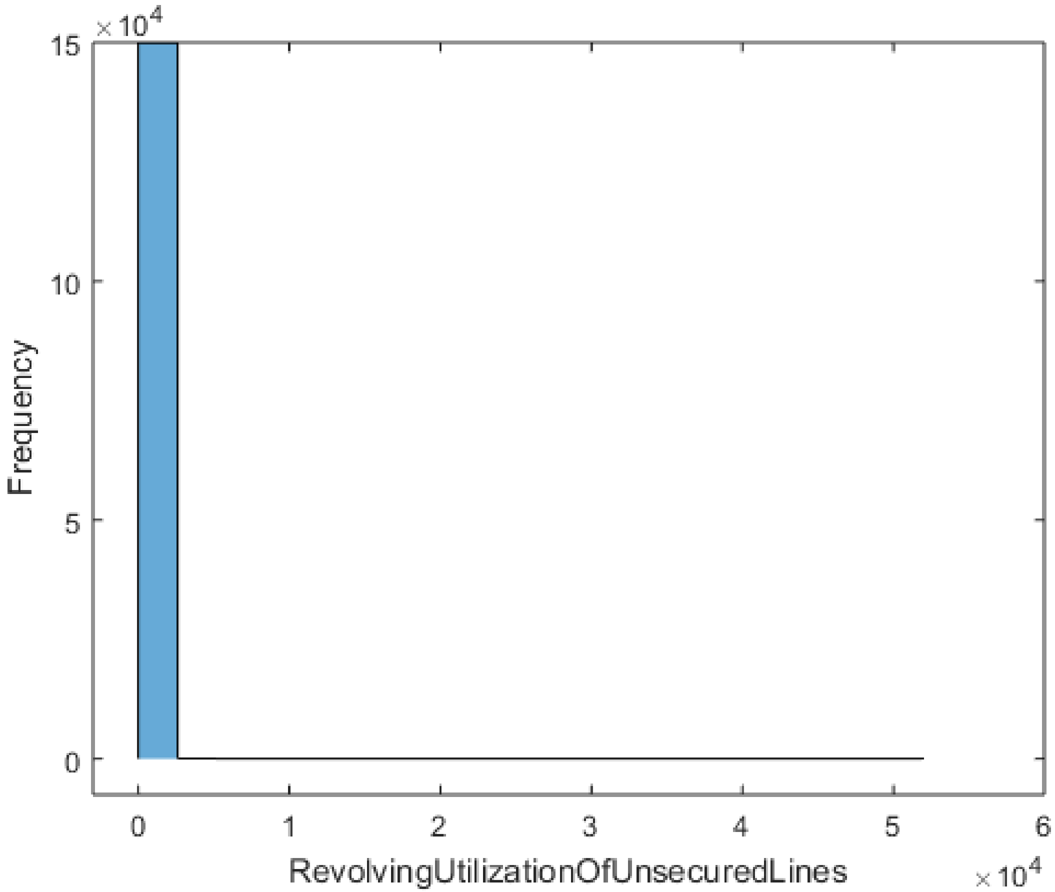
Min leaf size=5000
#Ensembles = 5000

- Except RUSBoost, other methods have considerably low AUC scores
- Would need a more careful look at the variables, after some data cleaning the results might be improved.

Let try Data Cleaning
Examining Each Feature

RevolvingUtilizationOfUnsecuredLines

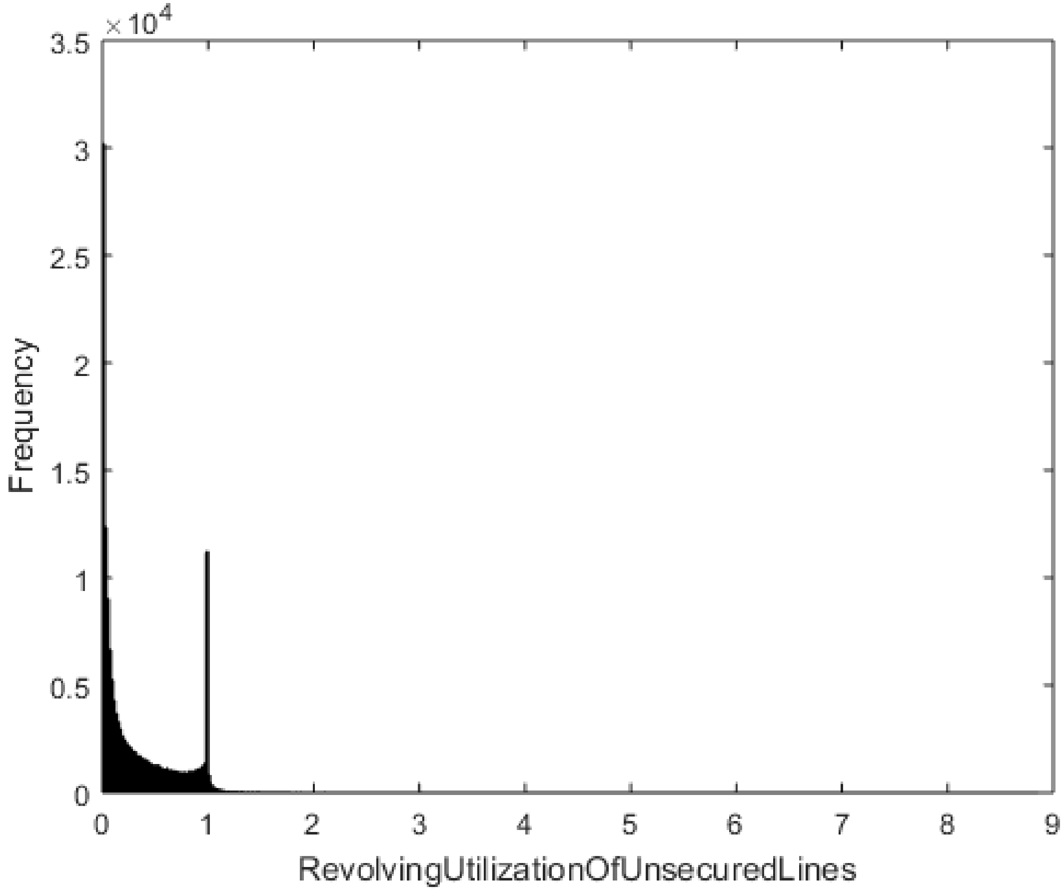
Before Cleaning



Skewed distribution, with 241 samples with value >10 which is unreasonable, **decided to drop these samples.**

RevolvingUtilizationOfUnsecuredLines

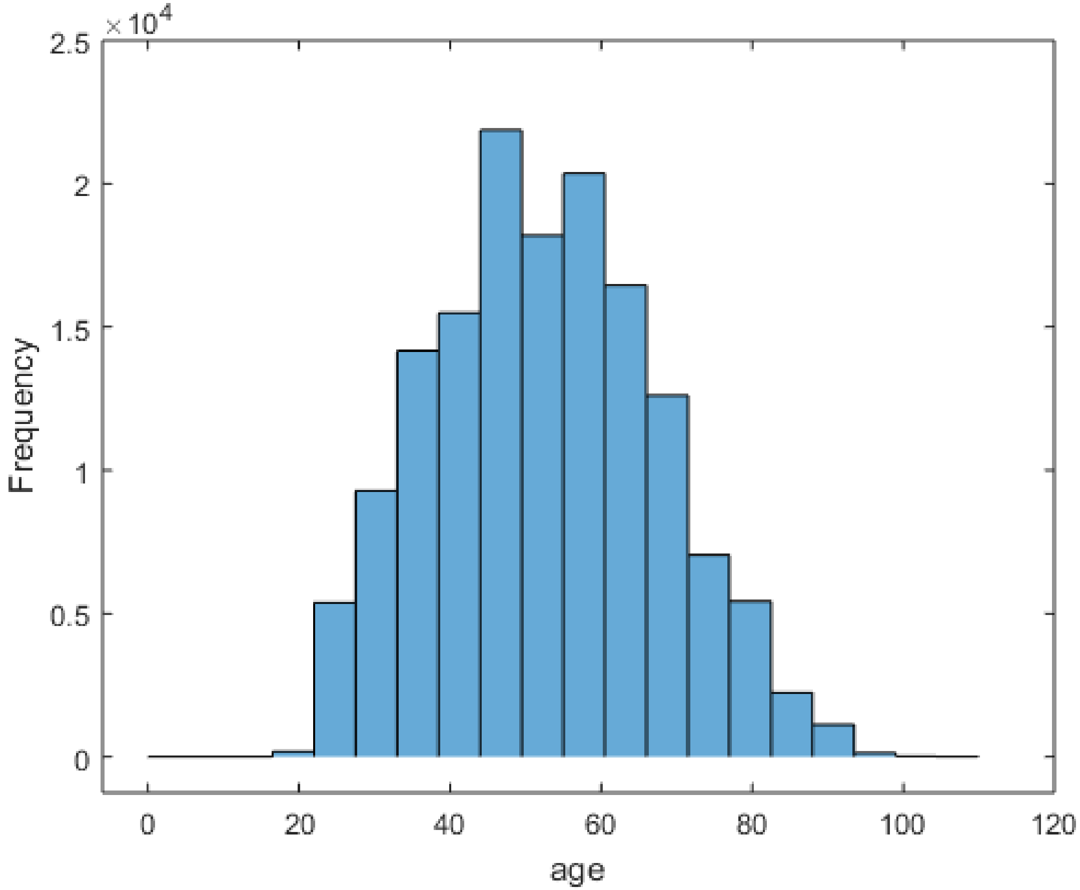
After Cleaning



Much better distribution

Age

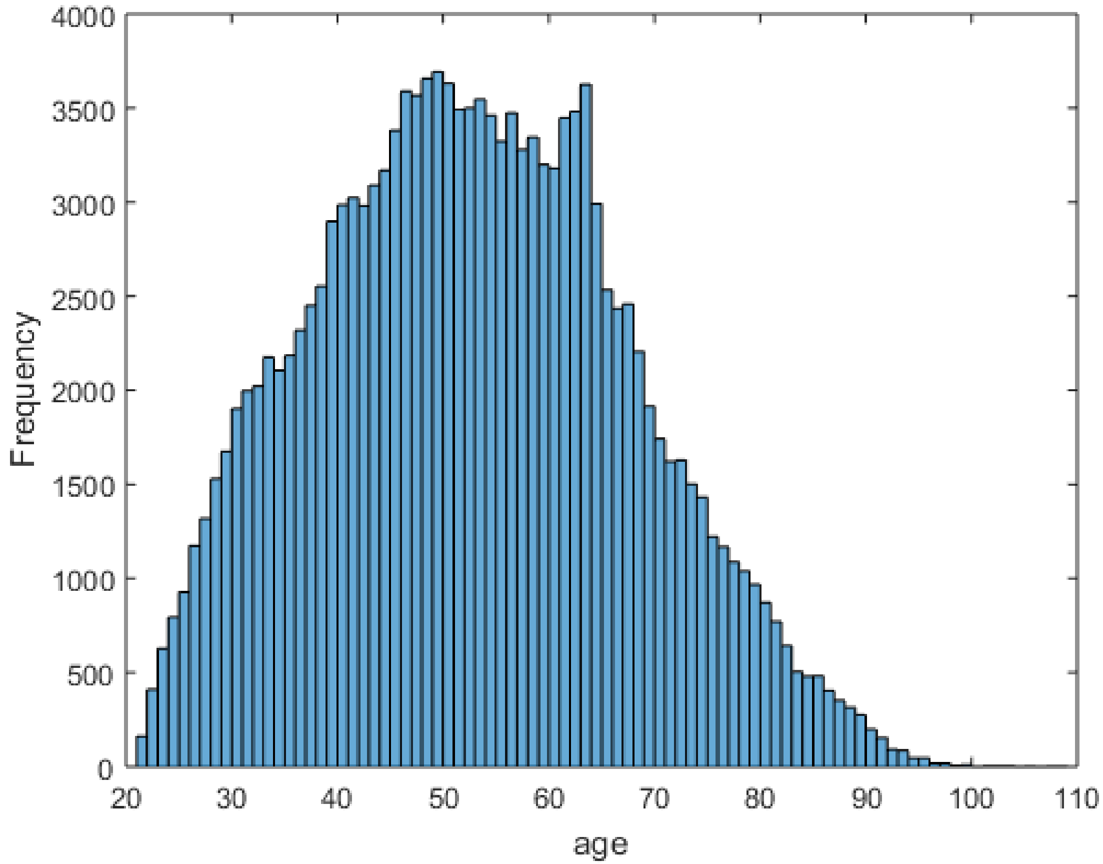
Before Cleaning



The distribution looks alright, except there is one outliner with zero value need to be removed

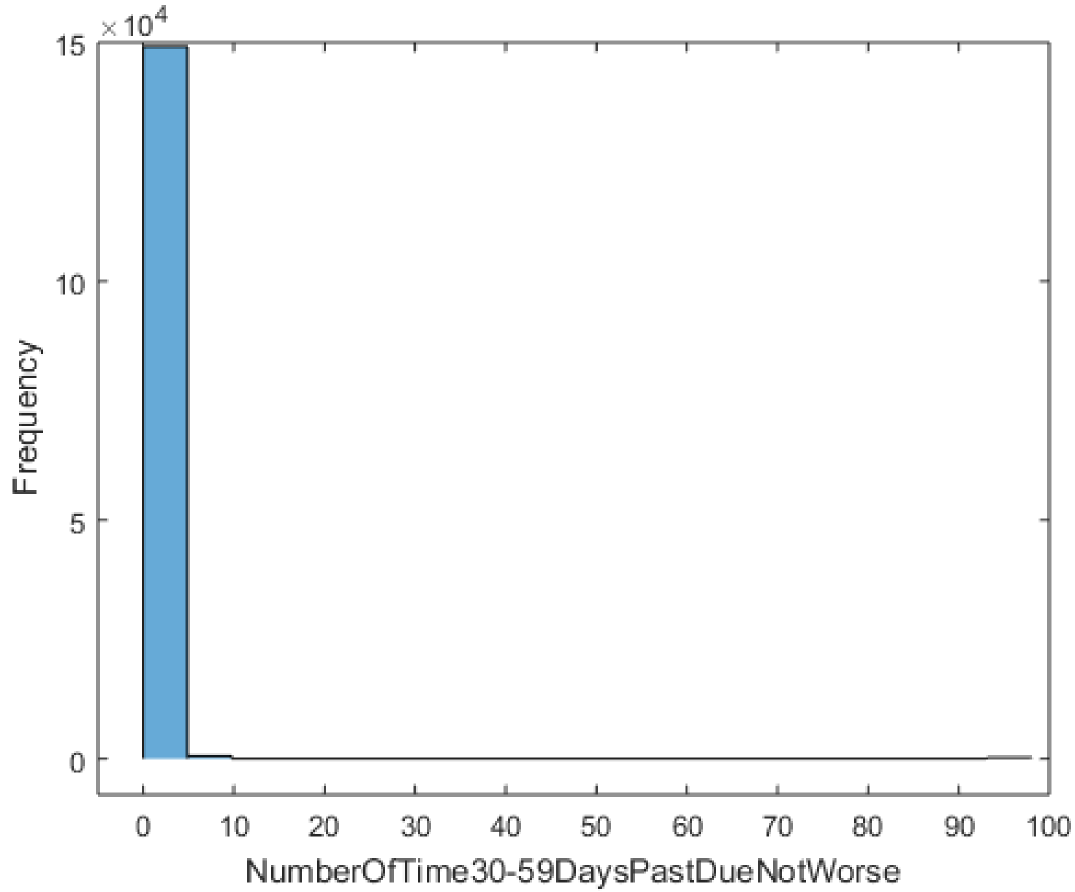
Age

After Cleaning



NumberOfTime30-59DaysPastDueNotWorse

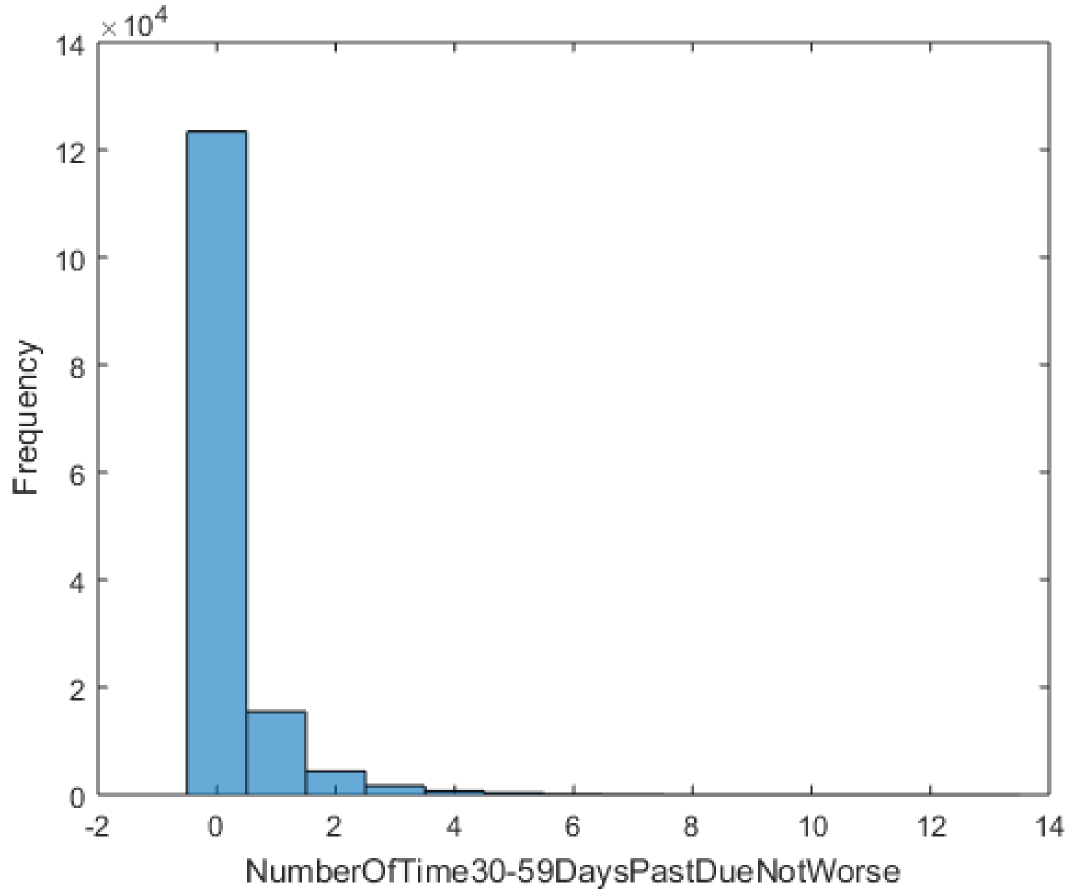
Before Cleaning



Skewed distribution with some outliners with distinct values >90, **decided to drop them**

NumberOfTime30-59DaysPastDueNotWorse

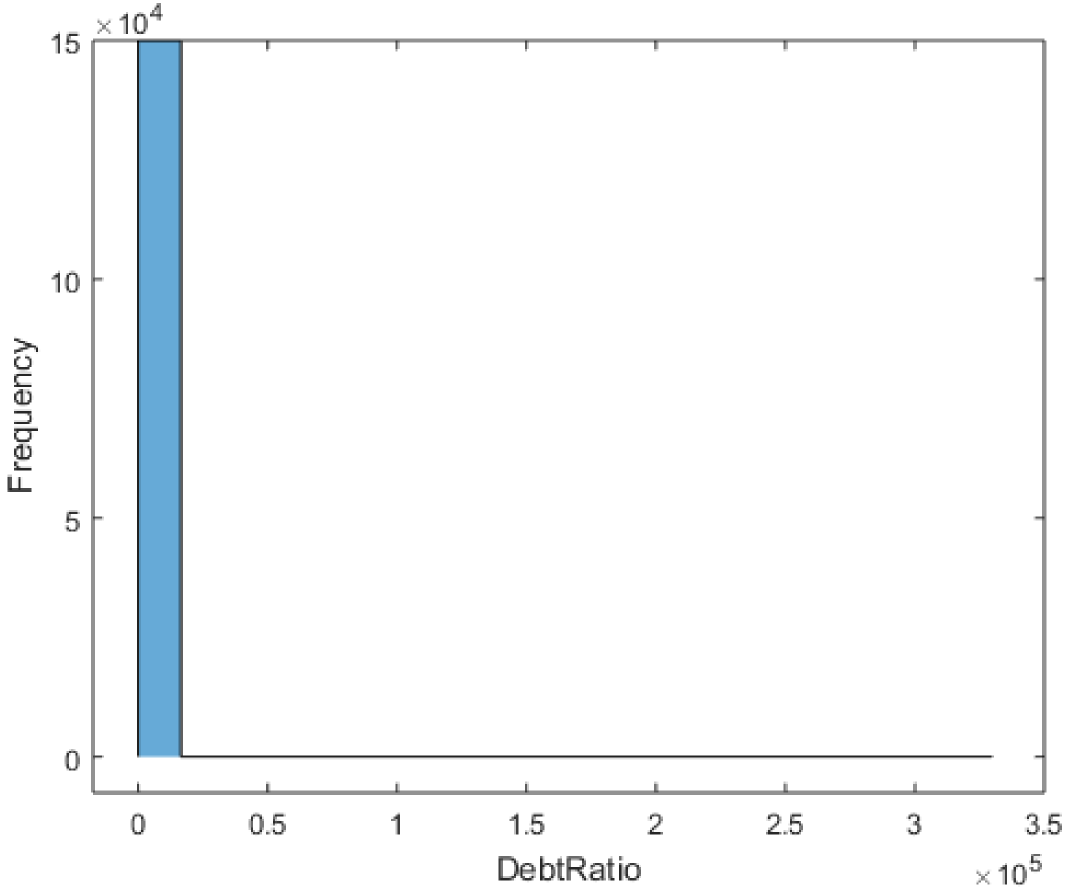
After Cleaning



Better distribution

DebtRatio

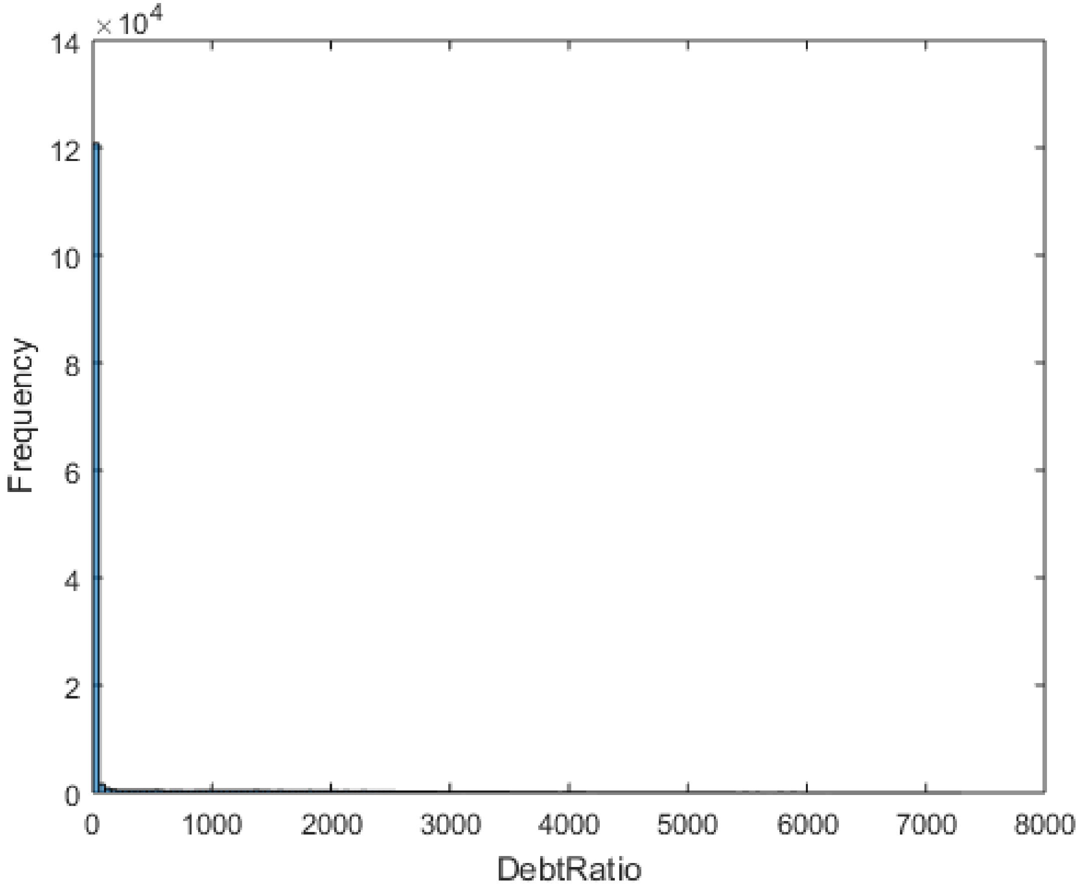
Before Cleaning



Trim 99.7% (3 standard deviations) on top to remove outliners

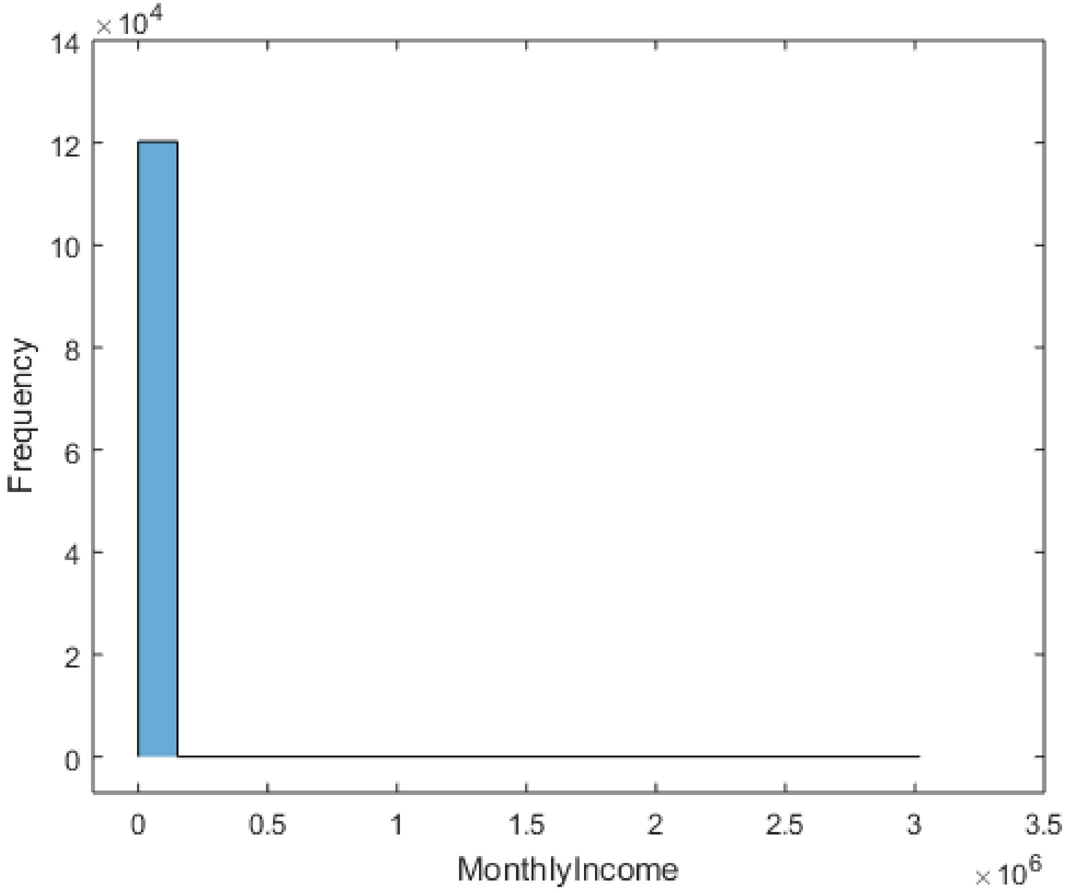
DebtRatio

After Cleaning



MonthlyIncome

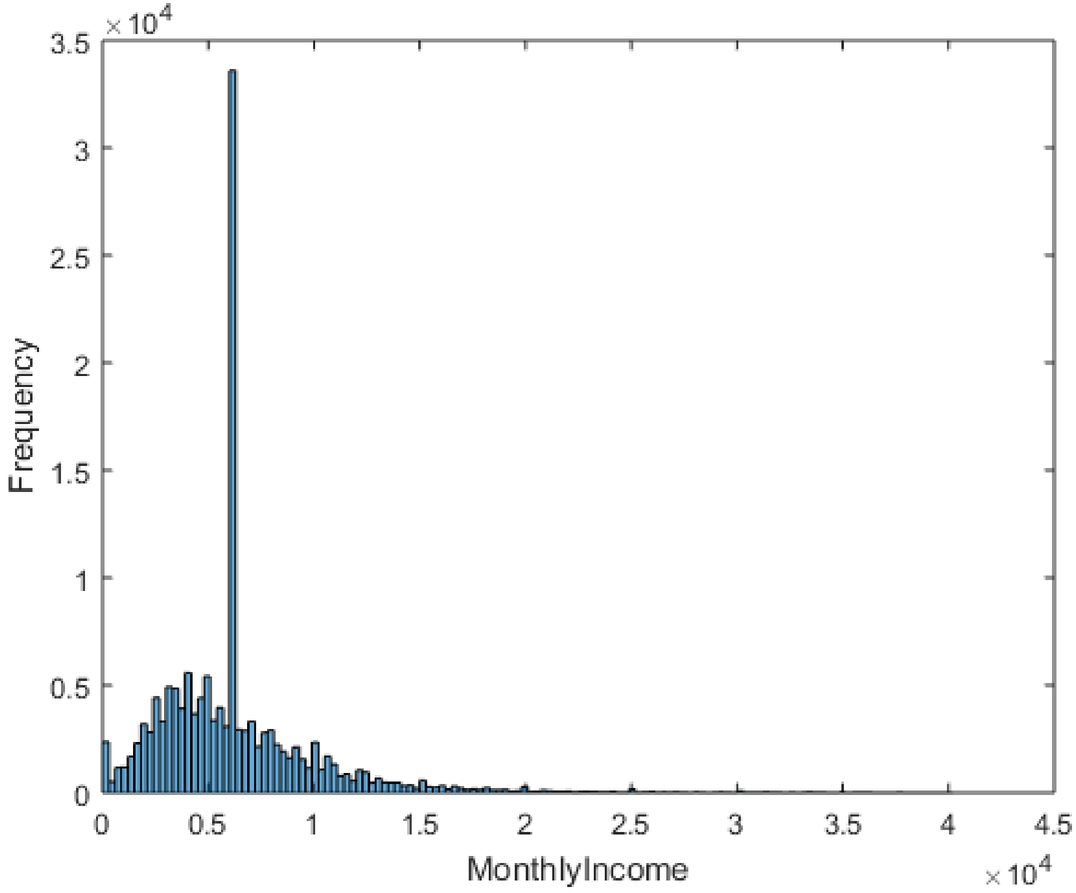
Before Cleaning



Trim 99.7% (3 standard deviations) on top to remove outliners
This also has some NA values which need to be replace by mean values of remaining samples.

MonthlyIncome

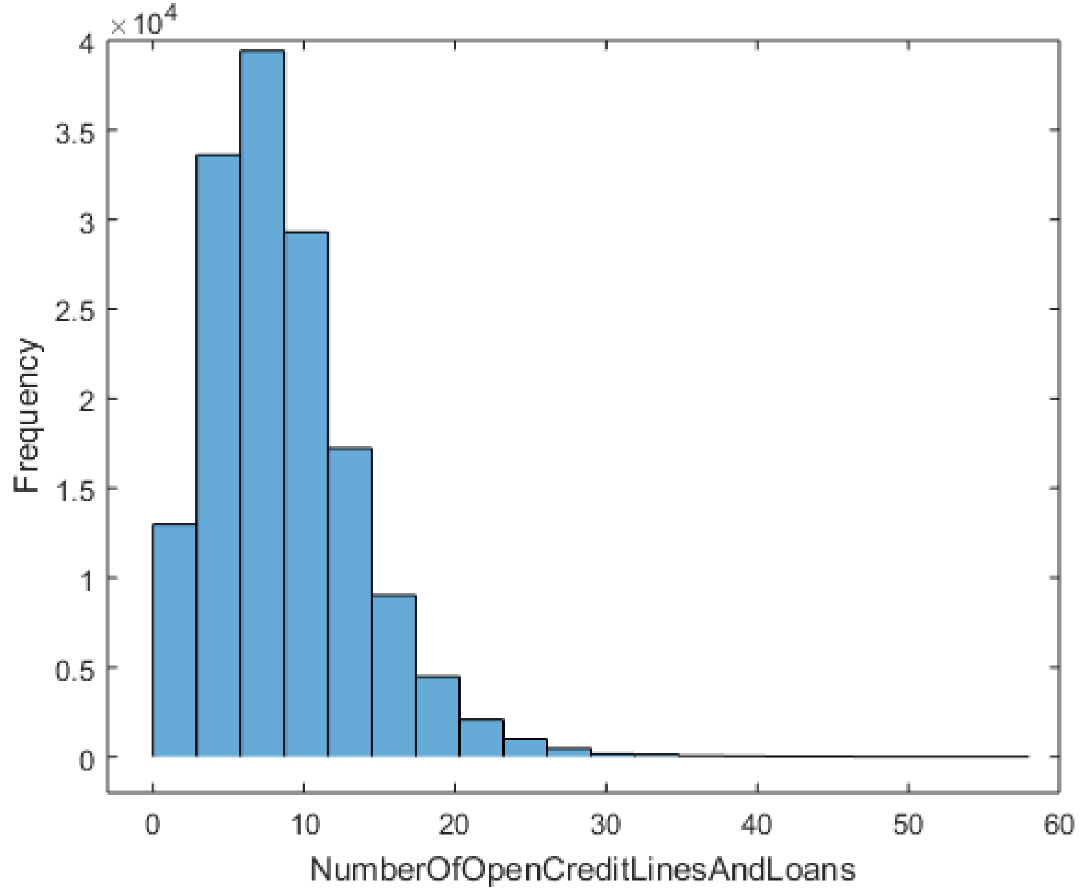
After Cleaning



Much better distribution except the hike at mean value

NumberOfOpenCreditLinesAndLoans

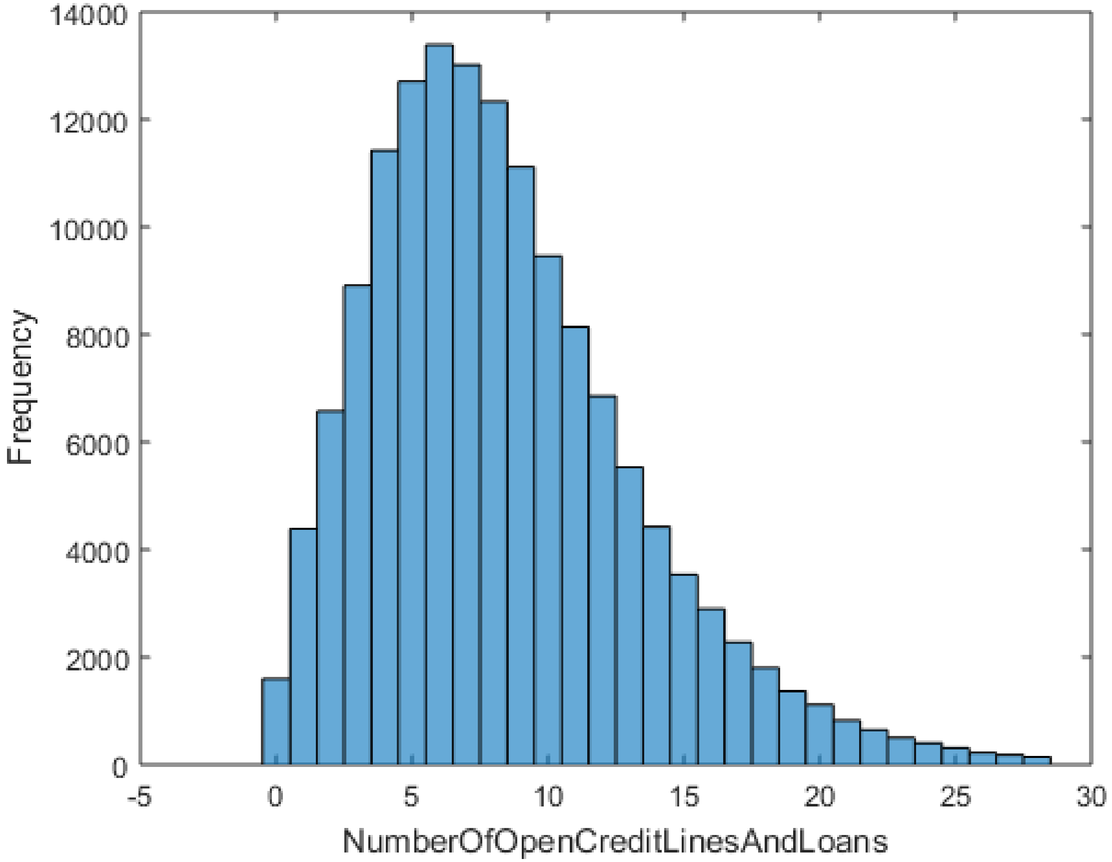
Before Cleaning



This distribution looks quite alright, further trim 99.7% on top to remove outliners

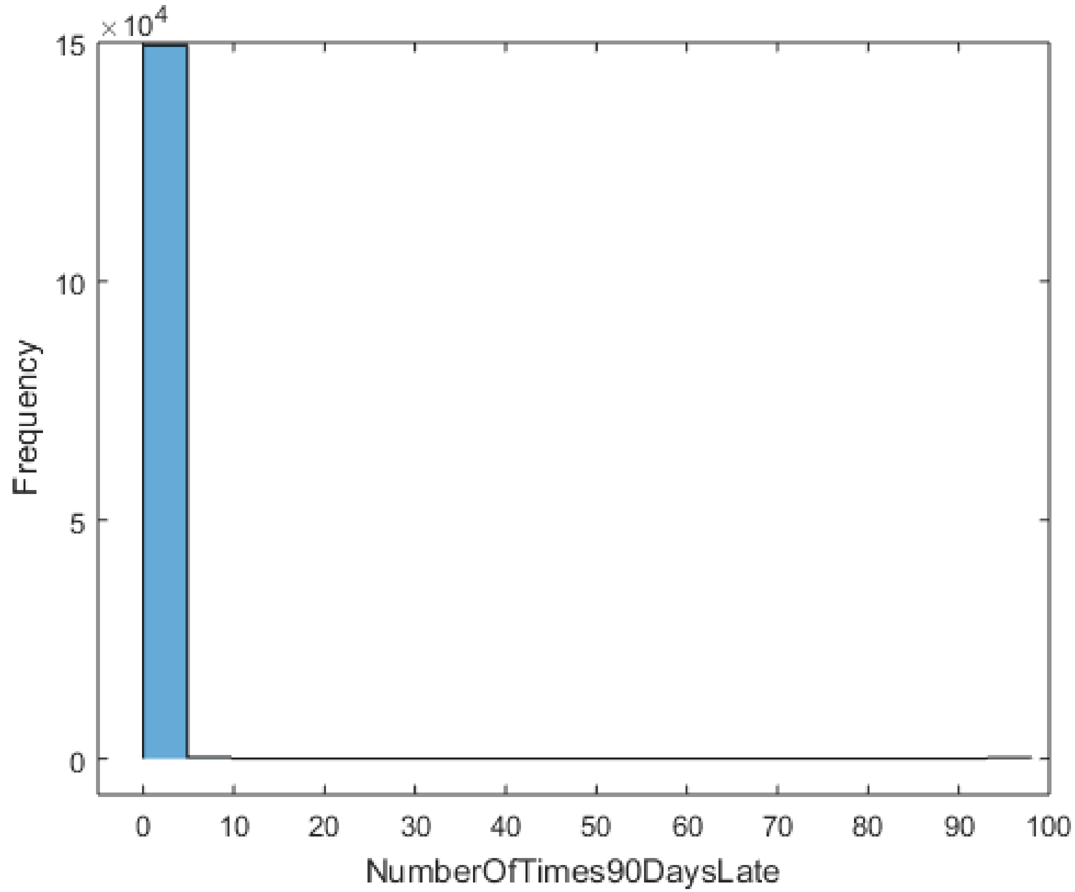
NumberOfOpenCreditLinesAndLoans

After Cleaning



NumberOfTimes90DaysLate

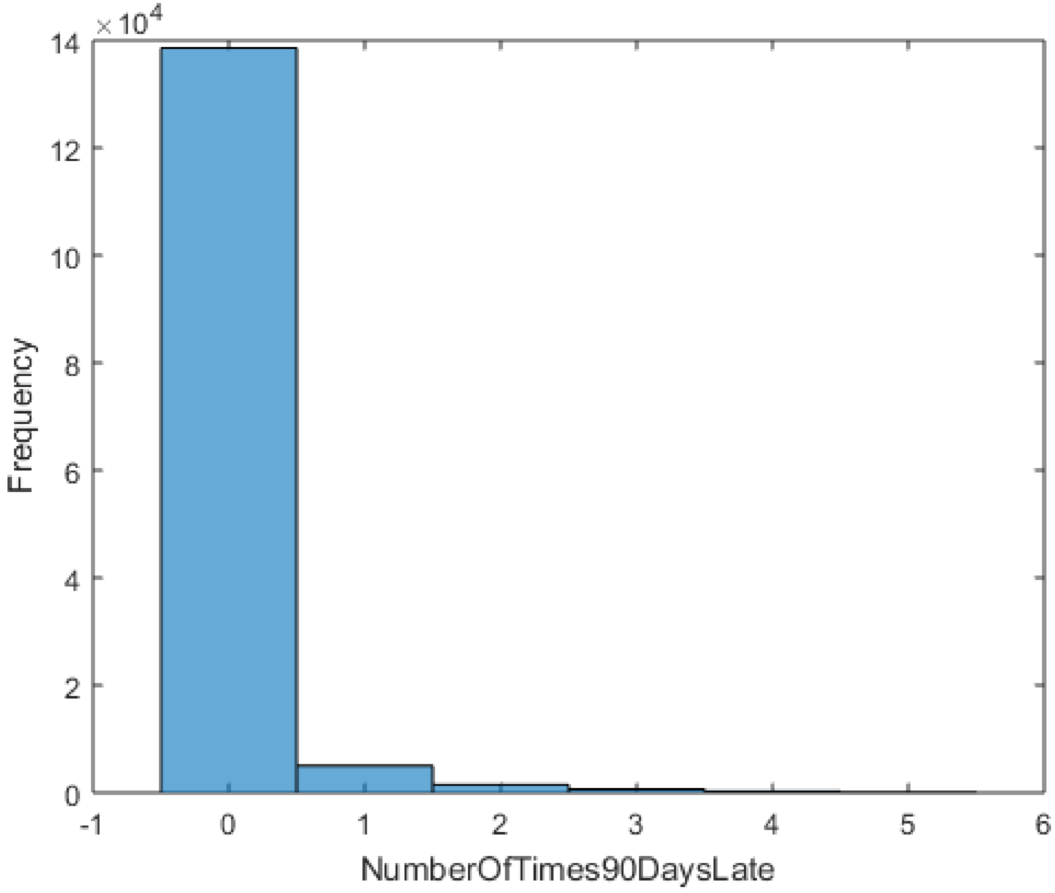
Before Cleaning



Trim 99.7% (3 standard deviations) on top to remove outliners

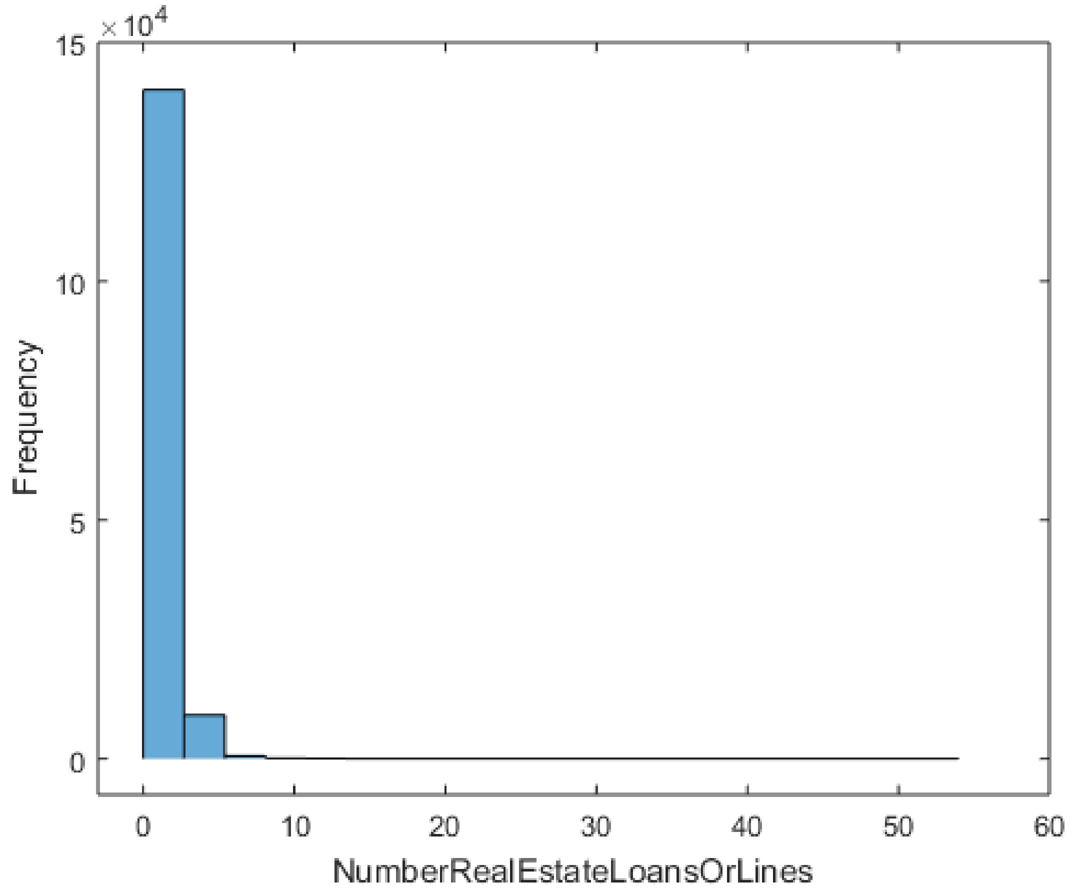
NumberOfTimes90DaysLate

After Cleaning



NumberRealEstateLoansOrLines

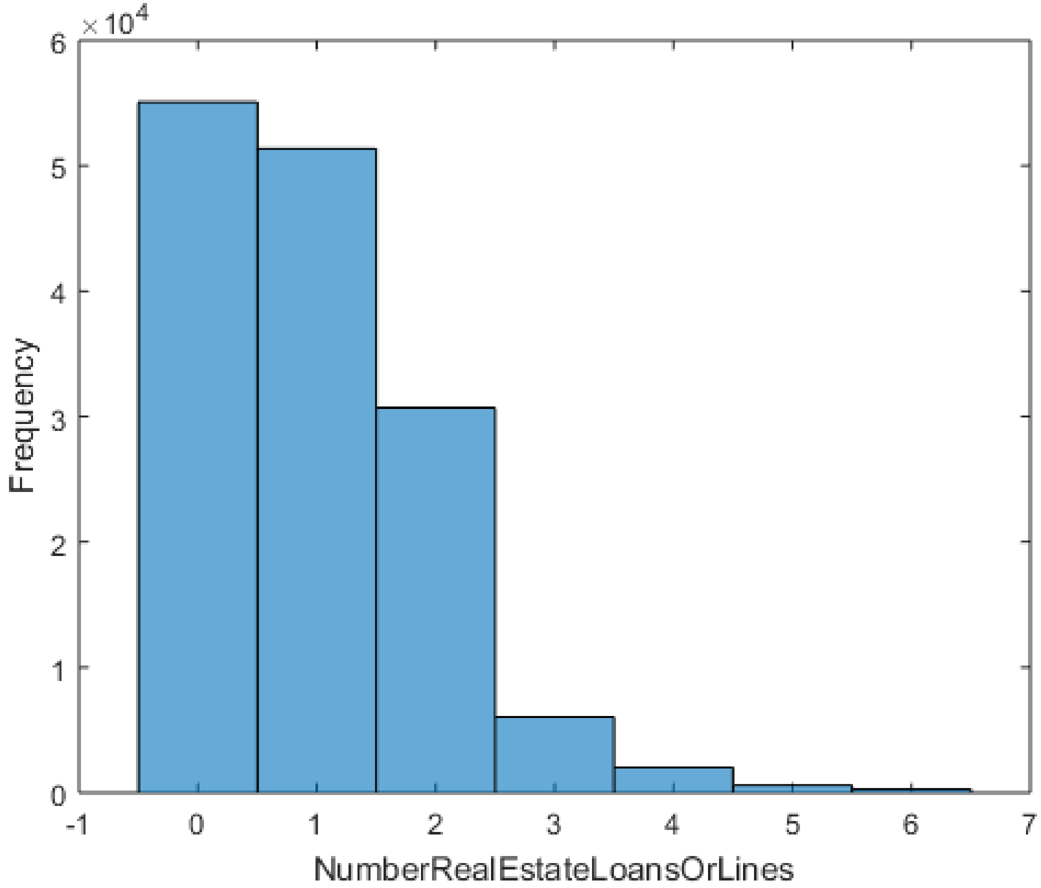
Before Cleaning



Trim 99.7% (3 standard deviations) on top to remove outliners

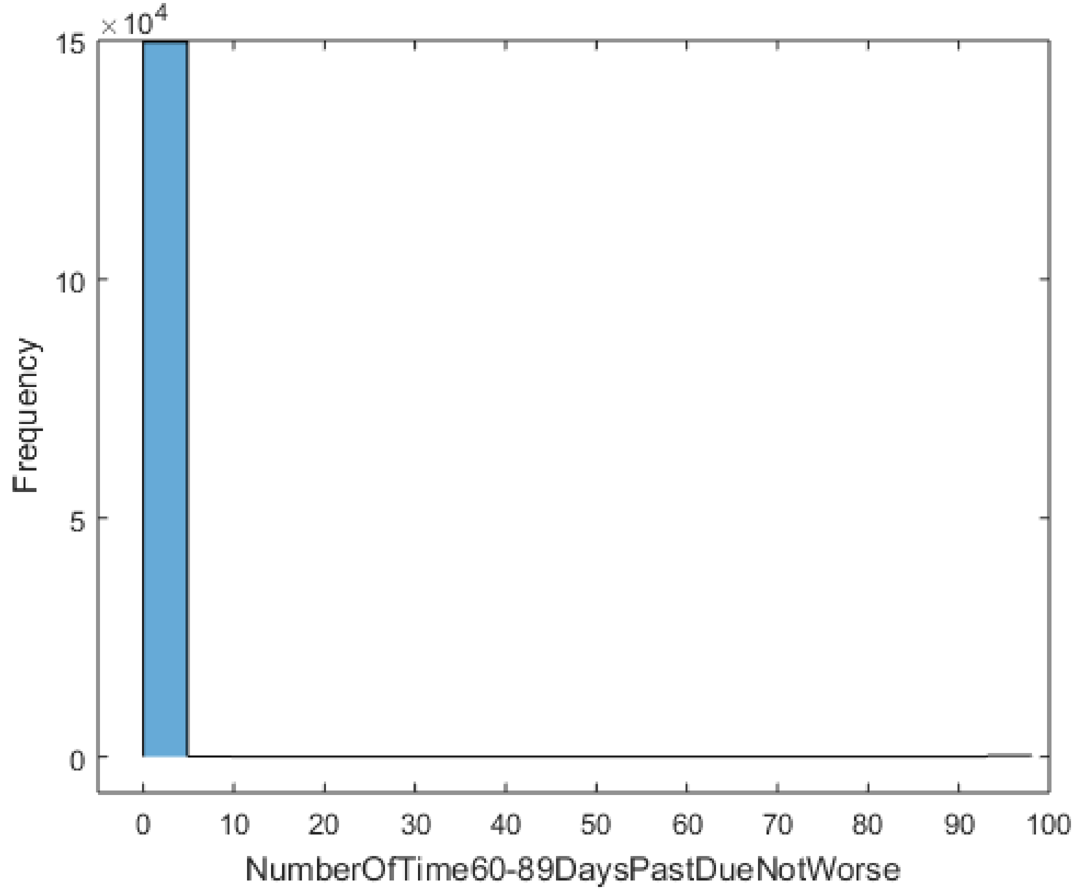
NumberRealEstateLoansOrLines

After Cleaning



NumberOfTime60-89DaysPastDueNotWorse

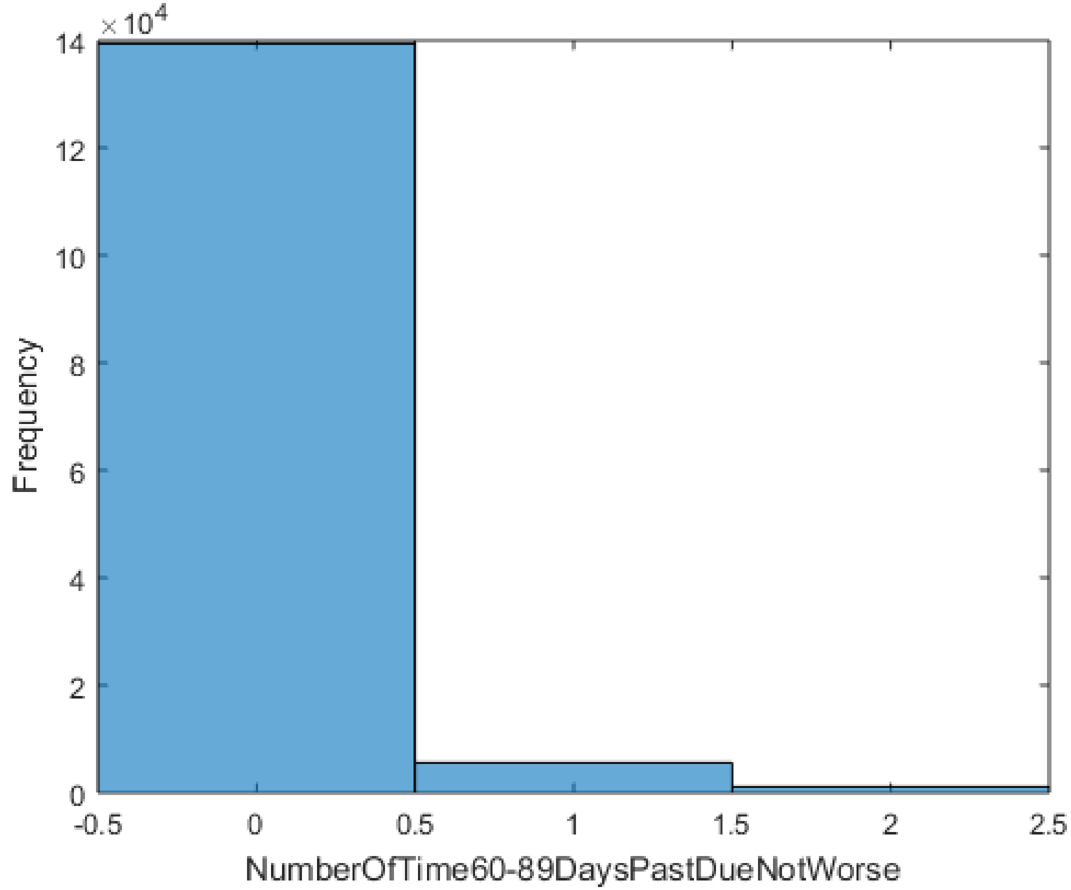
Before Cleaning



Trim 99.7% (3 standard deviations) on top to remove outliners

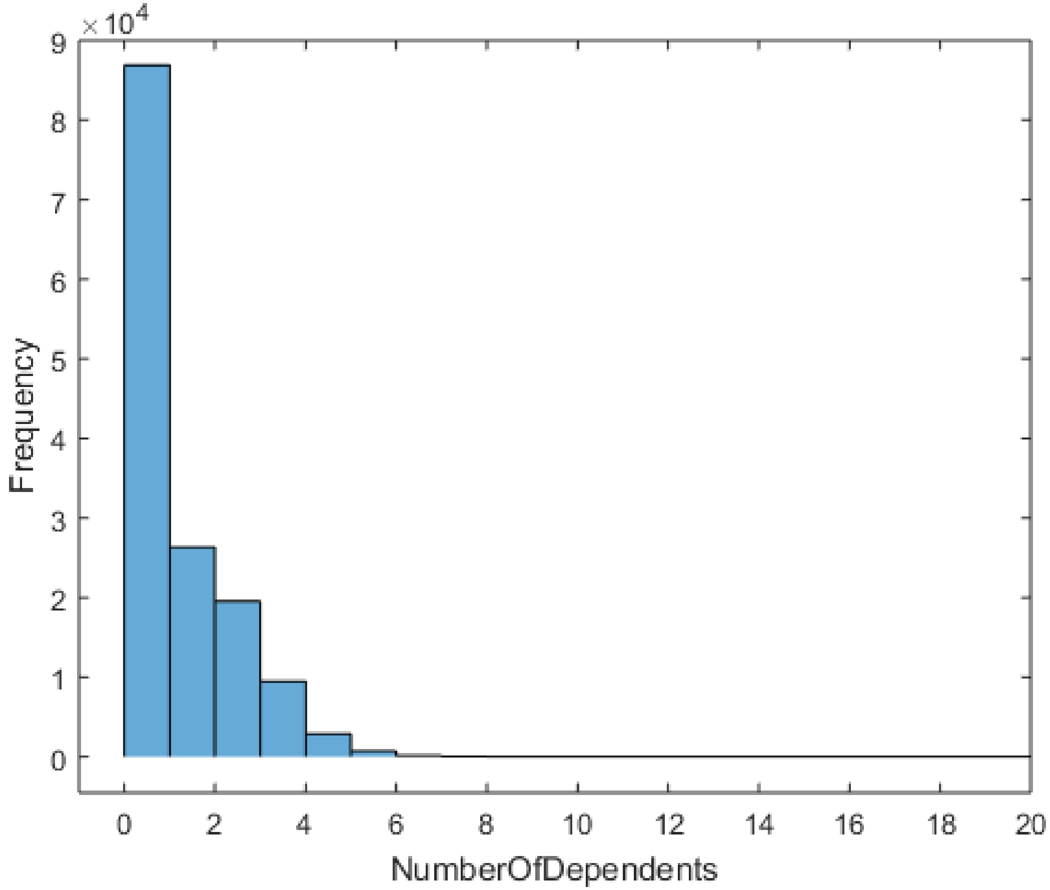
NumberOfTime60-89DaysPastDueNotWorse

After Cleaning



NumberOfDependents

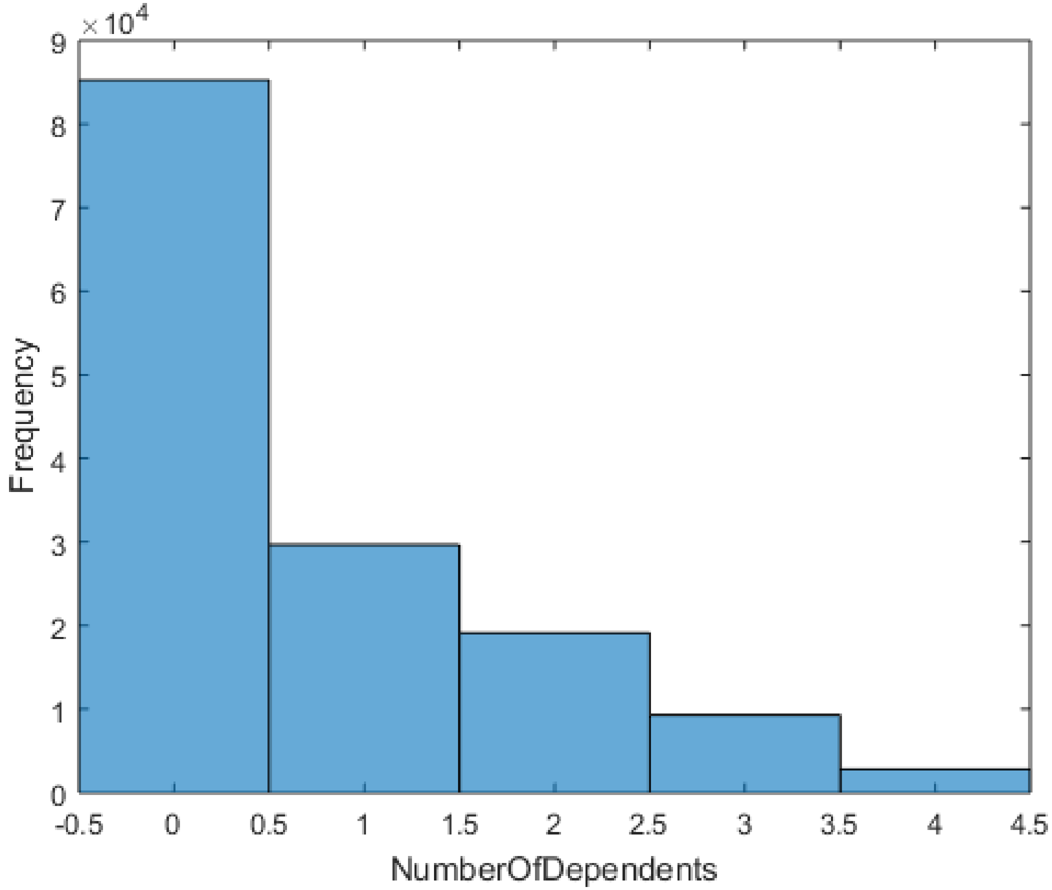
Before Cleaning



Trim 99.7% (3 standard deviations) on top to remove outliners
This also has some NA values which need to be replace by mean values of remaining samples.

NumberOfDependents

After Cleaning



After Cleaning

Feature	Min	Max	Mean	Std
RevolvingUtilizationOfUnsecuredLines	0	8.852	0.319	0.365
age	21	109	52.364	14.824
NumberOfTime30-59DaysPastDueNotWorse	0	13	0.237	0.678
DebtRatio	0	7299	310.837	908.995
MonthlyIncome	0	39999	6263.841	3864.021
NumberOfOpenCreditLinesAndLoans	0	28	8.333	4.868
NumberOfTimes90DaysLate	0	5	0.078	0.394
NumberRealEstateLoansOrLines	0	6	0.983	0.994
NumberOfTime60-89DaysPastDueNotWorse	0	2	0.053	0.254
NumberOfDependents	0	4	0.724	1.032

Value	**Count**	**Percent**
0	136859	93.7%
1	10026	6.3%

New Results

Logistic Regression

Before Cleaning

lambda	0.01	1	100
Training AUC	0.698524	0.698559	0.699888
Testing AUC	0.696193	0.696241	0.698956

After Cleaning

lambda	0.01	1	100
Training AUC	0.849737	0.849739	0.849889
Testing AUC	0.853484	0.853485	0.853622

Much better improvement than without data cleaning

Neural Network

Before Cleaning

lambda	0.01	1	100
Training AUC	0.832331	0.831945	0.811223
Testing AUC	0.835876	0.835416	0.813816

After Cleaning

lambda	0.01	1	100
Training AUC	0.859228	0.858895	0.855317
Testing AUC	0.862445	0.862503	0.860135

Slight improvement after data cleaning

Isolation Forest

Before Cleaning

#subsample	128	256	512	1024
Training AUC	0.809209	0.802698	0.791381	0.786107
Testing AUC	0.812078	0.805104	0.793656	0.788564

After Cleaning

#subsample	128	256	512	1024
Training AUC	0.802326	0.799242	0.796858	0.79121
Testing AUC	0.811797	0.807685	0.804366	0.79898

Very little improvement after data cleaning

RUSBoost

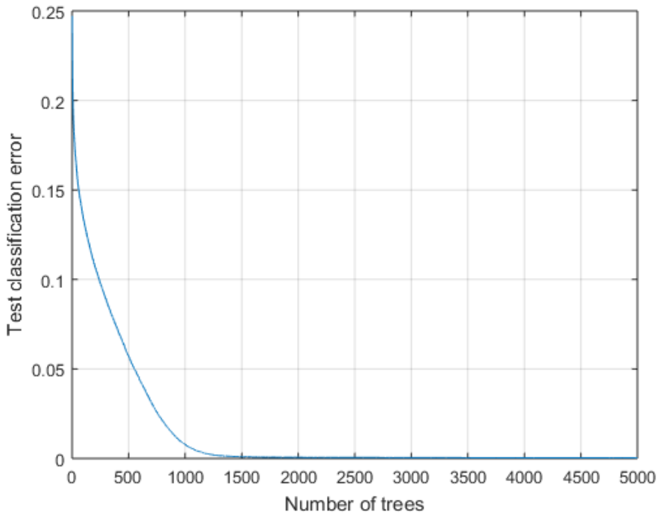
Before Cleaning

minleaf_#ensemble	**2_5000**	**50_5000**	**100_5000**	**300_5000**	**5000_5000**
Training AUC	0.999889	0.895696	0.879818	0.868595	0.822152
Testing AUC	0.827538	0.867186	**0.867329**	0.865336	0.823692

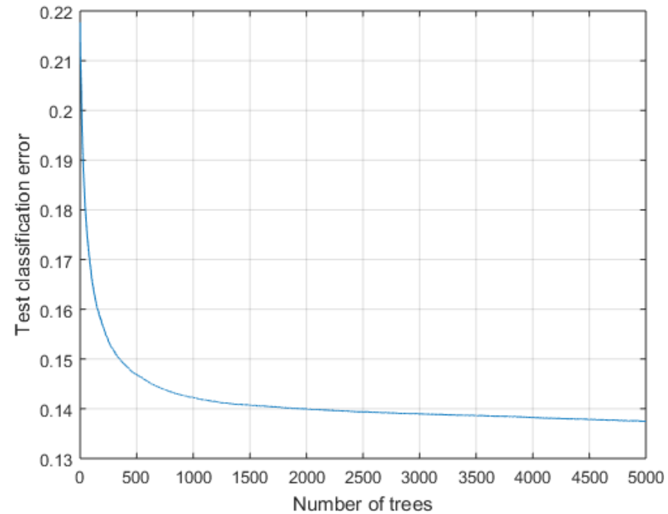
After Cleaning

minleaf_#ensemble	**2_5000**	**50_5000**	**100_5000**	**300_5000**	**5000_5000**
Training AUC	0.99992	0.892047	0.875529	0.864151	0.818085
Testing AUC	0.820189	0.866736	**0.867058**	0.865167	0.823405

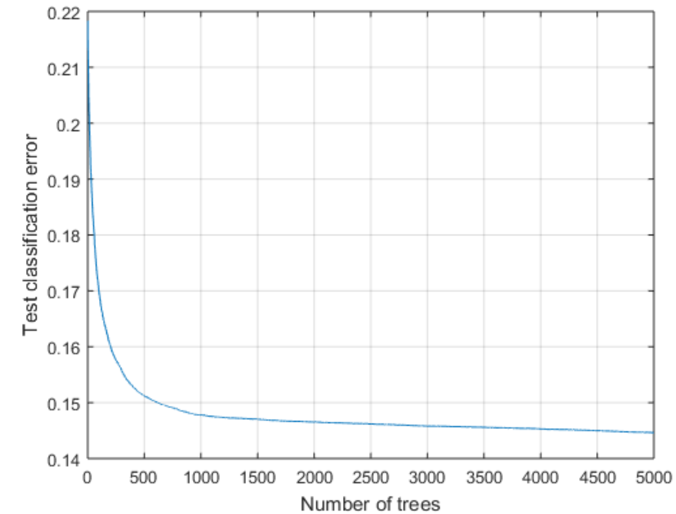
No improvement after data cleaning



Min leaf size=2
#Ensembles = 5000

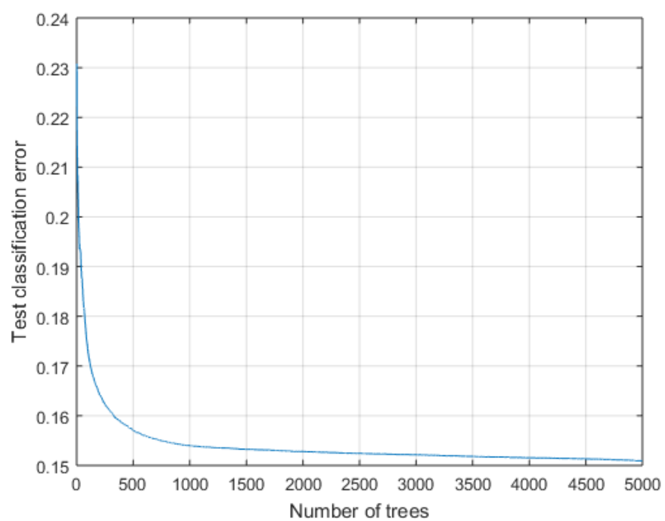


Min leaf size=50
#Ensembles = 5000

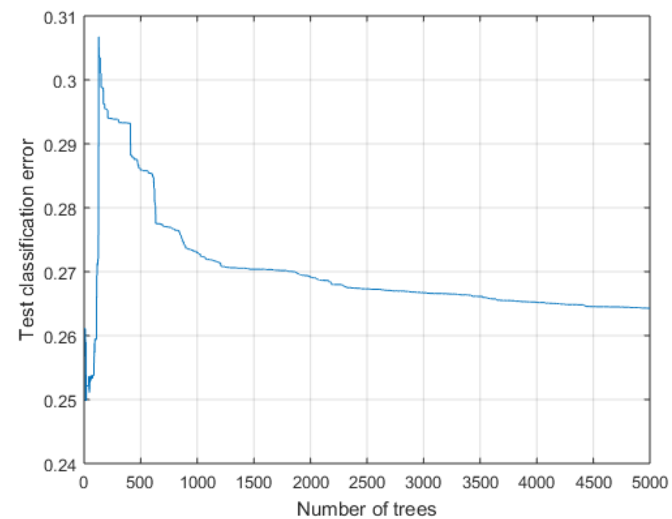


Min leaf size=100
#Ensembles = 5000

Loss function of RUSBoost



Min leaf size=300
#Ensembles = 5000



Min leaf size=5000
#Ensembles = 5000

Conclusions

With RUSBoost, min leaf size=100, it gives the best result. From the graph of its loss function, the error can be futher reduced with increasing number of ensembles.

Due to limitation of 8GB memory, the maxium number of ensembles I can run is 50000, giving the test AUC = 0.867693 (rank 70)

Data cleaning almost has very little effect on RUSBoost but does help to improve performance of other methods.