CAPSTONE PROJECT REPORT

XYZCORP Loan Default Prediction

Submitted By:

RAHUL S K
SIGAPRIYA SUBRAMANIAM
SARAVANA PRASHANTH R
ROSHINI RAMAKUMAR
SRI KRISHNA KANTH V
RESHMA S
ASHWINKUMAR J
ANUSHIKHA AGARWAL
VIKESH B S
BALAJI SANKAR A

Course and Batch: PGA09 APR - SEP 2019

Abstract

Keywords

*Disclaimer: *Data shared by the customer is confidential and sensitive, it should not be used for any purposes apart from capstone project submission for PGA. The Name and demographic details of the enterprise is kept confidential as per their owners' request and binding.*

Acknowledgements

We are using this opportunity to express our gratitude to everyone who supported us throughout the course of this group project. We are thankful for their aspiring guidance, invaluably constructive criticism and friendly advice during the project work. We are sincerely grateful to them for sharing their truthful and illuminating views on a number of issues related to the project.

We wish to thank, all the faculties, as this project utilized knowledge gained from every course that formed the PGA program.

We certify that the work done by us for conceptualizing and completing this project is original and authentic.

Date: Oct 16th, 2019

Place: Chennai

TABLE OF CONTENTS

INTRODUCTION

TITLE & OBJECTIVE OF THE STUDY

The objective of our project is to predict whether a loan will default or not based on client financial data thereby allowing investors to decide whether to approve loan to a customer Data from 2007-2015 is available to us.

NEED OF THE STUDY

In today's world, obtaining loans from financial institutions has become a very common phenomenon. Every day many people apply for loans, for a variety of purposes. But not all the applicants are reliable, and not everyone can be approved. Every year, there are cases where people do not repay the bulk of the loan amount to the bank which results in huge financial loss. The risk associated with making a decision on a loan approval is immense. Hence, the idea of this project is to gather loan data from the Lending Club website and use machine learning techniques on this data to extract important information and predict if a customer would be able to repay the loan or not. In other words, the goal is to predict if the customer would be a defaulter or not.

DATA SOURCES

The provided dataset corresponds to all loans issued to individuals in the past from 2007-2015. The dataset has 855969 observations and 73 features. The data contains the indicator of default, payment information, credit history, etc. Customers under 'current' status have been considered as non-defaulters in the dataset. We have also been provided with a Data dictionary that best describes the features.

The dataset has quite a lot of missing values and the figures can be considered as ground truth, but lots of columns are either irrelevant, very sparse or non informative. Moreover, the dataset is unbalanced, with approximately 6% of loans considered as defaulted.

TOOLS & TECHNIQUES

Tools: Python 3.7.2, Jupyter Notebook, Numpy, Pandas, Matplotlib, Seaborn, Scikit-learn

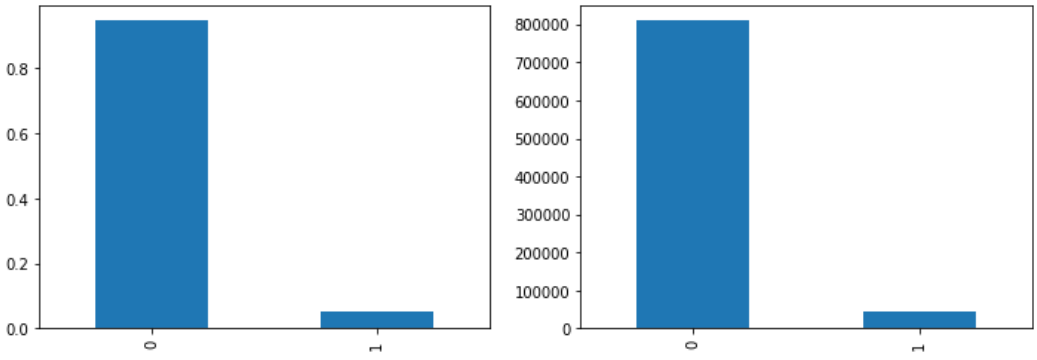
Techniques: Logistic regression

DATA PREPARATION AND MODEL BUILDING

Our first step was to outline the sequence that we will be following for our project. We had to understand the problem statement and then explore every piece of information we could get from the dataset. Each of these steps are elaborated.

UNDERSTANDING THE DATA

- We loaded the data given into the environment and tried to understand the basic appearance of the data. Our dataset has 855969 rows × 73 features including the target variable.
- The data had 21 objects, 49 float type variables and 3 integer type variables.
- Data was summarized to find basic statistical metrics.
- Data was segregated into numerical and categorical objects.



In our dataset our target shows that 94.5% have not defaulted and 5.5% are defaulters. Thus, we are dealing with an unbalanced dataset.

We measured the number of unique levels in each column to understand the data better. We found the percentage of each levels in each of the numerical variables. We found that "policy code" variable had just one level. It doesn't serve the model and so we had to remove the column.

	columns	unique
38	open_il_24m	16
47	inq_fi	16
42	open_rv_12m	15
37	open_il_12m	12
35	open_acc_6m	11
10	inq_last_6mths	9
32	acc_now_delinq	8
29	policy_code	1

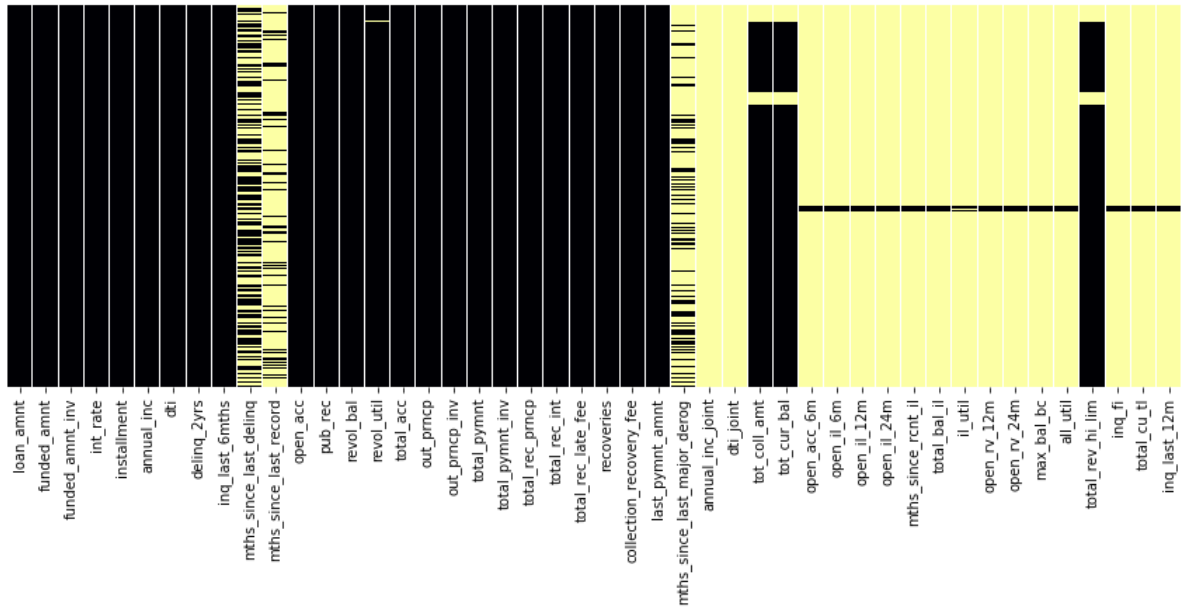
Similarly, we also removed the columns like “Id” and “Member Id” which had 855969 unique levels.

We did the same process on categorical variables and found that some variables had too many unique levels. So we had to remove them too.

	columns	unique
3	emp_title	290912
9	desc	120335
11	title	60991
12	zip_code	931
14	earliest_cr_line	697
7	issue_d	103
18	last_credit_pull_d	102
16	last_pymnt_d	97

MISSING VALUES

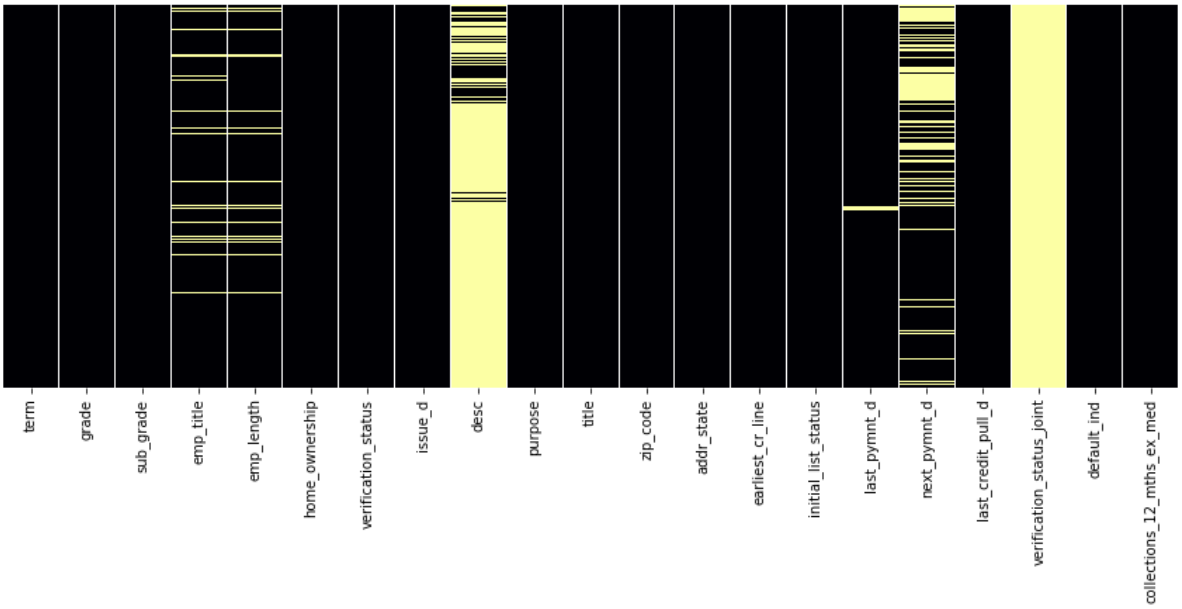
We understood from the data that there were plenty of columns with missing entries. We visualized it to get a better understanding of all the columns with missing values.



The following columns had more than 50% of missing values. We removed the variables.

	missing count	percent
dti_joint	855529	99.948596
annual_inc_joint	855527	99.948363
il_util	844360	98.643759
mths_since_rcnt_il	843035	98.488964
all_util	842681	98.447607
max_bal_bc	842681	98.447607
total_cu_tl	842681	98.447607
total_bal_il	842681	98.447607
open_rv_24m	842681	98.447607
open_rv_12m	842681	98.447607
open_il_6m	842681	98.447607
open_il_24m	842681	98.447607
open_il_12m	842681	98.447607
open_acc_6m	842681	98.447607
inq_last_12m	842681	98.447607
inq_fi	842681	98.447607
mths_since_last_record	724785	84.674211
mths_since_last_major_derog	642830	75.099682
mths_since_last_delinq	439812	51.381767

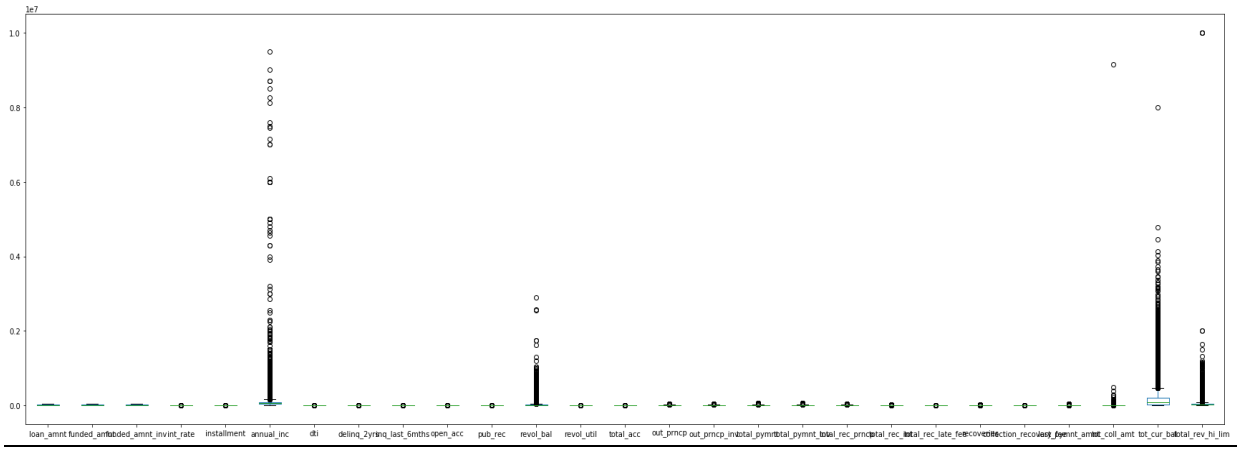
We did the same for categorical variables too.



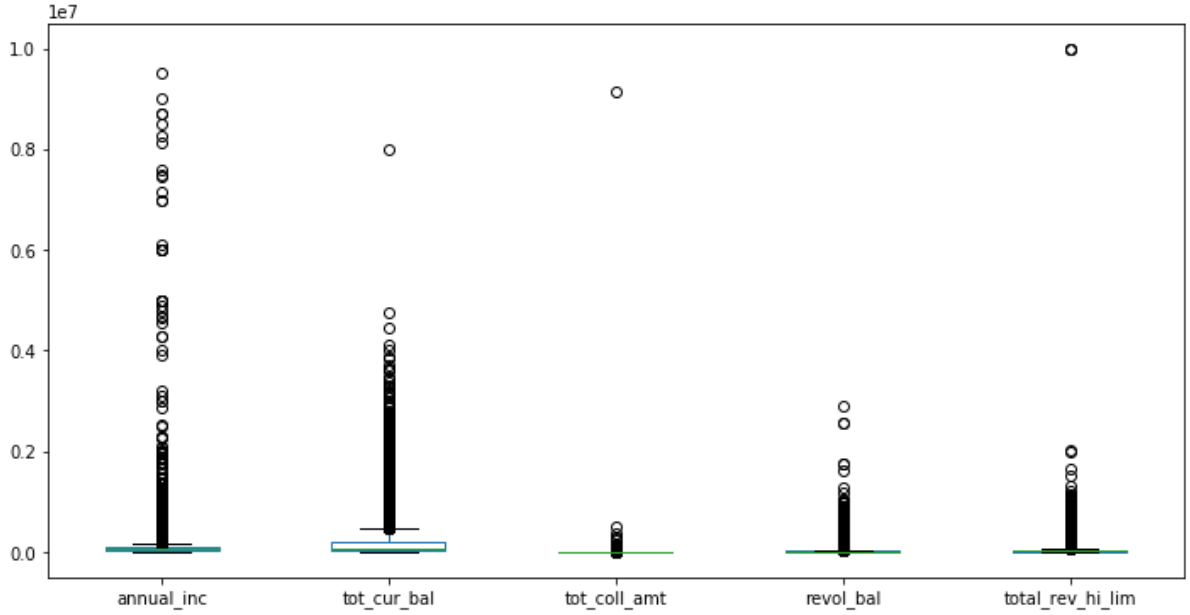
	missing count	percent
verification_status_joint	855527	99.948363
desc	734157	85.769111

After removing the respective columns, we have 27 numerical variables and 19 categorical variables which also includes the target variable.

OUTLIER TREATMENT

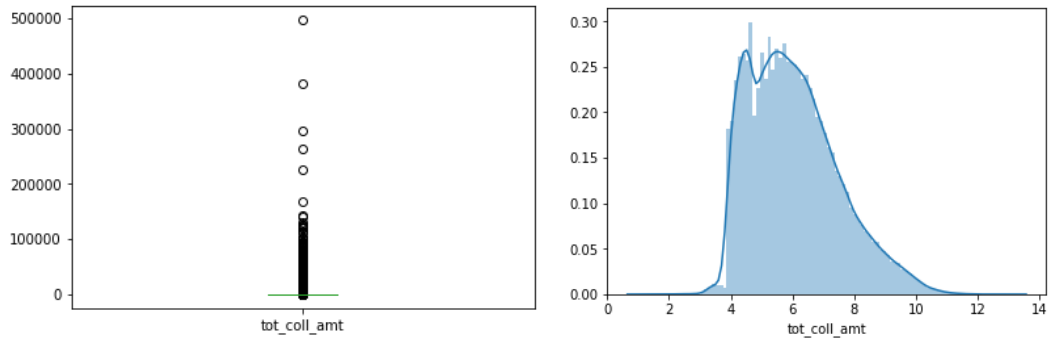


Once missing columns are removed, we checked for the distribution of all numerical variables. Normal distribution is an important assumption while running Logistic Regression and thus, it would be better to normalize the variables as much as possible. Outliers are a road block to any normal distribution and hence had to be treated. We also had to keep in mind that removing all outliers will provide a biased result and thus we had to treat them carefully. We removed only the extreme outliers which were few in numbers thus keeping the distribution intact.

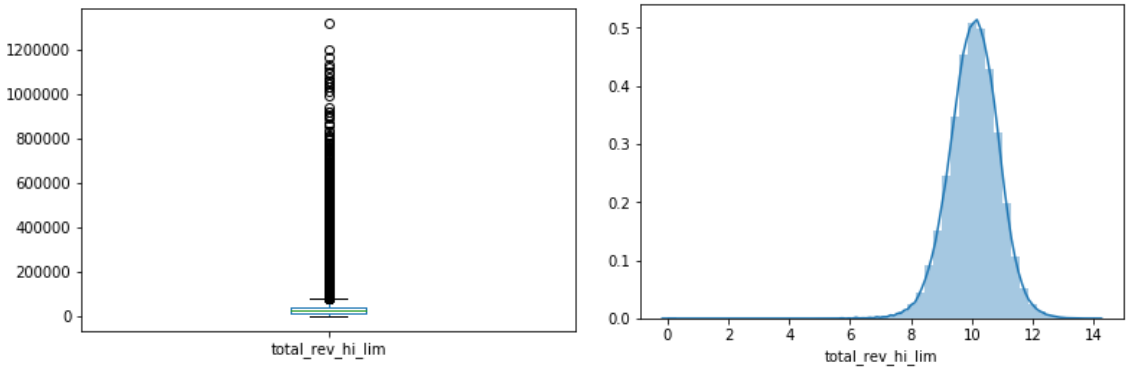


We found the above numerical columns to have a few extreme outliers and we decided to treat them while also not affecting their distribution.
We capped those few entries with the 99th percentile value of those columns.

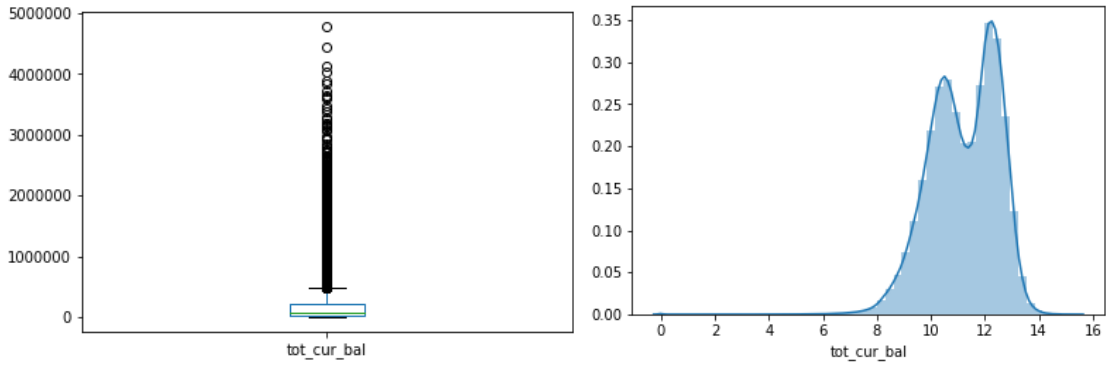
TOT_COLL_AMT:



TOTAL_REV_HI_LIM:



TOT_CUR_BAL:



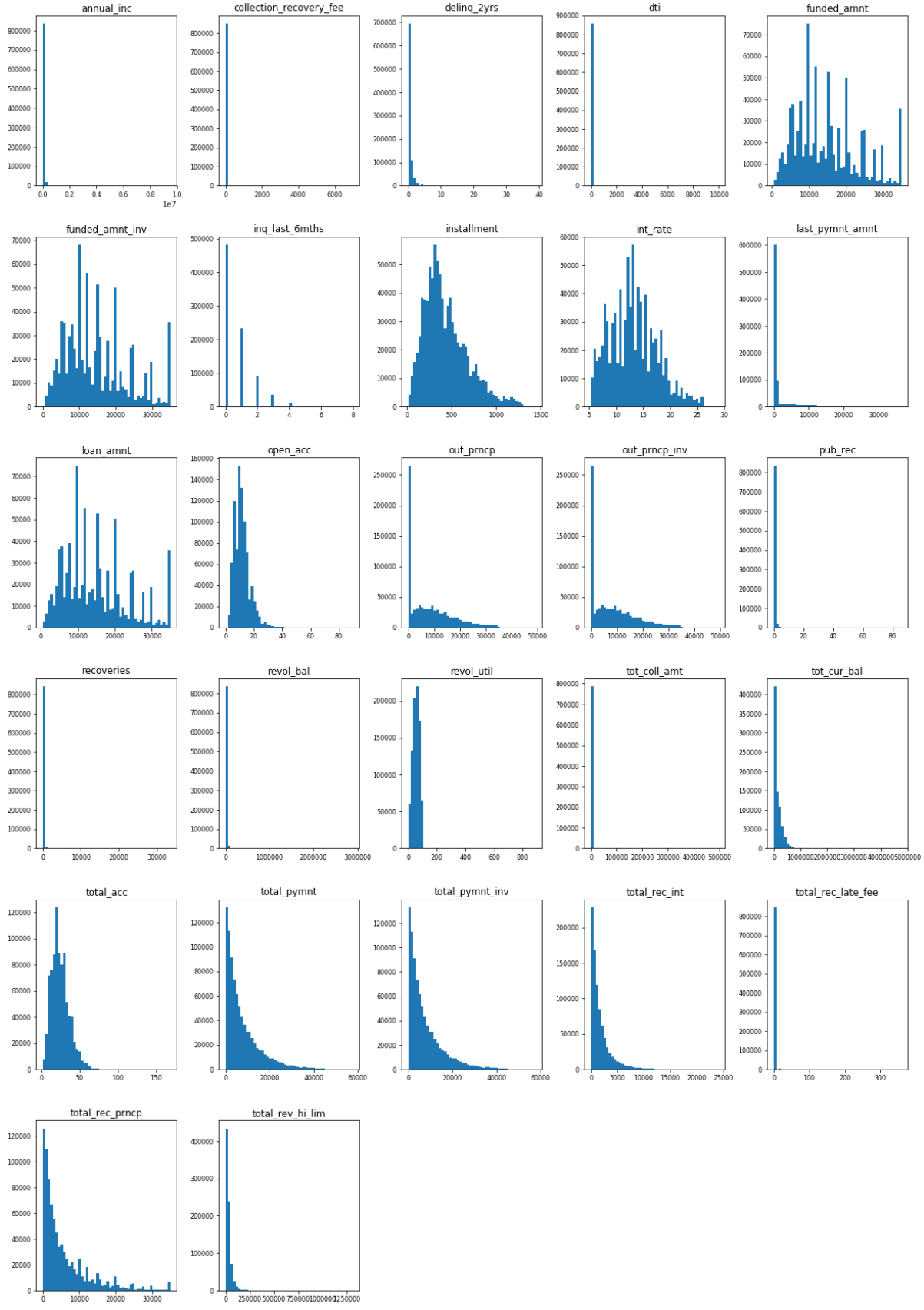
As shown above, we removed the extreme outliers and maintained the distribution. We applied log transformation to get a normal distribution of data.

SKEWNESS

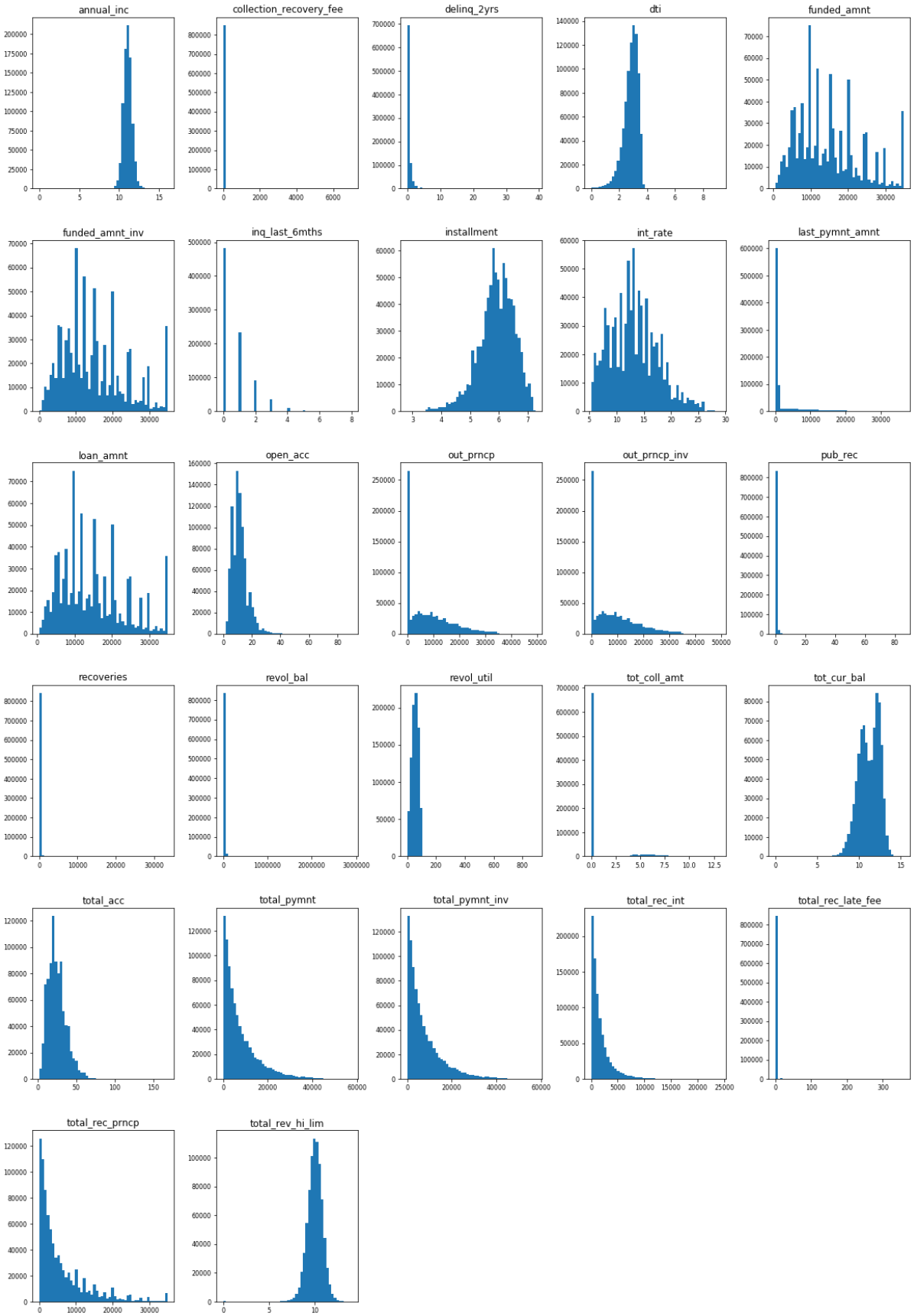
Data skewness affects the symmetry of the distribution. Any distribution with skewness value between -0.5 and 0.5 is considered fairly normal. Distributions with values beyond them are considered skewed. Most of our variables are right skewed.

We measured the skewness of each variables. For the variables with very high skewness values, we applied the log transformation to bring them to a fairly normal distribution.

BEFORE TRANSFORMATION:



AFTER TRANSFORMATION:



MISSING VALUE TREATMENT

The variables we have still have a few percentage of missing values which have to be treated. We can either remove those rows or impute them with mean, median or mode. Removing rows result in a huge loss of data and thus we have to impute them. For numerical variables, we can impute them with either mean or median. So, we measured the difference between the mean and median for each variable. The differences were negligible and so we imputed the mean.

```
total_rev_hi_lim -1.728123702633353e-13
tot_coll_amt 0.06513558155348072
tot_cur_bal -1.4925918509054854e-14
revol_util -0.0009868827696817727
```

For categorical variables, we imputed the missing values with mode.

STANDARDIZATION

Numerical columns were standardized using each column's mean, minimum and maximum values using the following formula.

```
In [49]: # Standardise each numerical variable with mean, max and min values
         data_num = ((data_num - data_num.mean())/(data_num.max() - data_num.min()))
```

Standardization is essential to bring down the range of all columns between -1 and 1.

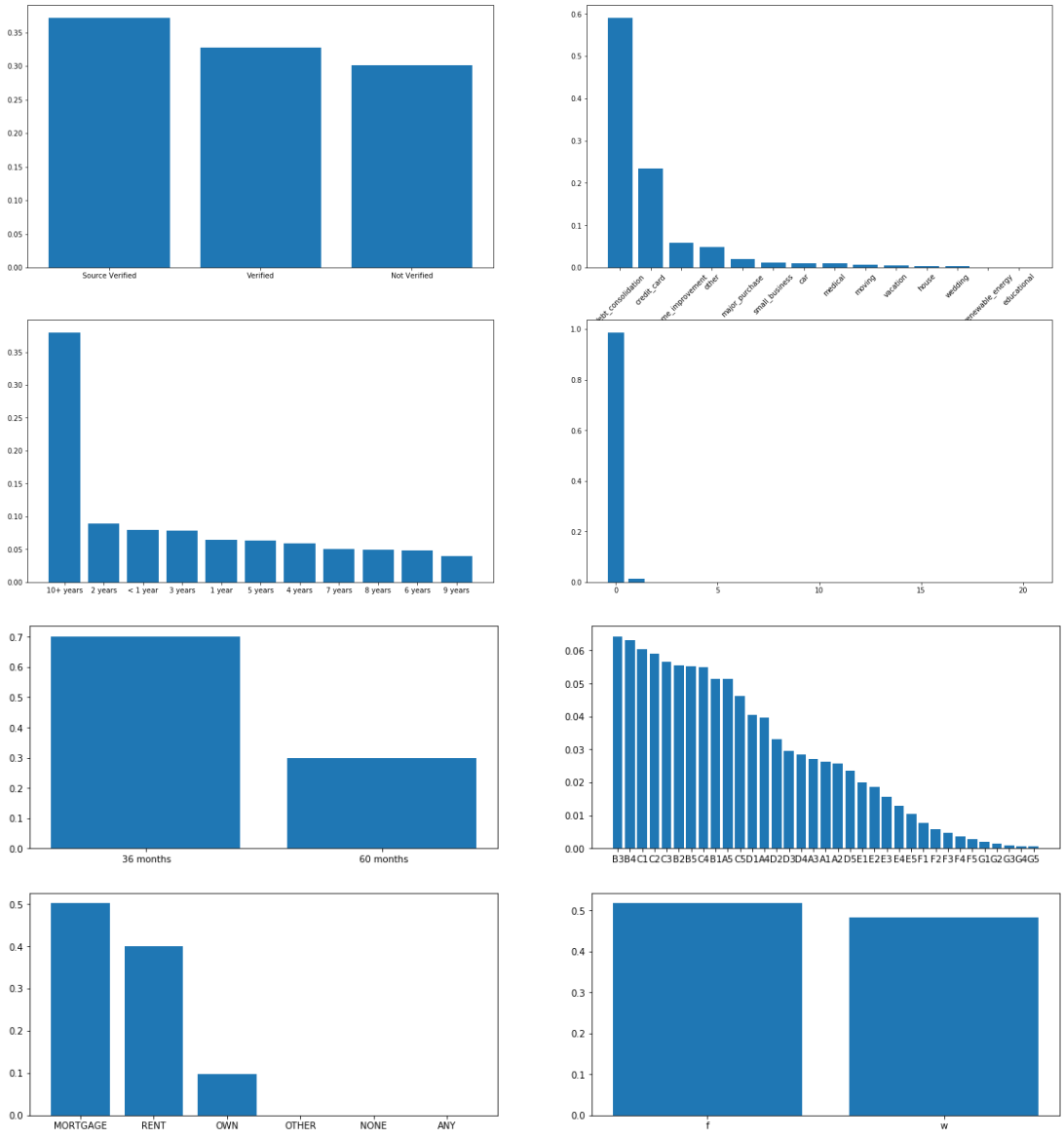
We removed the irrelevant variables from the categorical object. Columns which had too many levels are removed as they do not affect the model.

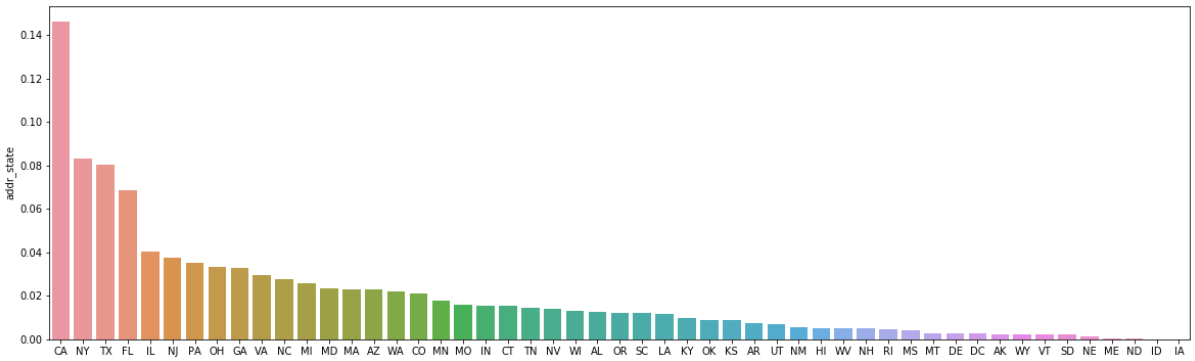
After removing irrelevant columns, our dataset had 27 numerical columns and 11 categorical columns.

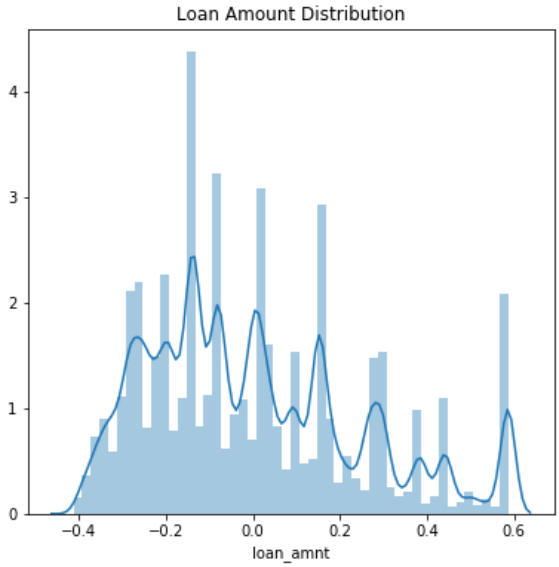
DATA VISUALIZATION

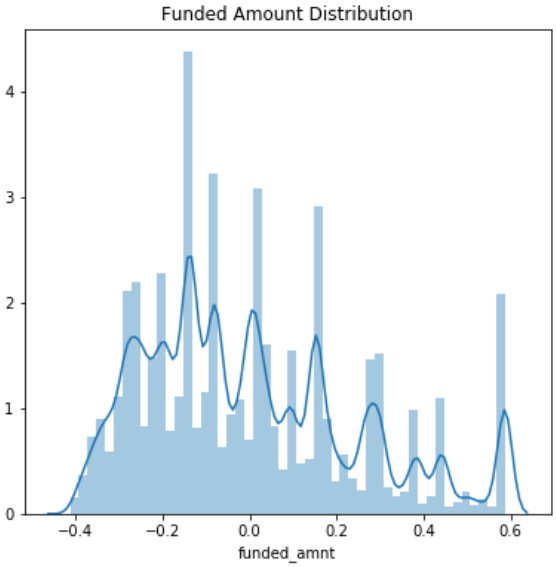
Data visualization can be powerful assets as they bring out important inferences from the data. Most of these inferences stay hidden and unnoticed and visualization helps us to focus on those unattended details in an aesthetic and appealing way.

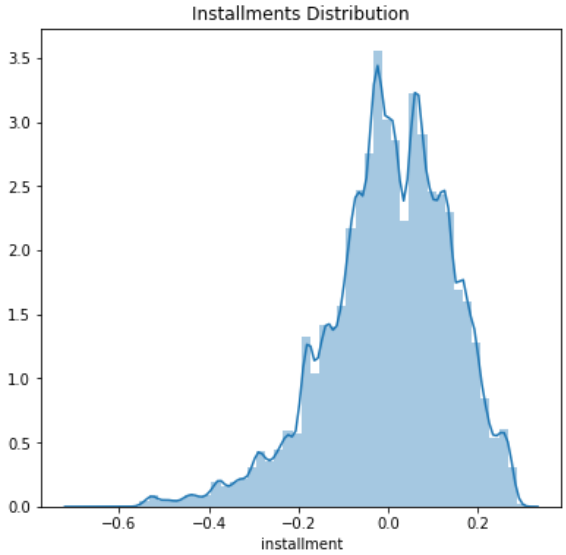
We visualized the categorical variables using bar charts and the numerical variables using histogram.

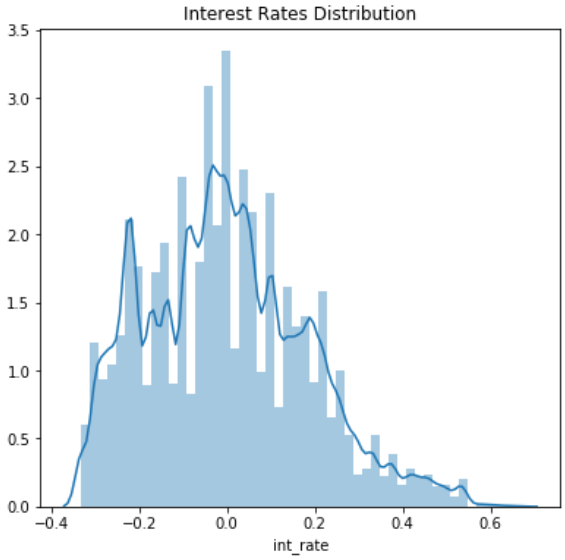


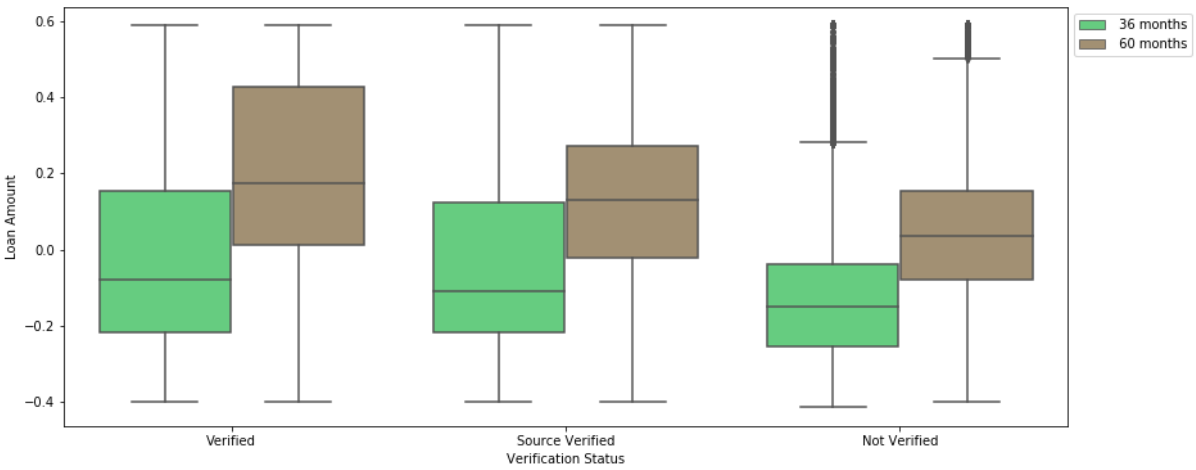

addr_state
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
CA NY TX FL IL NJ PA OH GA VA NC MI MD MA AZ WA CO MN MO IN CT TN NV WI AL OR SC LA KY OK KS AR UT NM HI WV NH RI MS MT DE DC AK WY VT SD NE ME ND ID IA


Loan Amount Distribution
loan_amnt

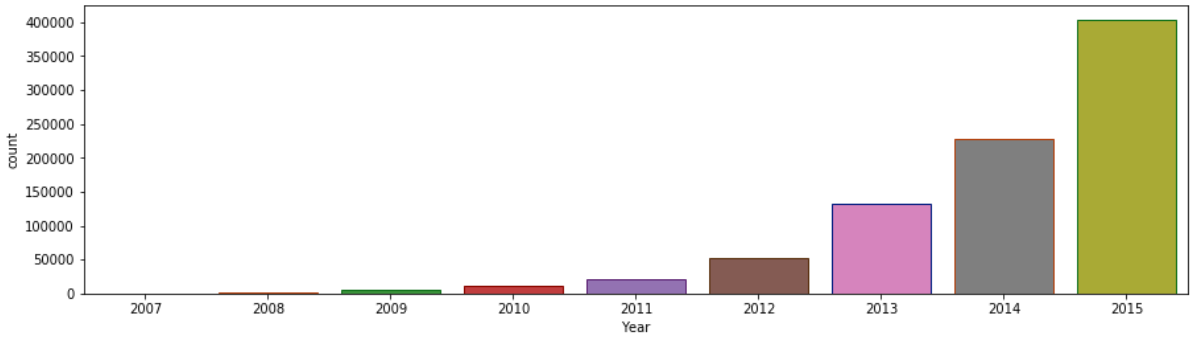

Funded Amount Distribution
funded_amnt

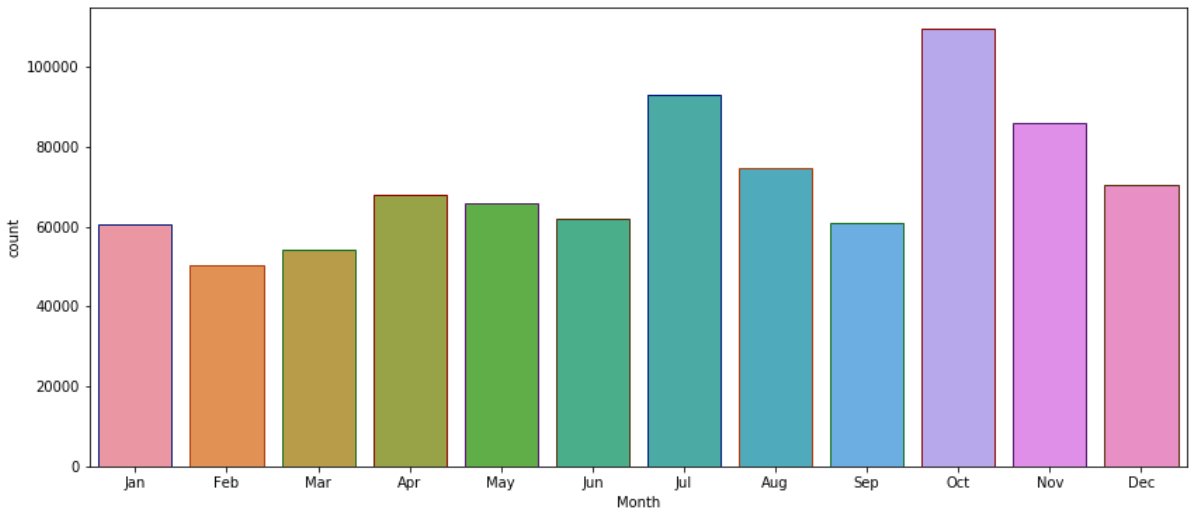

Installments Distribution
installment


Interest Rates Distribution
int_rate



We used the “issue_d” column to extract month and year of loan issued date. Using that, we tried to understand the distribution of loans across years and months to extract patterns. The following graphs show the distribution of loans across months.





ENCODING OF CATEGORICAL VARIABLES

Categorical variables had to be converted to numerical values to run regression model. Ordinal variables like "term" and "Emp_Length" were label encoded.
Each value were given a unique value and thus all categorical levels are converted to numbers.

The rest of the variables are not ordinal and so we used one-hot encoding on them. A new column is created for every level in a column which is either filled with 0's or 1's depending upon the value in the original column.

We created a new object called "Period" from the Month and Year extracted already. This new object helps us in splitting the data.

Once encoding is done, the original columns are dropped.

```
In [72]: #combining all columns into a new dataframe
         new_data=pd.concat([data_num,data_cat,dummies,Period],axis=1)
```

All the individual objects are now merged to form a complete cleaned dataset. The final dataset has 855969 rows and 143 columns.

DATA SPLIT

The problem statement requires us to use data from June 2007 to May 2015 as training data. Data from June 2015 are to be used as test data.

Hence, we combined the year and month to create a new column called Period. Period is now set as the index of the whole dataset. Now the entire dataset is sorted using period thus enabling us to split the data into test and train based on dates.

```
In [77]: train=new_data_sorted.loc['200706':'201505',:]

In [78]: test=new_data_sorted.loc['201506':,:]
```

```
In [81]: train.shape,test.shape
Out[81]: ((598978, 142), (256991, 142))
```

LOGISTIC REGRESSION

Logistic regression, despite its name, is a linear model for classification rather than regression. Logistic regression is also known in the literature as logit regression, maximum-entropy classification (MaxEnt) or the log-linear classifier. In this model, the probabilities describing the possible outcomes of a single trial are modeled using a logistic function.

The data is split into respective objects as shown below to be given as inputs to the regression model.

```
X_train=train.drop('default_ind',axis=1)
Y_train=train['default_ind']
X_test=test.drop('default_ind',axis=1)
Y_test=test['default_ind']
```

The required libraries are imported to run the regression model.

```
# Import required libraries
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, accuracy_score, \
classification_report
```

We ran a basic Logistic regression model by fitting the train data and the running the model on test data to predict the new values.

Our accuracy score was 0.9994

We obtained the following report:

Classification report

	precision	recall	f1-score	support
0	1.00	1.00	1.00	256680
1	0.78	0.79	0.78	311
micro avg	1.00	1.00	1.00	256991
macro avg	0.89	0.90	0.89	256991
weighted avg	1.00	1.00	1.00	256991

CONFUSION MATRIX:

Confusion Matrix

```
array([[256609,     71],
       [    65,    246]], dtype=int64)
```

TUNING THE MODEL

The default threshold value to segregate 0's and 1's is 0.5. We changed the value to 0.6 to see if it affects the grouping. It increased it by a very minimal percentage.

```
Model accuracy :  0.9994785809619792


Classification report

              precision    recall  f1-score   support

           0       1.00      1.00      1.00    256680
           1       0.78      0.78      0.78       311

   micro avg       1.00      1.00      1.00    256991
   macro avg       0.89      0.89      0.89    256991
weighted avg       1.00      1.00      1.00    256991


Confusion Matrix

array([[256613,     67],
       [    67,    244]], dtype=int64)
```

CONCLUSION

We have successfully built a machine learning algorithm to predict the people who might default on their loans.

Also, we might want to look into other techniques such as Decision Trees, Random Forest, Support Vector Machine and Neural Network.

REFERENCES

1) https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html
2) https://scikit-learn.org/stable/modules/model_evaluation.html#accuracy-score
3) https://scikit-learn.org/stable/model_selection.html#model-selection
4) https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier