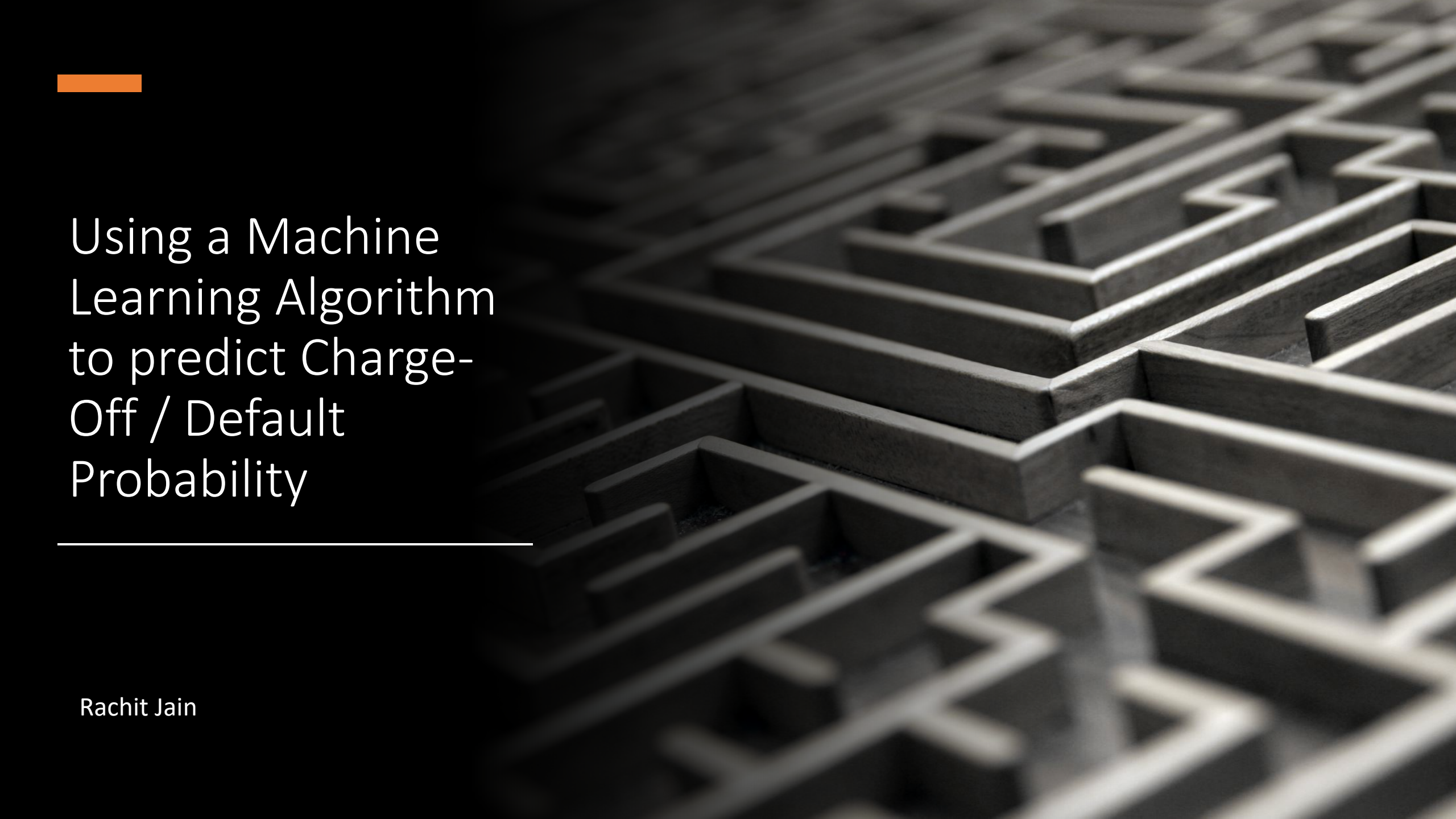
Using a Machine Learning Algorithm to predict Charge-Off / Default Probability

Rachit Jain



Step 1

Sourcing the data

```
# Data Extraction
train = pd.read_csv('lc_loan.csv')
test = pd.read_csv('lc_2016_2017.csv')
```

Problem Statement: We get the loan data for a US peer to peer lending firm called “LendingClub”. The data has been taken from Kaggle.com. The problem we are solving here is to model the probability of a loan to become a Charge-off/default in the future. We achieve that by developing a Machine Learning Algorithm to model to predict the status for a loan. We are basically trying to make a classification algorithm

Step 2

Data Munging

We replace all zero's with numpy.nan

```
loan_details = loan_details.replace(0.0,np.nan)
```

Drop all the columns with more than 20% missing values

```
loan_details = loan_details.dropna(axis=1,thresh = 0.8*len(loan_details))
```

We delete all the unimportant variables

```
to_be_del = ['policy_code','title','zip_code','addr_state','url','issue_d','pymnt_plan']
loan_details = loan_details.drop(columns = to_be_del)
```

Imputed the null values by mean for debt to income ratio & one for other columns with applicant-specific information. We fill with one to make the log transformation convenient in later stages.

```
# Imputing the missing values
cols = ['funded_amnt_inv','earliest_cr_line','open_acc','revol_bal','revol_util','total_acc','t
train.update(train[cols].fillna(1))
train['dti'].fillna(train['dti'].mean(), inplace=True)
train.isnull().sum()
```

Step 3

Data Exploration

We analyze the loan status value counts.

```
Current                                                 601779
Fully Paid                                              207723
Charged Off                                              45248
Late (31-120 days)                                       11591
Issued                                                    8460
In Grace Period                                           6253
Late (16-30 days)                                         2357
Does not meet the credit policy. Status:Fully Paid        1988
Default                                                   1219
Does not meet the credit policy. Status:Charged Off        761
Name: loan_status, dtype: int64
```

We look at the statistical properties of all numerical variables

	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	int_rate	installment	annual_in
count	8.873790e+05	8.873790e+05	887379.000000	887379.000000	887146.000000	887379.000000	887379.000000	8.873730e+0
mean	3.246513e+07	3.500182e+07	14755.264605	14741.877625	14706.325838	13.246740	436.717127	7.502776e+0
std	2.282734e+07	2.411335e+07	8435.455601	8429.897657	8439.851678	4.381867	244.186593	6.469828e+0
min	5.473400e+04	7.047300e+04	500.000000	500.000000	0.000121	5.320000	15.670000	1.200000e+0
25%	9.206643e+06	1.087713e+07	8000.000000	8000.000000	8000.000000	9.990000	260.705000	4.500000e+0
50%	3.443327e+07	3.709528e+07	13000.000000	13000.000000	13000.000000	12.990000	382.550000	6.500000e+0
75%	5.490814e+07	5.847135e+07	20000.000000	20000.000000	20000.000000	16.200000	572.600000	9.000000e+0
max	6.861706e+07	7.354484e+07	35000.000000	35000.000000	35000.000000	28.990000	1445.460000	9.500000e+0

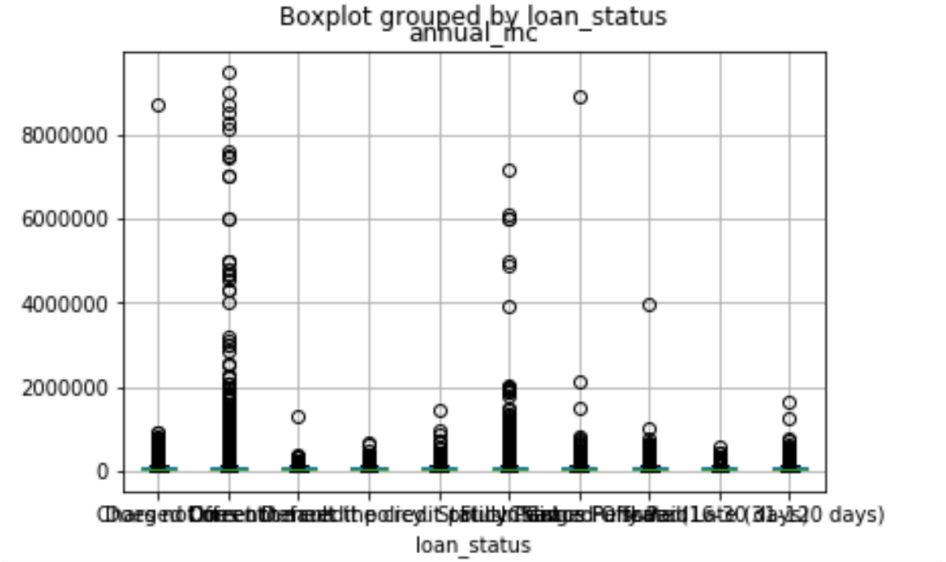
We look at the crosstab between Loan Status & Interest rates to judge relationship between both

loan_status	Charged Off	Current	Default	Does not meet the credit policy. Status:Charged Off	Does not meet the credit policy. Status:Fully Paid	Fully Paid	In Grace Period	Issued	Late (16-30 days)	Late (31-120 days)	All
int_rate											
5.32	0	9143	0	0	0	165	12	322	6	3	9651
5.42	20	0	0	0	0	553	0	0	0	0	573
5.79	16	0	0	0	0	394	0	0	0	0	410
5.93	1	1684	0	0	0	125	0	0	0	2	1812
5.99	19	0	0	0	0	328	0	0	0	0	347
...	...	...	...	...	...	...	...	...	...	...	...
27.88	0	193	1	0	0	14	3	2	0	9	222
27.99	0	3	0	0	0	0	0	2	0	0	5
28.49	1	113	0	0	0	11	4	3	2	5	139
28.99	2	83	0	0	0	6	4	3	3	11	112
All	45248	601779	1219	761	1988	207723	6253	8460	2357	11591	887379

Int Rates	Fully paid	Charged-Off
5-10	27%	10%
11-15	44%	37%
16-20	23%	38%
21-25	5%	14%
26-30	1%	2%

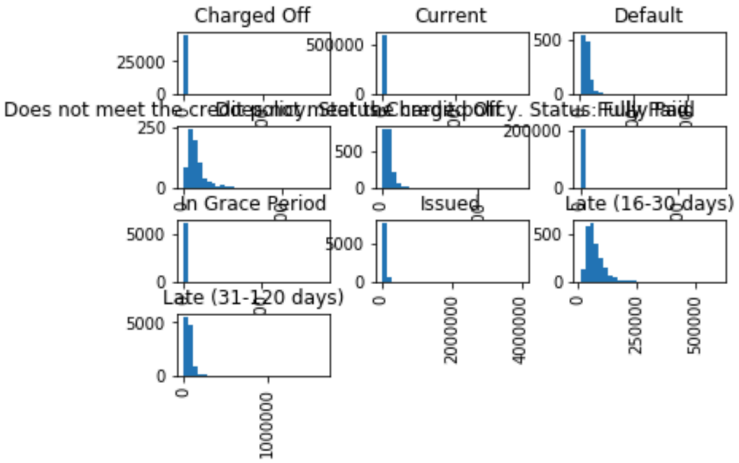
The information we get is that a higher number of charge-off loans were at higher interest rates than Fully-paid loans

Boxplot showing annual income of applicants grouped by loan status



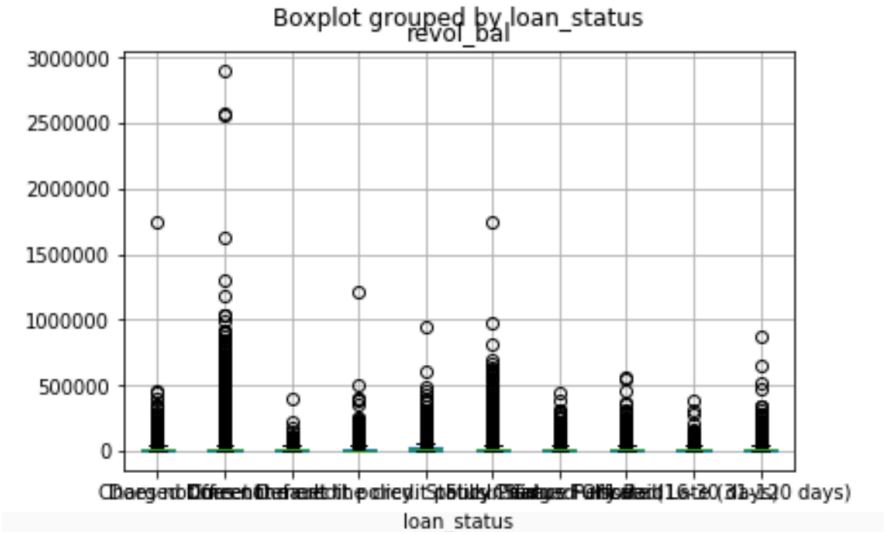
We can visualize and see the very low number of charged-off & default accounts were at higher than $2M annual income

We look at the distribution of annual income for each loan status category



The distribution of annual income for charged-off loans is concentrated at low income groups

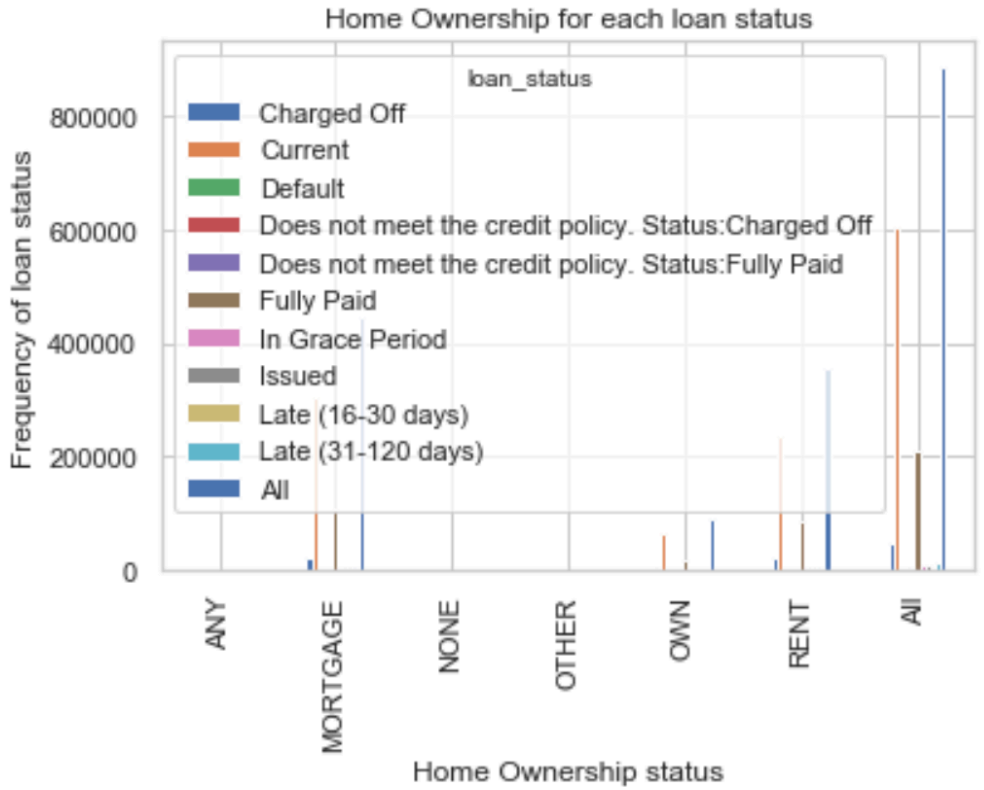
We also relate revolving balance with loan status categories



We can visualize and see that charged-off & default accounts had lowest revolving balances

We look at the crosstab between Loan Status & Home ownership to judge relationship between both

We clearly can articulate that more charged-off applicants were on rental homes or mortgage than Owned houses



loan_status	Charged Off	Current	Default	Does not meet the credit policy. Status:Charged Off	Does not meet the credit policy. Status:Fully Paid	Fully Paid	In Grace Period	Issued	Late (16-30 days)	Late (31-120 days)	All
home_ownership											
ANY	0	2	0	0	0	1	0	0	0	0	3
MORTGAGE	19878	303764	498	348	908	104966	2855	4220	1101	5019	443557
NONE	7	2	0	1	4	36	0	0	0	0	50
OTHER	27	3	0	11	27	114	0	0	0	0	182
OWN	4025	62041	110	49	138	17960	637	1038	260	1212	87470
RENT	21311	235967	611	352	911	84646	2761	3202	996	5360	356117
All	45248	601779	1219	761	1988	207723	6253	8460	2357	11591	887379

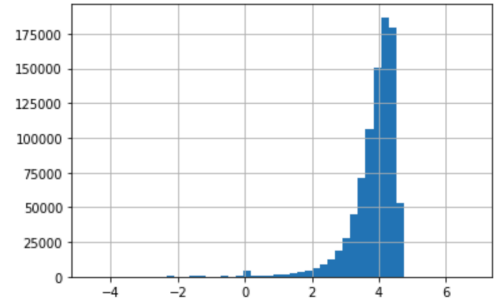
Step 4

Transforming the variables to better suit modelling

- **We choose to transform all the relevant numerical variables with log transformation (to remove effect of outliers). To make the selection of such variables, we refer the relationships that we analyzed in previous steps. We can see that log transformation are more normal and suitable for modelling.**

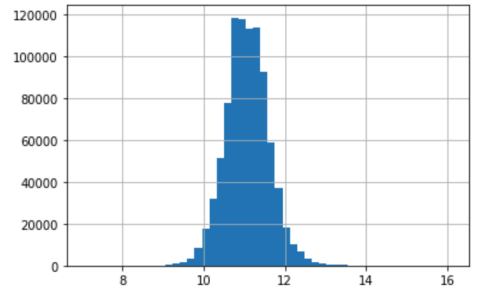
```
train['revol_util_log']= np.log(train['revol_util'])
print (train['revol_util_log'].hist(bins = 50))
```

```
AxesSubplot(0.125,0.125;0.775x0.755)
```



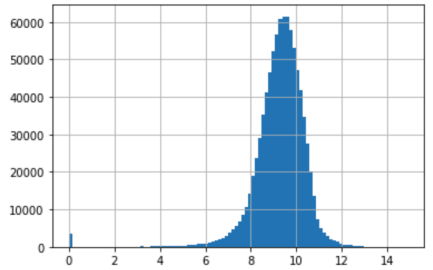

```
train['annual_inc_log']= np.log(train['annual_inc'])
print (train['annual_inc_log'].hist(bins = 50))
```

```
AxesSubplot(0.125,0.125;0.775x0.755)
```



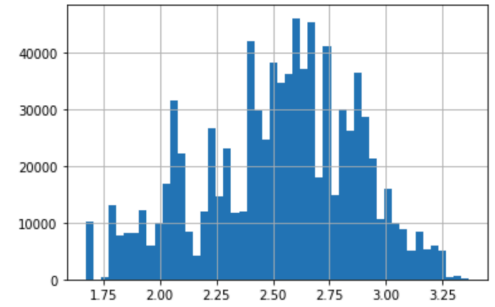

```
train['revol_bal_log']= np.log(train['revol_bal'])
# plt.hist(train[np.isfinite(train['revol_bal'])].values)
print (train['revol_bal_log'].hist(bins = 100))
```

```
AxesSubplot(0.125,0.125;0.775x0.755)
```




```
train['int_rate_log']= np.log(train['int_rate'])
print (train['int_rate_log'].hist(bins = 50))
```

```
AxesSubplot(0.125,0.125;0.775x0.755)
```



Step 5

Model Selection

Logistic Regression is a classification algorithm. It is used to predict a binary outcome (1 / 0, Yes / No, True / False) given a set of independent variables. To represent binary / categorical outcome, we use dummy variables. You can also think of logistic regression as a special case of linear regression when the outcome variable is categorical, where we are using log of odds as the dependent variable.
In simple words, it predicts the probability of occurrence of an event by fitting data to a logit function, read more about Logistic Regression.

```
## Model Building
from sklearn.preprocessing import LabelEncoder
#"Scikit Learn" library has a module called "LabelEncoder" which helps
# to label character labels into numbers so first import module "LabelEncoder".
number = LabelEncoder()
# Transform variable to dummy variables
train['term_new'] = number.fit_transform(train['term'].astype(str))
train['home_ownership_new'] = number.fit_transform(train['home_ownership'].astype(str))
train['grade_new'] = number.fit_transform(train['grade'].astype(str))
train['loan_status_new'] = number.fit_transform(train['loan_status'].astype(str))
```

Transformed the category variables to dummy variables

Step 6

Defining target and Predictors

Target Variable: Loan Status, it defines whether the loan is charged-off, defaulted, on Grace period or current.

Predictors:

1. Interest rate
2. Debt to Income ratio
3. Revolving utility - > The FICO credit score of the borrower days with credit line. The number of days the borrower has had a credit line. Revolving balance. Indicates whether the loan was not paid back in full (the borrower either defaulted or the borrower was deemed unlikely to pay it back).
4. Term
5. Home Ownership

Step 7

Logistic Regression Model Fitting

```
# Import linear model of sklearn
import sklearn.linear_model

# Create object of Logistic Regression
model=sklearn.linear_model.LogisticRegression()

# Select the predictors
predictors =['term_new','home_ownership_new','grade_new','int_rate_log','revol_util_log','dti_l

# Converting predictors and outcome to numpy array
x_train = train[predictors].values
y_train = train['loan_status_new'].values

# Model Building
model.fit(x_train, y_train)
```

Step 8

Model Fit

```
Accuracy of logistic regression classifier on train set: 0.68
```

Step 9

Model Testing

```
# Converting predictors and outcome to numpy array
predictors =['term_new','home_ownership_new','grade_new','int_rate_log','revol_util_log','dti_l
x_test = test[predictors].values
np.isnan(x_test)
np.where(np.isnan(x_test))
x_test = np.nan_to_num(x_test)

#Predict Output
y_pred= model.predict(x_test)

#Reverse encoding for predicted outcome
y_pred = number.inverse_transform(y_pred)

#Store it to test dataset
test['Loan_Status_new']=y_pred

#Output file to make submission
test.to_csv("TestFile_Submission.csv",columns=['id','Loan_Status_new'])
```

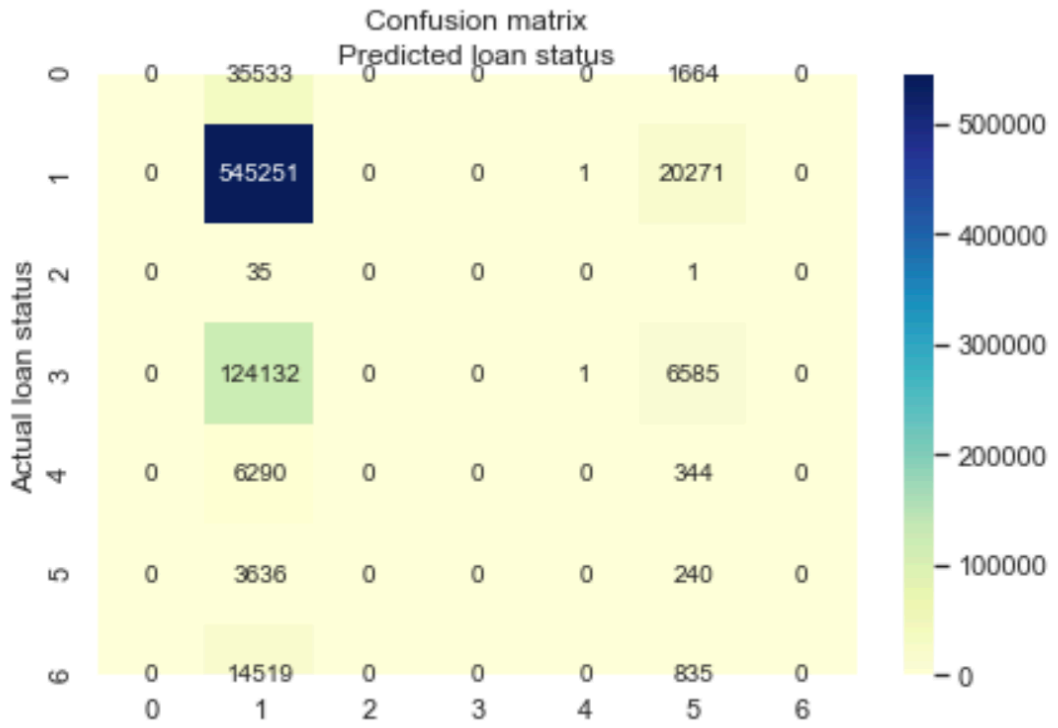
```
unique, counts = np.unique(y_pred, return_counts=True)
print (unique, counts)
```

```
['Current' 'In Grace Period' 'Late (16-30 days)'] [729396      2  29940]
```

Results

Step 9

Testing Performance




```
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
print("Precision:",metrics.precision_score(y_test, y_pred,pos_label='positive',
                                           average='micro'))
print("Recall:",metrics.recall_score(y_test, y_pred,pos_label='positive',
                                     average='micro'))
```

```
Accuracy: 0.7183770600180683
```

```
/Users/rachnish/opt/anaconda3/lib/python3.7/site-packages/sklearn/metrics/classificat
259: UserWarning: Note that pos_label (set to 'positive') is ignored when average !=
(got 'micro'). You may use labels=[pos_label] to specify a single positive class.
  % (pos_label, average), UserWarning)
```

```
Precision: 0.7183770600180683
Recall: 0.7183770600180683
```

The confusion matrix does not show a very good result of the model for the test data. On contrary, the accuracy of 71% is good for classification algorithms

Future Steps:

1. To improve the results of the model on Test Data.
2. Improvement in results seen through confusion matrix. A more diagonal scheme.
3. Testing other variables in the model to improve model score.