NAME OF THE PROJECT

MICRO CREDIT

Submitted by:

Rahul soni

ACKNOWLEDGMENT

A special thanks to my SME Mr. Harsh Ayush who gave me this opportunity to work on this project work we would also like to thanks Krish Naik for his tutorials on many sampling techniques which I used in this project work. I would also like to thanks my friends who helped and encouraged me during this project work.

Last but not the least thanks to Fliprobo technologies for sharing the data set of this project work.

INTRODUCTION

- ## Business Problem Framing

The growth in micro finance sector has been significant in last decades.
A Microfinance Institution (MFI) is an organization that offers financial services to low-income populations. MFS becomes very useful when targeting especially the unbanked poor families living in remote areas with not much sources of income.

Micro finance is widely accepted as a poverty-reduction tool.

In today's world communication is the key element for every person's life
In this project we are working and analyzing a data set provided by a company which is in telecom business and providing their customers airtime credit depending on their credit and usage history of their mobile number.
In this project work we will be predicting the class of the customers who have paid the airtime credit/loan amount with interest with in time or not.
Our target variable here is label which has 2 classes (1= paid the loan amount within time, 2=didn't paid the loan amount within time).

The main purpose of this project/analysis is to look at the credit risk problems that can arise so that the risk can be reduced.
In real-life scenarios financial institutes like banks, credit rating companies are very concerned with credit risk analysis. This project is also related to this as when a telecom company is providing airtime credit/loan, then it becomes the liability of the telecom company to repay the loan amount to the lending company (Micro Finance Institute).

Airtime-based credit scoring algorithms help MNOs better monetize their customer data to offer a wider range of products, including mobile money platforms either directly or in collaboration with banks.

- ## Conceptual Background of the Domain Problem

Mobile subscribers in the world are estimated to reach around 5.86 billion by 2025.
For a MNO it is very important to retain their most valued customers because of the competition in the telecom industry.
Out of the total subscribers' base 80% are on prepaid plans and mostly in rural, unbanked areas and churn rate is also more in prepaid plans, more than 80% of new

prepaid users leave within the first 90 days in hyper-competitive markets. Companies spends hundreds of millions of dollars to retain new prepaid customers, only to lose 60%+ of them within 30 days.
Now the question arises: Are the high costs of acquisition and constant churn in prepaid sustainable in emerging markets looking to reduce opex? Can we allow this model to take hold in developed markets? The answers can only be: no, it's not; and no, we can't.

So, what is the solution this? The answer to this is creating "SIM equity" -- a strong, compelling reason for subscribers to stick with their operator on the same SIM despite better short-term deals from competitors. It requires an approach that puts retention ahead of acquisition -- a monumental shift in MNO marketing strategy... especially in emerging markets.

- ## Review of Literature

For this project work I read an article by Steve Polsky (Founder and CEO, Juvo). The objective is to retain the prepaid users.
Prepaid users are invisible to MNOs beyond a phone number and an account balance. In many emerging markets, the average prepaid user will experience a "zero balance" day every week -- one day where they have no credit and cannot use their devices. In these markets, many customers do not have formal financial identities. This makes offering credit to encourage continued service usage nigh-on impossible -- if you take a traditional view of credit, which is to say "no" a lot more than "yes."

But it only takes an operator to say "yes" to, for example, a small interest-free airtime credit extension, to provide prepaid customers with the opportunity of a lifetime. Then, based on the subscriber's payback behaviour, machine learning can determine individualized lending criteria based on real-time data. Using this model, an individual can continue to borrow and pay back larger and larger amounts of credit -- slowly building trust, reducing risk and building an identity score that creates a compelling reason to stay with their operator.

- ## Motivation for the Problem Undertaken

The motivation for doing this project was primarily an interest in undertaking a challenging project in an interesting area of research. The opportunity to learn about a new area of computing not covered in lectures was appealing.

Analytical Problem Framing

- ## Mathematical/ Analytical Modeling of the Problem

    1. Absolute value conversion - some of the inputs were having negative values which was unrealistic for that variable, so an absolute data conversion is done there.
    2. Log transformation - for some input variables log transformation is done for normalizing the data.
    3. IQR – Interquartile Range is the difference between Q3 and Q1, this is used for replacing high var values (outliers).
    4. Correlation - Correlation is a statistic that measures the degree to which two variables move in relation to each other. It is used to check the correlation of input variables with other input variables and target variable and to check for multicollinearity.

- ## Data Sources and their formats

    ### 1. Data was provided by the client.

    In our data set we have 209593 observations and 37 variables/features including our target variable/feature (label).

2. Data Info: -

```
Data columns (total 37 columns):
 #   Column                Non-Null Count   Dtype
---  ------                --------------   -----
 0   Unnamed: 0            209593 non-null  int64
 1   label                 209593 non-null  int64
 2   msisdn                209593 non-null  object
 3   aon                   209593 non-null  float64
 4   daily_decr30          209593 non-null  float64
 5   daily_decr90          209593 non-null  float64
 6   rental30              209593 non-null  float64
 7   rental90              209593 non-null  float64
 8   last_rech_date_ma     209593 non-null  float64
 9   last_rech_date_da     209593 non-null  float64
 10  last_rech_amt_ma      209593 non-null  int64
 11  cnt_ma_rech30         209593 non-null  int64
 12  fr_ma_rech30          209593 non-null  float64
 13  sumamnt_ma_rech30     209593 non-null  float64
 14  medianamnt_ma_rech30  209593 non-null  float64
 15  medianmarechprebal30  209593 non-null  float64
 16  cnt_ma_rech90         209593 non-null  int64
 17  fr_ma_rech90          209593 non-null  int64
 18  sumamnt_ma_rech90     209593 non-null  int64
 19  medianamnt_ma_rech90  209593 non-null  float64
 20  medianmarechprebal90  209593 non-null  float64
 21  cnt_da_rech30         209593 non-null  float64
 22  fr_da_rech30          209593 non-null  float64
 23  cnt_da_rech90         209593 non-null  int64
 24  fr_da_rech90          209593 non-null  int64
 25  cnt_loans30           209593 non-null  int64
 26  amnt_loans30          209593 non-null  int64
 27  maxamnt_loans30       209593 non-null  float64
 28  medianamnt_loans30    209593 non-null  float64
 29  cnt_loans90           209593 non-null  float64
 30  amnt_loans90          209593 non-null  int64
 31  maxamnt_loans90       209593 non-null  int64
 32  medianamnt_loans90    209593 non-null  float64
 33  payback30             209593 non-null  float64
 34  payback90             209593 non-null  float64
 35  pcircle               209593 non-null  object
 36  pdate                 209593 non-null  object
dtypes: float64(21), int64(13), object(3)
```

Most of our variables are in float format, pdate, pcircle and msisdn are string type data.

3. Data Description: -

	Variable	Definition
1	label	Flag indicating whether the user paid back the credit amount within 5 days of issuing the loan{1:success, 0:failure}
2	msisdn	mobile number of user
3	aon	age on cellular network in days
4	daily_decr30	Daily amount spent from main account, averaged over last 30 days (in Indonesian Rupiah)
5	daily_decr90	Daily amount spent from main account, averaged over last 90 days (in Indonesian Rupiah)
6	rental30	Average main account balance over last 30 days
7	rental90	Average main account balance over last 90 days
8	last_rech_date_ma	Number of days till last recharge of main account
9	last_rech_date_da	Number of days till last recharge of data account
10	last_rech_amt_ma	Amount of last recharge of main account (in Indonesian Rupiah)
11	cnt_ma_rech30	Number of times main account got recharged in last 30 days
12	fr_ma_rech30	Frequency of main account recharged in last 30 days
13	sumamnt_ma_rech30	Total amount of recharge in main account over last 30 days (in Indonesian Rupiah)
14	medianamnt_ma_rech30	Median of amount of recharges done in main account over last 30 days at user level (in Indonesian Rupiah)
15	medianmarechprebal30	Median of main account balance just before recharge in last 30 days at user level (in Indonesian Rupiah)
16	cnt_ma_rech90	Number of times main account got recharged in last 90 days
17	fr_ma_rech90	Frequency of main account recharged in last 90 days
18	sumamnt_ma_rech90	Total amount of recharge in main account over last 90 days (in Indonasian Rupiah)
19	medianamnt_ma_rech90	Median of amount of recharges done in main account over last 90 days at user level (in Indonasian Rupiah)
20	medianmarechprebal90	Median of main account balance just before recharge in last 90 days at user level (in Indonasian Rupiah)
21	cnt_da_rech30	Number of times data account got recharged in last 30 days
22	fr_da_rech30	Frequency of data account recharged in last 30 days
23	cnt_da_rech90	Number of times data account got recharged in last 90 days
24	fr_da_rech90	Frequency of data account recharged in last 90 days
25	cnt_loans30	Number of loans taken by user in last 30 days
26	amnt_loans30	Total amount of loans taken by user in last 30 days
27	maxamnt_loans30	maximum amount of loan taken by the user in last 30 days
28	medianamnt_loans30	Median of amounts of loan taken by the user in last 30 days
29	cnt_loans90	Number of loans taken by user in last 90 days
30	amnt_loans90	Total amount of loans taken by user in last 90 days
31	maxamnt_loans90	maximum amount of loan taken by the user in last 90 days
32	medianamnt_loans90	Median of amounts of loan taken by the user in last 90 days
33	payback30	Average payback time in days over last 30 days
34	payback90	Average payback time in days over last 90 days
35	pcircle	telecom circle
36	pdate	date

4. Statistical description of data

Statistical description

```
In [6]: data.describe()
```

Out[6]:

	label	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_date_da	last_rech_amt_ma
count	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000
mean	0.875177	8112.343445	5381.402289	6082.515068	2692.581910	3483.406534	3755.847800	3712.202921	2064.452797
std	0.330519	75696.082531	9220.623400	10918.812767	4308.586781	5770.461279	53905.892230	53374.833430	2370.786034
min	0.000000	-48.000000	-93.012667	-93.012667	-23737.140000	-24720.580000	-29.000000	-29.000000	0.000000
25%	1.000000	246.000000	42.440000	42.692000	280.420000	300.260000	1.000000	0.000000	770.000000
50%	1.000000	527.000000	1469.175667	1500.000000	1083.570000	1334.000000	3.000000	0.000000	1539.000000
75%	1.000000	982.000000	7244.000000	7802.790000	3356.940000	4201.790000	7.000000	0.000000	2309.000000
max	1.000000	999860.755168	265926.000000	320630.000000	198926.110000	200148.110000	998650.377733	999171.809410	55000.000000

Out[6]:

cnt_ma_rech30	fr_ma_rech30	sumamnt_ma_rech30	medianamnt_ma_rech30	medianmarechprebal30	cnt_ma_rech90	fr_ma_rech90	sumamnt_ma_rech90	n
209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.00000	209593.000000	209593.000000	
3.978057	3737.355121	7704.501157	1812.817952	3851.927942	6.31543	7.716780	12396.218352	
4.256090	53643.625172	10139.621714	2070.864620	54006.374433	7.19347	12.590251	16857.793882	
0.000000	0.000000	0.000000	0.000000	-200.000000	0.00000	0.000000	0.000000	
1.000000	0.000000	1540.000000	770.000000	11.000000	2.00000	0.000000	2317.000000	
3.000000	2.000000	4628.000000	1539.000000	33.900000	4.00000	2.000000	7226.000000	
5.000000	6.000000	10010.000000	1924.000000	83.000000	8.00000	8.000000	16000.000000	
203.000000	999606.368132	810096.000000	55000.000000	999479.419319	336.00000	88.000000	953036.000000	

Out[6]:

medianamnt_ma_rech90	medianmarechprebal90	cnt_da_rech30	fr_da_rech30	cnt_da_rech90	fr_da_rech90	cnt_loans30	amnt_loans30	maxamnt_loans30
209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000
1864.595821	92.025541	262.578110	3749.494447	0.041495	0.045712	2.758981	17.952021	274.658747
2081.680664	369.215658	4183.897978	53885.414979	0.397556	0.951386	2.554502	17.379741	4245.264648
0.000000	-200.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
773.000000	14.600000	0.000000	0.000000	0.000000	0.000000	1.000000	6.000000	6.000000
1539.000000	36.000000	0.000000	0.000000	0.000000	0.000000	2.000000	12.000000	6.000000
1924.000000	79.310000	0.000000	0.000000	0.000000	0.000000	4.000000	24.000000	6.000000
55000.000000	41456.500000	99914.441420	999809.240107	38.000000	64.000000	50.000000	306.000000	99864.560864

medianamnt_loans30	cnt_loans90	amnt_loans90	maxamnt_loans90	medianamnt_loans90	payback30	payback90	date	month
209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.00000	209593.000000
0.054029	18.520919	23.645398	6.703134	0.046077	3.398826	4.321485	14.39894	6.797321
0.218039	224.797423	26.469861	2.103864	0.200692	8.813729	10.308108	8.43890	0.741435
0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.00000	6.000000
0.000000	1.000000	6.000000	6.000000	0.000000	0.000000	0.000000	7.00000	6.000000
0.000000	2.000000	12.000000	6.000000	0.000000	0.000000	1.666667	14.00000	7.000000
0.000000	5.000000	30.000000	6.000000	0.000000	3.750000	4.500000	21.00000	7.000000
3.000000	4997.517944	438.000000	12.000000	3.000000	171.500000	171.500000	31.00000	8.000000

5. Findings from statistical description: -

- Mean of all the variables is less than their respective standard deviation, which is not correct. It shows the data is not normally distributed.

- In aon, we see that minimum value is negative, it shouldn't be negative as age in days can never be negative.

- In daily_decr30 and daily_decr90 we see minimum value is in negative, it shouldn't be negative as the daily spending should be 0 or more than negative.

- In last_rech_date_ma and last_rech_date_da minimum value is also negative; it also needs correction as difference between the last recharge date and consideration date cannot be negative.

- In p-cirlce there are no variance among all the observation.

- In p-date format of the variable is object type, it should be datetime format to extract details.

- Most of the variables are highly skewed to right due to some unrealistic values.

- Our data is imbalanced, if we look at the target variable, we can see that the distribution is imbalanced.

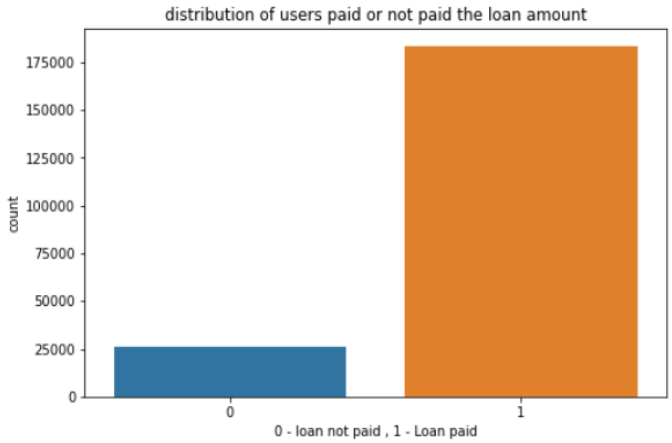
- Data Pre-processing Done

1. Firstly, we have converted the negative values in aon, daily_decr30, daily_decr90, last_rech_date_ma and last_rech_date_da into absolute values.

2. Now, as we see independent variables are highly skewed to right, to not to lose the data we have replaced the high values which are more than (Q3 + 1.5(IQR)) with Q3 + 1.5(IQR) and in some column's where values are less than (Q1-1.5(IQR)) with Q1-1.5(IQR).
3. In fr_ma_rech30 I have replaced the values which are greater than 38 with 38. Because there were many values which were far away from 38 so I replaced them by the max value (excluding far away values), In cnt_da_rech30 I replaced those values by 16, in fr_da_rech30 I replaced the values by 21 and in cnt_ma_rech90 I replaced the values by 55.
4. In maxamnt_loan30 and maxamnt_loan90 the maximum amount of loan should be 6 and 12 for both, here are some values which are very high so, here I replaced those values which were higher than 12 with the median of respective variables.
5. After correcting last_rech_date_da it appears that there is no variance there so removed the column from the dataset.
6. Log transformation using np.log1p done in most of the variable where the skewness was very hight. Log transformation helped us to make our input variables normally distributed and have less skeweness. Only with continuous values we have used log transformation.
7. Converted pdate variable into date time format and extracted the day and month details from it.
8. Removed variables (Unnamed: 0, msisdn, pcircle, pdate), Unnamed :0 and msisdn are not affecting our target variable as they are only representing the observation index, in pcircle there is no variance, pdate is not required as we have already extracted the day and month details from it.

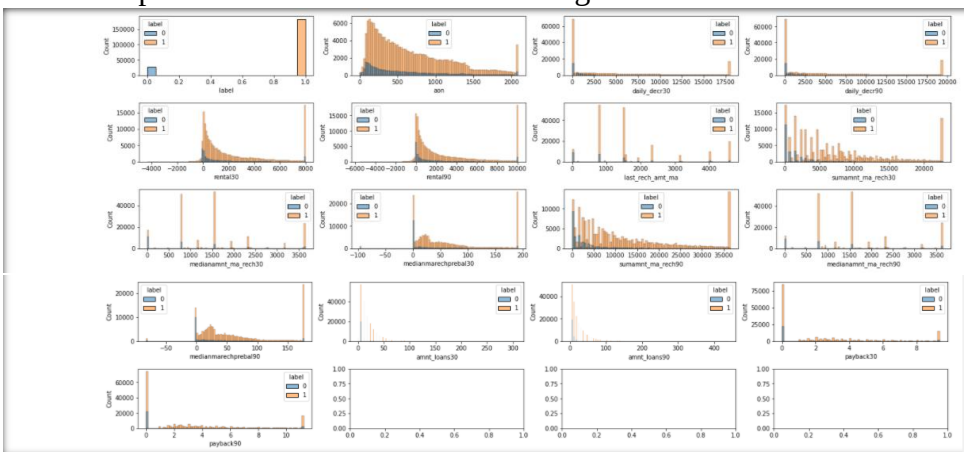
- ## Data Inputs- Logic- Output Relationships

I) Target variable (label) in our data set is highly imbalanced.



here, we can see that out of the total users 87.5 % are those who paid the loan amount with interest in time and 12.5 are those who didn’t paid the loan amount with interest within time.

2. Other independent variables are skewed to right.



- in aon we can see that the data is skewed to right, most of users are using number ranging from 0-500, it shows that there are many new users. From 0-300 approx. the number of defaulters is more than of non-defaulters, new users tend to fall in defaulter class.

- daily_decr30 and daily_decr90 shows almost same pattern, from 0 to 2500 the number of defaulters is more, when there is less amount spends on main account by the user, he/she is more likely to fall in defaulter's list.

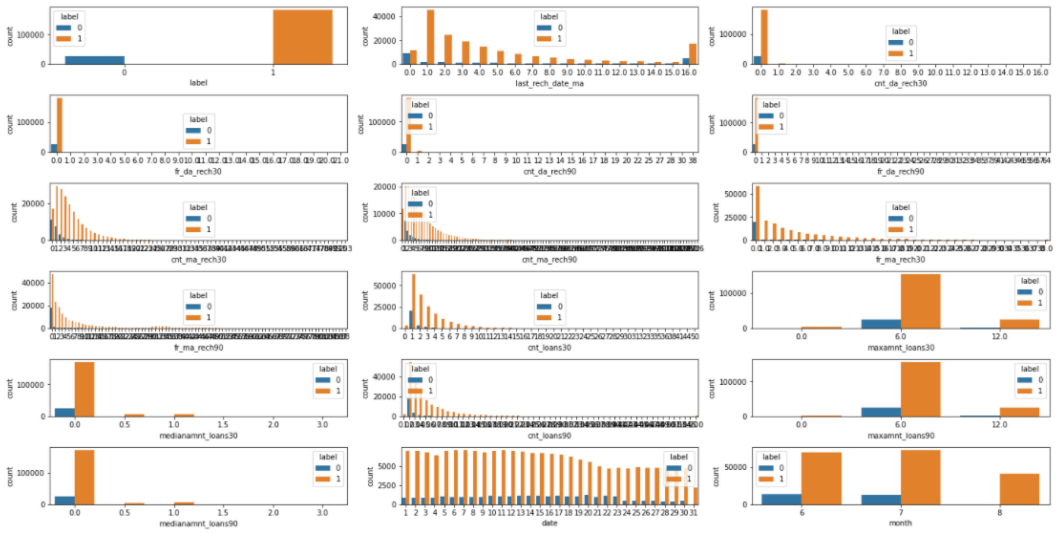
- rental30 and rental90 shows users with negative and less balance in the main account falls in defaulter's list. It is obvious that with having more and more balance in the account it is easy for the user to repay the loan amount.

- last_rech_amnt_ma - here we can see that when there is 0 the number of defaulters is more. Mean value for class 0 is 1075.73 and for class 1 is 1810.87.

- sum_amnt_ma_rech30 and sum_amnt_ma_rech90 is having 2155.96 and 3104.51 respectively for class 0 (defaulters) and 7532.14 and 11987.52 respectively for class 1(non-defaulters), when the total amount of recharge is less than the number of defaulters is more.

- amnt_loans30 and amnt_loans90 shows a right skewed data, from 0-25 in amnt_loans30 users are more. From 0-50 in amnt_loans90 users are more.

ANALYSIS OF CATEGORICAL VARIABLES



- last_rech_date_ma - the number of defaulters decreases as the number of recharge increases, at 0 when there is no recharge number of defaulters are at max.

- cnt_da_rech30 - here at 0 number of defaulters are more, most of the data is at 0.

- fr_da_rech30 - here at 0 number of defaulters are more, most of the data is at 0.

- cnt_da_rech90 - here at 0 number of defaulters are more, most of the data is at 0.

- fr_da_rech90 - here at 0 number of defaulters are more, most of the data is at 0.

- cnt_ma_rech30 - when there is a smaller number of recharges in last 30 days, we can see that the number of defaulters is more, as there is more and more recharge done in the number of defaulters decreases. At 0 max number of defaulters are around 12000.

- cnt_ma_rech90 - when there is a smaller number of recharges in last 30 days, we can see that the number of defaulters is more, as there is more and more recharge done in the number of defaulters decreases. At 0 max number of defaulters are at around 9000.

- fr_ma_rech30 and fr_ma_rech90 - these 2 also shows the same distribution, the number of defaulters is more when there is no or a smaller number of frequencies of recharge.

- cnt_loans30 = those who are taking loans from 1-4 times are tends to be defaulter, when the consumer is taking more loans, he is repaying them in time to build trust.

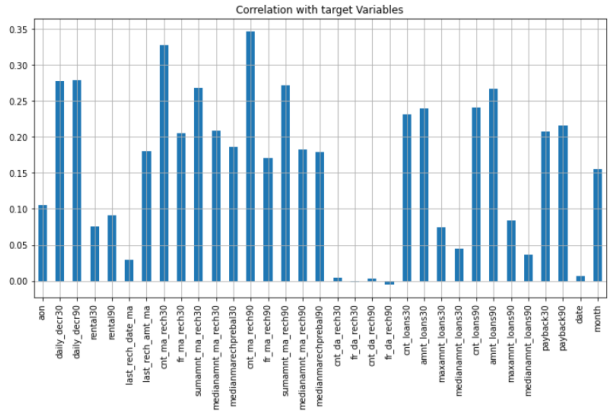
- maxamnt_loans30 - here we can see that maximum amount of loan 6 is mostly taken by the user. About 14% are defaulters when maxamnt_loans30 is 6 and 4% when maxamnt_loans30 is 12.

- medianamnt_loans30 - the numbers are ranging from 0-3, max number of defaulters is at 0

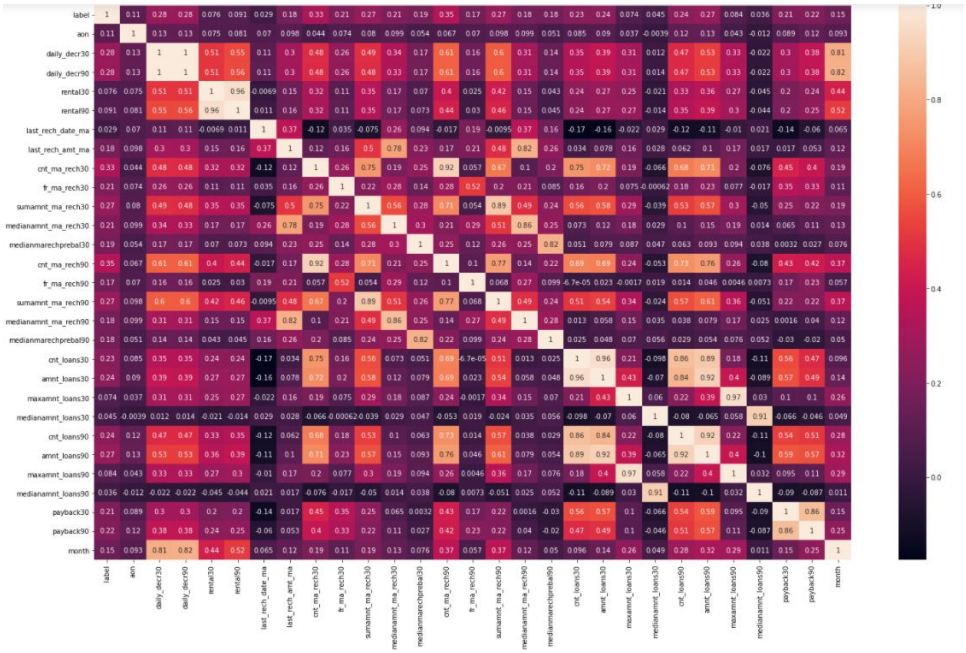
- cnt_loans90, maxamnt_loans90, and medianamnt_loans90 - these also shows the same distribution as it was for 30 days.

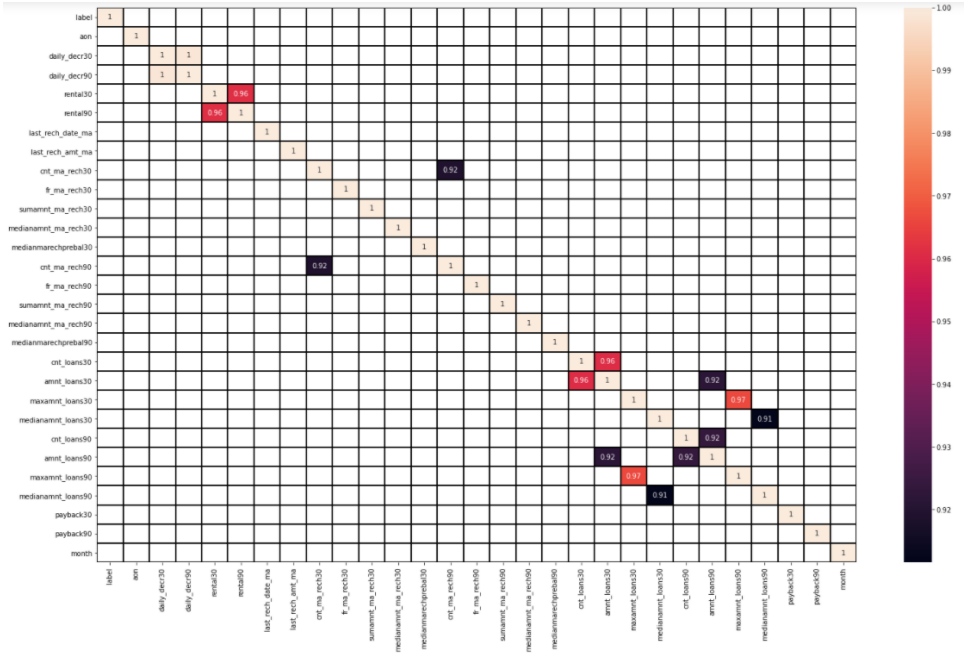
- month - here we can see that users who got their numbers in the 6th month have higher ration of defaulters. When month is 6 % of defaulters is 16% and when month is 7 % of defaulters is 15%

CORRELATION METRIX



here we can see that cnt_da_rech30, cnt_da_rech90, fr_da_rech30 and fr_da_rech90 are very less correlated to our target variable. date is also very less correlated. I will be removing these features for our model.





Here, we have filtered the correlation matrix in such a way that only the variables with correlation coefficient are more than equal to .90.
After checking the correlation of all the independent variables, we have removed one of the variables having .90 or more correlation coefficient.
For ex -
('daily_decr30','rental30','cnt_ma_rech30','cnt_loans30','amnt_loans30','maxamnt_loans30','medianamnt_loans30', 'cnt_loans90'), these are the variables where correlation was more than equal to .90.
This is done to reduce the chances of multicollinearity which will affect our model performance.

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

The 1st approach we did is to convert some columns like aon, daily_decr30, ddail_decr90, last_rech_date_ma and last_rech_date_da then corrected the outlier's values with IQR rule, if we had removed the outliers then there would have been big data loss. For normalising the data, we have used log transformation on those variables where skewness is high, using log transformation the data (independent variables) we have reduced the skewness.
This is an imbalanced data set so here we have also used oversampling technique to get better model performance.
Using Correlation matrix, we have removed the independent variables which are highly correlated with each other and having the same correlation with our target variable (label).

Splatted the data into inputs and output variable then scaled down the data inputs using standard scaler from sklearn.

- Testing of Identified Approaches (Algorithms)
    a. Logistic Regression.
    b. Random Forest Classifier.
    c. K-nearest Neighbor.
    d. Decision Tree.
    e. Gaussian NB.

- Run and Evaluate selected models
    1. **Logistic Regression.**

```
In [9]: log_class1 = LogisticRegression()
```

applying hyper parameter tuning

```
In [10]: grid_param = {'C':10.0**np.arange(-3,3),'penalty':['l1','l2']}
         cv = KFold(n_splits=5,shuffle=False,random_state=None)

In [11]: clf = GridSearchCV(log_class1,grid_param,cv=cv,n_jobs=-1,scoring='f1_macro')

In [17]: from imblearn.over_sampling import RandomOverSampler

In [18]: from collections import Counter
         Counter(y_train)

Out[18]: Counter({1: 122848, 0: 17579})

In [19]: os = RandomOverSampler(.4)
         X_train_sc_os,y_train_os = os.fit_sample(X_train_sc,y_train)
         print(' the number of classes before sampling{}'.format(Counter(y_train)))
         print(' the number of classes after sampling{}'.format(Counter(y_train_os)))

          the number of classes before samplingCounter({1: 122848, 0: 17579})
          the number of classes after samplingCounter({1: 122848, 0: 49139})

In [20]: training3 = log_class2.fit(X_train_sc_os,y_train_os)
         pred3 = log_class2.predict(X_test_sc)
         CM3 = confusion_matrix(y_test,pred3)
         CR3 = classification_report(y_test,pred3)
         acc3 = accuracy_score(y_test,pred3)

         print('confision metris :', '\n', CM3)
         print('classification report ', '\n', CR3)
         print('accuracy score: ', '\n' , acc3)

         confision metris :
          [[ 5118  3465]
          [ 6973 53610]]
         classification report
                       precision    recall  f1-score   support

                    0       0.42      0.60      0.50      8583
                    1       0.94      0.88      0.91     60583

             accuracy                           0.85     69166
            macro avg       0.68      0.74      0.70     69166
         weighted avg       0.88      0.85      0.86     69166

         accuracy score:
          0.849087702050403
```

2. Random Forest Classifier.

In [26]:
```
# RandomForestClassifier()
grid_param = {'criterion':['gini','entropy'],'max_features':['auto', 'sqrt', 'log2']}
cv = KFold(n_splits=3,shuffle=False,random_state=None)
```

In [27]:
```
rf_clf = GridSearchCV(RandomForestClassifier(),grid_param,cv=cv,n_jobs=-1,scoring='f1_weighted')
```

In [28]:
```
rf_clf.fit(X_train,y_train)
```

Out[28]:
```
GridSearchCV(cv=KFold(n_splits=3, random_state=None, shuffle=False),
             estimator=RandomForestClassifier(), n_jobs=-1,
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_features': ['auto', 'sqrt', 'log2']},
             scoring='f1_weighted')
```

In [29]:
```
rf_clf.best_params_
```

Out[29]:
```
{'criterion': 'entropy', 'max_features': 'auto'}
```

In [31]:
```
#trying with over sampling
from imblearn.over_sampling import RandomOverSampler
from collections import Counter
Counter(y_train)
```

Out[31]:
```
Counter({1: 122848, 0: 17579})
```

In [32]:
```
os = RandomOverSampler(.4)
X_train_os,y_train_os = os.fit_sample(X_train,y_train)
print(' the number of classes before sampling{}'.format(Counter(y_train)))
print(' the number of classes after sampling{}'.format(Counter(y_train_os)))
```

```
 the number of classes before samplingCounter({1: 122848, 0: 17579})
 the number of classes after samplingCounter({1: 122848, 0: 49139})
```

In [33]:
```
rf_model3 = rf_class2.fit(X_train_os,y_train_os)
rf_pred3 = rf_class2.predict(X_test)
rf_CM3 = confusion_matrix(y_test,rf_pred3)
rf_CR3 = classification_report(y_test,rf_pred3)
rf_acc3 = accuracy_score(y_test,rf_pred3)

print('confision metris :', '\n', rf_CM3)
print('classification report ', '\n', rf_CR3)
print('accuracy score: ', '\n' , rf_acc3)
```

```
confision metris :
 [[ 4542  4041]
 [ 1951 58632]]
classification report
              precision    recall  f1-score   support

           0       0.70      0.53      0.60      8583
           1       0.94      0.97      0.95     60583

    accuracy                           0.91     69166
   macro avg       0.82      0.75      0.78     69166
weighted avg       0.91      0.91      0.91     69166

accuracy score:
 0.9133678396900211
```

3. K- nearest Neighbor.

In [36]:
```
from sklearn.neighbors import KNeighborsClassifier
```

In [37]:
```
#setting grid search parameter for hyper tuning
param={'n_neighbors':np.arange(5,30),'weights':['uniform','distance']}
Knn_clf=GridSearchCV(KNeighborsClassifier(),param,cv=3,scoring='f1_weighted',n_jobs=-1)
```

In [38]:
```
Knn_clf.fit(X_train,y_train)
```

Out[38]:
```
GridSearchCV(cv=3, estimator=KNeighborsClassifier(), n_jobs=-1,
             param_grid={'n_neighbors': array([ 5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21,
       22, 23, 24, 25, 26, 27, 28, 29]),
                         'weights': ['uniform', 'distance']},
             scoring='f1_weighted')
```

In [39]:
```
Knn_clf.best_params_
```

Out[39]:
```
{'n_neighbors': 19, 'weights': 'distance'}
```

In [41]:
```
knn1 = KNeighborsClassifier(n_neighbors=19,weights='distance')
knn_model2 = knn1.fit(X_train_os,y_train_os)
knn_pred2 = knn1.predict(X_test)
knn_CM2 = confusion_matrix(y_test,knn_pred2)
knn_CR2 = classification_report(y_test,knn_pred2)
knn_acc2 = accuracy_score(y_test,knn_pred2)

print('confision metris :', '\n', knn_CM2)
print('classification report ', '\n', knn_CR2)
print('accuracy score: ', '\n' , knn_acc2)
```

```
confision metris :
 [[ 4787  3796]
 [ 6165 54418]]
classification report
              precision    recall  f1-score   support

           0       0.44      0.56      0.49      8583
           1       0.93      0.90      0.92     60583

    accuracy                           0.86     69166
   macro avg       0.69      0.73      0.70     69166
weighted avg       0.87      0.86      0.86     69166

accuracy score:
 0.8559841540641355
```

4. Decision Tree Classifier.

In [42]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import RandomizedSearchCV
```

In [43]:
```
#grid param for grid search cv
param={'max_depth':np.arange(1,50),'criterion':['entropy','gini'],'min_samples_leaf':np.arange(3,20),'max_features':["auto", "sqr
dt_clf=RandomizedSearchCV(DecisionTreeClassifier(),param,cv=3,scoring='f1_weighted')
```

In [44]:
```
dt_clf.fit(X_train,y_train)
```

Out[44]:
```
RandomizedSearchCV(cv=3, estimator=DecisionTreeClassifier(),
                   param_distributions={'criterion': ['entropy', 'gini'],
                                        'max_depth': array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17,
       18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34,
       35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49]),
                                        'max_features': ['auto', 'sqrt',
                                                         'log2'],
                                        'min_samples_leaf': array([ 3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17,
18, 19])},
                   scoring='f1_weighted')
```

In [45]:
```
dt_clf.best_params_
```

Out[45]:
```
{'min_samples_leaf': 19,
 'max_features': 'auto',
 'max_depth': 38,
 'criterion': 'gini'}
```

In [46]:
```
#building model
dt1 = DecisionTreeClassifier(min_samples_leaf=16,max_features='log2',max_depth=39,criterion='entropy')
dt_model1 = dt1.fit(X_train,y_train)
dt_pred1 = dt1.predict(X_test)
dt_CM1 = confusion_matrix(y_test,dt_pred1)
dt_CR1 = classification_report(y_test,dt_pred1)
dt_acc1 = accuracy_score(y_test,dt_pred1)

print('confision metris :', '\n', dt_CM1)
print('classification report ', '\n', dt_CR1)
print('accuracy score: ', '\n' , dt_acc1)
```

```
confision metris :
 [[ 3993  4590]
 [ 2075 58508]]
classification report
              precision    recall  f1-score   support

           0       0.66      0.47      0.55      8583
           1       0.93      0.97      0.95     60583

    accuracy                           0.90     69166
   macro avg       0.79      0.72      0.75     69166
weighted avg       0.89      0.90      0.90     69166

accuracy score:
 0.9036376254228956
```

5. Guassian NB.

In [50]:
```
from sklearn.naive_bayes import GaussianNB
```

In [51]:
```
gnb = GaussianNB()
gn_model1 = gnb.fit(X_train,y_train)
gn_pred1 = gnb.predict(X_test)
```

In [52]:
```
gn_CM1 = confusion_matrix(y_test,gn_pred1)
gn_CR1 = classification_report(y_test,gn_pred1)
gn_acc1 = accuracy_score(y_test,gn_pred1)

print('confision metris :', '\n', gn_CM1)
print('classification report ', '\n', gn_CR1)
print('accuracy score: ', '\n' , gn_acc1)
```

```
confision metris :
 [[ 6493  2090]
 [14581 46002]]
classification report
              precision    recall  f1-score   support

           0       0.31      0.76      0.44      8583
           1       0.96      0.76      0.85     60583

    accuracy                           0.76     69166
   macro avg       0.63      0.76      0.64     69166
weighted avg       0.88      0.76      0.80     69166

accuracy score:
 0.7589711708064656
```

- ## Key Metrics for success in solving problem under consideration

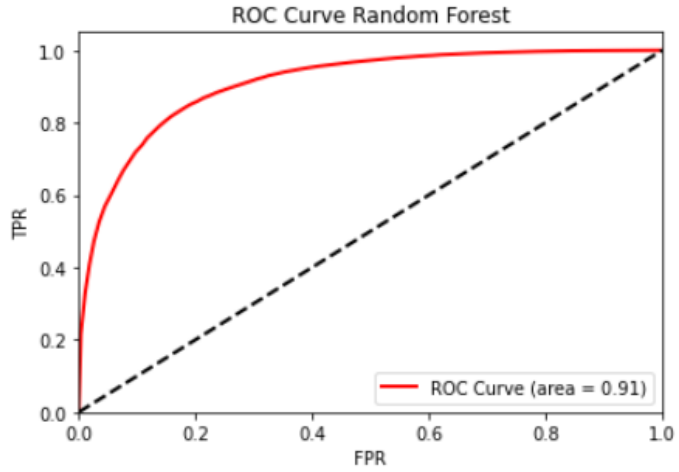
  3 key metrics/ performance metrics were used to solve and evaluate this project work.
  - Confusion matrix.
  - Classification report.
  - Accuracy score.

  Accuracy score will show the overall results of our model; however, our data is highly imbalanced so our main focus will be on precision score, recall score and f1 score that is why we are using classification report and confusion matrix is used to check the counts in TP, TN, FP, FN.

- ## Interpretation of the Results

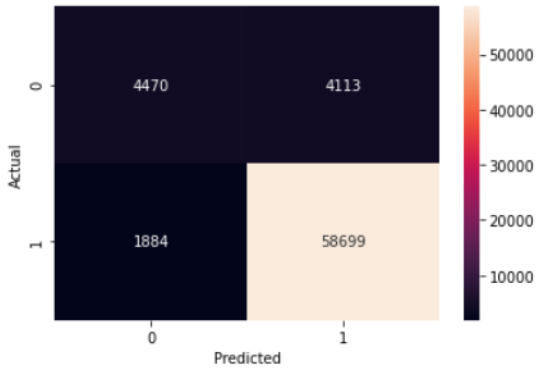
  Out of all the models created Random forest classification performs well.
  An accuracy of 91.32% with f1-score of class 0 (60) and class 1 (95).
  Random forest classifier deals and performs better when the data is imbalanced.

```
ROC AUC = 0.909733673498488
```



Above roc curve shows the performance of Random forest classifier.

```
In [78]: sns.heatmap(rf_CM3,annot=True, fmt='d')
         plt.ylabel('Actual')
         plt.xlabel('Predicted')
         plt.show()
```



above plot shows the distribution of True positive, True negative, False positive and False negative data, it is a heatmap of confusion matrix.

CONCLUSION

- ## Key Findings and Conclusions of the Study

We have seen that features like recharge done on data account and other features related to data account recharge are not much correlated to our target variable, users having more and more balance in their main account are more likely to fall in non-defaulter's list. Most of the users are taking loan of 6 instead of 12.
Users at initial phase (new users) are more likely to fall in defaulter’s lists, long term customers are loyal to the company and pay the loan amount with interest within time.

- ## Learning Outcomes of the Study in respect of Data Science

  From visualisation we were able to check the distribution of defaulters and non-defaulters with respect to every independent variable. We were able to check the correlation of each variable.
  Visualising Model performance using roc curve and confusion matrix.
  We have concluded that for imbalanced data set ensemble techniques and sampling techniques can be used for better model performance. In our final model we have used Random forest classifier with over sampling technique to get the best results.

- ## Limitations of this work and Scope for Future Work

  Due to some unrealistic values and imbalanced data the model may needs changes to predict future data. XGboost classification can be done for better results. In this data set there are a lot of outliers which should have been checked before as by removing the outliers data set losses 15-18% data which is not a good approach for an imbalanced data.