Micro-credit Defaulters Project

Submitted by:

Jayashri Dandare

ACKNOWLEDGMENT

- I would like to thank Flip Robo Technologies for providing me with the opportunity to work on this project from which I have learned a lot. I am also grateful to Miss Khushboo Garg for her constant guidance and support.

- I have primarily referred to various articles scattered across various websites for the purpose of getting an idea on “Micro-credit defaulters” project.

- I would like to thank the technical support team also for helping me out and reaching out to me on clearing all my doubts as early as possible.

TABLE OF CONTENTS

INTRODUCTION

- ## Business Problem Framing:

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. The Microfinance services (MFS) provided by MFI are Group Loans, Agricultural Loans, Individual Business Loans and so on.

Many microfinance institutions (MFI), experts and donors are supporting the idea of using mobile financial services (MFS) which they feel are more convenient and efficient, and cost saving, than the traditional high-touch model used since long for the purpose of delivering microfinance services. Though, the MFI industry is primarily focusing on low income families and are very useful in such areas, the implementation of MFS has been uneven with both significant challenges and successes.

Today, microfinance is widely accepted as a poverty-reduction tool, representing $70 billion in outstanding loans and a global outreach of 200 million clients.

- ## Conceptual Background of the Domain Problem:

We are working with client that is in Telecom Industry .They have launched various products and have developed its business and organization based on the budget operator model, offering better products at Lower Prices to all value conscious customers through a strategy of disruptive innovation that focuses on the subscriber.

Telecom Industry understand the importance of communication and how it affects a person's life, thus, focusing on providing their services and products to low income families and poor customers that can help them in the need of hour.

They are collaborating with an MFI to provide micro-credit on mobile balances to be paid back in 5 days. The Consumer is believed to be defaulter if he deviates from the path of paying back the loaned amount within the time

duration of 5 days. For the loan amount of 5 (in Indonesian Rupiah), payback amount should be 6 (in Indonesian Rupiah), while, for the loan amount of 10 (in Indonesian Rupiah), the payback amount should be 12 (in Indonesian Rupiah). The sample data is provided to us from our client database. It is hereby given to you for this exercise. In order to improve the selection of customers for the credit, the client wants some predictions that could help them in further investment and improvement in selection of customers.

- # Review of Literature:

An attempt has been made in this report to review the available literature in the area of microfinance. Approaches to microfinance, issues related to measuring social impact versus profitability of MFIs, issue of sustainability, variables impacting sustainability, which affect the regulations of profitability and impact assessment of MFIs have been summarized in the below report. We hope that the below report of literature will provide a platform for further research and help the industry to combine theory and practice to take microfinance forward and contribute to alleviating the poor from poverty.

The various applications and methods which inspired us to build our project. We did a background survey regarding the basic ideas of our project and used those ideas for the collection of information like the technological stack, algorithms, and shortcomings of our project which led us to build a better project.

I have built a model which can be used to predict in terms of a probability for each loan transaction, whether the customer will be paying back the loaned amount within 5 days of insurance of loan. In this case, Label ‘1’ indicates that the loan has been payed i.e. Nondefaulter, while, Label ‘0’ indicates that the loan has not been payed i.e. defaulter.

- # Motivation for the Problem Undertaken:

I have to model the micro credit defaulters with the available independent variables. This model will then be used by the management to understand how the customer is considered as defaulter or non-defaulter based on the independent variables. They can accordingly manipulate the strategy of the firm and concentrate on areas that will yield high returns. Further, the model will be a good way for the management to understand whether the customer will be paying back the loaned amount within 5 days of insurance of loan. The relationship between predicting defaulter and the economy is an important motivating factor for predicting micro credit defaulter model.

Analytical Problem Framing

- ## Mathematical/ Analytical Modeling of the Problem:

I am working with the micro credit defaulters dataset that contains various features and information about it. Using the data in form of ‘read_csv’ function provided by the Pandas package, which can import the data into our python environment. After importing the data, I have used the ‘head’ function to get a glimpse of our dataset.

In this project we have performed various mathematical and statistical analysis such as we checked description or statistical summary of the data using describe, checked correlation using corr and also visualized it using heatmap.

Summary Statistics

```
In [14]: df.describe()
```

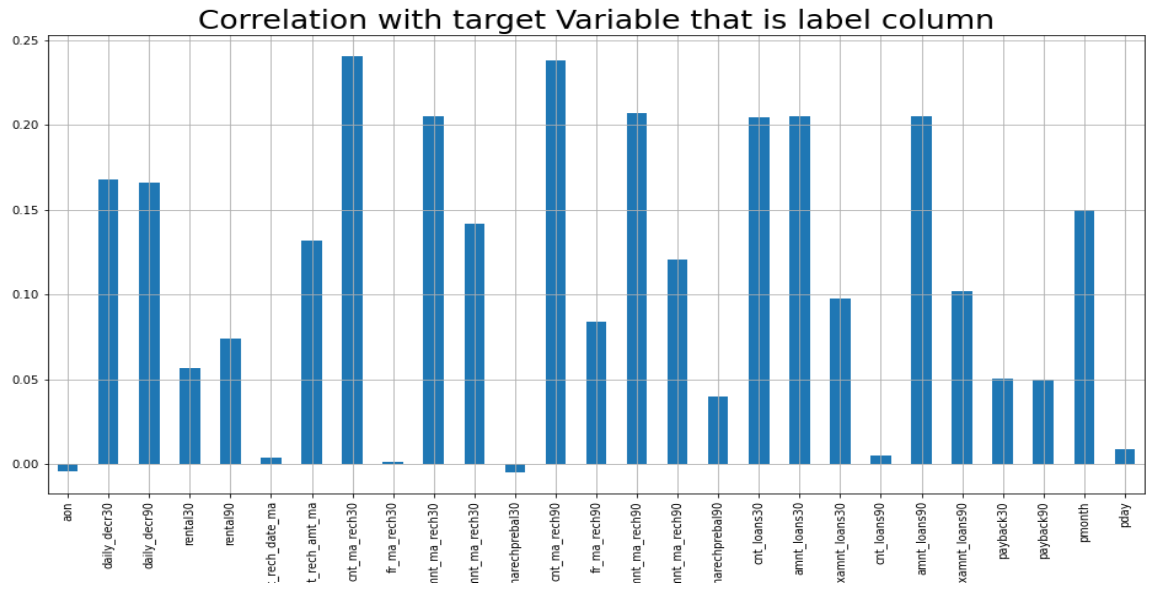
Out[14]:

	label	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_amt_ma	cnt_ma_rech30	fr_m
count	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	20959
mean	0.875177	8112.343445	5381.402289	6082.515068	2692.581910	3483.406534	3755.847800	2064.452797	3.978057	373
std	0.330519	75696.082531	9220.623400	10918.812767	4308.586781	5770.461279	53905.892230	2370.786034	4.256090	5364
min	0.000000	-48.000000	-93.012667	-93.012667	-23737.140000	-24720.580000	-29.000000	0.000000	0.000000	
25%	1.000000	246.000000	42.440000	42.692000	280.420000	300.260000	1.000000	770.000000	1.000000	
50%	1.000000	527.000000	1469.175667	1500.000000	1083.570000	1334.000000	3.000000	1539.000000	3.000000	
75%	1.000000	982.000000	7244.000000	7802.790000	3356.940000	4201.790000	7.000000	2309.000000	5.000000	
max	1.000000	999860.755168	265926.000000	320630.000000	198926.110000	200148.110000	998650.377733	55000.000000	203.000000	9996

Correlation

In [25]:
```
#Checking the correlation with target variable
plt.figure(figsize=(16,8))
df.drop('label', axis=1).corrwith(df['label']).plot(kind='bar',grid=True)
plt.xticks(rotation='vertical')
plt.title("Correlation with target Variable that is label column",fontsize=25)
```

Out[25]:
```
Text(0.5, 1.0, 'Correlation with target Variable that is label column')
```



- # Data Sources and their formats:

Dataset has been provided by internship company – Flip Robo technologies in excel format. The sample data is provided from our client database. Data given is only for academic use, not for any commercial. In order to improve the selection of customers for the credit, the client wants some predictions that could help them in further investment and improvement in selection of customers. Also, dataset was having 209593 rows and 36 columns including target. In this particular datasets I have object, float and integer types of data. The dataset is in both numerical as well as categorical data. There may be some customers with no loan history. The dataset is imbalanced. Label '1' has approximately 87.5% records, while, label '0' has approximately 12.5% records.

- # Data Preprocessing Done:

After loading all the required libraries, we loaded the data into our jupyter notebook.

In [2]:
```
# Loading the data set
df=pd.read_csv(r'C:\Users\JAYASHRI\Downloads\Micro-Credit-Project--1-\Micro Credit Project\Data file.csv')
df.head(10)
```

Out[2]:

	Unnamed: 0	label	msisdn	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_date_da	last_rech_amt_ma	cnt_ma_rech3
0	1	0	21408I70789	272.0	3055.050000	3065.150000	220.13	260.13	2.0	0.0	1539	
1	2	1	76462I70374	712.0	12122.000000	12124.750000	3691.26	3691.26	20.0	0.0	5787	
2	3	1	17943I70372	535.0	1398.000000	1398.000000	900.13	900.13	3.0	0.0	1539	
3	4	1	55773I70781	241.0	21.228000	21.228000	159.42	159.42	41.0	0.0	947	
4	5	1	03813I82730	947.0	150.619333	150.619333	1098.90	1098.90	4.0	0.0	2309	
5	6	1	35819I70783	568.0	2257.362667	2261.460000	368.13	380.13	2.0	0.0	1539	
6	7	1	96759I84459	545.0	2876.641667	2883.970000	335.75	402.90	13.0	0.0	5787	
7	8	1	09832I90846	768.0	12905.000000	17804.150000	900.35	2549.11	4.0	55.0	3178	
8	9	1	59772I84450	1191.0	90.695000	90.695000	2287.50	2287.50	1.0	0.0	1539	
9	10	1	56331I70783	536.0	29.357333	29.357333	612.96	612.96	11.0	0.0	773	

In [4]:
```
# Checking any null value present in dataset

df.isnull().sum()
```

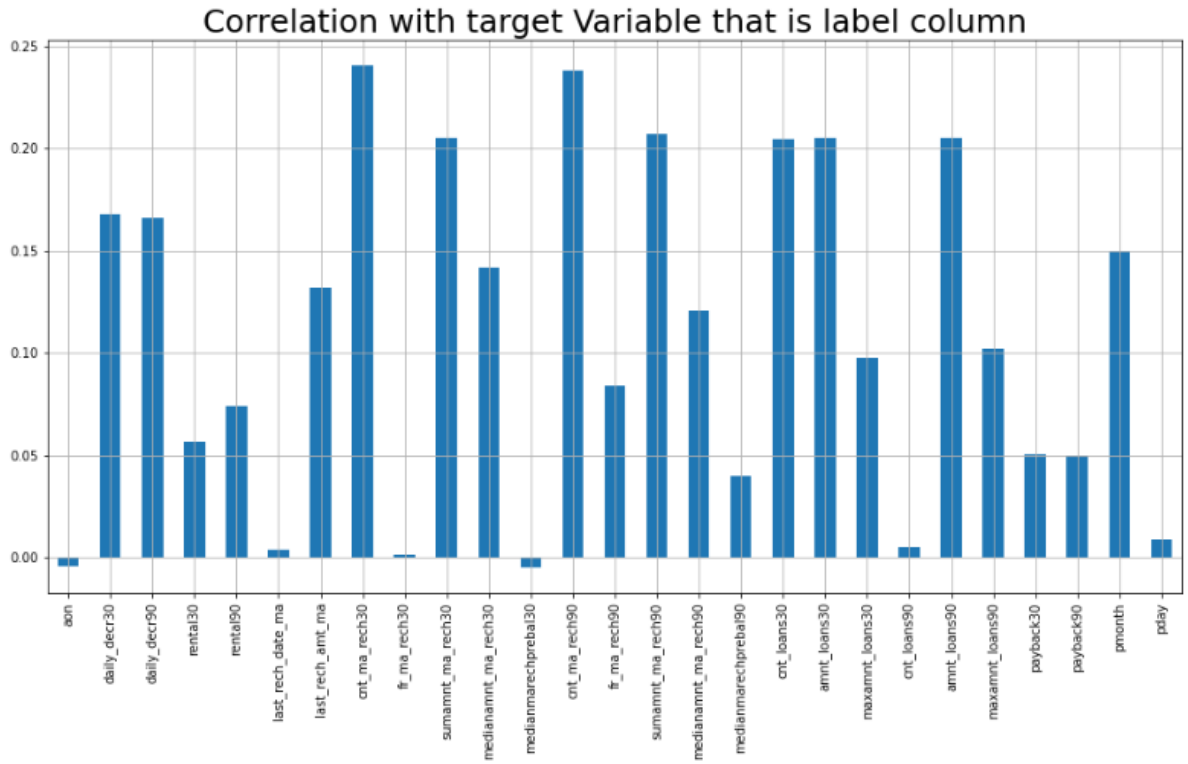
Out[4]:
```
Unnamed: 0              0
label                   0
msisdn                  0
aon                     0
daily_decr30            0
daily_decr90            0
rental30                0
rental90                0
last_rech_date_ma       0
last_rech_date_da       0
last_rech_amt_ma        0
cnt_ma_rech30           0
fr_ma_rech30            0
sumamnt_ma_rech30       0
medianamnt_ma_rech30    0
medianmarechprebal30    0
cnt_ma_rech90           0
fr_ma_rech90            0
sumamnt_ma_rech90       0
medianamnt_ma_rech90    0
medianmarechprebal90    0
cnt_da_rech30           0
fr_da_rech30            0
cnt_da_rech90           0
fr_da_rech90            0
cnt_loans30             0
amnt_loans30            0
maxamnt_loans30         0
medianamnt_loans30      0
cnt_loans90             0
amnt_loans90            0
maxamnt_loans90         0
medianamnt_loans90      0
payback30               0
payback90               0
pcircle                 0
pdate                   0
dtype: int64
```

Then we checked the correlation with target variable.

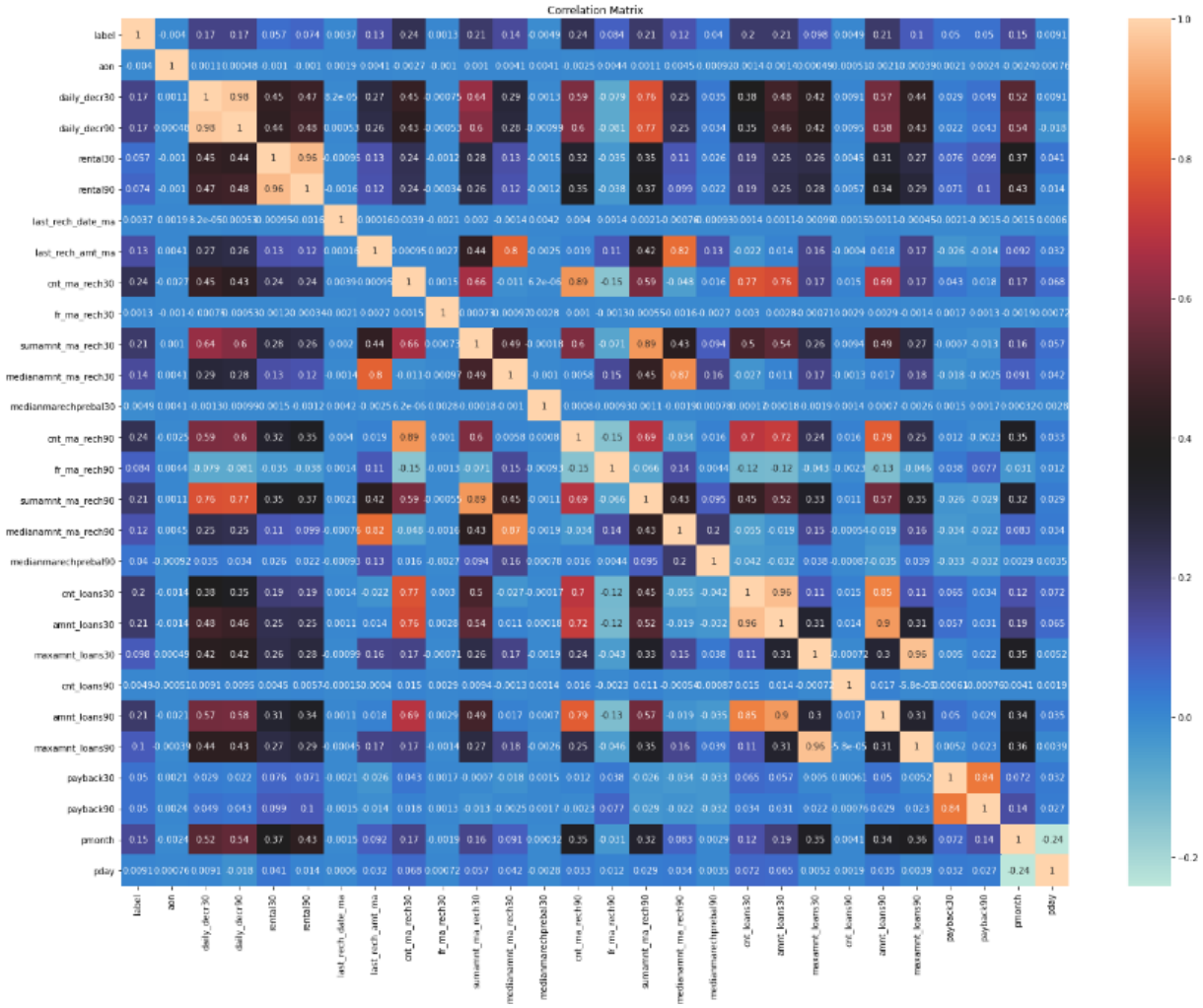
Correlation

In [25]:
```
#Checking the correlation with target variable
plt.figure(figsize=(16,8))
df.drop('label', axis=1).corrwith(df['label']).plot(kind='bar',grid=True)
plt.xticks(rotation='vertical')
plt.title("Correlation with target Variable that is label column",fontsize=25)
```

Out[25]: Text(0.5, 1.0, 'Correlation with target Variable that is label column')

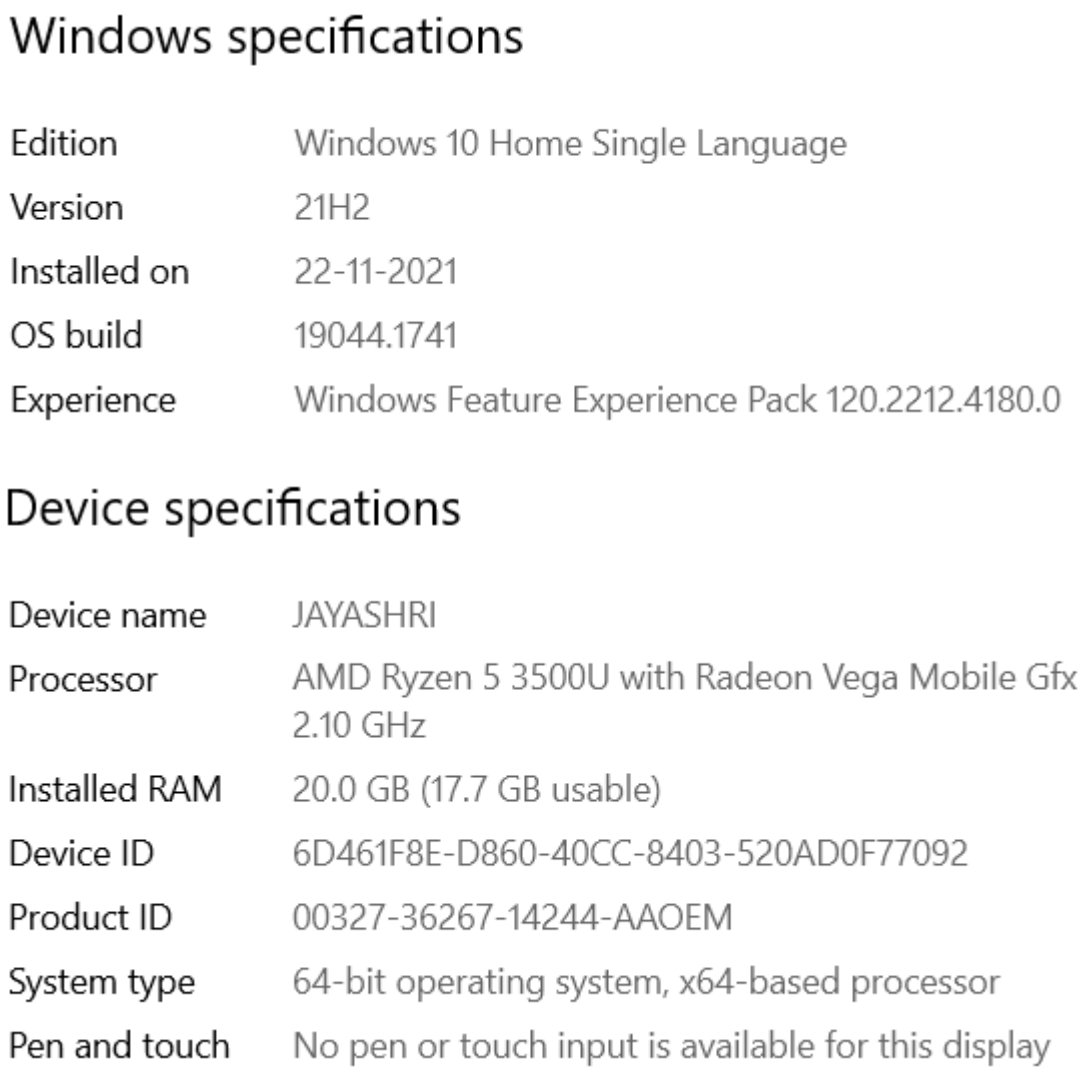


In [34]:
```
plt.figure(figsize=(24,18))
sns.heatmap(corr_matrix, annot=True, cmap='icefire')
plt.title("Correlation Matrix")
plt.show()
```



Hardware and Software Requirements and Tools Used

- **Hardware:**



Windows specifications

Edition	Windows 10 Home Single Language
Version	21H2
Installed on	22-11-2021
OS build	19044.1741
Experience	Windows Feature Experience Pack 120.2212.4180.0

Device specifications

Device name	JAYASHRI
Processor	AMD Ryzen 5 3500U with Radeon Vega Mobile Gfx 2.10 GHz
Installed RAM	20.0 GB (17.7 GB usable)
Device ID	6D461F8E-D860-40CC-8403-520AD0F77092
Product ID	00327-36267-14244-AAOEM
System type	64-bit operating system, x64-based processor
Pen and touch	No pen or touch input is available for this display

- **SOFTWARE:**

- Jupyter Notebook (Anaconda 3) – Python 3.7.6
- Microsoft Excel 2010

- # LIBRARIES:

- The tools, libraries and packages we used for accomplishing this project are pandas, numpy, matplotlib, seaborn, scipy stats, sklearn.preprocessing sklearn standardscaler, GridSearchCV.

Importing the needed Libraries

In [1]:
```
# Importing Libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import time
from datetime import datetime
pd.set_option('display.max_columns',None)
pd.set_option('display.max_rows',None)
import warnings
warnings.filterwarnings('ignore')
from scipy.stats import zscore

# Machine Learning Library

from sklearn.preprocessing import PowerTransformer
from sklearn.preprocessing import MinMaxScaler
from sklearn.linear_model import LogisticRegression,SGDClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from xgboost import XGBClassifier
from sklearn.model_selection import train_test_split,GridSearchCV,cross_val_score,KFold
from sklearn.metrics import confusion_matrix,classification_report,f1_score,accuracy_score,roc_auc_score,roc_curve
from imblearn.over_sampling import SMOTE
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.ensemble import RandomForestClassifier,AdaBoostClassifier,GradientBoostingClassifier,BaggingClassifier
from collections import Counter
from imblearn.combine import SMOTETomek
```

From sklearn.preprocessing import StandardScaler

As these columns are different in scale, they are standardized to have common scale while building machine learning model. This is useful when you want to compare data that correspond to different units.

from sklearn.preprocessing import Label Encoder

Label Encoder and One Hot Encoder. These two encoders are parts of the SciKit Learn library in Python, and they are used to convert categorical data, or text data, into numbers, which our predictive models can better understand.

from sklearn.model_selection import train_test_split,cross_val_score

- Train_test_split is a function in Sklearn model selection for splitting data arrays into two subsets: for training data and for testing data. With this function, you don't need to divide the dataset manually. By default, Sklearn train_test_split will make random partitions for the two subsets.

- Through pandas library we loaded our csv file ‘Data file’ into dataframe and performed data manipulation and analysis.
- With the help of numpy we worked with arrays.
- With the help of matplotlib and seaborn we did plot various graphs and figures and done data visualization.
- With scipy stats we treated outliers through winsorization technique.
- With sklearn’s standardscaler package we scaled all the feature variables onto single scale.

from sklearn.linear_model import LogisticRegression

The library sklearn can be used to perform logistic regression in a few lines as shown using the LogisticRegression class. It also supports multiple features.

from sklearn.tree import DecisionTreeRegressor

Decision Tree is a white box type of ML algorithm. It shares internal decision-making logic, which is not available in the black box type of algorithms such as Neural Network. Its training time is faster compared to the neural network algorithm. The time complexity of decision trees is a function of the number of records and number of attributes in the given data.

from sklearn.ensemble import RandomForestRegressor

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is controlled with the max_samples parameter if bootstrap=True (default), otherwise the whole dataset is used to build each tree.

Through GridSearchCV we were able to find the right parameters for hyperparameter tuning. Through joblib we saved our model in csv format.

Model/s Development and Evaluation

- ## Identification of possible problem-solving approaches (methods):

- The factors need to be found which can impact the micro credit. This can be done by analysing the various factors and the stores the respondent prefers. This will be done by checking each of the factors impacts the respondents in decision making.

- Machine learning-based systems are growing in popularity in research applications in most disciplines. Considerable decision-making knowledge from data has been acquired in the broad area of machine learning, in which decision-making tree-based ensemble techniques are recognized for supervised classification problems. Thus, classification is an essential form of data analysis in data mining that formulates models while describing significant data.

- # Testing of Identified Approaches (Algorithms):

We utilized several classic and state-of-the-art methods, including ensemble learning techniques, with a 90% - 10% split for the training and test data. To reduce the time required for training, we used over 20 thousand examples in our dataset with 209593 rows and 36 columns.

Z-score

In [30]:
```
z_score = zscore(df[['label']])
abs_zscore = np.abs(z_score)

filtering_entry = (abs_zscore < 3).all(axis=1)

df = df[filtering_entry]


# the data now seems much better than before.

df.describe()
```

Out[30]:

	label	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_amt_ma	cnt_ma_rech30	fr_m
count	206334.000000	206334.000000	206334.000000	206334.000000	206334.000000	206334.000000	206334.000000	206334.000000	206334.000000	2063
mean	0.873206	8083.775238	5336.954688	6021.166259	2664.367446	3431.671363	3747.842543	2051.860580	4.000456	37
std	0.332743	75547.449680	9212.622214	10903.651687	4262.266027	5690.281853	53835.726141	2359.664033	4.270021	535
min	0.000000	-48.000000	-93.012667	-93.012667	-23737.140000	-24720.580000	-29.000000	0.000000	0.000000	
25%	1.000000	246.000000	41.300000	41.484667	276.130000	299.700000	1.000000	770.000000	1.000000	
50%	1.000000	526.000000	1389.035667	1407.000000	1070.795000	1308.900000	3.000000	1539.000000	3.000000	
75%	1.000000	981.000000	7170.000000	7671.877500	3314.222500	4136.532500	7.000000	2309.000000	5.000000	
max	1.000000	999860.755168	265926.000000	320630.000000	198926.110000	200148.110000	998650.377733	55000.000000	203.000000	9996

So, from the above we can see that we have somewhat cleaned the datset and now we will proceed for model building.

Skewness

In [36]:
```
# Remove skewness
fetr=['aon', 'daily_decr30', 'daily_decr90', 'rental30', 'rental90', 'last_rech_date_ma', 'last_rech_amt_ma', 'cnt_ma_rech30', 't
```

In [37]:
```
pt = PowerTransformer(method='yeo-johnson')

df[fetr] = pt.fit_transform(df[fetr].values)
```

In [38]:
```
df[fetr].skew()
```

Out[38]:
```
aon                    1.681185
daily_decr30          -6.827117
daily_decr90          -7.346018
rental30              -1.002671
rental90              -0.982559
last_rech_date_ma     -5.330198
last_rech_amt_ma      -0.064244
cnt_ma_rech30         -0.000629
fr_ma_rech30           0.162386
sumamnt_ma_rech30     -0.290936
medianamnt_ma_rech30  -0.187796
medianmarechprebal30  -0.135088
cnt_ma_rech90         -0.002572
fr_ma_rech90           0.145152
sumamnt_ma_rech90     -0.194656
medianamnt_ma_rech90  -0.048127
medianmarechprebal90   7.467146
cnt_loans30            0.223040
amnt_loans30           0.148341
maxamnt_loans30        0.233352
cnt_loans90            0.232384
amnt_loans90           0.132503
maxamnt_loans90        0.000000
payback30              0.283898
payback90              0.202892
dtype: float64
```

The algorithms we used for the training and testing are as follows: -

- AdaBoost Classifier
- Gradient Boost Classifier
- Bagging Classifier
- Decision Tree Regressor
- Random Forest Regressor
- SGD Regressor
- XGB Classifier

• Run and Evaluate selected models:

Accuracy_score of train-test

In [45]:
```
lg=LogisticRegression()
grid={'C':10.0**np.arange(-2,3),'penalty':['l1','l2']}
cv=KFold(n_splits=5,shuffle=False,random_state=None)
clf=GridSearchCV(lg,grid,cv=cv,n_jobs=-1,scoring='f1_macro')
clf.fit(x_train,y_train)
```

Out[45]:

```
GridSearchCV(cv=KFold(n_splits=5, random_state=None, shuffle=False),
             estimator=LogisticRegression(), n_jobs=-1,
             param_grid={'C': array([1.e-02, 1.e-01, 1.e+00, 1.e+01, 1.e+02]),
                         'penalty': ['l1', 'l2']},
             scoring='f1_macro')
```

estimator: LogisticRegression

```
LogisticRegression()
```

LogisticRegression

```
LogisticRegression()
```

In [46]:
```
y_pred=clf.predict(x_test)
print(accuracy_score(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

```
0.8782087526857402
[[  862  6943]
 [  596 53500]]
              precision    recall  f1-score   support

           0       0.59      0.11      0.19      7805
           1       0.89      0.99      0.93     54096

    accuracy                           0.88     61901
   macro avg       0.74      0.55      0.56     61901
weighted avg       0.85      0.88      0.84     61901
```

Models:

In [50]:
```
dtc=DecisionTreeClassifier()
sgd=SGDClassifier()
xgb=XGBClassifier(eval_metric='logloss')
```

In [51]:
```
def classifiers(f):
    f.fit(X_train_sm,Y_train_sm)
    print(f,'\n',f.score(X_train_sm,Y_train_sm)*100)
    f_pred=f.predict(x_test)
    scores=cross_val_score(f,X_train_sm,Y_train_sm,cv=5).mean()*100
    print('\n cross value score',scores)
    print('ACCURACY SCORE:\n',accuracy_score(y_test,f_pred)*100)
    print('ROC AUC SCORE:\n',roc_auc_score(y_test,f_pred)*100)
    print('CONFUSION MATRIX:\n',confusion_matrix(y_test,f_pred))
    print('CLASSIFICATION REPORT:\n',classification_report(y_test,f_pred))
    return
```

Decision Tree Classifier

In [52]:
```
classifiers(dtc)
```

```
DecisionTreeClassifier()
 99.99915502002602

 cross value score 90.26881024267679
ACCURACY SCORE:
 87.18599053327087
ROC AUC SCORE:
 76.59575469883802
CONFUSION MATRIX:
 [[ 4873  2932]
 [ 5000 49096]]
CLASSIFICATION REPORT:
               precision    recall  f1-score   support

           0       0.49      0.62      0.55      7805
           1       0.94      0.91      0.93     54096

    accuracy                           0.87     61901
   macro avg       0.72      0.77      0.74     61901
weighted avg       0.89      0.87      0.88     61901
```

SGD Classifier

In [53]:
```
classifiers(sgd)
```

```
SGDClassifier()
 75.47952613523059

 cross value score 75.412350485272
ACCURACY SCORE:
 76.66273565855157
ROC AUC SCORE:
 74.40132009130423
CONFUSION MATRIX:
 [[ 5571  2234]
 [12212 41884]]
CLASSIFICATION REPORT:
               precision    recall  f1-score   support

           0       0.31      0.71      0.44      7805
           1       0.95      0.77      0.85     54096

    accuracy                           0.77     61901
   macro avg       0.63      0.74      0.64     61901
weighted avg       0.87      0.77      0.80     61901
```

XGB Classifier

In [54]:
```
classifiers(xgb)
```

```
XGBClassifier(base_score=0.5, booster='gbtree', callbacks=None,
              colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
              early_stopping_rounds=None, enable_categorical=False,
              eval_metric='logloss', gamma=0, gpu_id=-1,
              grow_policy='depthwise', importance_type=None,
              interaction_constraints='', learning_rate=0.300000012,
              max_bin=256, max_cat_to_onehot=4, max_delta_step=0, max_depth=6,
              max_leaves=0, min_child_weight=1, missing=nan,
              monotone_constraints='()', n_estimators=100, n_jobs=0,
              num_parallel_tree=1, predictor='auto', random_state=0,
              reg_alpha=0, reg_lambda=1, ...)
 95.0999611309212

 cross value score 93.3470924976303
ACCURACY SCORE:
 91.22308201806109
ROC AUC SCORE:
 80.00739260414636
CONFUSION MATRIX:
 [[ 5074  2731]
 [ 2702 51394]]
CLASSIFICATION REPORT:
              precision    recall  f1-score   support

           0       0.65      0.65      0.65      7805
           1       0.95      0.95      0.95     54096

    accuracy                           0.91     61901
   macro avg       0.80      0.80      0.80     61901
weighted avg       0.91      0.91      0.91     61901
```

In [55]:
```
ensemble=[RandomForestClassifier(),AdaBoostClassifier(),GradientBoostingClassifier(),BaggingClassifier()]
for i in ensemble:
    i.fit(X_train_sm,Y_train_sm)
    print(i,'\n score: ',i.score(X_train_sm,Y_train_sm)*100)
    pred=i.predict(x_test)
    print(' F1 score:',f1_score(y_test,pred)*100)
    print('Accuracy scoer:',accuracy_score(y_test,pred)*100)
    scores=cross_val_score(i,X_train_sm,Y_train_sm,cv=5).mean()*100
    print('\n Cross value score',scores)
    print('Confusion_matrix:\n',confusion_matrix(y_test,pred))
    print('Classification report:\n',classification_report(y_test,pred))
    print('\n')
```

```
RandomForestClassifier()
 score:  99.99746506007807
 F1 score: 94.78597492081721
Accuracy scoer: 90.90483190901601

 Cross value score 94.60355671531902
Confusion_matrix:
 [[ 5097  2708]
 [ 2922 51174]]
Classification report:
              precision    recall  f1-score   support

           0       0.64      0.65      0.64      7805
           1       0.95      0.95      0.95     54096

    accuracy                           0.91     61901
   macro avg       0.79      0.80      0.80     61901
weighted avg       0.91      0.91      0.91     61901
```

```
AdaBoostClassifier()
 score:  85.07385124972538
 F1 score: 89.62709592200854
Accuracy scoer: 83.0002746320738

 Cross value score 85.00626398306169
Confusion_matrix:
 [[ 5916  1889]
 [ 8634 45462]]
Classification report:
               precision    recall  f1-score   support

           0       0.41      0.76      0.53      7805
           1       0.96      0.84      0.90     54096

    accuracy                           0.83     61901
   macro avg       0.68      0.80      0.71     61901
weighted avg       0.89      0.83      0.85     61901


GradientBoostingClassifier()
 score:  90.09598972504351
 F1 score: 92.2311636929341
Accuracy scoer: 86.93074425292

 Cross value score 89.67647594474755
Confusion_matrix:
 [[ 5789  2016]
 [ 6074 48022]]
Classification report:
               precision    recall  f1-score   support

           0       0.49      0.74      0.59      7805
           1       0.96      0.89      0.92     54096

    accuracy                           0.87     61901
   macro avg       0.72      0.81      0.76     61901
weighted avg       0.90      0.87      0.88     61901


 BaggingClassifier()
  score:  99.60750680208878
  F1 score: 94.1254706554521
 Accuracy scoer: 89.86769195974217

  Cross value score 93.23849152794749
 Confusion_matrix:
  [[ 5382  2423]
  [ 3849 50247]]
 Classification report:
               precision    recall  f1-score   support

           0       0.58      0.69      0.63      7805
           1       0.95      0.93      0.94     54096

    accuracy                           0.90     61901
   macro avg       0.77      0.81      0.79     61901
weighted avg       0.91      0.90      0.90     61901
```

- # Key Metrics for success in solving problem under consideration

  Using the sklearn.metrics calculated Adjusted R2 squared ,Mean Absolute Error (MAE),Mean Squared Error (MSE),Root Mean Squared Error (RMSE)

  - Precision can be seen as a measure of quality, higher precision means that an algorithm returns more relevant results than irrelevant ones.
  - Recall is used as a measure of quantity and high recall means that an algorithm returns most of the relevant results.
  - Accuracy score is used when the True Positives and True negatives are more important. Accuracy can be used when the class distribution is similar.
  - F1-score is used when the False Negatives and False Positives are crucial. While F1- score is a better metric when there are imbalanced classes.
  - Cross_val_score: To run cross-validation on multiple metrics and also to return train scores, fit times and score times. Get predictions from each split of cross-validation for diagnostic purposes. Make a scorer from a performance metric or loss function.
  - AUC_ROC_score: ROC curve. It is a plot of the false positive rate (x-axis) versus the true positive rate (y-axis) for a number of different candidate threshold values between 0.0 and 1.0

  Using Hyper-parameter : model parameters are estimated from data automatically and model hyper-parameters are set manually and are used in processes to help estimate model and Grid search is a basic method for hyper-parameter tuning. It performs an exhaustive search on the hyper-parameter set specified by users.

VIF (Inspect VIF Factors)

In [44]:
```
# For each X, calculate VIF and save in dataframe


vif = pd.DataFrame()
vif["VIF Factor"] = [variance_inflation_factor(x1.values, i) for i in range(x1.shape[1])]
vif["features"] = x.columns

vif.round(1)
```

Out[44]:

	VIF Factor	features
0	110.3	aon
1	2650698.9	daily_decr30
2	2684678.2	daily_decr90
3	95568.6	rental30
4	96076.1	rental90
5	190.3	last_rech_date_ma
6	33.5	last_rech_amt_ma
7	218.1	cnt_ma_rech30

- **Visualizations:**

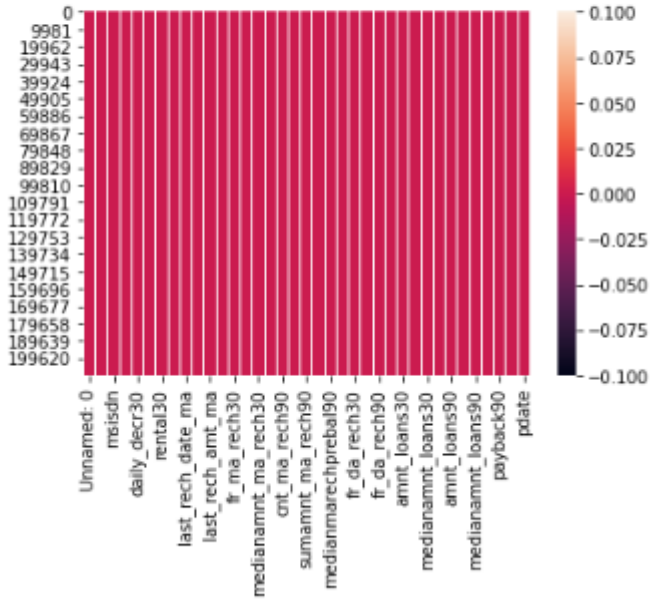
  • As the value counts observation I find imbalance dataset in which defaulter values is less and Non defaulter values is high. About to 15% and 85% respectively

  • Dataset is imbalanced. Label ‘1’ has approximately 87.5% records, while, label ‘0’ has approximately 12.5% records. Need to balance.

Data Visualization

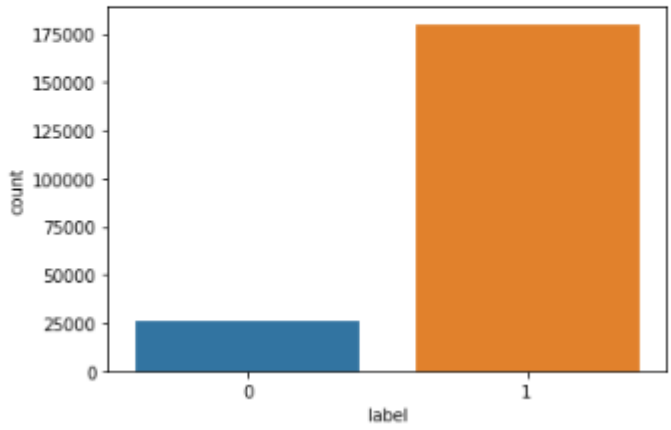
```
In [7]: sns.heatmap(df.isnull())
```

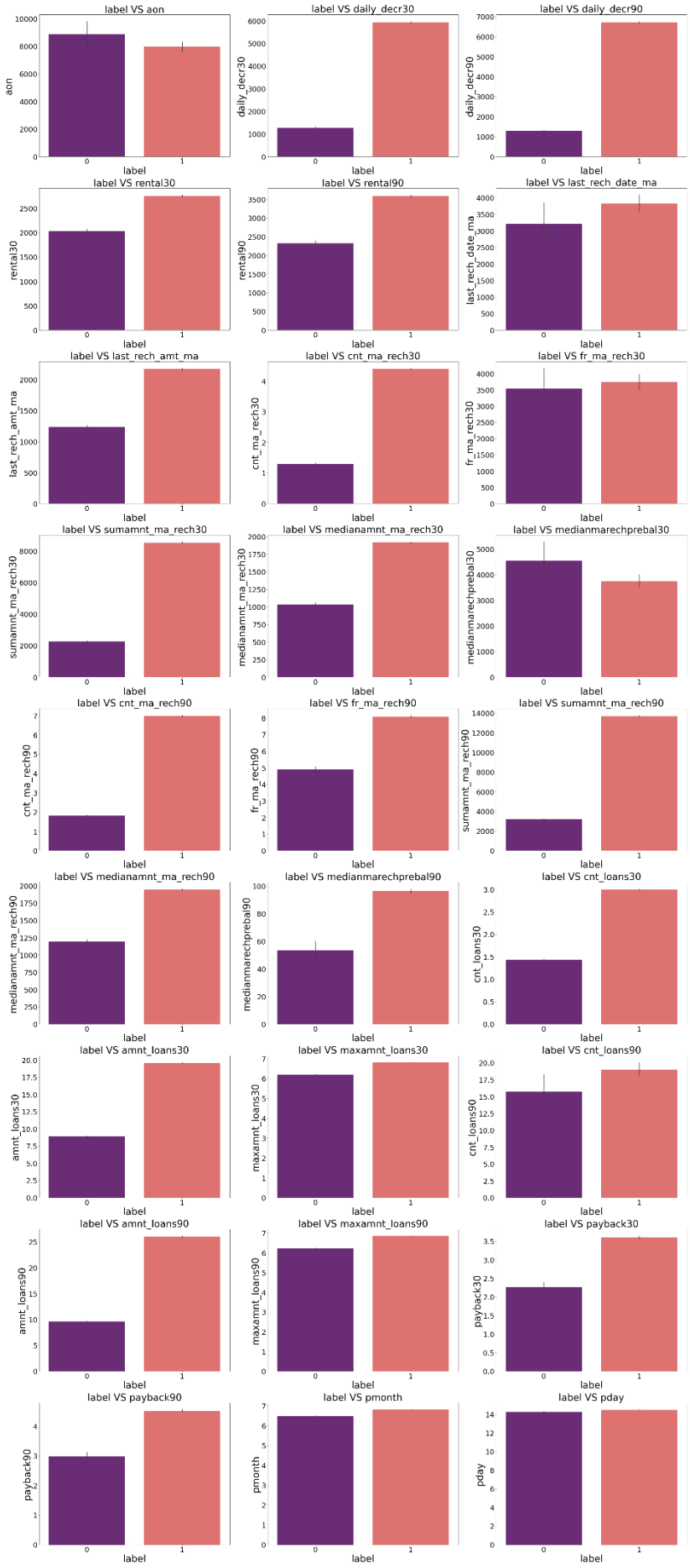
```
Out[7]: <AxesSubplot:>
```



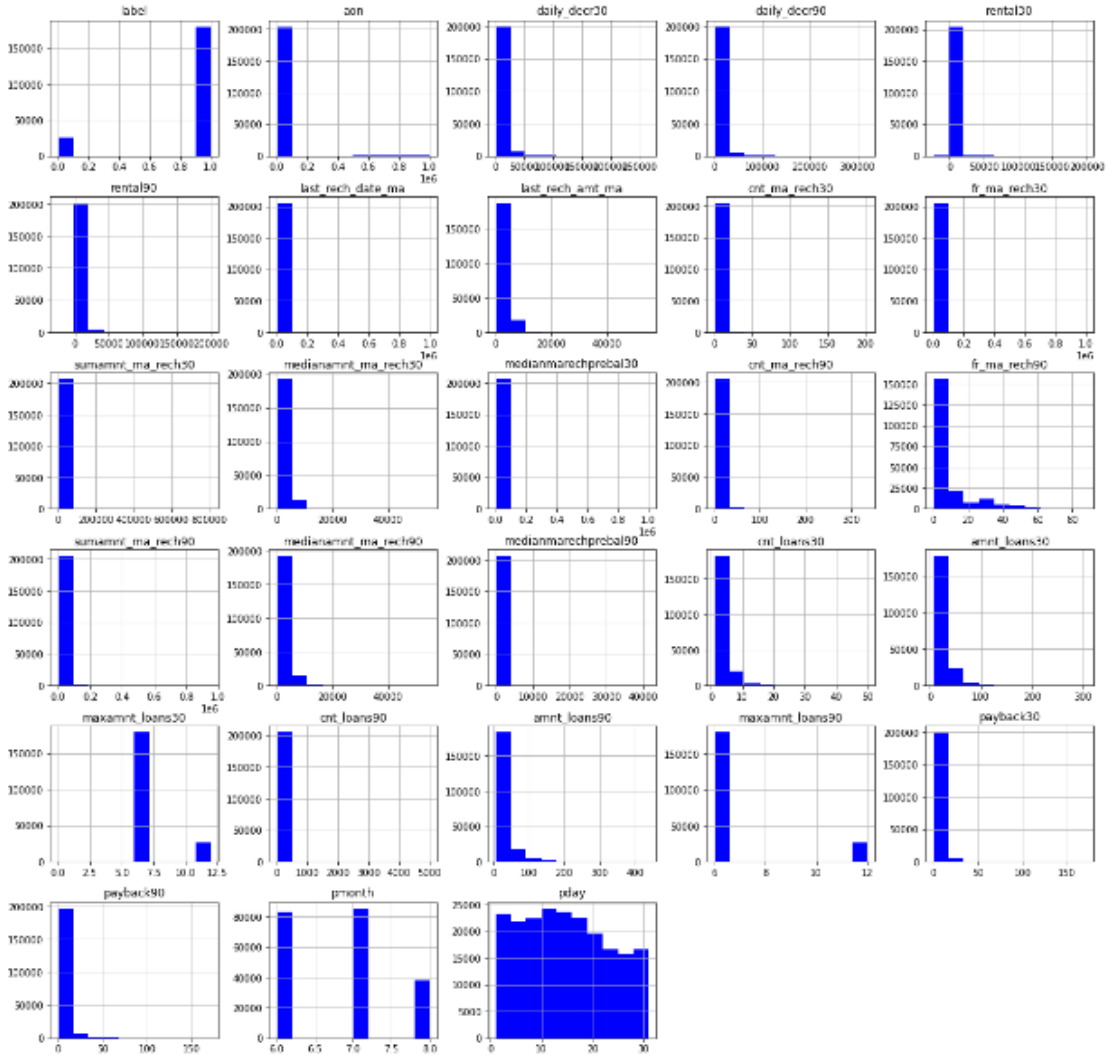

```
In [22]: #count for target column
         sns.countplot(df['label'])
```

```
Out[22]: <AxesSubplot:xlabel='label', ylabel='count'>
```




label VS aon
label VS daily_decr30
label VS daily_decr90
label VS rental30
label VS rental90
label VS last_rech_date_ma
label VS last_rech_amt_ma
label VS cnt_ma_rech30
label VS fr_ma_rech30
label VS sumamnt_ma_rech30
label VS medianamnt_ma_rech30
label VS medianmarechprebal30
label VS cnt_ma_rech90
label VS fr_ma_rech90
label VS sumamnt_ma_rech90
label VS medianamnt_ma_rech90
label VS medianmarechprebal90
label VS cnt_loans30
label VS amnt_loans30
label VS maxamnt_loans30
label VS cnt_loans90
label VS amnt_loans90
label VS maxamnt_loans90
label VS payback30
label VS payback90
label VS pmonth
label VS pday
label

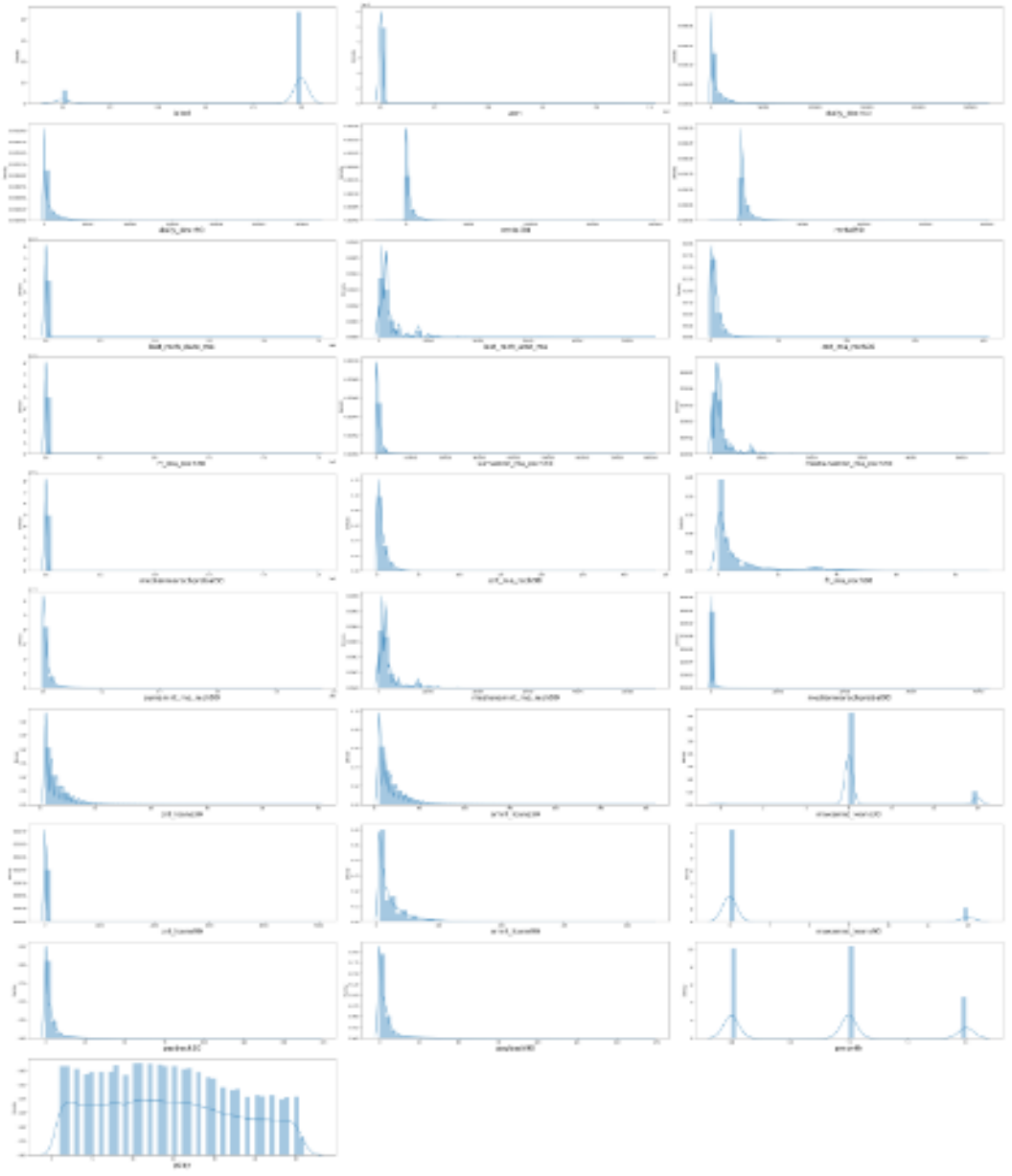
In [28]:
```
#Plotting the Histogram
df.hist(figsize=(20,20),color='b')
plt.show()
```



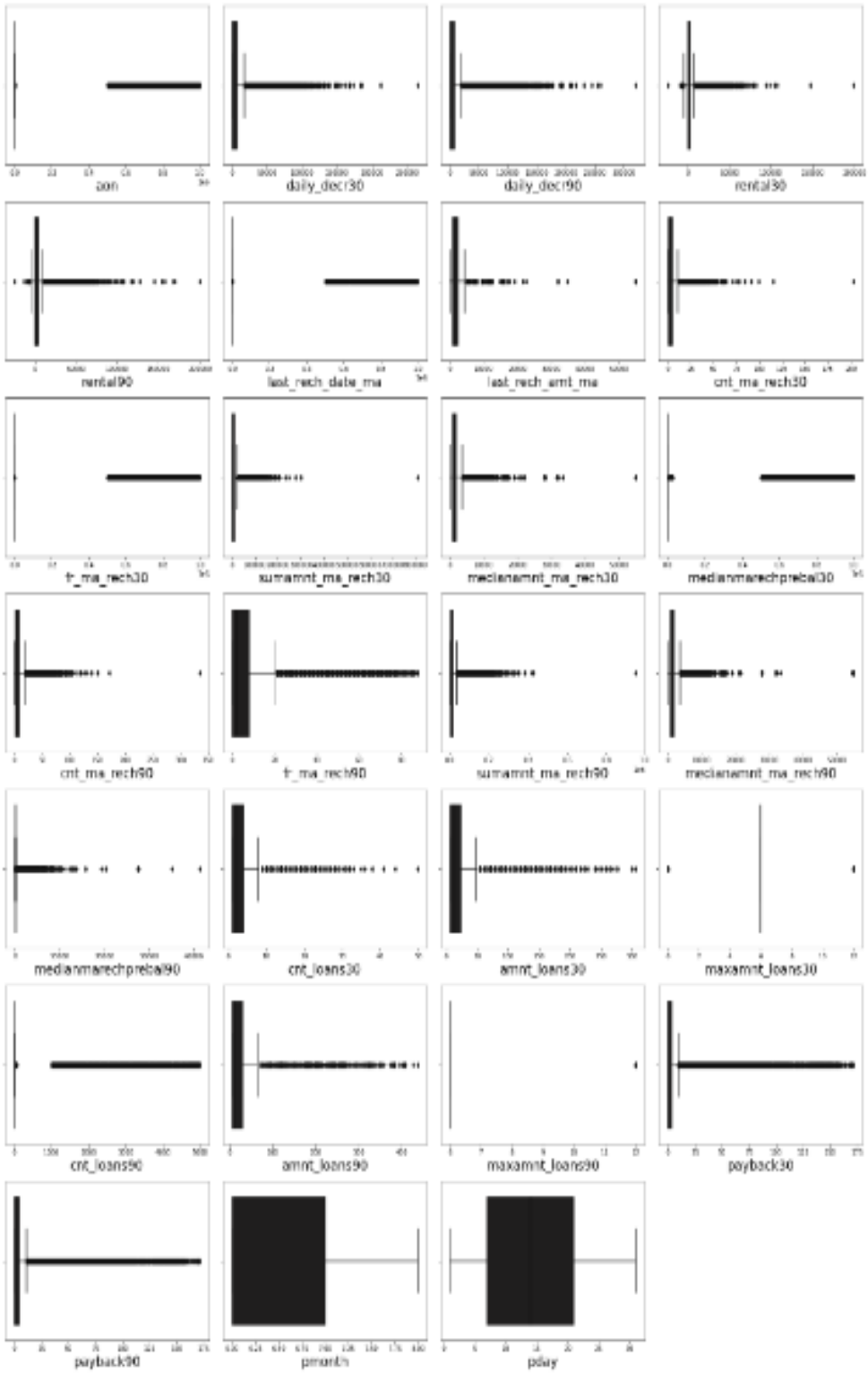
- To remove outliers I have used percentile method. And to remove skewness I have used yeo-johnson method. We have dropped all the unnecessary columns in the dataset according to our understanding. Use of Pearson's correlation coefficient to check the correlation between dependent and independent features. Also I have used Normalization to scale the data. After scaling we have to balance the target column using oversampling. Then followed by model building with all Classification algorithms. I have used oversampling (SMOTETomek) to get rid of data unbalancing. Bar plots to see the relation of numerical feature with target and 2 types of plots for numerical columns like distribution plot for univariate and bar plot for bivariate analysis.

Univariate Analysis

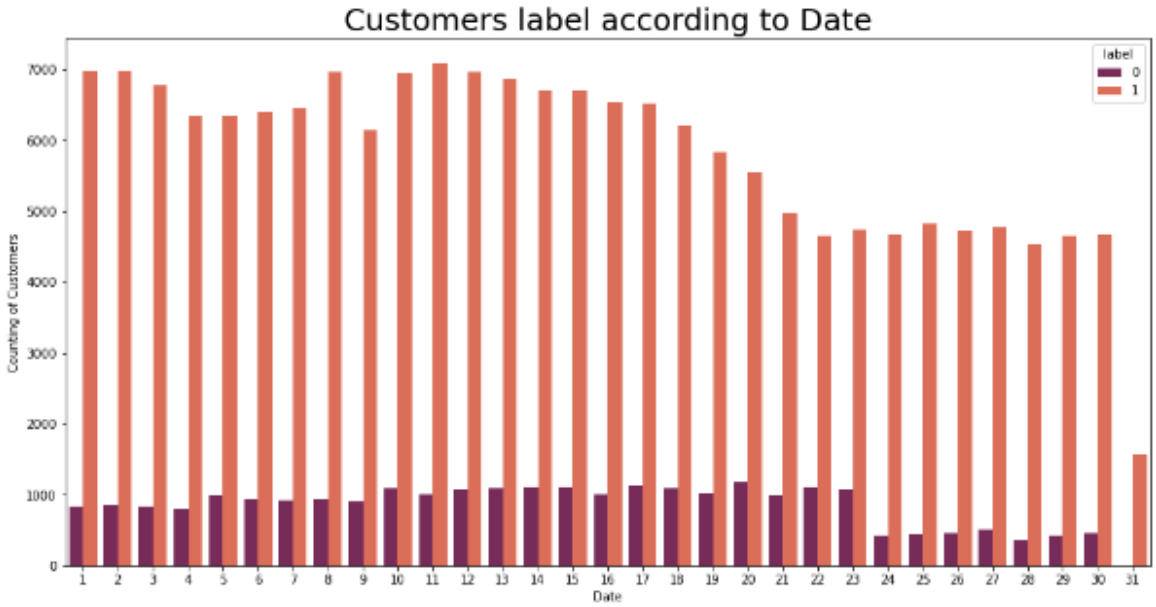
In [21]:
```
# distribution plot of numerical features
plt.figure(figsize=(40,50))
plotnumber=1
for column in numerical_features:
    if plotnumber <=30:
        ax=plt.subplot(10,3,plotnumber)
        sns.distplot(df[column])
        plt.xlabel(column,fontsize=20)
    plotnumber+=1
plt.tight_layout()
```



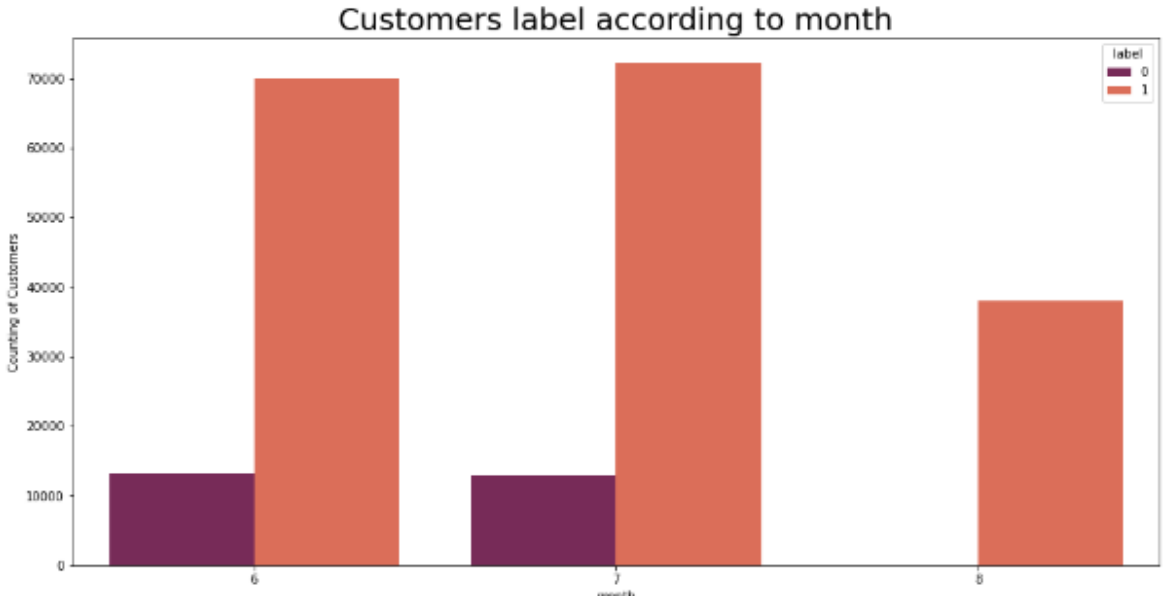
In [24]:
```
# Traget varible vs other column :

plt.figure(figsize=(20,30))
for i in range(len(col)):
    plt.subplot(8,4,i+1)
    sns.boxplot(df[col[i]], palette='icefire')
    plt.xlabel(col[i],fontsize =20)
    plt.tight_layout()
```



In [26]:
```
#Customer label according to per day:
plt.figure(figsize=(16,8))
sns.countplot(x="pday", hue='label', data=df, palette='rocket')
plt.title("Customers label according to Date", fontsize=25)
plt.xlabel('Date')
plt.ylabel('Counting of Customers')
plt.show()
```



In [27]:
```
#Customer label according to month (date):
plt.figure(figsize=(16,8))
sns.countplot(x="pmonth", hue='label', data=df, palette='rocket')
plt.title("Customers label according to month", fontsize=25)
plt.xlabel('month')
plt.ylabel('Counting of Customers')
plt.show()
```



Hyperparameter Tuning

In [56]:
```
n_estimators=[int(x) for x in np.linspace(start=0,stop=100,num=10)]
max_features=['auto','sqrt','log2']
criterion=['gini','entropy']
max_depth=[2,4]
min_samples_split=[1,2]
bootstrap=[True,False]
```

In [57]:
```
para_grid={'n_estimators':n_estimators,
           'max_features':max_features,
           'criterion':criterion,
           'max_depth':max_depth,
           'min_samples_split':min_samples_split,
           'bootstrap':bootstrap}
```

In [58]:
```
grid=GridSearchCV(RandomForestClassifier(),para_grid,n_jobs=-1)
grid.fit(X_train_sm,Y_train_sm)
grid.best_params_
```

Out[58]:
```
{'bootstrap': True,
 'criterion': 'gini',
 'max_depth': 4,
 'max_features': 'auto',
 'min_samples_split': 2,
 'n_estimators': 11}
```

ROC & AUC

• Present the receiver operating characteristic (ROC) curves and their respective areas under the curve (AUCs). ROC curves and AUCs are used to measure the quality of a classifier's output; thus, they measure how correctly a classifier has been tuned. Movement along the ROC curve is typically a trade-off between the classifier's sensitivity (true positive rate (TPR)) and specificity (TNR), and the steeper the curve, the better. For the ROC curve, sensitivity increases as we move up, and specificity decreases as we move right. The ROC curve along a 45_ angle.

ROC & AUC SCORE:

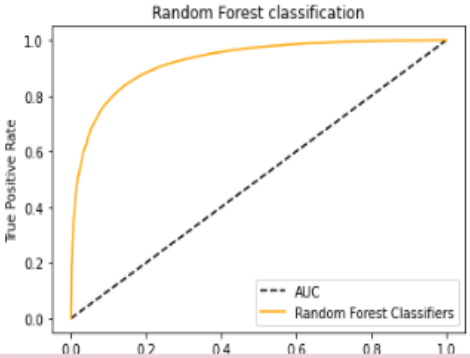
In [59]:
```
rf=RandomForestClassifier(bootstrap=False,criterion='gini',max_depth=None,max_features='sqrt',min_samples_split=7,n_estimators=
rf.fit(X_train_sm,Y_train_sm)
pred_rf=rf.predict(x_test)
print(accuracy_score(y_test,pred_rf))
print(confusion_matrix(y_test,pred_rf))
print(classification_report(y_test,pred_rf))
```

```
0.9111646015411706
[[ 4908  2897]
 [ 2602 51494]]
              precision    recall  f1-score   support

           0       0.65      0.63      0.64      7805
           1       0.95      0.95      0.95     54096

    accuracy                           0.91     61901
   macro avg       0.80      0.79      0.80     61901
weighted avg       0.91      0.91      0.91     61901
```

In [60]:
```
y_pred_prob=rf.predict_proba(x_test)[:,1]

fpr,tpr,thresold=roc_curve(y_test,y_pred_prob)
```

In [61]:
```
#Ploting ROC curve for final model

plt.plot([0,1],[0,1],'k--',label='AUC')
plt.plot(fpr,tpr,label='Random Forest Classifiers',color='orange',)
print("ROC AUC SCORE: ",roc_auc_score(y_test,pred_rf)*100)
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Random Forest classification')
plt.legend()
plt.show()
```

```
ROC AUC SCORE:  79.03639999575577
```



- **Interpretation of the Results**

  - In this research, two experiments were performed, the first experiment was validating and filtering data using all the variables available in the dataset after pre-processing, while the second experiment was conducted using most important variables and the goal of this is to be able to improve the model's performance using fewer variables.
  - Requirement of train and test and building of many models to get accuracy of the model.
  - There are multiple of matric which decide the best fit model like as : R-squared ,RMSE value, VIF, CDF & PDF Z-score , Roc & Auc and etc.
  - Database helped in making perfect model and will help in understanding Indonesian micro finance services (MFS) And use multiple metrics like F1_score, precision, recall and accuracy_score which will help to decide the best model.
  - Random forest Classifier as the best model with 91.18% accuracy_score..
  - Lastly predicted wheather the loan is paid back or not using saved model. It was good!! that was able to get the predictions near to actual values.

CONCLUSION

- **Key Findings and Conclusions of the Study:**

  This research evaluated individuals' credit risk performance in a micro-finance environment using machine learning and deep learning techniques. While traditional methods utilizing models such as linear regression are commonly adopted to estimate reasonable accuracy nowadays, these models have been succeeded by extensive employment of machine and deep learning models that have been broadly applied and produce prediction outcomes with greater precision. Using real data, we compared the various machine learning algorithms' accuracy by performing detailed experimental analysis while classifying individuals' requesting a loan into three classes, namely, good, average, and poor.

  In this project report, we have used machine learning algorithms to predict the micro credit defaulters. We have mentioned the step-by-step procedure to analyse the dataset and finding the correlation between the features. Thus, we can select the features which are correlated to each other and are independent in nature. These feature set were then given as an input to four algorithms and a hyper parameter tuning was done to the best model and the accuracy has been improved. Calculated the performance of each model using different performance metrics and compared them based on these metrics. Then we have also saved the best fit model and predicted the label. This is interesting that predicted and actual values were almost same.

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

- Dataset is imbalanced. Label ‘1’ has approximately 87.5% records, while, label ‘0’ has approximately 12.5% records, and defaulter are higher.
- This model will be a good way for the management to understand whether the customer will be paying back the loaned amount within 5 days of insurance of loan. The relationship between predicting defaulter and the economy is an important motivating factor for predicting micro credit defaulter

- # Limitations of this work and Scope for Future Work

- The length of the dataset it is very huge and hard to handle.
- Number of outliers and skewness these two will reduce our model accuracy.
- Also, we have tried best to deal with outliers, skewness and zero values. So it looks quite good that we have achieved a accuracy of 91.32% even after dealing all these drawbacks.
- This study will not cover all Classification algorithms instead, it is focused on the chosen algorithm, starting from the basic assembling techniques to the advanced ones.