Micro-Credit Defaulter Model

Submitted by:

SRIVIDYA.V

ACKNOWLEDGMENT

I would like to express my deepest gratitude to my SME (Subject Matter Expert) Shubam Yadav as well as Flip Robo Technologies who gave me the opportunity to do this project on 'Micro-Credit Defaulter Model' & also helping me to gain in-depth knowledge of Machine Learning and DataScience to derive insights for organizational goals or to meet business needs.

Also, I have utilized a few external resources that helped me to complete this project. All the external resources that were used in creating this project are listed below:

https://stackoverflow.com/questions

https://medium.com/

https://www.kaggle.com/

https://www.geeksforgeeks.org/

https://www.codegrepper.com/

https://www.analyticsvidhya.com/

https://towardsdatascience.com/

https://github.com/

INTRODUCTION

Business Problem Framing

Problem Overview

Micro-Credit Defaulter Model

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.

We are working with one such client that is in Telecom Industry. They are a fixed wireless telecommunications network provider. 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.

They 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.

Prediction:

Build 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. Non- defaulter, while, Label '0' indicates that the loan has not been payed i.e. defaulter.

Conceptual Background of the Domain Problem

MACHINE LEARNING AND DATA SCIENCE FOR BUSINESS:

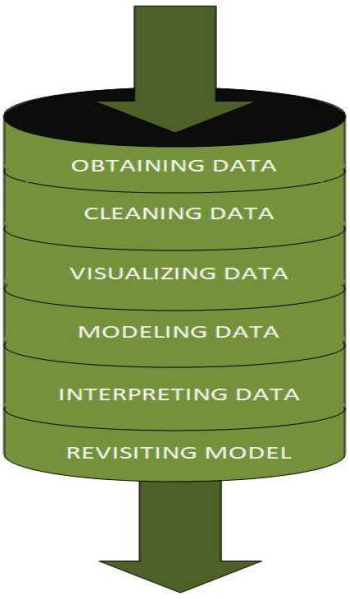
Machine learning is a branch of artificial intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the way that humans learn from experience, make predictions and gradually improving its accuracy. It is an important component of the growing field of data science. Through the use of statistical methods, algorithms are trained to make classifications or predictions, uncovering key insights within data mining projects. These insights subsequently drive decision making within applications and businesses, ideally impacting key growth metrics. As big data continues to expand and grow, the market demand for data science will increase, requires to assist in the identification of the most relevant business questions and subsequently the data to answer them. Following are the ways Data science can add value to Business :

- Empowering management and officers to make better decision
- Directing actions based on trends—which in turn help to define goals
- Challenging the staff to adopt best practices and focus on issues that matter
- Identifying opportunities
- Decision making with quantifiable, data-driven evidence
- Testing these decisions
- Identification and refining of target audiences

DATASCIENCE PIPELINE:

The data science pipeline is a collection of connected tasks that aims at delivering an insightful data science product or service to the business organization. The responsibilities include collecting,

cleaning, exploring, modeling, interpreting the data, and other processes of the launching of the product. This final product can be used for to achieve Business Goals.



Exploratory Data Analysis:

The main purpose of EDA is to help look at data before making any assumptions. It can help identify obvious errors, as well as better understand patterns within the data, detect outliers or anomalous events, find interesting relations among the variables.

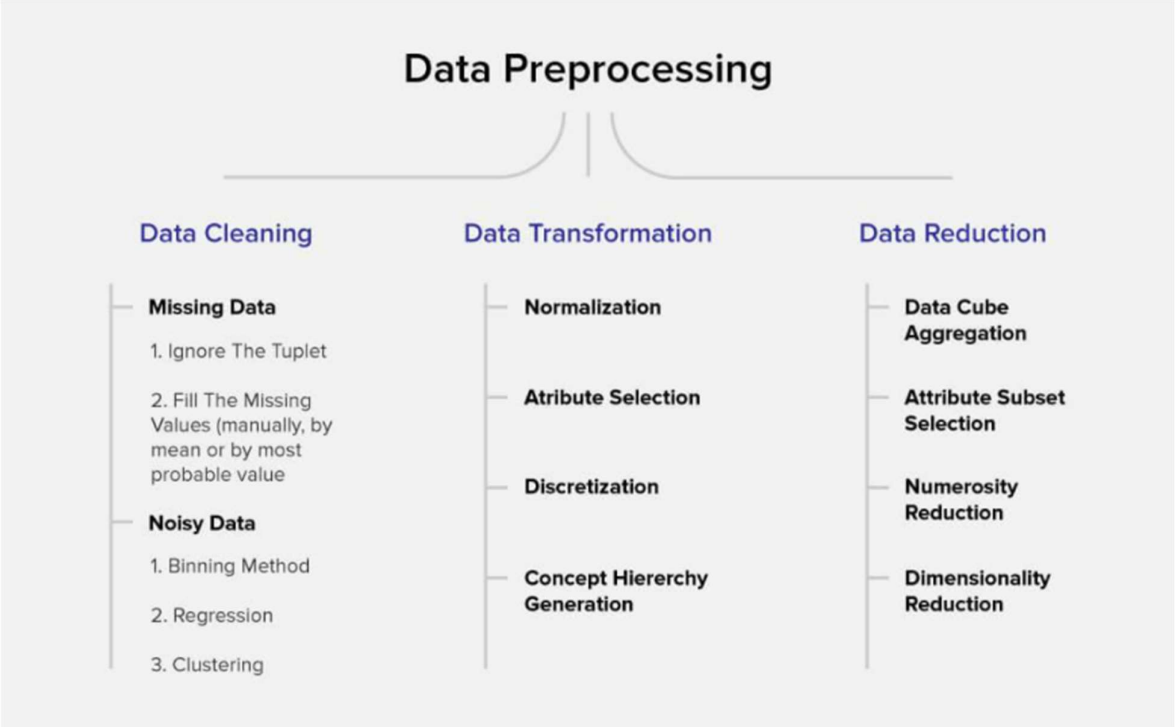
Data scientists can use exploratory analysis to ensure the results they produce are valid and applicable to any desired business outcomes and goals. EDA also helps stakeholders by confirming they are asking the right questions

TYPES OF EXPLORATORY DATA ANALYSIS:

- Univariate Non-graphical
- Multivariate Non-graphical
- Univariate graphical
- Multivariate graphical

DATA PRE-PROCESSING & FEATURE ENGINEERING:

Preprocessing simply refers to perform series of operations to transform or change data. It is transformation applied to our data before feeding it to algorithm. When creating a machine learning project, and doing any operation with data, it is mandatory to clean it and put in a formatted way. So for this, we use data preprocessing task.



Data pre-processing is a very vital input to machine learning models, It is to prepare the raw data & make it suitable for efficient machine learning model. These are the methods of data preprocessing and we are going to use the required ones in our project.

FEATURE ENGINEERING:

Feature engineering is the process of selecting, manipulating, and transforming raw data into features that can be used in supervised learning. In order to make machine learning work well on new tasks, it might be necessary to design and train better features. As you may know, a "feature" is any measurable input that can be used in a predictive model.
Feature engineering, in simple terms, is the act of converting raw observations into desired features using statistical or machine learning approaches. It can produce new features for both supervised and unsupervised learning, with the goal of simplifying and speeding up data transformations while also enhancing model accuracy.

Feature Engineering Techniques for Machine Learning

- Imputation
- Handling Outliers
- Log Transform
- One-hot encoding/Label Encoding
- Scaling

Data Transformation:

Label Encoding:

As we mentioned above in library installation, Label Encoder is used to encode labels by assigning them numbers. It is used to encode single or multiple columns. Thus, if the feature is color with values such as ['white', 'red', 'black', 'blue']., using Label Encoder may encode color string label as [0, 1, 2, 3]

Handling Outliers:

The most important phase in Feature Engineering is handling outliers because it ensures that our model is trained on accurate data which leads to accurate models. An outlier may occur due to the variability in the data. It may indicate an experimental error or heavy skewness in the data(heavy-tailed distribution). We have three measures of central tendency namely Mean, Median, and Mode. They help us describe the data.

Below are some of the techniques of detecting outliers

- Boxplots
- Z-score
- Quantile method
- Percentile method

Variance Inflation Factor (VIF)

Variance Inflation Factors (VIFs) measure the correlation among independent variables in least squares regression models. Statisticians refer to this type of correlation as multicollinearity. Excessive multicollinearity can cause problems for regression models. The stats models package has VIF library, Let us import the package.

SKEWNESS REMOVAL-(POWER-TRANSFORM):

Key step prior to initiating Machine learning models, optimizing, scaling the data to provide it as a input to start the modelling.

A power transform will make the probability distribution of a variable more Gaussian. This is often described as removing a skew in the distribution, although more generally is described as stabilizing the variance of the distribution. The log transform is a specific example of a family of transformations known as power transforms. The power_transform library present in the Sklearn. Pre-processing package.

MINMAX SCALER:

MinMax Scaler shrinks the data within the given range, usually of 0 to 1. It transforms data by scaling features to a given range. It scales the values to a specific value range without changing the shape of the original distribution.
Before scaling we have to train test split the data.since we have to do skewness removal and scaling only on input data.

TRAIN TEST SPLIT:

The scikit-learn Python machine learning library provides an implementation of the train-test split evaluation procedure via the train_test_split() function. The function takes a loaded dataset as input and returns the dataset split into two subsets.train_test_split() will split arrays data into random subsets. The ideal split is said to be 80:20 for training and testing.

PERCENTILE METHOD:

The IQR can then be calculated as the difference between the 75th and 25th percentiles. We can then calculate the cutoff for outliers as **1.5 times the IQR and subtract this cut-off from the** 25th percentile and add it to the 75th percentile to give the actual limits on the data. Use percentile-based approach. For Example, Data points that are **far from 99% percentile and less than 1 percentile** are considered an outlier.

Review of Literature

ABSTRACT:

Micro finance Solution has been seen very beneficial to unbanked poor families living in remote areas by providing them with Group, Agricultural, Individual business loans. We are working with a telecommunication client out of Indonesia, that is Telecom Industry (Indonesia). They understand the importance of communication & how it will affect a person's life in this modern era. They are providing a plan in tie-up with MFI. i.e. to provide micro-credit on mobile balances which needs to be paid back in 5 days. The Consumer is believed to be defaulter if he deviates this duration. Our purpose is to build a model which can enable them to predict the customers & decide on further investments & improvements with customer selection.

Motivation for the Problem Undertaken

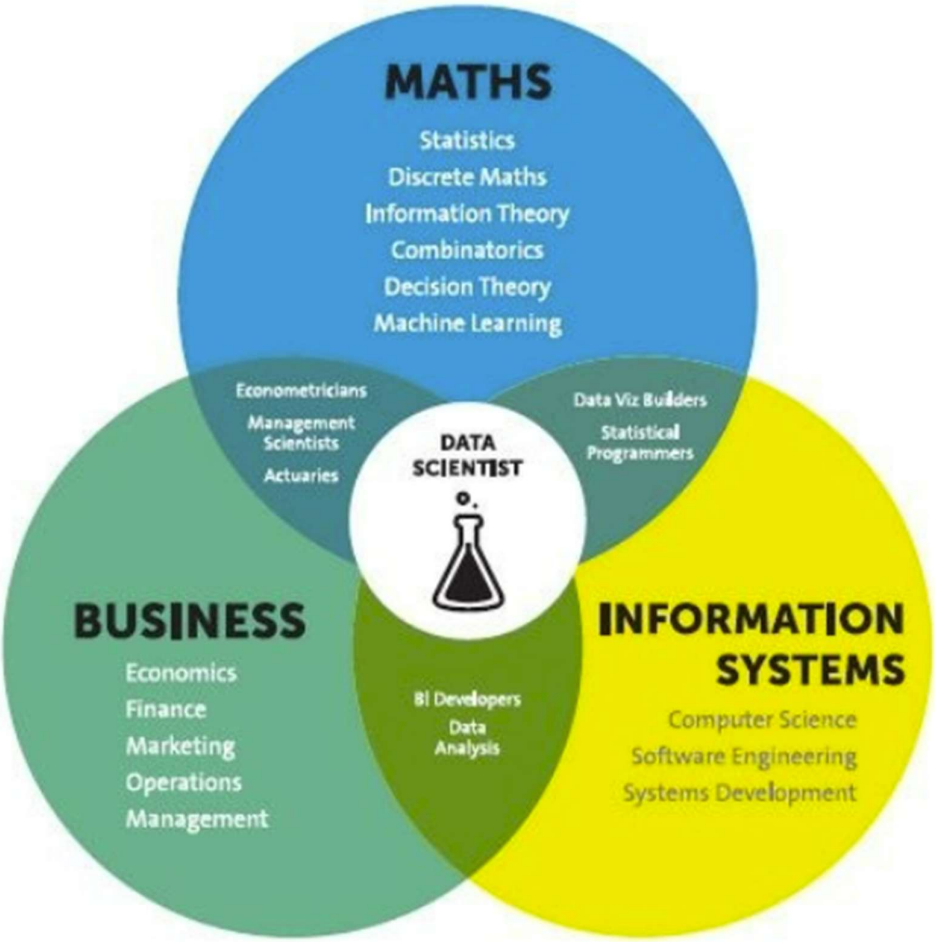
Business Goal:

We are provided with data from out client database with clear details on customer transactions done over such period. We need to build a model & understand the factors which defines defaulter or non-defaulter based on the criteria of repaying the loan amount in 5 days timeframe. This will help them to decide on further investment & improvements with customer selection.

Analytical Problem Framing

Mathematical/ Statistical /Analytical Modeling of the Problem

Mathematics, Statistics and Analytics are three of the most important concepts of Data Science. Data Science revolves around these three fields and draws their concepts to operate on the data.we will explore its practical usages in this field. So let's first explore how much these three are required for data science.



Mathematical Modelling

Mathematical models are important, selecting the right one to answer the business question can bring tremendous value to the organization. Machine Learning is a field that focuses on computers having the ability to learn/operate without being programmed to do so.

Mathematics is playing an essential role in the latest technologies like Machine Learning, Artificial Intelligence, Data Science and Deep Learning,

etc., It is because every algorithm built in the latest technologies has a mathematical function behind it and aid in identifying patterns.

The understanding of various notions of Statistics and Probability Theory are key for the implementation of such algorithms in data science. Notions include: Regression, Maximum Likelihood Estimation, the understanding of distributions (Binomial, Bernoulli, Gaussian (Normal)) and Bayes' Theorem.

The main reason for a greater significance of mathematics is because of its various concepts like: –

· Linear Algebra

· Probability

· Calculus

· Statistics

Linear Algebra & Calculus

Deep learning requires us to understand linear algebra & calculus, to understand how it works, for example forward propagation, backward propagation, parameters setting etc. For linear algebra, there are matrix operations (plus, minus, times, divide), scalar product, dot product, eigen-vectors and eigenvalues.

It is a branch of Mathematics for studying systems of equations. it can be one, two, and multi-dimensional equations. it helps us to solve numerical data or relations between two or more variables by establishing relations or equations between them. for example,

here' one basic algebraic equation:

$$y = a + bx + cx2$$

linear-algebra has a wide range of applications such as statics and matrices calculations, linear regression equations, descriptive statistics, graphic image vectors, Fourier series, graphs, and network establishment.

machine-learning algorithms like linear regression, logistic regression uses linear algebra to solve our target variables with given inputs/attributes or feature vectors given in the data set.

Calculus

Calculus is used essentially in optimization techniques. Using calculus, you can carry out mathematical modeling of artificial neural networks and also increase their accuracy and performance. For calculus, the data scientist need to understand various differentiation (to second-order derivative), integration, partial differentiation.

Differential Calculus

Differential Calculus studies the rate at which the quantities change. Derivates are most widely used for finding the maxima and minima of the functions. Derivates are used in optimization techniques where we have to find the minima in order to minimize the error function.

Integral Calculus

It is the mathematical study of the accumulation of quantities and for finding the area under the curve. Integrals are further divided into definite integrals and indefinite integrals.

Probability

The probability theory is very much helpful for making the prediction and Estimation.With the help of statistical methods,

we make estimates for the further analysis. Thus, statistical methods are largely dependent on the theory of probability.

Probability is a very important mathematical concept for data science, used in validating hypothesis, bayes theorem and interpreting outputs in machine learning.

Bases on these we try to estimate various events, and the likelihood of the outcome. sometimes we wat graphical representations of probable outcomes which we call probability density functions or density curves.

Concepts of probability help us estimate expected value from given variables, to solve confusion matrix in classification algorithms, information entropy, evidence of particular attributes in naive Bayes classification, and even in statistics for hypothesis testings.

Statistics

A statistical model is a mathematical representation (or mathematical model) of observed data. When data analysts apply various statistical models to the data they are investigating, they are able to understand and interpret the information more strategically.

So the areas in statistics are simple statistics like measurement of centrality, distributions and different probability distributions (Weibull, Poisson etc), Baye's Theorem

statistics is divided into two –

- Descriptive Statistics
- Inferential Statistics

Descriptive Statistics

Descriptive Statistics or summary statistics is used for describing the data. It deals with the quantitative summarization of data. This summarization is performed through graphs or numerical representations.

Descriptive Statistics:

1) Mean, Median, Mode

2) IQR, percentiles

3) Std deviation and Variance

4) Normal Distribution

5) Z-statistics and T-statistics

6) correlation and linear regression

Inferential Statistics

It is the procedure of inferring or concluding from the data. Through inferential statistics, we make a conclusion about the larger population by running several tests and deductions from the smaller sample.

Inferential Statistics:

1) Sampling distributions

2) confidence interval

3) chi-square test

4) Advanced regression

5) ANOVA

The mathematical concepts noted above are key in understanding/implementing the following Machine Learning techniques.

- Supervised learning, including regression and classification models.
- Unsupervised learning, including clustering algorithms and association rules.

Regression Models

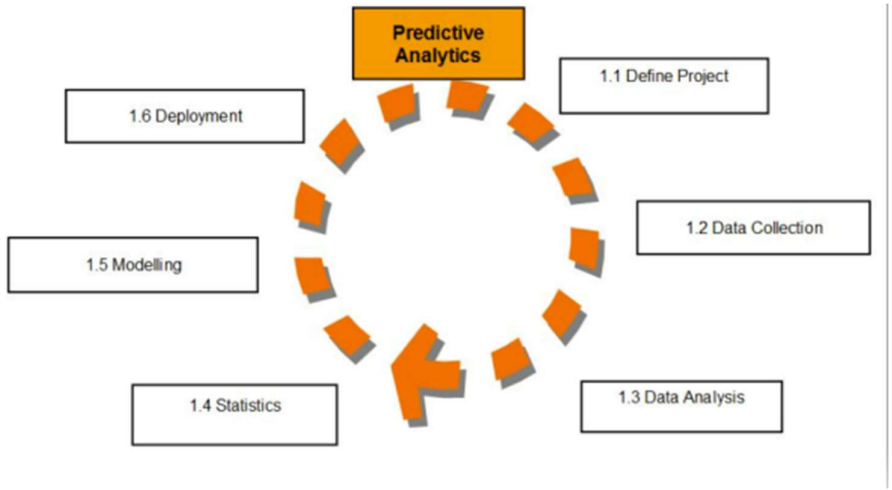
Data analysts use **regression models** to examine relationships between variables. Regression models are often used by organizations to determine which independent variables hold the most influence over dependent variables—information that can be leveraged to make essential business decisions.

Classification Models

Classification is a process in which an algorithm is used to analyze an existing data set of known points. The understanding achieved through that analysis is then leveraged as a means of appropriately classifying the data. Classification is a form of machine learning that can be particularly helpful in analyzing very large, complex sets of data to help make more accurate predictions.

Analytical Models:

An analytical model estimates or classifies data values by essentially drawing a line through data points. When applied to new data or records, a model can predict outcomes based on historical patterns.



. An analytical model is quantitative in nature, and used to answer a specific question or make a specific design decision. Different analytical models are used to address different aspects of the system, such as its performance, reliability, or mass properties.Data analysis comes with the fundamental types of data analytics encounter in data science: Descriptive, Diagnostic, Predictive, and Prescriptive.

- Descriptive analytics is a statistical method that is used to search and summarize historical data in order to identify patterns or meaning.
- Descriptive analysis is often used when reviewing any past or present data. This is because raw data is difficult to consume and interpret, while the metrics offered by descriptive analysis are much more focused.
- The example of descriptive statistics or analytics is to calculate the mean, median mode, standard deviation, and similar kinds of statistical calculation on finance or sales data.

- Diagnostic analytics takes it a step further to uncover the reasoning behind certain results. Diagnostic analytics is usually performed using such techniques as data discovery, drill-down, data mining, and different type of bivariant data analysis like correlations.etc.,
- Predictive Analytics is a **statistical method that utilizes algorithms and machine learning to identify trends in data and predict future behaviors**. Predictive Analytics can take both past and current data and offer predictions of what could happen in the future.
- Predictive models typically utilize variability in data to make the correct prediction and more variability of ingredient data that shows the relationship with what is possible to predict that united together into a prediction or valid score.
- Prescriptive analytics automatically synthesizes big data, mathematical sciences, business rules, algorithms, and machine learning to make predictions and then suggests decision options to take advantage of the predictions. Prescriptive means (optimization and simulation).

Data Sources and their formats

Technical Requirements:

- There are no null values in the dataset.
- 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.
- For some features, there may be values which might not be realistic. We may have to observe them and treat them with a suitable explanation.
- We might come across outliers in some features which we need to handle as per our understanding. Keep in mind that data is expensive and we cannot lose more than 7-8% of the data.
- Data contains 209593 entries each having 37 variables.
- Data set doesn't contains Null values. We treated them using the domain knowledge and our own understanding.
- Extensive EDA has been performed to gain relationships of important variable and labels.
- Data contains one numerical and all others as categorical variable. We handled them accordingly.
- We built Machine Learning models, applied regularization and determined the optimal values of Hyper Parameters.
- We found important features which affect the labels positively or negatively.
- The dataset is enclosed in notebook file.
- The dataset is provided to us by FlipRobo Technologies.And the dataset is in excel file format.

Data Description:

Following are the descriptions for data variables.

```
In [235]: df.columns #checking column names

Out[235]: Index(['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_rech30', 'fr_ma_rech30',
                 'sumamnt_ma_rech30', 'medianamnt_ma_rech30', 'medianmarechprebal30',
                 'cnt_ma_rech90', 'fr_ma_rech90', 'sumamnt_ma_rech90',
                 'medianamnt_ma_rech90', 'medianmarechprebal90', 'cnt_da_rech30',
                 'fr_da_rech30', 'cnt_da_rech90', 'fr_da_rech90', 'cnt_loans30',
                 'amnt_loans30', 'maxamnt_loans30', 'medianamnt_loans30', 'cnt_loans90',
                 'amnt_loans90', 'maxamnt_loans90', 'medianamnt_loans90', 'payback30',
                 'payback90', 'pcircle', 'pdate'],
                dtype='object')
```

Shape of the dataset is

In [234]:
```
df.shape
```

Out[234]:
```
(209593, 37)
```

Here pdate,msisdn are the categorical ordinal data type column and pcircle is the categorical nominal data type. And all other columns are having continous type values.Our target column is label and it is having categorical nominal data type.Hence it is a Classification Problem.

There is only two unique values in target column so its a Binary classification problem.

Attributes Information

In [236]:
```
import pandas as pd
df1 = pd.read_excel (r'C:\Users\Srividya\Downloads\Micro-Credit-Project--3- (1)\Micro Credit Project\Data_Description.xlsx', shee
```

In [237]:
```
info=pd.DataFrame({})
info=df1
info
```

Out[237]:

	Variable	Definition	Comment
0	label	Flag indicating whether the user paid back the...	NaN
1	msisdn	mobile number of user	NaN
2	aon	age on cellular network in days	NaN
3	daily_decr30	Daily amount spent from main account, averaged...	NaN
4	daily_decr90	Daily amount spent from main account, averaged...	NaN
5	rental30	Average main account balance over last 30 days	Unsure of given definition
6	rental90	Average main account balance over last 90 days	Unsure of given definition
7	last_rech_date_ma	Number of days till last recharge of main account	NaN
8	last_rech_date_da	Number of days till last recharge of data account	NaN
9	last_rech_amt_ma	Amount of last recharge of main account (in In...	NaN
10	cnt_ma_rech30	Number of times main account got recharged in ...	NaN
11	fr_ma_rech30	Frequency of main account recharged in last 30...	Unsure of given definition
12	sumamnt_ma_rech30	Total amount of recharge in main account over ...	NaN
13	medianamnt_ma_rech30	Median of amount of recharges done in main acc...	NaN

14	medianmarechprebal30	Median of main account balance just before rec...	NaN
15	cnt_ma_rech90	Number of times main account got recharged in ...	NaN
16	fr_ma_rech90	Frequency of main account recharged in last 90...	Unsure of given definition
17	sumamnt_ma_rech90	Total amount of recharge in main account over ...	NaN
18	medianamnt_ma_rech90	Median of amount of recharges done in main acc...	NaN
19	medianmarechprebal90	Median of main account balance just before rec...	NaN
20	cnt_da_rech30	Number of times data account got recharged in ...	NaN
21	fr_da_rech30	Frequency of data account recharged in last 30...	NaN
22	cnt_da_rech90	Number of times data account got recharged in ...	NaN
23	fr_da_rech90	Frequency of data account recharged in last 90...	NaN
24	cnt_loans30	Number of loans taken by user in last 30 days	NaN
25	amnt_loans30	Total amount of loans taken by user in last 30...	NaN
26	maxamnt_loans30	maximum amount of loan taken by the user in la...	There are only two options: 5 & 10 Rs., for wh...
27	medianamnt_loans30	Median of amounts of loan taken by the user in...	NaN
28	cnt_loans90	Number of loans taken by user in last 90 days	NaN
29	amnt_loans90	Total amount of loans taken by user in last 90...	NaN
30	maxamnt_loans90	maximum amount of loan taken by the user in la...	NaN
31	medianamnt_loans90	Median of amounts of loan taken by the user in...	NaN
32	payback30	Average payback time in days over last 30 days	NaN
33	payback90	Average payback time in days over last 90 days	NaN
34	pcircle	telecom circle	NaN
35	pdate	date	NaN

DATA ACQUISITION

In [232]:
```
#stroing into csv file
df = pd.read_csv(r'C:\Users\Srividya\Downloads\Micro-Credit-Project--3- (1)\Micro Credit Project\Data file.csv')
df.to_csv("Micro-Credit Defaulter Model.csv",sep='\t')
```

In [233]:
```
#create dataframe
df
```

Out[233]:

	Unnamed: 0	label	msisdn	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_date_da	...	maxamnt_loans30	r
0	1	0	21408I70789	272.0	3055.050000	3065.150000	220.13	260.13	2.0	0.0	...	6.0	
1	2	1	76462I70374	712.0	12122.000000	12124.750000	3691.26	3691.26	20.0	0.0	...	12.0	
2	3	1	17943I70372	535.0	1398.000000	1398.000000	900.13	900.13	3.0	0.0	...	6.0	
3	4	1	55773I70781	241.0	21.228000	21.228000	159.42	159.42	41.0	0.0	...	6.0	
4	5	1	03813I82730	947.0	150.619333	150.619333	1098.90	1098.90	4.0	0.0	...	6.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
209588	209589	1	22758I85348	404.0	151.872333	151.872333	1089.19	1089.19	1.0	0.0	...	6.0	

FEATURE DESCRIPTION:

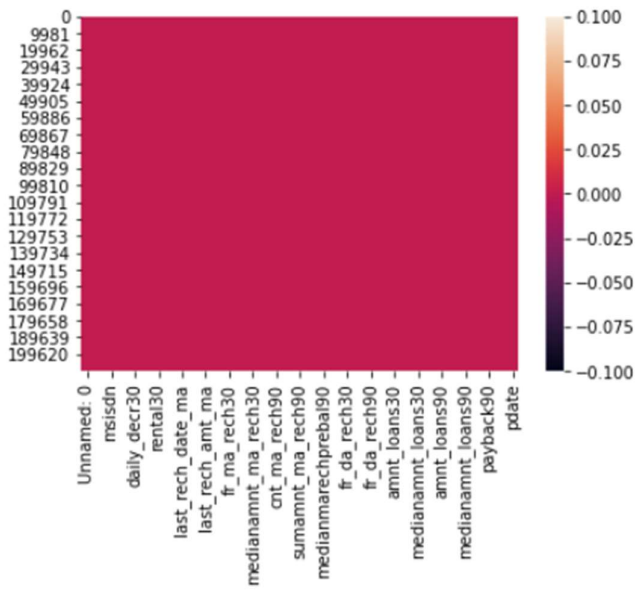
Following are the features and their data type details,

```
In [238]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 209593 entries, 0 to 209592
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
```

This info() method gives the information about the dataset which includes indexing type, column type, no-null values and memory usage.The dataset contains 3 different types of data, integer, object and float. There are 3 columns with object datatype.Here all the columns are numeric excecpt pcircle,pdate and msisdn.pcircle and msisdn features are object datatype and pdate feature is in datetime datatype.The dataset consist of 209593 rows and 36 columns. We will encode the object datatypes using appropriate encoding techniques before building machine learning models.

```
In [240]: #visualizing the null values using heatmap
          sns.heatmap(df.isnull())

Out[240]: <AxesSubplot:>
```



It is clear from the heat map that there are no null values in any of the column.

Exploratory Data Analysis and Preprocessing Done

```
In [14]: df.isnull().sum().any()

Out[14]: False
```

There are no null values in the dataset.

In [241]:
```
# Checking number of unique values in each column of dataset
df.nunique().to_frame("No of Unique Values")
```

Out[241]:

	No of Unique Values
Unnamed: 0	209593
label	2
msisdn	186243
aon	4507
daily_decr30	147025
daily_decr90	158669
rental30	132148
rental90	141033
last_rech_date_ma	1186
last_rech_date_da	1174
last_rech_amt_ma	70
cnt_ma_rech30	71
fr_ma_rech30	1083
sumamnt_ma_rech30	15141
medianamnt_ma_rech30	510
medianmarechprebal30	30428
cnt_ma_rech90	110
fr_ma_rech90	89
sumamnt_ma_rech90	31771
medianamnt_ma_rech90	608
medianmarechprebal90	29785
cnt_da_rech30	1066
fr_da_rech30	1072
cnt_loans30	40
amnt_loans30	48
maxamnt_loans30	1050
medianamnt_loans30	6
cnt_loans90	1110
amnt_loans90	69
maxamnt_loans90	3
medianamnt_loans90	6
payback30	1363
payback90	2381
pcircle	1
pdate	82

Above are the number of unique values present in the columns of the dataset.

In [242]:
```
# Checking the uniqueness of target column
print("The unique value present in label is:",df['label'].unique())
```

```
The unique value present in label is: [0 1]
```

In [243]:
```
round(df['label'].value_counts(normalize=True)*100,2)
```

Out[243]:
```
1    87.52
0    12.48
Name: label, dtype: float64
```

From the above we can see that it is an imbalanced Dataset.

There are only two unique values present in the label.No of defaulter's percentage is very low and it is only 12.48% in overall

In [244]:
```
# Checking value count of the label
print("The value count of the label is:\n",df["label"].value_counts())
```

```
The value count of the label is:
 1    183431
0     26162
Name: label, dtype: int64
```

This gives the list of values in the label. As we can see Label '1' indicates that the loan has been payed i.e. Non- defaulter which has 183431 values, while, Label '0' indicates that the loan has not been payed i.e. defaulter which has 26162 values. So from this we can notice there is a class imbalance issue exists which need to be balanced in the later part.

In [245]:
```
# Checking value count of pcircle
print("Value count of pcircle:",df["pcircle"].value_counts())
```

```
Value count of pcircle: UPW    209593
Name: pcircle, dtype: int64
```

Number of unique value in pcircle: 1

Here pcircle column contains only one unique value so we can drop that column because it seems that the dataset have contains only one circle area data. So it have not any impact in our model if we drop this feature.

Note:

pdate variable needs to be changed to datetime format to extract the useful values.Then we can remove pdate column "Unnamed: 0" contains only index values and "msisdn" contains mobile number of the user.
Unnamed: 0 and msisdn are there only to locate the indivisual observation. Unnamed: 0, pcircle, and msisdn are not necessary for our prediction. Whereas pcircle has only one value in all the observation, there is no variance so we can remove all of this

In [246]:
```
df.drop('Unnamed: 0',axis=1,inplace=True)
```

Since Index value is already available to represent the total number of data.

Date Type conversion to Datetime format

The column "pdate" has object data type which is in the form of dd/mm/yy so we need to convert it into datetime type. And we will extract day, month and year from the column "pdate".

In [247]:
```
df['pdate']=pd.to_datetime(df['pdate'])
df['Year']=df['pdate'].dt.year
df['Month']=df['pdate'].dt.month
df['Day']=df['pdate'].dt.day
```

In [248]:
```
#Checking the number of months
df['Month'].unique()
```

Out[248]:
```
array([7, 8, 6], dtype=int64)
```

In [249]:
```
df[['Day','Month','Year']]
```

Out[249]:

	Day	Month	Year
0	20	7	2016
1	10	8	2016
2	19	8	2016
3	6	6	2016
4	22	6	2016
...	...	...	...
209588	17	6	2016
209589	12	6	2016
209590	29	7	2016
209591	25	7	2016
209592	7	7	2016

209593 rows × 3 columns

In [250]:
```
# checkig the unique value present in column 'Year'
print("No of unique values present in Year column:",df['Year'].nunique())
# Checking value counts of Year column
print("\nValue count of Year column:",df["Year"].value_counts())
```

```
No of unique values present in Year column: 1

Value count of Year column: 2016    209593
Name: Year, dtype: int64
```

As we can see there is only one unique value that means all user data present in the dataset is collected in the same year 2016. We can also observe the value counts of the column Year.We can drop this column 'Year' since it won't affect our predictions.

In [251]:
```
df.drop('msisdn',axis=1,inplace=True)
df.drop('pdate',axis=1,inplace=True)
df.drop('pcircle',axis=1,inplace=True)
df.drop('Year',axis=1,inplace=True)
```

"pdate" is dropped,as the date column has been converted to integer and seperate date month and year columns created.We have successfully extracted day, month and year from pdate column and dropped pdate feature after extraction. "pcircle" is also dropped, as it contain only one value which will not effect the data set much "msisdn" is dropped as it only points to individual observations. "Year" is dropped as the it contains same value in all the rows.

In [252]:
```
#Lets check for duplicate values if there are any
df.duplicated().sum()
```

Out[252]:
```
31
```

There are duplicate entries present in the dataset.

31 rows of data are duplicate

There are only 209562 unique records. Assumption: It is possible because same customer might taken loan multiple times in the same month or in different months of year.One time he might be paid in time, and other time he might not paid in time, etc.Each time he applied for loan will be evaluated and stored into different row.So let us keep the duplicate customers.

Handling zero values¶

In [253]:
```
Zero_value = pd.DataFrame(df.isin([0]).sum().sort_values(ascending=False))
Zero_value.columns=['Count of Zero values']
Zero_value['% of Zero values']=Zero_value['Count of Zero values']/2095.93    # 209593/100 = 2095.93
Zero_value
```

Out[253]:

	Count of Zero values	% of Zero values
fr_da_rech90	208728	99.587295
fr_da_rech30	208014	99.246635
cnt_da_rech30	205479	98.037148
cnt_da_rech90	204226	97.439323
last_rech_date_da	202861	96.788061
medianamnt_loans90	197424	94.193985
medianamnt_loans30	195445	93.249775
payback30	106712	50.913914
payback90	95699	45.659445
fr_ma_rech30	78683	37.540853
fr_ma_rech90	65753	31.371754
medianmarechprebal30	30680	14.637893
cnt_ma_rech30	27979	13.349205
medianamnt_ma_rech30	27979	13.349205
sumamnt_ma_rech30	27979	13.349205
label	26162	12.482287
medianmarechprebal90	23391	11.160201
last_rech_amt_ma	20995	10.017033
medianamnt_ma_rech90	20950	9.995563
sumamnt_ma_rech90	20950	9.995563

cnt_ma_rech90	20950	9.995563
last_rech_date_ma	20743	9.896800
rental30	7566	3.609853
rental90	6918	3.300683
daily_decr30	4144	1.977165
daily_decr90	4063	1.938519
cnt_loans30	3259	1.554918
amnt_loans30	3259	1.554918
maxamnt_loans30	3244	1.547762
amnt_loans90	2043	0.974746
maxamnt_loans90	2043	0.974746
cnt_loans90	2036	0.971406
aon	0	0.000000
Month	0	0.000000
Day	0	0.000000

By looking into the above method of finding zero values we can observe the features

fr_da_rech90 fr_da_rech30 cnt_da_rech30 cnt_da_rech90 last_rech_date_da medianamnt_loans90 medianamnt_loans30

In all the above columns i found more than 90% zeros so they will create skewness in our dataset.

These are the column values which are having zero values more than 90% ,they will create skewness in our dataset and are not required for the predictions so dropping these columns.

In [254]:
```
# Dropping columns having more than 90% of zero values
df.drop(columns=["fr_da_rech90","fr_da_rech30","cnt_da_rech30","cnt_da_rech90","last_rech_date_da","medianamnt_loans90","medianar
```

In [255]:
```
# checkig the value count in column 'maxamnt_loans30' = maximum amount of loan taken by the user in last 30 days
df['maxamnt_loans30'].value_counts()
```

Out[255]:
```
6.000000        179193
12.000000        26109
0.000000          3244
94122.633158         1
59668.008360         1
                 ...
66749.081149         1
55723.858041         1
31545.936341         1
66847.875001         1
96775.751803         1
Name: maxamnt_loans30, Length: 1050, dtype: int64
```

We know that the maximum amount taken by the user in last 30 days.

There are only two options: 5 & 10 Rs., for which the user needs to pay back 6 & 12 Rs. respectively. Also we can see zero entries which means the user haven't taken any kind of loan. But from the value counts we can observe the data other than 6,12 and 0. So the data other than these values need to be converted into 0 assuming the user haven't taken any kind of loan, because there is no other amount to be return other than 6 & 12.

In [256]:
```
# Checking the values which have entries other than 6,12,0
df.loc[(df['maxamnt_loans30'] != 6.0) & (df['maxamnt_loans30'] != 12.0) & (df['maxamnt_loans30']!=0.0), 'maxamnt_loans30']
```

Out[256]:
```
118       61907.697372
125       22099.413732
146       98745.934048
369       58925.364061
374       78232.464324
              ...
209189    50824.996349
209262    17324.994582
209331    92864.501728
209392    54259.265687
209424    96927.243252
Name: maxamnt_loans30, Length: 1047, dtype: float64
```

We can observe out of 1050 rows there are 1047 rows having values other than 6,12 & 0. Let's convert them into 0 values.

In [257]:
```
# Converting the values having other than 6,12 & 0 into 0 values
df.loc[(df['maxamnt_loans30'] != 6.0) & (df['maxamnt_loans30'] != 12.0) & (df['maxamnt_loans30']!=0.0), 'maxamnt_loans30'] = 0
.0
```

Out[257]:
```
0.0
```

In [258]:
```
# Let's check the value count of the column maxamnt_loans30 after converting values into 0
df['maxamnt_loans30'].value_counts()
```

Out[258]:
```
6.0     179193
12.0     26109
0.0       4291
Name: maxamnt_loans30, dtype: int64
```

Now we have entries 6,12 and 0. We can observe there 4291 rows are having 0 values. That means user have not taken any kind of loan.

In [259]:
```
# Checking the value count in column 'maxamnt_loans90' = maximum amount of loans taken by the user in last 90 days
df['maxamnt_loans90'].value_counts()
```

Out[259]:
```
6     180945
12     26605
0       2043
Name: maxamnt_loans90, dtype: int64
```

The column maxamnt_loans90 has entries 0,6 and 12. Here we can observe there are 2043 rows having 0 values where the maximum amount of loan taken by the user in last 90 days. That means user have not taken any king of loan.

In [260]:
```
# Checking the value count in column 'amnt_loans30' = Total amount of loans taken by the user in last 30 days
df['amnt_loans30'].value_counts().head(15)
```

Out[260]:
```
6     76620
12    44384
18    26379
24    18403
30    11999
36     8559
42     5580
48     3994
0      3259
54     2660
60     2043
66     1402
72     1101
78      742
84      580
Name: amnt_loans30, dtype: int64
```

In [261]:
```
# Checking the value count in column 'amnt_loans90' = Total amount of loans taken by the user in last 90 days
df["amnt_loans90"].value_counts().head(15)
```

Out[261]:
```
6     69131
12    38908
18    23867
24    17216
30    12503
36     9589
42     7388
48     5627
54     4429
60     3503
66     2799
72     2254
0      2043
78     1730
84     1438
Name: amnt_loans90, dtype: int64
```

From the value counts of amnt_loans30 and amnt_loans90 we can notice there are 3259 and 2043 rows have zero values. Let's remove the zero values in the column amnt_loans90 as it gives the sum of loans taken by the user in 90 days (It covers loan taken by the user in 30 days as well).

In [262]:
```
#droping the rows from amnt_loans90 data where loan values are 0.
#droping the people who haven't taken any loan
df.drop(df.loc[(df['amnt_loans90']== 0.0)].index,inplace=True)
```

Droping people who haven't taken any loans as we don't have any use from them.

In [263]:
```
# Checking value counts after removing the rows containing 0 values
df["amnt_loans90"].value_counts().head(15)
```

Out[263]:
```
6     69131
12    38908
18    23867
24    17216
30    12503
36     9589
42     7388
48     5627
54     4429
60     3503
66     2799
72     2254
78     1730
84     1438
90     1184
Name: amnt_loans90, dtype: int64
```

In [264]:
```
# Checking shape of the dataset after dropping the rows
print('After Dropping rows new  shape is: ',df.shape)
```

```
After Dropping rows new  shape is:  (207550, 28)
```

In [265]:
```
# Checking data loss after removing rows having 0 values
data_loss = (209593-207550)/209593*100
data_loss
```

Out[265]:
```
0.9747462940079105
```

After removing the rows having zero entries we are losing only 0.97% of data which is acceptable.

Description of Dataset

In [266]:
```
# Statistical summary of dataset
df.describe().T
```

Out[266]:

	count	mean	std	min	25%	50%	75%	max
label	207550.0	0.873948	0.331908	0.000000	1.0000	1.000000	1.0000	1.000000
aon	207550.0	8095.156177	75605.569198	-48.000000	246.0000	527.000000	982.0000	999860.755168
daily_decr30	207550.0	5352.424286	9208.694592	-93.012667	41.7600	1414.400000	7200.0000	265926.000000
daily_decr90	207550.0	6044.967417	10902.815812	-93.012667	41.9795	1443.355000	7723.9975	320630.000000
rental30	207550.0	2674.303807	4272.953526	-23737.140000	278.1300	1074.880000	3330.5700	198926.110000
rental90	207550.0	3451.557712	5714.928132	-24720.580000	299.8300	1318.480000	4163.5700	200148.110000
last_rech_date_ma	207550.0	3744.288249	53813.277038	-29.000000	1.0000	3.000000	7.0000	998650.377733
last_rech_amt_ma	207550.0	2057.044751	2363.829442	0.000000	770.0000	1539.000000	2309.0000	55000.000000
cnt_ma_rech30	207550.0	3.991477	4.264318	0.000000	1.0000	3.000000	5.0000	203.000000
fr_ma_rech30	207550.0	3728.164201	53603.753070	0.000000	0.0000	2.000000	6.0000	999606.368132
sumamnt_ma_rech30	207550.0	7719.908547	10154.119795	0.000000	1543.0000	4629.000000	10013.0000	810096.000000
medianamnt_ma_rech30	207550.0	1809.560378	2065.621490	0.000000	770.0000	1539.000000	1924.0000	55000.000000
medianmarechprebal30	207550.0	3856.541717	54049.919466	-200.000000	11.0000	33.930000	83.0000	999479.419319
cnt_ma_rech90	207550.0	6.324110	7.203957	0.000000	2.0000	4.000000	9.0000	336.000000
fr_ma_rech90	207550.0	7.707916	12.594178	0.000000	0.0000	2.000000	8.0000	88.000000
sumamnt_ma_rech90	207550.0	12380.958497	16849.059437	0.000000	2317.0000	7218.000000	16000.0000	953036.000000
medianamnt_ma_rech90	207550.0	1856.325770	2071.485509	0.000000	773.0000	1539.000000	1924.0000	55000.000000
medianmarechprebal90	207550.0	91.041710	355.399706	-200.000000	14.6000	36.000000	79.0000	41456.500000
cnt_loans30	207550.0	2.786138	2.552263	0.000000	1.0000	2.000000	4.0000	50.000000
amnt_loans30	207550.0	18.128730	17.373116	0.000000	6.0000	12.000000	24.0000	306.000000
maxamnt_loans30	207550.0	6.689790	2.107794	0.000000	6.0000	6.000000	6.0000	12.000000
cnt_loans90	207550.0	18.610723	225.235874	1.000000	1.0000	2.000000	5.0000	4997.517944

amnt_loans90	207550.0	23.878150	26.495145	6.000000	6.0000	12.000000	30.0000	438.000000
maxamnt_loans90	207550.0	6.769116	2.005785	6.000000	6.0000	6.000000	6.0000	12.000000
payback30	207550.0	3.421201	8.796510	0.000000	0.0000	0.000000	3.8000	171.500000
payback90	207550.0	4.350039	10.294639	0.000000	0.0000	1.714286	4.5000	171.500000
Month	207550.0	6.787560	0.737225	6.000000	6.0000	7.000000	7.0000	8.000000
Day	207550.0	14.429159	8.421331	1.000000	7.0000	14.000000	21.0000	31.000000

The describe() method gives the statistical information of the dataset. The summary of this dataset is not perfect since there are some features have negative minimum values which are invalid values which we need to take care of by converting negative values into positive values. The following features have negative values:

aon - age on cellular network in days daily_decr30 - Daily amount spent from main account, averaged over last 30 days (in Indonesian Rupiah) daily_decr90 - Daily amount spent from main account, averaged over last 90 days (in Indonesian Rupiah) rental30 - Average main account balance over last 30 days rental90 - Average main account balance over last 30 days last_rech_date_ma - Number of days till last recharge of main account medianmarechprebal30 - Median of main account balance just before recharge in last 30 days at user level (in Indonesian Rupiah) medianmarechprebal90 - Median of main account balance just before recharge in last 90 days at user level (in Indonasian Rupiah)

From the above discription we see that almost our data is imbalanced, we have more counts of 1 than 0 in or target variable. Almost every independent variable is highly skewed to right and has outliers.And some kind of skewness can also be seen. aon shows negative value at minimum as age (in terms of days) cannot be negative, daily_decr30 and daily_decr90 also shows genative value at minimum that too needs to be corrected. last_rech_date_ma and last_rech_date_da - minimum value also here should be corrected.

I will convert negative value columns into absolute values to remove the negative value.

In [267]:
```
# Converting negative values present in the above features into positive values
df["aon"] = abs(df["aon"])
df["daily_decr30"] = abs(df["daily_decr30"])
df["daily_decr90"] = abs(df["daily_decr90"])
df["rental30"] = abs(df["rental30"])
df["rental90"] = abs(df["rental90"])
df["last_rech_date_ma"] = abs(df["last_rech_date_ma"])
df["medianmarechprebal30"] = abs(df["medianmarechprebal30"])
df["medianmarechprebal90"] = abs(df["medianmarechprebal90"])
```

In [268]:
```
# Checking statistical summary of the dataset after converting negative to positive
df.describe().T
```

In [268]:
```
# Checking statistical summary of the dataset after converting negative to positive
df.describe().T
```

Out[268]:

	count	mean	std	min	25%	50%	75%	max
label	207550.0	0.873948	0.331908	0.0	1.0000	1.000000	1.0000	1.000000
aon	207550.0	8095.625616	75605.518933	1.0	246.0000	527.000000	982.0000	999860.755168
daily_decr30	207550.0	5352.453575	9208.677568	0.0	41.7600	1414.400000	7200.0000	265926.000000
daily_decr90	207550.0	6044.996776	10902.799534	0.0	41.9835	1443.355000	7723.9975	320630.000000
rental30	207550.0	2697.321812	4258.460653	0.0	299.6900	1088.165000	3334.7500	198926.110000
rental90	207550.0	3477.845090	5698.968928	0.0	326.3400	1332.425000	4167.7625	200148.110000
last_rech_date_ma	207550.0	3744.568567	53813.257533	0.0	1.0000	3.000000	7.0000	998650.377733
last_rech_amt_ma	207550.0	2057.044751	2363.829442	0.0	770.0000	1539.000000	2309.0000	55000.000000
cnt_ma_rech30	207550.0	3.991477	4.264318	0.0	1.0000	3.000000	5.0000	203.000000
fr_ma_rech30	207550.0	3728.164201	53603.753070	0.0	0.0000	2.000000	6.0000	999606.368132
sumamnt_ma_rech30	207550.0	7719.908547	10154.119795	0.0	1543.0000	4629.000000	10013.0000	810096.000000
medianamnt_ma_rech30	207550.0	1809.560378	2065.621490	0.0	770.0000	1539.000000	1924.0000	55000.000000
medianmarechprebal30	207550.0	3858.821549	54049.756747	0.0	11.8000	35.000000	85.0000	999479.419319
cnt_ma_rech90	207550.0	6.324110	7.203957	0.0	2.0000	4.000000	9.0000	336.000000
fr_ma_rech90	207550.0	7.707916	12.594178	0.0	0.0000	2.000000	8.0000	88.000000
sumamnt_ma_rech90	207550.0	12380.958497	16849.059437	0.0	2317.0000	7218.000000	16000.0000	953036.000000
medianamnt_ma_rech90	207550.0	1856.325770	2071.485509	0.0	773.0000	1539.000000	1924.0000	55000.000000
medianmarechprebal90	207550.0	93.437946	354.777243	0.0	15.2500	37.000000	81.1000	41456.500000
cnt_loans30	207550.0	2.786138	2.552263	0.0	1.0000	2.000000	4.0000	50.000000
amnt_loans30	207550.0	18.128730	17.373116	0.0	6.0000	12.000000	24.0000	306.000000
maxamnt_loans30	207550.0	6.689790	2.107794	0.0	6.0000	6.000000	6.0000	12.000000
cnt_loans90	207550.0	18.610723	225.235874	1.0	1.0000	2.000000	5.0000	4997.517944
amnt_loans90	207550.0	23.878150	26.495145	6.0	6.0000	12.000000	30.0000	438.000000
maxamnt_loans90	207550.0	6.769116	2.005785	6.0	6.0000	6.000000	6.0000	12.000000
payback30	207550.0	3.421201	8.796510	0.0	0.0000	0.000000	3.8000	171.500000
payback90	207550.0	4.350039	10.294639	0.0	0.0000	1.714286	4.5000	171.500000
Month	207550.0	6.787560	0.737225	6.0	6.0000	7.000000	7.0000	8.000000
Day	207550.0	14.429159	8.421331	1.0	7.0000	14.000000	21.0000	31.000000

We have successfully converted negative values and the dataset looks perfect and have no invalid values.

From the above description we can observe the following things:

The counts of all the columns are same which means there are no missing values present in the dataset. In all the features, the mean value is greater than the median(50%) which means all the columns are skewed to right except the target column label which has mean value bit less than median that means it is skewed to left. By summarizing the data I can notice there is huge difference between maximum value and 75% percentile that leads to large number of outliers present in the dataset. We will remove these outliers using appropriate methods. From the above summary we can notice the mean value of label is 0.873948.

It contains only three month data.

Data Inputs- Logic- Output Relationships

In Classification, the output variable must be **a discrete value**.

In classification, **inputs are divided into two or more classes**, and the learner must produce a model that assigns unseen inputs to one (or multi-label classification) or more of these classes. This is typically tackled in a supervised way.

In machine learning, classification refers to **a predictive modeling problem where a class label is predicted for a given example of input data**. A classification model attempts to draw some conclusion from observed values. Given one or more inputs a classification model will try to predict the value of one or more outcomes. The output variables are often called **labels or categories**. ... A classification can have real-valued or discrete input variables. A problem with two classes is often called a two-class or binary classification problem. A problem with more than two classes is often called a multi-class classification problem. A classification algorithm, in general, is a function that weighs the input features so that the output **separates one class into positive values and the other into negative values**. Classification is a data mining function that **assigns items in a collection to target categories or classes**. The goal of classification is to accurately predict the target class for each case in the data. For example, a classification model could be used to identify loan applicants as low, medium, or high credit risks.

Classification analysis is a data analysis task within data-mining, **that identifies and assigns categories to a collection of data to allow for more accurate analysis**. ... Classification analysis can be used to question, make a decision, or predict behavior through the use of an algorithm.

DATA PREPROCESSING AND FEATURE ENGINEERING

Correlation with Heatmap:

The correlation coefficient is a statistical measure of the strength of the relationship between the relative movements of two variables. The values range between -1.0 and 1.0. A calculated number greater than 1.0 or less than -1.0 means that there was an error in the correlation measurement. A correlation of -1.0 shows a perfect negative correlation, while a correlation of 1.0 shows a perfect positive correlation. A correlation of 0.0 shows no linear relationship between the movement of the two variables.Correlation statistics can be used in finance and investing. Pearson correlation is the one most commonly used in statistics. This measures the strength and direction of a linear relationship between two variables.

It can also be defined as the measure of dependence between two different variables. If there are multiple variables and the goal is to find correlation between all of these variables and store them using appropriate data structure, the **matrix data structure** is used. Such matrix is called as **correlation matrix.**

In [290]:
```
#find correlation co.efficient of all variables in table
cor=df.corr()
cor
```

Out[290]:

	label	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_amt_ma	cnt_ma_rech30	fr_ma_rech
label	1.000000	-0.003900	0.168263	0.166017	0.057860	0.075098	0.003676	0.131744	0.239399	0.001
aon	-0.003900	1.000000	0.000987	0.000319	-0.000561	-0.000504	0.001818	0.003749	-0.002831	-0.001
daily_decr30	0.168263	0.000987	1.000000	0.977689	0.445827	0.464531	0.000014	0.273527	0.452708	-0.000
daily_decr90	0.166017	0.000319	0.977689	1.000000	0.438313	0.477283	0.000460	0.261557	0.428065	-0.000
rental30	0.057860	-0.000561	0.445827	0.438313	1.000000	0.955513	-0.001257	0.132898	0.239774	-0.001
rental90	0.075098	-0.000504	0.464531	0.477283	0.955513	1.000000	-0.001940	0.126216	0.237016	-0.000
last_rech_date_ma	0.003676	0.001818	0.000014	0.000460	-0.001257	-0.001940	1.000000	-0.000092	0.003980	-0.002
last_rech_amt_ma	0.131744	0.003749	0.273527	0.261557	0.132898	0.126216	-0.000092	1.000000	-0.000700	0.003
cnt_ma_rech30	0.239399	-0.002831	0.452708	0.428065	0.239774	0.237016	0.003980	-0.000700	1.000000	0.001
fr_ma_rech30	0.001274	-0.001077	-0.000623	-0.000437	-0.001283	-0.000606	-0.002067	0.003252	0.001394	1.000
sumamnt_ma_rech30	0.204252	0.000837	0.635692	0.603036	0.288577	0.274095	0.001934	0.442556	0.657067	0.000
medianamnt_ma_rech30	0.142047	0.003893	0.292393	0.280007	0.137194	0.126616	-0.001564	0.796149	-0.011597	-0.000
medianmarechprebal30	-0.004833	0.004045	-0.001173	-0.000768	-0.001036	-0.000842	0.004171	-0.002389	-0.000033	0.002
cnt_ma_rech90	0.237831	-0.002467	0.589230	0.594874	0.318113	0.352330	0.003948	0.017956	0.886799	0.000
fr_ma_rech90	0.084565	0.004268	-0.079029	-0.080269	-0.033735	-0.036867	0.001344	0.105883	-0.152790	-0.001
sumamnt_ma_rech90	0.206712	0.001096	0.762831	0.768453	0.354596	0.372562	0.002008	0.418604	0.585541	-0.000
medianamnt_ma_rech90	0.120616	0.004160	0.255064	0.247378	0.116462	0.107093	-0.000941	0.818551	-0.049341	-0.001
medianmarechprebal90	0.037722	-0.001128	0.034721	0.033351	0.027343	0.023614	-0.000991	0.125352	0.014821	-0.002
cnt_loans30	0.201600	-0.001604	0.373957	0.348302	0.189217	0.181793	0.001439	-0.024569	0.769089	0.002
amnt_loans30	0.202318	-0.001513	0.480214	0.456706	0.242995	0.242898	0.001135	0.011884	0.755241	0.002
maxamnt_loans30	0.087468	-0.000008	0.401807	0.396058	0.239718	0.256000	-0.000746	0.151791	0.168277	-0.000
cnt_loans90	0.004902	-0.000558	0.009180	0.009700	0.004435	0.005622	-0.000181	0.000159	0.014733	0.002
amnt_loans90	0.204055	-0.002142	0.572071	0.576183	0.307896	0.338620	0.001070	0.017058	0.687172	0.002
maxamnt_loans90	0.101247	-0.000499	0.435069	0.433017	0.264363	0.287473	-0.000468	0.168291	0.173439	-0.001
payback30	0.049668	0.002174	0.028252	0.020861	0.075283	0.069854	-0.002093	-0.026841	0.043146	0.001
payback90	0.050577	0.002456	0.048500	0.042248	0.097797	0.102944	-0.001420	-0.013628	0.016416	0.001
Month	0.151680	-0.002268	0.522744	0.543263	0.365532	0.430502	-0.001595	0.094308	0.162367	-0.002
Day	0.008241	0.000998	0.008077	-0.019652	0.040602	0.012918	0.000718	0.030666	0.068376	0.001

28 rows × 28 columns

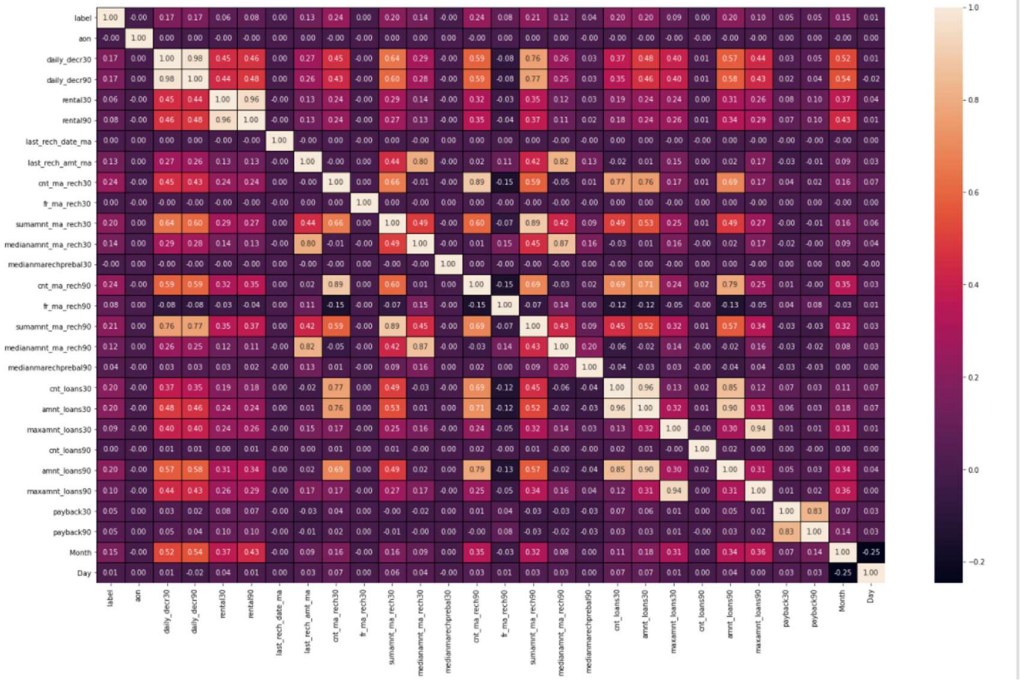
This gives the correlation between the denpendent and independent variables. We can visualize this by plotting heat map.

From the above we can cinclude that much of the variables have somewhat relations with each other. We will use Feature selection to select the best features.

Correlation heatmap is graphical representation of correlation matrix representing correlation between different variables.

For to do feature selection and make feature ready for the model building.we check correlation of variables using heatmap.And describe method for the census data set.

Out[291]: <AxesSubplot:>



This heatmap shows the correlation matrix of the data. We can observe the relation between one feature to other and relation between features and label. Here we can notice there is no strong relation between features and label. Dark shades are highly positively correlated with the label and light shades are highly negatively correlated with the label.

The features

sumamnt_ma_rech30: Total amount of recharge in main account over last 30 days (in Indonesian Rupiah) sumamnt_ma_rech90:Total amount of recharge in main account over last 90 days (in Indonesian Rupiah-) daily_decr30: Daily amount spent from main account, averaged over last 30 days (in Indonesian Rupiah) daily_decr90: Daily amount spent from main account, averaged over last 90 days (in Indonesian Rupiah) cnt_ma_rech30: Number of times main account got recharged in last 30 days cnt_ma_rech90: Number of times main account got recharged in last 90 days

cnt_loans30 & cnt_loans90: Number of loans taken by user in last 30 days & 90 days respectively. amnt_loans30 & amnt_loans90: Total amount of loans taken by user in last 30 days and 90 days

These features have somewhat strong correlation with the label of defaulters and non-defaulters data. Also, we can observe there are no negative correlation between label and features. Most of the features are correlated with each other.

1-daily_decr30 and daily_decr90 features are highly correlated with each other.

2-rental30 and rental90 features are highly correlated with each other.

3-cnt_loans30 and amount_loans30 columns are highly correlated with each other.

4-amount_loans30 is also highly correlated with amount_loans90 column.

5-medianamnt_loans30 and medianamnt_loans90 is highly correlated with each other.

6-We have to drop one of the features which are highly correlated with other feayures. And if we dont do this then our model will face multicolinearity problem.

```
In [292]: cor['label'].sort_values(ascending=False)
```

```
Out[292]: label                   1.000000
cnt_ma_rech30           0.239399
cnt_ma_rech90           0.237831
sumamnt_ma_rech90       0.206712
sumamnt_ma_rech30       0.204252
amnt_loans90            0.204055
amnt_loans30            0.202318
cnt_loans30             0.201600
daily_decr30            0.168263
daily_decr90            0.166017
Month                   0.151680
medianamnt_ma_rech30    0.142047
last_rech_amt_ma        0.131744
medianamnt_ma_rech90    0.120616
maxamnt_loans90         0.101247
maxamnt_loans30         0.087468
fr_ma_rech90            0.084565
rental90                0.075098
rental30                0.057860
payback90               0.050577
payback30               0.049668
medianmarechprebal90    0.037722
Day                     0.008241
cnt_loans90             0.004902
last_rech_date_ma       0.003676
fr_ma_rech30            0.001274
aon                    -0.003900
medianmarechprebal30   -0.004833
Name: label, dtype: float64
```

These are the most positively correlated column with the target column 'label'

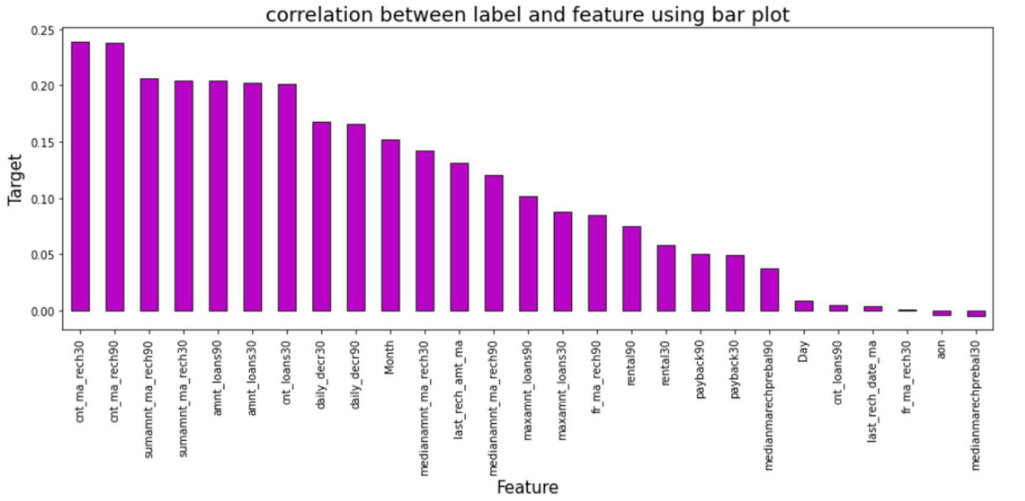
cnt_ma_rech30
cnt_ma_rech90
sumamnt_ma_rech90
sumamnt_ma_rech30
amnt_loans90
amnt_loans30
cnt_loans30

Correlation model:

Graph depicts clearly the positive and negative correlation of each variables with target column, justifies the outcome outlined in Multivariate analysis, that higher the education higher the gain & vice-versa

Visualizing the correlation between label and features

```
In [293]: plt.figure(figsize=(15,5))
          df.corr()['label'].sort_values(ascending=False).drop(['label']).plot(kind='bar',color='m',edgecolor=".2")
          plt.xlabel('Feature',fontsize=15)
          plt.ylabel('Target',fontsize=15)
          plt.title('correlation between label and feature using bar plot',fontsize=18)
          plt.show()
```



From the bar plot we can clearly observe the positive correlation between the label and features.

cnt_ma_rech30 cnt_ma_rech90 sumamnt_ma_rech90 sumamnt_ma_rech30 amnt_loans90 amnt_loans30 cnt_loans30

Are mostly positively correlated features with Target.

Here the column fr_ma_rech30 is less correlated with the label compared to others, we can drop these columns if necessary. We don't find any columns with negative correlation.

Hardware and Software Requirements and Tools Used

HARDWARE & SOFTWARE TOOLS, LIBRARIES AND PACKAGES USED:

Hardware :Intel i7,RAM 16GB used.

Software: Jupyter Notebook (Anaconda 3)

Language: Python

Libraries:

1. Pandas
2. Numpy
3. Matplotlib
4. Seaborn
5. Sklean
6. Scipy
7. Statsmodels
8. Pip-Package install Manager

```
#import all libaries
import pandas as pd
import numpy as np
import statistics
import seaborn as sns
from matplotlib import pyplot as plt
import sklearn
from sklearn.preprocessing import LabelEncoder,OneHotEncoder
import warnings
warnings.filterwarnings('ignore')
url="https://raw.githubusercontent.com/dsrscientist/dataset1/master/census_income.csv"
```

Category	Tool	Function
Data loading and analysis	Import pandas as pd	Pandas is a Python library that is used for faster data analysis, data cleaning and data pre-processing. Pandas is built on top of numpy. So, numpy gets some superpower with pandas. It offers data structures and operations for manipulating numerical tables and time series.
	Import numpy as np	NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.It has Quantile method too for removing outliers. It is the fundamental package for scientific computing with Python
Data visualization	Import matplotlib.pyplot as plt	Matplotlib is a plotting library used for data visualization.
	Import seaborn as sns	Seaborn is also a plotting library. It is more advanced than matplotlib but works with matplotlib
Scikit Learn Preprocessing Libraries	Sklearn.preprocessing	Package provides several common utility functions and transformer classes to change raw feature vectors into a representation that is more suitable for the downstream estimators.Has power transformer to remove skewness. In general, learning algorithms benefit from standardization of the data set. If some outliers are present in the set, robust scalers or transformers are more appropriate. It has MinMaxScaler to scale the data.
	Sklearn.preprocessing import LabelEncoder	Label Encoding in Python can be implemented using the Sklearn Library. Sklearn furnishes a very effective method for encoding the categories of categorical features into numeric values. Label encoder encodes labels with credit between 0 and n-1 classes where n is the number of diverse labels.
Import statistics	Import statsmodels.api as sm	From scipy import stats This module provides functions for calculating mathematical statistics of numeric (Real-valued) data. This library provides a number of common functions and types useful in statistics. It focus on high performance, numerical robustness, and use of good algorithms

In [55]:
```
#VIF calculation
import statsmodels.api as sm
from scipy import stats
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

Variance Inflation Factors (VIFs) measure the correlation among independent variables in least squares regression models. Statisticians refer to this type of correlation as multicollinearity. Excessive multicollinearity can cause problems for regression models. The stats models package has VIF library and we can import this library.

```
In [75]: #train test split
         from sklearn.model_selection import train_test_split

         x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=40)
```

The scikit-learn Python machine learning library provides an implementation of the train-test split evaluation procedure via the train_test_split() function. The function takes a loaded dataset as input and returns the dataset split into two subsets.train_test_split() will split arrays data into random subsets. The ideal split is said to be 80:20 for training and testing.

Many learning algorithms have been proposed. It is often valuable to assess the efficacy of an algorithm. In many cases, such assessment is relative, that is, evaluating which of several alternative algorithms is best suited to a specific application.

People even end up creating metrics that suit the application. In this article, we will see some of the most common metrics in a classification setting of a problem.

Choice of metrics influences how the performance of machine learning algorithms is measured and compared.

The most commonly used Performance metrics for classification problem are as follows,

- Accuracy.
- Confusion Matrix.
- Precision, Recall, and F1 score.
- ROC AUC.
- Log-loss.

Accuracy:

Accuracy is a good measure when the target variable classes in the data are nearly balanced. Accuracy in classification problems is the number of correct predictions made by the model over all kinds predictions made. It is the simple ratio between the number of correctly classified points to the total number of points.

Confusion Matrix:

It is used for Classification problem where the output can be of two or more types of classes. *For a sensible model, the principal diagonal element values will be high and the off-diagonal element values will be below i.e., TP, TN will be high.*

Precision:

Precision is the fraction of the correctly classified instances from the total classified instances. Precision helps us understand how useful the results are.

Recall or Sensitivity:

Recall is the fraction of the correctly classified instances from the total classified instances. Recall helps us understand how complete the results are.

F1 Score:

The F-score is often used in the field of information retrieval for measuring search, document classification, and query classification performance.

The F-score has been widely used in the natural language processing literature, such as the evaluation of named entity recognition and word segmentation

Specificity:

Specificity is the exact opposite of Recall.

Log Loss

Logarithmic loss (or log loss) measures the performance of a classification model where the prediction is a probability value between 0 and 1. Log loss increases as the predicted probability diverge from the actual label.

Lower the log-loss value, better are the predictions of the model.

ROC AUC

A Receiver Operating Characteristic curve or **ROC curve** is created by plotting the True Positive (TP) against the False Positive (FP) at various threshold settings. The ROC curve is generated by plotting the cumulative distribution function of the True Positive in the y-axis versus the cumulative distribution function of the False Positive on the x-axis.The area under the ROC curve (ROC AUC) is the single-valued metric used for evaluating the performance.

The higher the AUC, the better the performance of the model at distinguishing between the classes.

```
In [93]: from sklearn.linear_model import LogisticRegression
         from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
```

```
In [115]: # multi-class classification
          from sklearn.multiclass import OneVsRestClassifier
          from sklearn.metrics import roc_curve,auc
          from sklearn.metrics import roc_auc_score
```

```
#perform gridsearchcv and cross val score on LinearRegression
from sklearn.model_selection import GridSearchCV
```

Grid search is used as an approach to hyper-parameter tuning that will methodically build and evaluate a model for each combination of algorithm parameters specified in a grid. GridSearchCV helps us combine an estimator with a grid search preamble to tune hyper-parameters.

```
from sklearn.model_selection import cross_val_score
```

Cross-validation is a technique in which we train our model using the subset of the data-set and then evaluate using the complementary subset of the data-set.
The three steps involved in cross-validation are as follows :

1. Reserve some portion of sample data-set.
2. Using the rest data-set train the model.
3. Test the model using the reserve portion of the data-set.

```
In [87]: #importing and fitting the data to the pca
         from sklearn.decomposition import PCA
```

The most important use of PCA is to **represent a multivariate data table as smaller set of variables** (summary indices) in order to observe trends, jumps, clusters and outliers. This overview may uncover the relationships between observations and variables, and among the variables.

Also import all the required algorithms for classification purpose below.

Model/s Development and Evaluation

Identification of possible problem-solving approaches (methods)

Checking data types for encoding

In [294]:
```
df
```

Out[294]:

	label	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_amt_ma	cnt_ma_rech30	fr_ma_rech30	...	cnt_loans30	amı
0	0	272.0	3055.050000	3065.150000	220.13	260.13	2.0	1539	2	21.0	...	2	
1	1	712.0	12122.000000	12124.750000	3691.26	3691.26	20.0	5787	1	0.0	...	1	
2	1	535.0	1398.000000	1398.000000	900.13	900.13	3.0	1539	1	0.0	...	1	
3	1	241.0	21.228000	21.228000	159.42	159.42	41.0	947	0	0.0	...	2	
4	1	947.0	150.619333	150.619333	1098.90	1098.90	4.0	2309	7	2.0	...	7	
...	...	...	...	...	...	...	...	...	...	...	...	...	
209588	1	404.0	151.872333	151.872333	1089.19	1089.19	1.0	4048	3	2.0	...	2	
209589	1	1075.0	36.936000	36.936000	1728.36	1728.36	4.0	773	4	1.0	...	3	
209590	1	1013.0	11843.111667	11904.350000	5861.83	8893.20	3.0	1539	5	8.0	...	4	
209591	1	1732.0	12488.228333	12574.370000	411.83	984.58	2.0	773	5	4.0	...	2	
209592	1	1581.0	4489.362000	4534.820000	483.92	631.20	13.0	7526	2	1.0	...	2	

207550 rows × 28 columns

In features, the values which are not realistic like negative values,high range values,zero values and unwanted columns etc., we treated that and the data is ready for model now

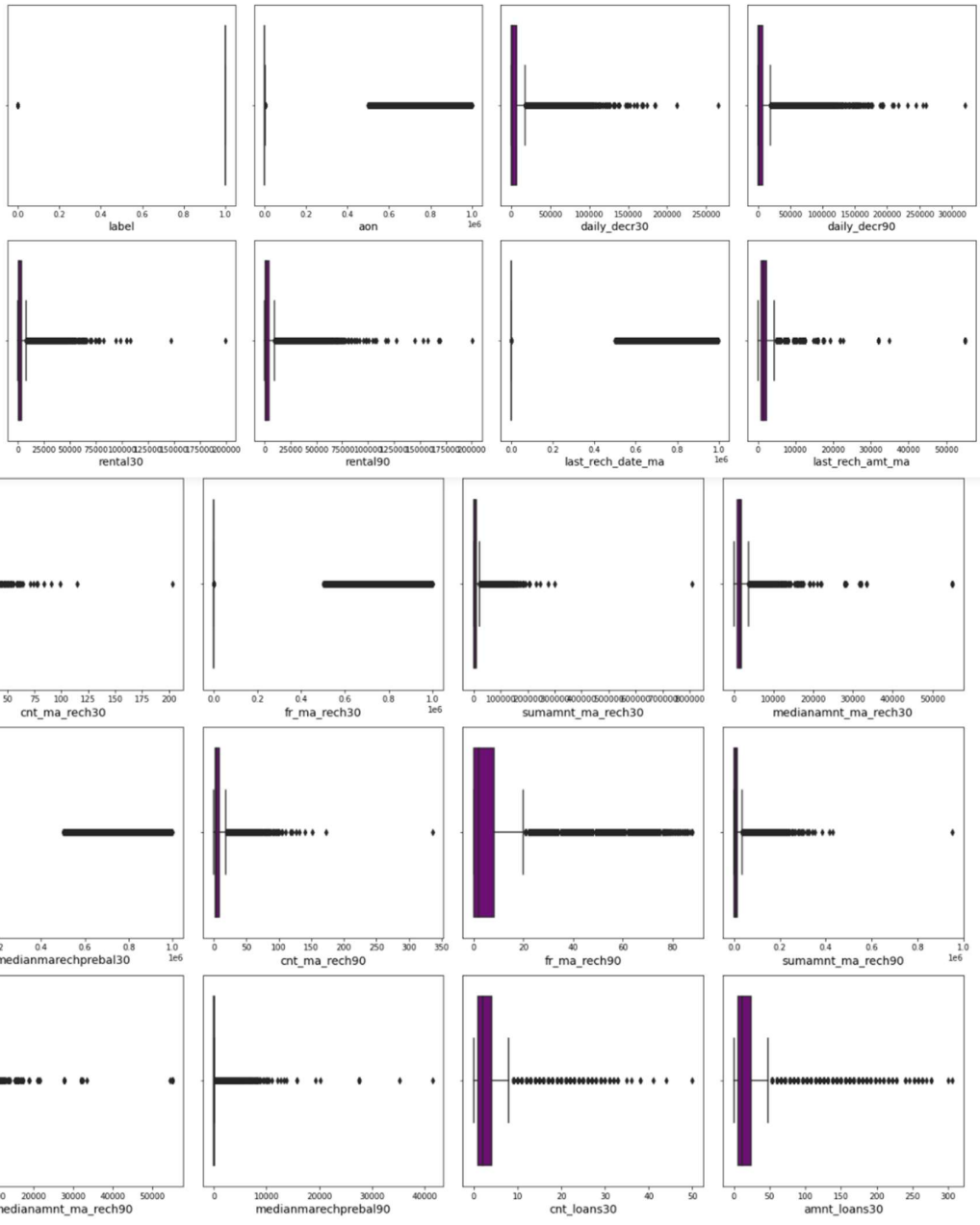
In [295]:
```
df.dtypes
```

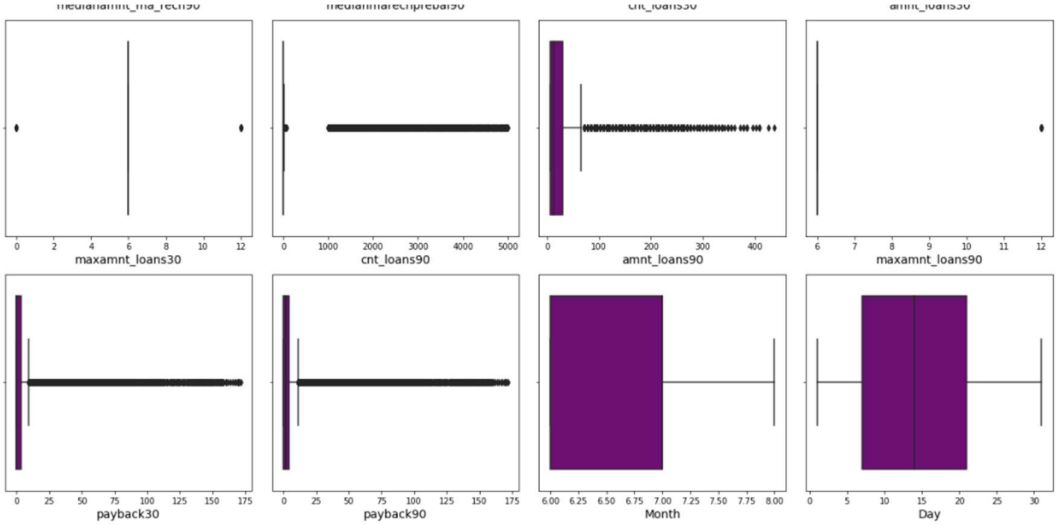
Out[295]:
```
label                   int64
aon                   float64
daily_decr30          float64
daily_decr90          float64
rental30              float64
rental90              float64
last_rech_date_ma     float64
last_rech_amt_ma        int64
cnt_ma_rech30           int64
fr_ma_rech30          float64
sumamnt_ma_rech30     float64
medianamnt_ma_rech30  float64
medianmarechprebal30  float64
cnt_ma_rech90           int64
fr_ma_rech90            int64
sumamnt_ma_rech90       int64
medianamnt_ma_rech90  float64
medianmarechprebal90  float64
cnt_loans30             int64
amnt_loans30            int64
maxamnt_loans30       float64
cnt_loans90           float64
amnt_loans90            int64
maxamnt_loans90         int64
payback30             float64
payback90             float64
Month                   int64
Day                     int64
dtype: object
```

Since there are no objec data types present.Hence no need to make Label Encoding

Identifying the outliers

In [296]:
```
# Identifying the outliers using boxplot
plt.figure(figsize=(18,30),facecolor='white')
plotnumber=1
for column in df:
    if plotnumber<=28:
        ax=plt.subplot(7,4,plotnumber)
        sns.boxplot(df[column],color="purple")
        plt.xlabel(column,fontsize=14)
    plotnumber+=1
plt.tight_layout()
```





From the above box plot we can notice the outliers present in all the features except Day and Month columns. Let's remove the outliers in these columns except Day, Month and label. Since label is our target columns we should not loose any data by removing outliers in this column.

Removing outliers

Zscore method

In [87]:
```
df
```

Out[87]:

	label	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_amt_ma	cnt_ma_rech30	fr_ma_rech30	...	cnt_loans30	amt
0	0	272.0	3055.050000	3065.150000	220.13	260.13	2.0	1539	2	21.0	...	2	
1	1	712.0	12122.000000	12124.750000	3691.26	3691.26	20.0	5787	1	0.0	...	1	
2	1	535.0	1398.000000	1398.000000	900.13	900.13	3.0	1539	1	0.0	...	1	
3	1	241.0	21.228000	21.228000	159.42	159.42	41.0	947	0	0.0	...	2	
4	1	947.0	150.619333	150.619333	1098.90	1098.90	4.0	2309	7	2.0	...	7	
...	...	...	...	...	...	...	...	...	...	...	...	...	
209588	1	404.0	151.872333	151.872333	1089.19	1089.19	1.0	4048	3	2.0	...	2	
209589	1	1075.0	36.936000	36.936000	1728.36	1728.36	4.0	773	4	1.0	...	3	
209590	1	1013.0	11843.111667	11904.350000	5861.83	8893.20	3.0	1539	5	8.0	...	4	
209591	1	1732.0	12488.228333	12574.370000	411.83	984.58	2.0	773	5	4.0	...	2	
209592	1	1581.0	4489.362000	4534.820000	483.92	631.20	13.0	7526	2	1.0	...	2	

207550 rows × 28 columns

In [297]:
```
# Feature containing outliers
features= df[['aon', 'daily_decr30', 'daily_decr90', 'rental30', 'rental90',
       'last_rech_date_ma', 'last_rech_amt_ma', 'cnt_ma_rech30',
       'fr_ma_rech30', 'sumamnt_ma_rech30', 'medianamnt_ma_rech30',
       'medianmarechprebal30', 'cnt_ma_rech90', 'fr_ma_rech90',
       'sumamnt_ma_rech90', 'medianamnt_ma_rech90', 'medianmarechprebal90',
       'cnt_loans30', 'amnt_loans30', 'maxamnt_loans30', 'cnt_loans90',
       'amnt_loans90', 'maxamnt_loans90', 'payback30', 'payback90']]
```

In [90]:
```
# Using zscore to remove outliers
from scipy.stats import zscore
```

In [91]:
```
z=np.abs(zscore(features))
# Creating new dataframe by setting z to 3
new_df = df[(z<3).all(axis=1)]
new_df
```

Out[91]:

	label	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_amt_ma	cnt_ma_rech30	fr_ma_rech30	...	cnt_loans30	amı
0	0	272.0	3055.050000	3065.150000	220.13	260.13	2.0	1539	2	21.0	...	2	
1	1	712.0	12122.000000	12124.750000	3691.26	3691.26	20.0	5787	1	0.0	...	1	
2	1	535.0	1398.000000	1398.000000	900.13	900.13	3.0	1539	1	0.0	...	1	
3	1	241.0	21.228000	21.228000	159.42	159.42	41.0	947	0	0.0	...	2	
4	1	947.0	150.619333	150.619333	1098.90	1098.90	4.0	2309	7	2.0	...	7	
...	...	...	...	...	...	...	...	...	...	...	...	...	
209588	1	404.0	151.872333	151.872333	1089.19	1089.19	1.0	4048	3	2.0	...	2	
209589	1	1075.0	36.936000	36.936000	1728.36	1728.36	4.0	773	4	1.0	...	3	
209590	1	1013.0	11843.111667	11904.350000	5861.83	8893.20	3.0	1539	5	8.0	...	4	
209591	1	1732.0	12488.228333	12574.370000	411.83	984.58	2.0	773	5	4.0	...	2	
209592	1	1581.0	4489.362000	4534.820000	483.92	631.20	13.0	7526	2	1.0	...	2	

170071 rows × 28 columns

This is the new dataframe after removing the outliers. Here we have removed the outliers whose Zscore is less than 3.

In [92]:
```
# Checking the shape of dataset before and after removing outliers
print("Shape of dataset before removing outliers:", df.shape)
print("Shape of dataset after removing outliers:", new_df.shape)
```

```
Shape of dataset before removing outliers: (207550, 28)
Shape of dataset after removing outliers: (170071, 28)
```

In [93]:
```
# Checking the the data loss after removing outliers
data_loss = (207550-170071)/207550*100
data_loss
```

Out[93]:
```
18.05781739339918
```

By using Zscore I am losing around 18% of data which is not acceptable. The acceptable range of data loss is 7-8%. So let's try to remove outliers using IQR method.

IQR (Inter Quartile Range) method

In [229]:
```
# 1st quantile
Q1=features.quantile(0.25)

# 3rd quantile
Q3=features.quantile(0.75)

# IQR
IQR=Q3 - Q1

df1=df[~((df < (Q1 - 1.5 * IQR)) |(df > (Q3 + 1.5 * IQR))).any(axis=1)]

print("Shape of data after using IQR method:", df1.shape)
```

```
Shape of data after using IQR method: (78654, 28)
```

In [230]:
```
# Checking the the data loss after removing outliers
data_loss = (207550-78654)/207550*100
data_loss
```

Out[230]:
```
62.103589496506864
```

By using Zscore and IQR method I am losing large amount of data, let's use percentile method to remove the outliers by setting the dataloss to 2%.

Percentile Method

In [298]:
```
# Checking the shape of data we have before treating of outliers.
print('before removing outliers shape was:', df.shape)
```

```
before removing outliers shape was: (207550, 28)
```

In [299]:
```
#Removing outliers using percentile method in train dataset
for col in features:
    if df[col].dtypes != 'object':
        percentile = df[col].quantile([0.01,0.98]).values
        df[col][df[col]<=percentile[0]]=percentile[0]
        df[col][df[col]>=percentile[1]]=percentile[1]
```

I have successfully removed the outliers present in dataset using percentile method.

In [300]:
```
# Checking the shape of data we have after treating of outliers.
print('after removing outliers shape is: ', df.shape)
```

```
after removing outliers shape is:  (207550, 28)
```

No data loss happened

Skewness Removal

In [301]:
```
# Checking for skewness in dataset
df.skew()
```

Out[301]:
```
label                 -2.253346
aon                    0.934791
daily_decr30           1.978547
daily_decr90           2.098290
rental30               2.117210
rental90               2.205817
last_rech_date_ma      2.565623
last_rech_amt_ma       2.016661
cnt_ma_rech30          1.410702
fr_ma_rech30           1.703431
sumamnt_ma_rech30      1.749207
medianamnt_ma_rech30   2.122065
medianmarechprebal30   2.799234
cnt_ma_rech90          1.566573
fr_ma_rech90           1.987801
sumamnt_ma_rech90      1.863681
medianamnt_ma_rech90   2.143777
medianmarechprebal90   2.631175
cnt_loans30            1.597669
amnt_loans30           1.752260
maxamnt_loans30        1.634976
cnt_loans90            2.000454
amnt_loans90           1.910837
maxamnt_loans90        2.224471
payback30              2.635055
payback90              2.826565
Month                  0.358219
Day                    0.184762
dtype: float64
```

Here we can observe the skewness present in all the columns except Day and Month. Since label is our target variable no need to remove skewness in this column as well. Lets remove the columns having skewness more than +0.5 & -0.5 using power transformation method(yeo-johnson method).

Removing skewness using yeo-johnson method

In [302]:
```
# Removing skewness using yeo-johnson  method to get better prediction
features= ['aon', 'daily_decr30', 'daily_decr90', 'rental30', 'rental90',
       'last_rech_date_ma', 'last_rech_amt_ma', 'cnt_ma_rech30',
       'fr_ma_rech30', 'sumamnt_ma_rech30', 'medianamnt_ma_rech30',
       'medianmarechprebal30', 'cnt_ma_rech90', 'fr_ma_rech90',
       'sumamnt_ma_rech90', 'medianamnt_ma_rech90', 'medianmarechprebal90',
       'cnt_loans30', 'amnt_loans30', 'maxamnt_loans30', 'cnt_loans90',
       'amnt_loans90', 'maxamnt_loans90', 'payback30', 'payback90']
```

In [303]:
```
from sklearn.preprocessing import PowerTransformer
scaler = PowerTransformer(method='yeo-johnson')
'''
parameters:
method = 'box-cox' or 'yeo-johnson'
'''
```

Out[303]:
```
"\nparameters:\nmethod = 'box-cox' or 'yeo-johnson'\n"
```

In [304]:
```
df[features] = scaler.fit_transform(df[features].values)
df[features]
```

Out[304]:

	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_amt_ma	cnt_ma_rech30	fr_ma_rech30	sumamnt_ma_rech30	.
0	-0.698192	0.502635	0.480196	-0.815119	-0.801254	-0.334457	0.085229	-0.297080	1.714875	-0.194777	.
1	0.387246	1.123583	1.076317	0.794042	0.630619	1.542566	1.549424	-0.817015	-1.126969	0.218868	.
2	0.041209	0.187550	0.173246	-0.123498	-0.219266	-0.005757	0.085229	-0.817015	-1.126969	-0.562266	.
3	-0.819499	-1.120125	-1.117515	-0.947896	-0.995637	2.044467	-0.297189	-1.677889	-1.126969	-1.901468	.
4	0.754265	-0.582302	-0.581745	-0.008841	-0.112338	0.235710	0.461086	1.012640	0.158382	1.322032	.
...	...	...	...	...	...	...	...	...	...	...	.
209588	-0.278836	-0.579762	-0.579232	-0.014038	-0.117181	-0.833241	1.082242	0.079341	0.158382	0.685801	.
209589	0.924703	-0.980848	-0.978014	0.268899	0.145770	0.235710	-0.438006	0.375692	-0.272800	-0.192092	.
209590	0.844267	1.112351	1.067842	1.155109	1.285978	-0.005757	0.085229	0.620705	1.117370	0.592674	.
209591	1.608771	1.137984	1.093184	-0.531384	-0.171704	-0.334457	-0.438006	0.620705	0.639283	0.824974	.
209592	1.472514	0.667319	0.642136	-0.452366	-0.399816	1.212794	1.933945	-0.297080	-0.272800	0.568129	.

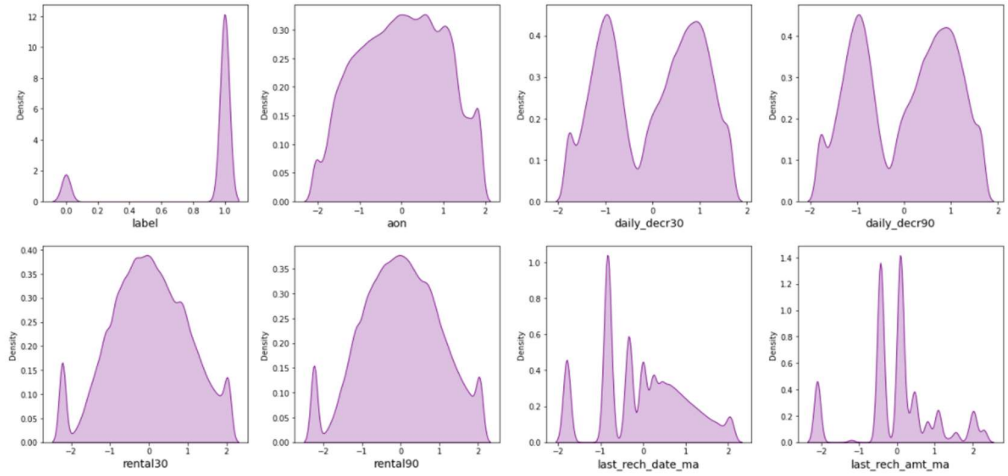
207550 rows × 25 columns

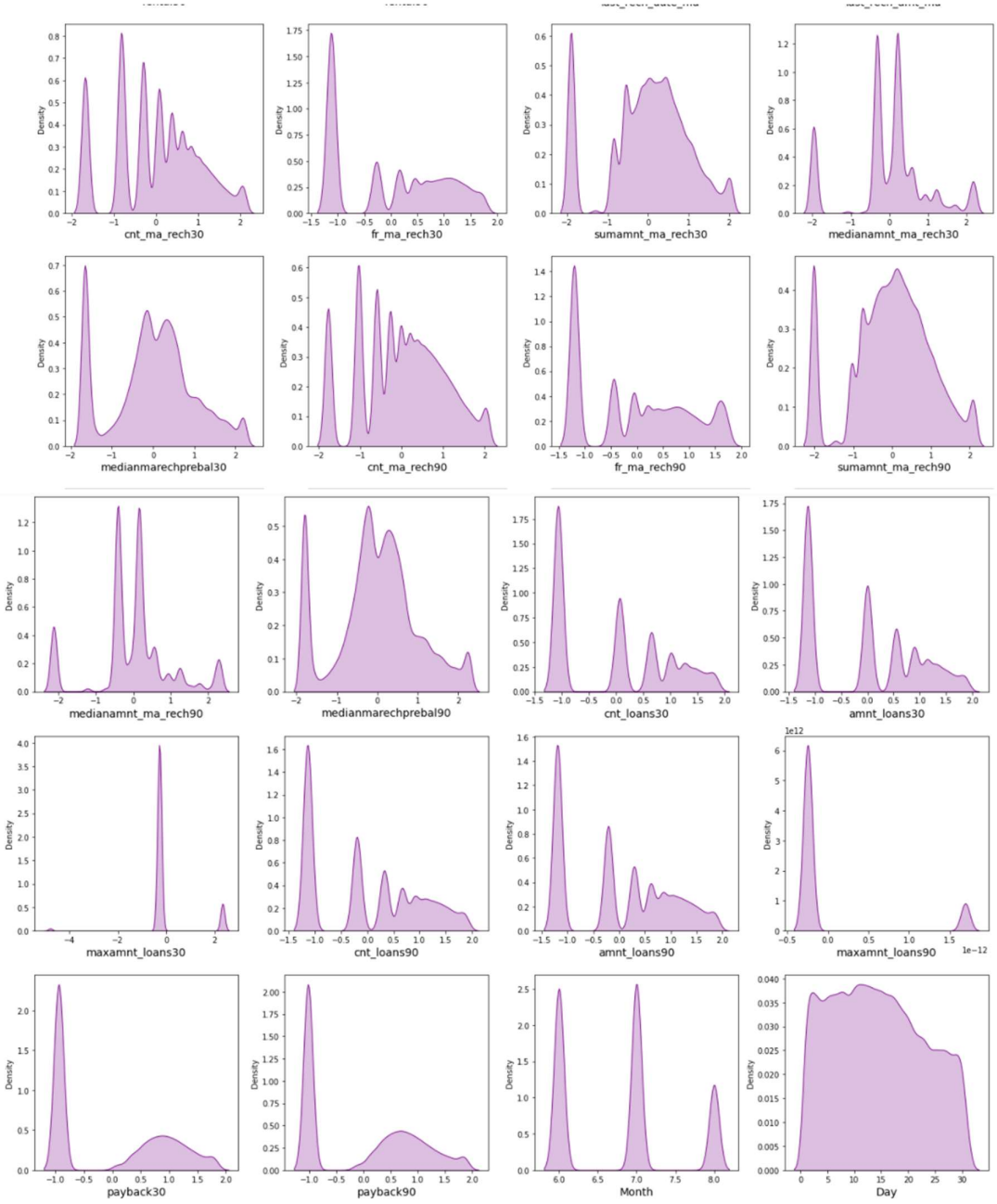
In [305]:
```
# Checking skewness after using yeo-johnson method
df[features].skew()
```

Out[305]:
```
aon                    -0.059261
daily_decr30           -0.137650
daily_decr90           -0.127335
rental30               -0.062522
rental90               -0.062946
last_rech_date_ma       0.043916
last_rech_amt_ma       -0.106643
cnt_ma_rech30          -0.010536
fr_ma_rech30            0.131926
sumamnt_ma_rech30      -0.369147
medianamnt_ma_rech30   -0.237104
medianmarechprebal30   -0.046085
cnt_ma_rech90          -0.012334
fr_ma_rech90            0.141522
sumamnt_ma_rech90      -0.266852
medianamnt_ma_rech90   -0.101431
medianmarechprebal90   -0.029793
cnt_loans30             0.219771
amnt_loans30            0.146844
maxamnt_loans30         0.381337
cnt_loans90             0.201541
amnt_loans90            0.129122
maxamnt_loans90         0.000000
payback30               0.278283
payback90               0.193524
dtype: float64
```

We can observe the skewness has almost been reduced in all the columns.

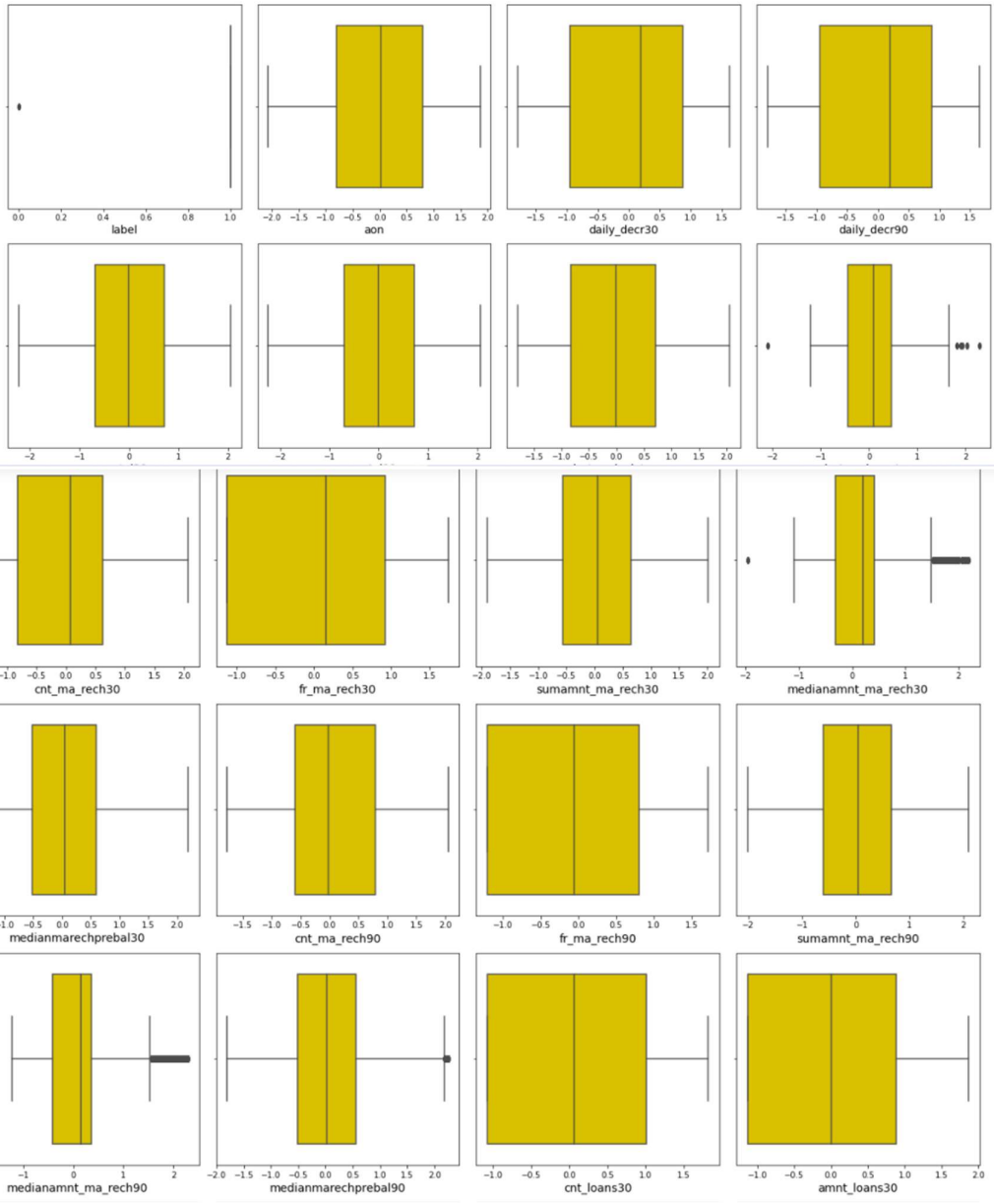
In [306]:
```
# Checking how the data has been distriubted in each column after removing skewness
plt.figure(figsize=(18,30),facecolor='white')
plotnumber=1
for column in df:
    if plotnumber<=28:
        ax=plt.subplot(7,4,plotnumber)
        sns.distplot(df[column],color="purple",kde_kws={"shade": True},hist=False)
        plt.xlabel(column,fontsize=14)
    plotnumber+=1
plt.tight_layout()
```

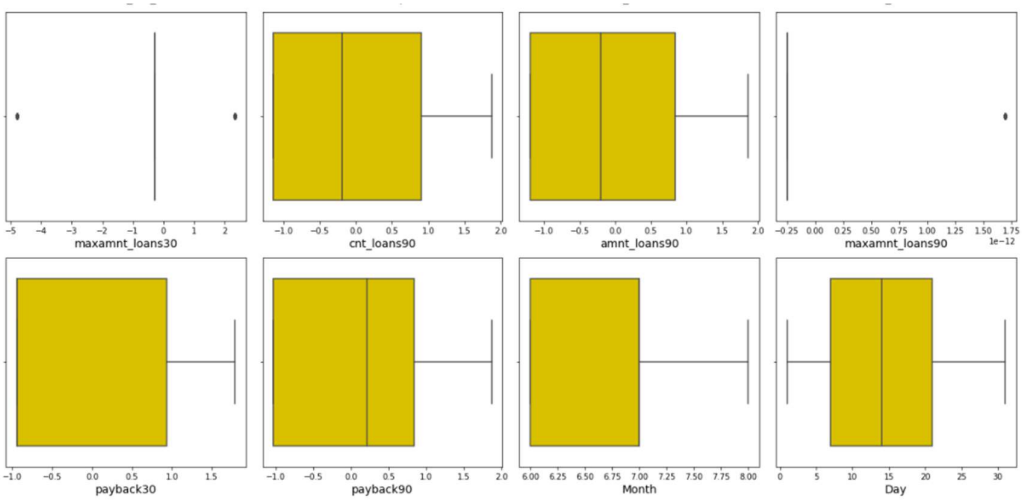



Density
cnt_ma_rech30
fr_ma_rech30
sumamnt_ma_rech30
medianamnt_ma_rech30
medianmarechprebal30
cnt_ma_rech90
fr_ma_rech90
sumamnt_ma_rech90
medianamnt_ma_rech90
medianmarechprebal90
cnt_loans30
amnt_loans30
maxamnt_loans30
cnt_loans90
amnt_loans90
maxamnt_loans90
1e12
payback30
payback90
Month
Day

From the above dist plots we can see that the data has been distributes normally in some of the columns and the skewness is also reduced compared to the previous data.

In [307]:
```
# Checking the outliers after using percentile method
plt.figure(figsize=(18,30),facecolor='white')
plotnumber=1
for column in df:
    if plotnumber<=28:
        ax=plt.subplot(7,4,plotnumber)
        sns.boxplot(df[column],color="gold")
        plt.xlabel(column,fontsize=14)
    plotnumber+=1
plt.tight_layout()
```





Its good to see that the outliers are almost removed in many columns after using percentile method and after removing skewness.

After cleaning the data we got only numerical data throughout the dataset. Since all the features in the dataset are numerical so no need to encode the data.

In [308]:
```
# Separating the feature and label into x and y
x = df.drop("label", axis=1)
y = df["label"]
```

In [310]:
```
# Dimension of x and y
x.shape,y.shape
```

Out[310]: ((207550, 27), (207550,))

Since skewness present in one column and some of the columns have outliers and the data is not distributed normally in some of the columns, so let's use MinMaxScaler method to scale the data.

SCALING

In [311]:
```
#train test split
from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=40)
```

In [312]:
```
#importing module-------scaling
from sklearn.preprocessing import MinMaxScaler
# creating normalization object
norm = MinMaxScaler()
# fit data
norm_fit = norm.fit(x_train)
new_xtrain = norm_fit.transform(x_train)
new_xtest = norm_fit.transform(x_test)
# display values
print(new_xtrain)
print(new_xtest)
```

```
[[0.5763921  0.28771649 0.28857396 ... 0.51285055 0.         0.56666667]
 [0.56252888 0.50296551 0.49937295 ... 0.         0.5        0.66666667]
 [0.28042557 0.33832555 0.33847051 ... 0.         0.         0.7       ]
 ...
 [0.19537869 0.82925502 0.81368381 ... 0.61975601 0.5        0.8       ]
 [0.33834804 0.16908682 0.26691043 ... 0.         0.5        0.7       ]
 [0.44508948 0.34193349 0.34201931 ... 0.         0.         0.76666667]]
[[0.53321084 0.81207771 0.79770795 ... 0.61975601 0.5        0.8       ]
 [1.         0.91187579 0.8937214  ... 0.46201044 0.5        0.8       ]
 [0.3610981  0.75436876 0.74251318 ... 0.71685134 0.5        0.6       ]
 ...
 [0.42211626 0.78653928 0.77231807 ... 0.66909828 0.5        0.36666667]
 [0.1101204  0.49030891 0.48715846 ... 0.         0.5        0.7       ]
 [0.14594951 0.46817765 0.46554344 ... 0.         0.5        0.        ]]
```

Resampling Technique

Oversampling

In [313]:
```
##The classes are ordered and not balanced (e.g. there are many more normal wines than excellent or poor ones)
###so we are balancing the classes with smote technique

print("Before OverSampling, counts of label '1': {}".format(sum(y_train == 1)))
print("Before OverSampling, counts of label '0': {} \n".format(sum(y_train == 0)))

# import SMOTE module from imblearn library
# pip install imblearn (if you don't have imblearn in your system)
from imblearn.over_sampling import SMOTE
sm = SMOTE(k_neighbors=1)
x_train_res, y_train_res = sm.fit_resample(new_xtrain, y_train.ravel())

print('After OverSampling, the shape of train_X: {}'.format(x_train_res.shape))
print('After OverSampling, the shape of train_y: {} \n'.format(y_train_res.shape))

print("After OverSampling, counts of label '1': {}".format(sum(y_train_res == 1)))
print("After OverSampling, counts of label '0': {}".format(sum(y_train_res == 0)))
```

```
Before OverSampling, counts of label '1': 145165
Before OverSampling, counts of label '0': 20875

After OverSampling, the shape of train_X: (290330, 27)
After OverSampling, the shape of train_y: (290330,)

After OverSampling, counts of label '1': 145165
After OverSampling, counts of label '0': 145165
```

Testing of Identified Approaches (Algorithms)

These are all the Algorithms used for Model Building and Prediction.We did Hyper Parameter Tuning with these algorithms using the GridSearchCV.
RandomForestClassifier
GaussianNB
KNeighborsClassifier
AdaBoost Classifier
SVC(Support Vector Classifier)
LogisticRegression
Soft Voting Classifier
Hard Voting classifier
DecisionTreeClassifier
Gradient Boosting Classifier
LightGradientBoostingClassifier
CatBoostClassifier
ExtraTreesClassifier
XGBoost Classifier

These algorithms has been used for both Training and Testing purpose and got evaluated with classification metrics such as f1score,confusion matrix,precision,recall and AUC ROC curve etc.,

Run and Evaluate selected models

Logistic Regression:

Logistic regression is used **to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables**. Logistic regression is a **simple and more efficient method for binary and linear classification problems**. It is a classification model, which is very easy to realize and achieves very good performance with linearly separable classes. It is an extensively employed algorithm for classification.Logistic Regression is used **when the dependent variable(target) is categorical**.

MODEL PREDICTION

LOGSITIC REGRESSION

In [92]:
```
#train test split
from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=40)
```

In [93]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
from sklearn.model_selection import train_test_split
```

Hyperparameter Tuning Done Using GridSearchCV:

Hyper parameter Tuning

In [318]:
```
#performs GridsearchCV logistic regression
from sklearn.model_selection import GridSearchCV
parameters={'dual':[False,True],'fit_intercept':[True,False],'random_state':list(range(0,1)),'max_iter':[100,50], 'tol':[0.001,0.
lr=LogisticRegression()
clf=GridSearchCV(lr,parameters)
clf.fit(x_train_res,y_train_res)
print(clf.best_params_)
```

```
{'dual': False, 'fit_intercept': True, 'max_iter': 100, 'random_state': 0, 'tol': 0.001}
```

In [319]:
```
lr=LogisticRegression(fit_intercept= True, dual=False, max_iter= 100, random_state=0, tol= 0.001)
lr.fit(x_train_res,y_train_res.ravel())
pred_test_lr=lr.predict(new_xtest)
pred_train_lr=lr.predict(x_train_res)
lr_score = lr.score(x_train_res,y_train_res)
lr_acc_score=accuracy_score(y_test,pred_test)
print("Accuracy score is:",lr_acc_score*100)
print("score of model is:",lr_score*100)
```

```
Accuracy score is: 75.90219224283305
score of model is: 77.12051803120586
```

In [320]:
```
cv_score_lr=cross_val_score(lr,x,y,cv=5)
cv_mean_lr=cv_score_lr.mean()
print("cv_mean is:",cv_mean_lr*100)
```

```
cv_mean is: 88.00337268128162
```

In [321]:
```
print(classification_report(y_test, pred_test))
```

```
              precision    recall  f1-score   support

           0       0.32      0.78      0.45      5287
           1       0.96      0.76      0.85     36223

    accuracy                           0.76     41510
   macro avg       0.64      0.77      0.65     41510
weighted avg       0.88      0.76      0.80     41510
```

In [322]:
```
print(confusion_matrix(y_test,pred_test))
```

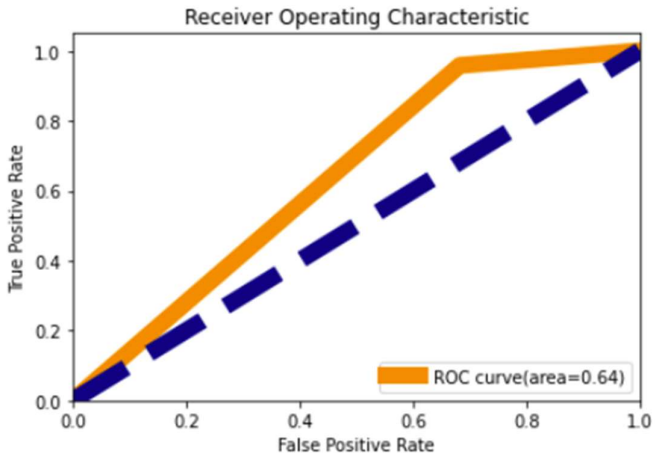
```
[[ 4098  1189]
 [ 8814 27409]]
```

In [323]:
```
print(accuracy_score(y_test,pred_test)*100)
```

```
75.90219224283305
```

AUc-ROC Curve

In [324]:
```
from sklearn.metrics import roc_curve,auc
fpr,tpr,thresholds=roc_curve(pred_test,y_test)
roc_auc=auc(fpr,tpr)
plt.figure()
plt.plot(fpr,tpr,color='darkorange',lw=10,label="ROC curve(area=%0.2f)" %roc_auc)
plt.plot([0,1],[0,1],color='navy',lw=10,linestyle='--')
plt.xlim([0.0,1.0])
plt.ylim([0.0,1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title("Receiver Operating Characteristic")
plt.legend(loc='lower right')
plt.show()
```



area under the curve is 64%

RANDOM FOREST CLASSIFIER

The random forest is a **classification algorithm consisting of many decisions trees**. It uses bagging and feature randomness when building each individual tree to try to create an uncorrelated forest of trees whose prediction by committee is more accurate than that of any individual tree. It **can perform both regression and classification tasks**. A random forest produces good predictions that can be understood easily. It can handle large datasets efficiently. The random forest algorithm provides a higher level of accuracy in predicting outcomes over the decision tree algorithm.

Random forest **adds additional randomness to the model, while growing the trees**. Instead of searching for the most important feature while splitting a node, it searches for the best feature among a random subset of features. This results in a wide diversity that generally results in a better model.

RANDOM FOREST CLASSIFIER

In [335]:
```
# importing modules

from sklearn.ensemble import RandomForestClassifier

#creating RandomForestClassifier constructor
rf = RandomForestClassifier(random_state=5)
# fit data
rf.fit(x_train_res,y_train_res.ravel())
# predicting score
pred_test_rf=rf.predict(new_xtest)
pred_train_rf=rf.predict(x_train_res)
#calculates score of the model using score method
rf_score = rf.score(new_xtest,y_test)
print('score of model is : ',rf_score*100)
```

```
score of model is :  91.47916164779572
```

In [336]:
```
print("Accuracy Score of Training Data is:",accuracy_score(y_train_res.ravel(),pred_train_rf)*100)
print("Accuracy Score of Testing Data is:",accuracy_score(y_test,pred_test_rf)*100)
```

```
Accuracy Score of Training Data is: 99.9975889505046
Accuracy Score of Testing Data is: 91.47916164779572
```

In [337]:
```
#choosing best Random state
for i in range(0,4):
    rf.fit(x_train_res, y_train_res.ravel())
    pred_train_rf=rf.predict(x_train_res)
    pred_test_rf=rf.predict(new_xtest)
    if round(accuracy_score(y_train_res.ravel(),pred_train_rf)*100,1)== round(accuracy_score(y_test,pred_test_rf)*100,1):
        print("At random state ",i,"model performs well")
        print("At random state:-",i)
        print("Training r2_score is :-",accuracy_score(y_train_res.ravel(),pred_train_rf)*100)
        print("Testing r2_score is :-",accuracy_score(y_test,pred_test_rf)*100)
```

In [338]:
```
print("accuracy score is:",accuracy_score(y_test,pred_test_rf)*100)
```

```
accuracy score is: 91.47916164779572
```

In [340]:
```
#cross validation
from sklearn.model_selection import cross_val_score
acc_score_rf=accuracy_score(y_test,pred_test_rf)
for j in range(2,5):
    cross_V_score=cross_val_score(rf,x,y,cv=j)
    print("At cv:-",j)
    print("cross validation score is:",acc_score_rf*100)
    print("accuracy score is:",acc_score_rf*100)
    print("\n")
```

```
At cv:- 2
cross validation score is: 91.47916164779572
accuracy score is: 91.47916164779572


At cv:- 3
cross validation score is: 91.47916164779572
accuracy score is: 91.47916164779572


At cv:- 4
cross validation score is: 91.47916164779572
accuracy score is: 91.47916164779572
```

parameter tuning

In [345]:
```
#performs GridSearchCV on RandomForestClassifier
from sklearn.model_selection import GridSearchCV
parameters={'criterion':['gini', 'entropy'],'n_jobs':[1,-1],'random_state':list(range(0,1)),'min_weight_fraction_leaf':[0.1,0.2]
rf=RandomForestClassifier()
clf=GridSearchCV(rf,parameters)
clf.fit(x_train_res,y_train_res.ravel())
print(clf.best_params_)
```

```
{'bootstrap': True, 'criterion': 'entropy', 'min_weight_fraction_leaf': 0.1, 'n_jobs': 1, 'random_state': 0}
```

In [347]:
```
rf=RandomForestClassifier(criterion="entropy",random_state=0,bootstrap="True",min_weight_fraction_leaf= 0.1,n_jobs= 1)
rf.fit(x_train_res,y_train_res.ravel())
pred_test_rf=rf.predict(new_xtest)
pred_train_rf=rf.predict(x_train_res)
rf_score = rf.score(x_train_res,y_train_res)
rf_acc_score=accuracy_score(y_test,pred_test)
print("Accuracy score is:",rf_acc_score*100)
print("score of model is:",rf_score*100)
```

```
Accuracy score is: 75.90219224283305
score of model is: 76.64106361726311
```

In [348]:
```
cv_score_rf=cross_val_score(rf,x,y,cv=5)
cv_mean_rf=cv_score_rf.mean()
print("cv_mean is:",cv_mean_rf*100)
```

```
cv_mean is: 87.39484461575525
```

In [349]:
```
print(confusion_matrix(y_test,pred_test_rf))
```

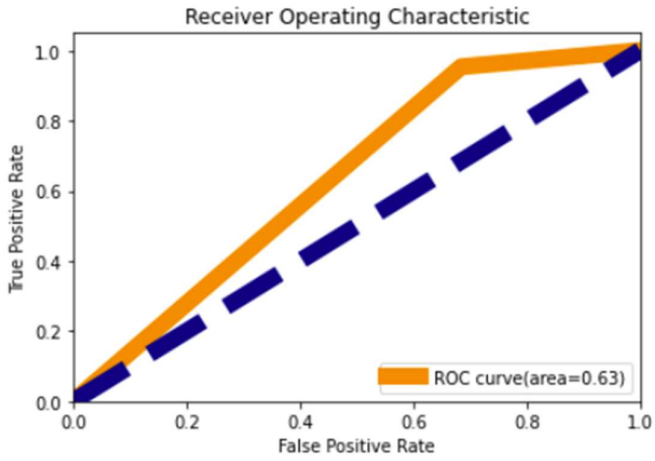
```
[[ 3947  1340]
 [ 8558 27665]]
```

In [350]:
```
print(classification_report(y_test,pred_test_rf))
```

```
              precision    recall  f1-score   support

           0       0.32      0.75      0.44      5287
           1       0.95      0.76      0.85     36223

    accuracy                           0.76     41510
   macro avg       0.63      0.76      0.65     41510
weighted avg       0.87      0.76      0.80     41510
```

AUC-ROC CURVE:

In [351]:
```
from sklearn.metrics import roc_curve,auc
fpr,tpr,thresholds=roc_curve(pred_test_rf,y_test)
roc_auc=auc(fpr,tpr)
plt.figure()
plt.plot(fpr,tpr,color='darkorange',lw=10,label="ROC curve(area=%0.2f)" %roc_auc)
plt.plot([0,1],[0,1],color='navy',lw=10,linestyle='--')
plt.xlim([0.0,1.0])
plt.ylim([0.0,1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title("Receiver Operating Characteristic")
plt.legend(loc='lower right')
plt.show()
```



area under the curve is 63%

DECISION TREE CLASSIFIER

The main advantage of the decision tree classifier is **its ability to using different feature subsets and decision rules at different stages of classification**. Decision tree often involves higher time to train the model. Decision tree training is relatively expensive as the complexity and time has taken are more. The Decision Tree algorithm **is inadequate for applying regression and predicting continuous values**.In this, **the data is continuously split according to a certain parameter**. The tree can be explained by two entities, namely decision nodes and leaves.

parameter tuning

In [355]:
```
#perform gridsearchcv and cross val score on Decison Tree DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeClassifier
parameters={'criterion':['gini', 'entropy'],'splitter':['best','random'],'max_features':['auto', 'sqrt'],'random_state':list(rang
dt=DecisionTreeClassifier()
clf=GridSearchCV(dt,parameters)
clf.fit(x_train_res,y_train_res.ravel())
print(clf.best_params_)
```

```
{'criterion': 'entropy', 'max_depth': 11, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 2, 'random_stat
e': 0, 'splitter': 'best'}
```

In [356]:
```
dt=DecisionTreeClassifier(criterion='entropy',max_features= 'auto',max_depth= 11 , random_state= 0, splitter= 'best',min_samples_
dt.fit(x_train_res,y_train_res.ravel())
pred_test_dt=dt.predict(new_xtest)
pred_train_dt=dt.predict(x_train_res)
dt_score = dt.score(x_train_res,y_train_res.ravel())
dt_acc_score=accuracy_score(y_test,pred_test_dt)
print("Accuracy score is:",dt_acc_score*100)
print("score of model is:",dt_score*100)
```

```
Accuracy score is: 82.89568778607564
score of model is: 84.56824992250198
```

In [357]:
```
cv_score_dt=cross_val_score(dt,x,y,cv=5)
cv_mean_dt=cv_score_dt.mean()
print("cv_mean is:",cv_mean_dt*100)
```

```
cv_mean is: 90.75837147675259
```

In [358]:
```
print(confusion_matrix(y_test,pred_test_dtc))
```

```
[[ 3123  2164]
 [ 2994 33229]]
```

In [359]:
```
print(classification_report(y_test,pred_test_dtc))
```

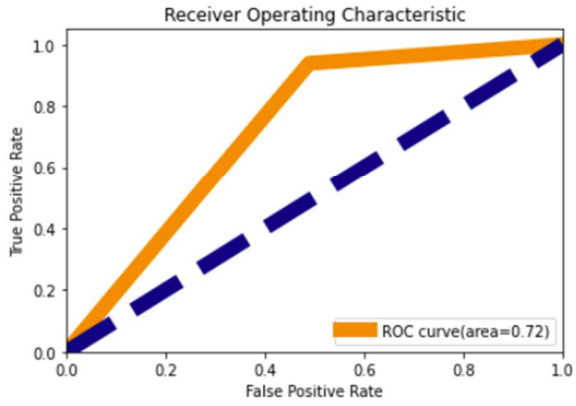
```
              precision    recall  f1-score   support

           0       0.51      0.59      0.55      5287
           1       0.94      0.92      0.93     36223

    accuracy                           0.88     41510
   macro avg       0.72      0.75      0.74     41510
weighted avg       0.88      0.88      0.88     41510
```

AUC-ROC CURVE:

In [360]:
```
# ROC_AUC CURVE
from sklearn.metrics import roc_curve,auc
fpr,tpr,thresholds=roc_curve(pred_test_dtc,y_test)
roc_auc=auc(fpr,tpr)
plt.figure()
plt.plot(fpr,tpr,color='darkorange',lw=10,label="ROC curve(area=%0.2f)" %roc_auc)
plt.plot([0,1],[0,1],color='navy',lw=10,linestyle='--')
plt.xlim([0.0,1.0])
plt.ylim([0.0,1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title("Receiver Operating Characteristic")
plt.legend(loc='lower right')
plt.show()
```

AUC-ROC CURVE:

In [360]:
```
# ROC_AUC CURVE
from sklearn.metrics import roc_curve,auc
fpr,tpr,thresholds=roc_curve(pred_test_dtc,y_test)
roc_auc=auc(fpr,tpr)
plt.figure()
plt.plot(fpr,tpr,color='darkorange',lw=10,label="ROC curve(area=%0.2f)" %roc_auc)
plt.plot([0,1],[0,1],color='navy',lw=10,linestyle='--')
plt.xlim([0.0,1.0])
plt.ylim([0.0,1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title("Receiver Operating Characteristic")
plt.legend(loc='lower right')
plt.show()
```



area under the curve is 72%

GAUSSIANNB

In [131]:
```
from sklearn.naive_bayes import GaussianNB
gnb= GaussianNB()
gnb.fit(x_train_res, y_train_res.ravel())
pred_test_gnb = gnb.predict(new_xtest)
pred_train_gnb = gnb.predict(x_train_res)
gnb_score = gnb.score(new_xtest,y_test)
print('score of model is : ',gnb_score*100)
```

```
score of model is :  74.09539869910866
```

In [132]:
```
print("Accuracy Score of Training Data is:",accuracy_score(y_train_res.ravel(),pred_train_gnb)*100)
print("Accuracy Score of Testing Data is:",accuracy_score(y_test,pred_test_gnb)*100)
```

```
Accuracy Score of Training Data is: 75.18857851410465
Accuracy Score of Testing Data is: 74.09539869910866
```

parameter tuning

In [133]:
```
#perform gridsearchcv and cross val score on GaussianNB
from sklearn.model_selection import GridSearchCV
params_NB = {'var_smoothing': np.logspace(0,-9, num=100)}
gnb=GaussianNB()
clf = GridSearchCV(estimator=gnb,
                   param_grid=params_NB,
                   cv=5,   # use any cross validation technique
                   verbose=1,
                   scoring='accuracy')
clf.fit(x_train_res,y_train_res.ravel())
print(clf.best_params_)
```

```
Fitting 5 folds for each of 100 candidates, totalling 500 fits
{'var_smoothing': 0.0533669923120631}
```

In [134]:
```
gnb=GaussianNB(var_smoothing=0.15199110829529336)
gnb.fit(x_train_res,y_train_res.ravel())
pred_test_gnb=gnb.predict(new_xtest)
pred_train_gnb=gnb.predict(x_train_res)
gnb_score = gnb.score(x_train_res,y_train_res.ravel())
gnb_acc_score=accuracy_score(y_test,pred_test_gnb)
print("Accuracy score is:",gnb_acc_score*100)
print("score of model is:",gnb_score*100)
```

```
Accuracy score is: 73.78463020958806
score of model is: 75.13209106878381
```

In [135]:
```
cv_score_gnb=cross_val_score(gnb,x,y,cv=5)
cv_mean_gnb=cv_score_gnb.mean()
print("cv_mean is:",cv_mean_gnb*100)
```

```
cv_mean is: 88.30643218501567
```

In [136]:
```
print(confusion_matrix(y_test,pred_test_gnb))
```

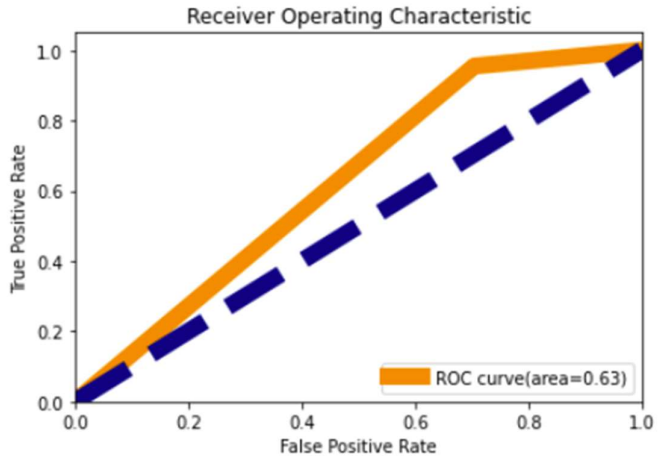
```
[[ 4034  1253]
 [ 9629 26594]]
```

In [137]:
```
print(classification_report(y_test,pred_test_gnb))
```

```
              precision    recall  f1-score   support

           0       0.30      0.76      0.43      5287
           1       0.96      0.73      0.83     36223

    accuracy                           0.74     41510
   macro avg       0.63      0.75      0.63     41510
weighted avg       0.87      0.74      0.78     41510
```

AUC-ROC CURVE:

In [138]:
```
from sklearn.metrics import roc_curve,auc
fpr,tpr,thresholds=roc_curve(pred_test_gnb,y_test)
roc_auc=auc(fpr,tpr)
plt.figure()
plt.plot(fpr,tpr,color='darkorange',lw=10,label="ROC curve(area=%0.2f)" %roc_auc)
plt.plot([0,1],[0,1],color='navy',lw=10,linestyle='--')
plt.xlim([0.0,1.0])
plt.ylim([0.0,1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title("Receiver Operating Characteristic")
plt.legend(loc='lower right')
plt.show()
```



area under the curve is 63%

GRADIENT BOOSTING CLASSIFIER

In [163]:
```
from sklearn.ensemble import GradientBoostingClassifier

gb = GradientBoostingClassifier(random_state=7)
gb.fit(x_train_res, y_train_res)
pred_test_gb = gb.predict(new_xtest)
pred_train_gb = gb.predict(x_train_res)
gb_score = gb.score(new_xtest,y_test)
print('score of model is : ',gb_score*100)
```

```
score of model is :  87.02481329800048
```

In [164]:
```
print("Accuracy Score of Training Data is:",accuracy_score(y_train_res.ravel(),pred_train_gb)*100)
print("Accuracy Score of Testing Data is:",accuracy_score(y_test,pred_test_gb)*100)
```

```
Accuracy Score of Training Data is: 89.95625667344058
Accuracy Score of Testing Data is: 87.02481329800048
```

In [172]:
```
print(confusion_matrix(y_test,pred_test_gb))
```

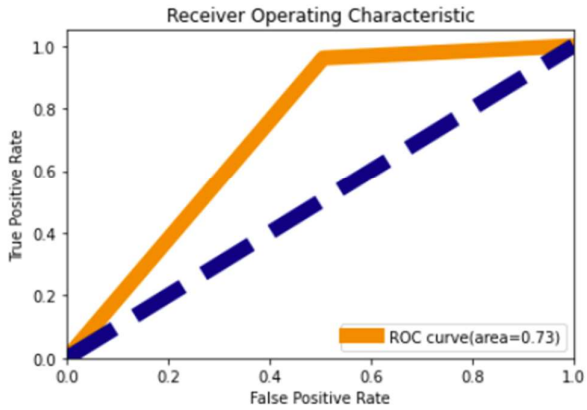
```
[[ 3987  1300]
 [ 4086 32137]]
```

In [173]:
```
print(classification_report(y_test,pred_test_gb))
```

```
              precision    recall  f1-score   support

           0       0.49      0.75      0.60      5287
           1       0.96      0.89      0.92     36223

    accuracy                           0.87     41510
   macro avg       0.73      0.82      0.76     41510
weighted avg       0.90      0.87      0.88     41510
```

AUC-ROC CURVE:

In [174]:
```
from sklearn.metrics import roc_curve,auc
fpr,tpr,thresholds=roc_curve(pred_test_gb,y_test)
roc_auc=auc(fpr,tpr)
plt.figure()
plt.plot(fpr,tpr,color='darkorange',lw=10,label="ROC curve(area=%0.2f)" %roc_auc)
plt.plot([0,1],[0,1],color='navy',lw=10,linestyle='--')
plt.xlim([0.0,1.0])
plt.ylim([0.0,1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title("Receiver Operating Characteristic")
plt.legend(loc='lower right')
plt.show()
```



area under the curve is 73%

LIGHT GRADIENT BOOSTING CLASSIFIER

In [140]:
```
!pip install lightgbm
```

```
Requirement already satisfied: lightgbm in c:\users\srividya\anaconda3\lib\site-packages (3.2.1)
Requirement already satisfied: numpy in c:\users\srividya\appdata\roaming\python\python38\site-packages (from lightgbm) (1.21.
4)
Requirement already satisfied: scipy in c:\users\srividya\anaconda3\lib\site-packages (from lightgbm) (1.6.2)
Requirement already satisfied: wheel in c:\users\srividya\anaconda3\lib\site-packages (from lightgbm) (0.36.2)
Requirement already satisfied: scikit-learn!=0.22.0 in c:\users\srividya\anaconda3\lib\site-packages (from lightgbm) (1.0.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\srividya\anaconda3\lib\site-packages (from scikit-learn!=0.22.0
->lightgbm) (2.1.0)
Requirement already satisfied: joblib>=0.11 in c:\users\srividya\anaconda3\lib\site-packages (from scikit-learn!=0.22.0->lightg
bm) (1.0.1)

WARNING: Ignoring invalid distribution -atplotlib (c:\users\srividya\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -atplotlib (c:\users\srividya\anaconda3\lib\site-packages)
```

In [141]:
```
#Light Gradient Boosting Classifier
```

In [142]:
```
from lightgbm import LGBMClassifier

lgbm = LGBMClassifier()
lgbm.fit(x_train_res, y_train_res.ravel())
pred_test_lgbm = lgbm.predict(new_xtest)
pred_train_lgbm = lgbm.predict(x_train_res)
lgbm_score = lgbm.score(new_xtest,y_test)
print('score of model is : ',lgbm_score*100)
```

```
score of model is :  91.00216815225247
```

In [143]:
```
print("Accuracy Score of Training Data is:",accuracy_score(y_train_res.ravel(),pred_train_lgbm)*100)
print("Accuracy Score of Testing Data is:",accuracy_score(y_test,pred_test_lgbm)*100)
```

```
Accuracy Score of Training Data is: 94.61474873419901
Accuracy Score of Testing Data is: 91.00216815225247
```

In [144]:
```
print(confusion_matrix(y_test,pred_test_lgbm))
```

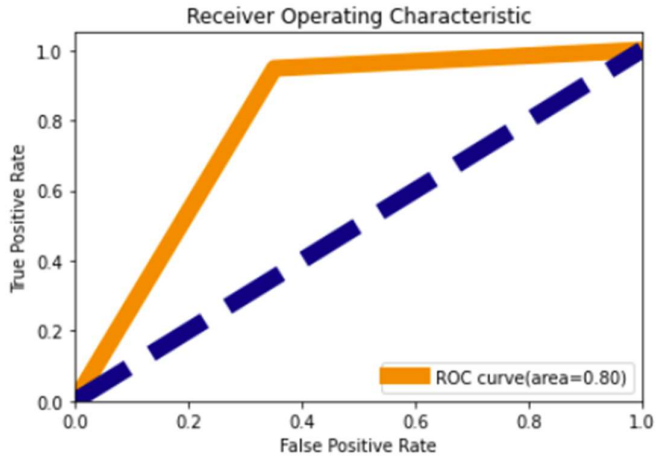
```
[[ 3398  1889]
 [ 1846 34377]]
```

In [145]:
```
print(classification_report(y_test,pred_test_lgbm))
```

```
              precision    recall  f1-score   support

           0       0.65      0.64      0.65      5287
           1       0.95      0.95      0.95     36223

    accuracy                           0.91     41510
   macro avg       0.80      0.80      0.80     41510
weighted avg       0.91      0.91      0.91     41510
```

AUC-ROC CURVE:

In [146]:
```
from sklearn.metrics import roc_curve,auc
fpr,tpr,thresholds=roc_curve(pred_test_lgbm,y_test)
roc_auc=auc(fpr,tpr)
plt.figure()
plt.plot(fpr,tpr,color='darkorange',lw=10,label="ROC curve(area=%0.2f)" %roc_auc)
plt.plot([0,1],[0,1],color='navy',lw=10,linestyle='--')
plt.xlim([0.0,1.0])
plt.ylim([0.0,1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title("Receiver Operating Characteristic")
plt.legend(loc='lower right')
plt.show()
```



area under the curve is 80%

CAT BOOST CLASSIFIER

In [147]:
```
!pip install catboost
```

```
Requirement already satisfied: catboost in c:\users\srividya\anaconda3\lib\site-packages (0.26.1)
Requirement already satisfied: matplotlib in c:\users\srividya\appdata\roaming\python\python38\site-packages (from catboost)
(3.5.0)
Requirement already satisfied: graphviz in c:\users\srividya\anaconda3\lib\site-packages (from catboost) (0.17)
Requirement already satisfied: scipy in c:\users\srividya\anaconda3\lib\site-packages (from catboost) (1.6.2)
Requirement already satisfied: numpy>=1.16.0 in c:\users\srividya\appdata\roaming\python\python38\site-packages (from catboost)
(1.21.4)
Requirement already satisfied: six in c:\users\srividya\appdata\roaming\python\python38\site-packages (from catboost) (1.16.0)
Requirement already satisfied: plotly in c:\users\srividya\anaconda3\lib\site-packages (from catboost) (5.3.1)
Requirement already satisfied: pandas>=0.24.0 in c:\users\srividya\anaconda3\lib\site-packages (from catboost) (1.2.4)
Requirement already satisfied: pytz>=2017.3 in c:\users\srividya\anaconda3\lib\site-packages (from pandas>=0.24.0->catboost) (2
021.1)
```

In [148]:
```
from catboost import CatBoostClassifier

cat = CatBoostClassifier()
cat.fit(x_train_res, y_train_res.ravel())
pred_test_cat = cat.predict(new_xtest)
pred_train_cat = cat.predict(x_train_res)
cat_score = cat.score(new_xtest,y_test)
print('score of model is : ',cat_score*100)
```

```
Learning rate set to 0.116035
0:      learn: 0.5829817        total: 216ms    remaining: 3m 35s
1:      learn: 0.5269479        total: 274ms    remaining: 2m 16s
2:      learn: 0.4913842        total: 321ms    remaining: 1m 46s
3:      learn: 0.4653059        total: 361ms    remaining: 1m 29s
4:      learn: 0.4371351        total: 397ms    remaining: 1m 19s
5:      learn: 0.4215135        total: 434ms    remaining: 1m 11s
6:      learn: 0.4070544        total: 472ms    remaining: 1m 6s
7:      learn: 0.3897583        total: 508ms    remaining: 1m 2s
8:      learn: 0.3768554        total: 544ms    remaining: 59.9s
9:      learn: 0.3652616        total: 582ms    remaining: 57.6s
10:     learn: 0.3569460        total: 618ms    remaining: 55.5s
11:     learn: 0.3497498        total: 657ms    remaining: 54.1s
12:     learn: 0.3431596        total: 692ms    remaining: 52.5s
13:     learn: 0.3377325        total: 727ms    remaining: 51.2s
14:     learn: 0.3277177        total: 762ms    remaining: 50s
15:     learn: 0.3202762        total: 796ms    remaining: 48.9s
16:     learn: 0.3141886        total: 829ms    remaining: 48s
17:     learn: 0.3081185        total: 866ms    remaining: 47.2s
```

In [149]:
```
print("Accuracy Score of Training Data is:",accuracy_score(y_train_res.ravel(),pred_train_cat)*100)
print("Accuracy Score of Testing Data is:",accuracy_score(y_test,pred_test_cat)*100)
```

```
Accuracy Score of Training Data is: 96.04484552061447
Accuracy Score of Testing Data is: 91.97060949168875
```

In [150]:
```
print(confusion_matrix(y_test,pred_test_cat))
```

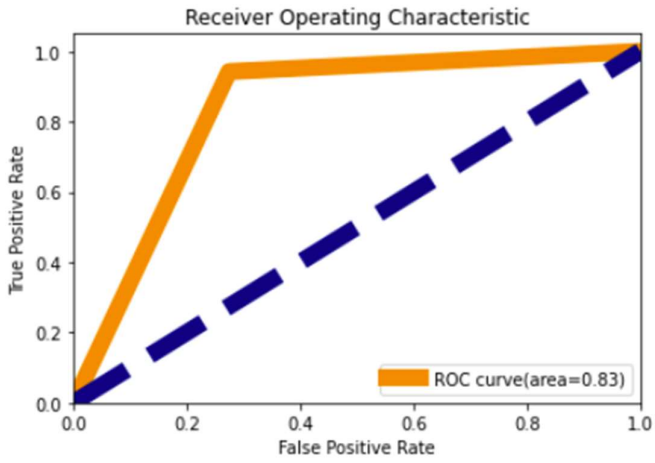
```
[[ 3143  2144]
 [ 1189 35034]]
```

In [151]:
```
print(classification_report(y_test,pred_test_cat))
```

```
              precision    recall  f1-score   support

           0       0.73      0.59      0.65      5287
           1       0.94      0.97      0.95     36223

    accuracy                           0.92     41510
   macro avg       0.83      0.78      0.80     41510
weighted avg       0.91      0.92      0.92     41510
```

AUC-ROC CURVE:

In [152]:
```
from sklearn.metrics import roc_curve,auc
fpr,tpr,thresholds=roc_curve(pred_test_cat,y_test)
roc_auc=auc(fpr,tpr)
plt.figure()
plt.plot(fpr,tpr,color='darkorange',lw=10,label="ROC curve(area=%0.2f)" %roc_auc)
plt.plot([0,1],[0,1],color='navy',lw=10,linestyle='--')
plt.xlim([0.0,1.0])
plt.ylim([0.0,1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title("Receiver Operating Characteristic")
plt.legend(loc='lower right')
plt.show()
```



area under the curve is 83%

XGBOOST

In [156]:
```
!pip install xgboost
```

```
Requirement already satisfied: xgboost in c:\users\srividya\anaconda3\lib\site-packages (1.4.2)
Requirement already satisfied: scipy in c:\users\srividya\anaconda3\lib\site-packages (from xgboost) (1.6.2)
Requirement already satisfied: numpy in c:\users\srividya\appdata\roaming\python\python38\site-packages (from xgboost) (1.21.4)
WARNING: Ignoring invalid distribution -atplotlib (c:\users\srividya\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -atplotlib (c:\users\srividya\anaconda3\lib\site-packages)
```

In [157]:
```
import xgboost as xgb
xgb = xgb.XGBClassifier()
xgb.fit(x_train_res, y_train_res.ravel())
pred_test_xgb = xgb.predict(new_xtest)
pred_train_xgb = xgb.predict(x_train_res)
xgb_score = xgb.score(new_xtest,y_test)
print('score of model is : ',xgb_score*100)
```

```
[05:51:23] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.0/src/learner.cc:1095: Starting in XGBoost 1.3.
0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly se
t eval_metric if you'd like to restore the old behavior.
score of model is :  91.63815947964345
```

In [158]:
```
print(confusion_matrix(y_test,pred_test_xgb))
```

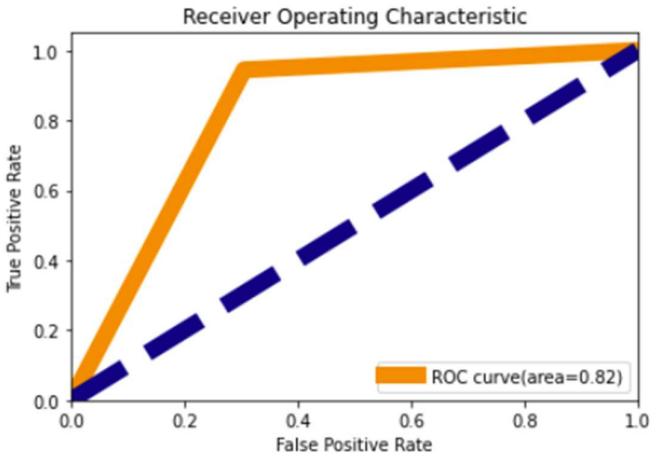
```
[[ 3241  2046]
 [ 1425 34798]]
```

In [159]:
```
print(classification_report(y_test,pred_test_xgb))
```

```
              precision    recall  f1-score   support

           0       0.69      0.61      0.65      5287
           1       0.94      0.96      0.95     36223

    accuracy                           0.92     41510
   macro avg       0.82      0.79      0.80     41510
weighted avg       0.91      0.92      0.91     41510
```

AUC-ROC CURVE:

In [160]:
```
from sklearn.metrics import roc_curve,auc
fpr,tpr,thresholds=roc_curve(pred_test_xgb,y_test)
roc_auc=auc(fpr,tpr)
plt.figure()
plt.plot(fpr,tpr,color='darkorange',lw=10,label="ROC curve(area=%0.2f)" %roc_auc)
plt.plot([0,1],[0,1],color='navy',lw=10,linestyle='--')
plt.xlim([0.0,1.0])
plt.ylim([0.0,1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title("Receiver Operating Characteristic")
plt.legend(loc='lower right')
plt.show()
```



area under the curve is 82%

These are some of the algorithms used and it described here with the snapshot of their code and the results observed over different evaluation metrics are also mentioned.
The evaluation metrics used here is classification metrics.

Key Metrics for success in solving problem under consideration

An evaluation metric **quantifies the performance of a predictive model**. This typically involves training a model on a dataset, using the model to make predictions on a holdout dataset not used during training, then comparing the predictions to the expected values in the holdout dataset. We got Good accuracy with Cat Boost Classifier when comparing with other model's performance.

CatBoost is **based on gradient boosted decision trees**. During training, a set of decision trees is built consecutively. Each successive tree is built with reduced loss compared to the previous trees. The number of trees is controlled by the starting parameters.

We use F-Score as our performane metrics for our prediction.

Feature Selection Using F-score

F1 Score:

The F-score is often used in the field of information retrieval for measuring search, document classification, and query classification performance.
The F-score has been widely used in the natural language processing literature, such as the evaluation of named entity recognition and word segmentation

In [187]:
```
from sklearn import feature_selection as fs
num_features = 27
fs_fit_fscore = fs.SelectKBest(fs.f_classif, k=num_features)
fs_fit_fscore.fit_transform(x_train,y_train)
fs_indices_fscore =np.argsort(np.nan_to_num(fs_fit_fscore.scores_))[::-1][0:num_features]
fs_indices_fscore
```

Out[187]:
```
array([14, 12,  9,  7, 21, 20, 18, 10, 17,  2, 11,  1,  6, 16, 15, 24,  8,
       23, 13,  4, 25,  3, 22,  0, 19,  5, 26], dtype=int64)
```

In [188]:
```
best_features_fscore = df.columns[fs_indices_fscore].values
best_features_fscore
```

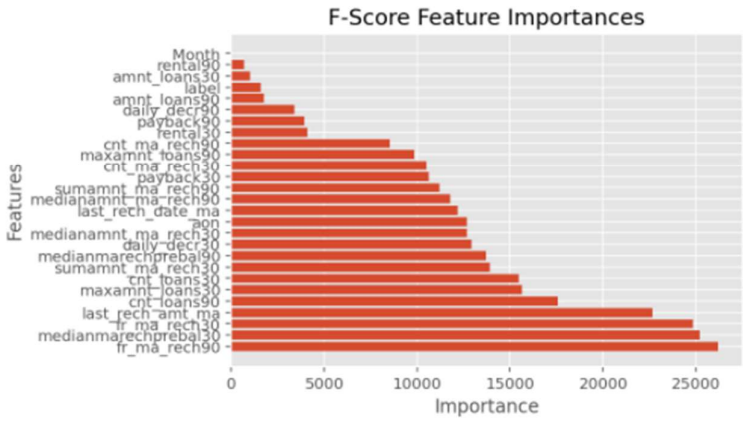
Out[188]:
```
array(['fr_ma_rech90', 'medianmarechprebal30', 'fr_ma_rech30',
       'last_rech_amt_ma', 'cnt_loans90', 'maxamnt_loans30',
       'cnt_loans30', 'sumamnt_ma_rech30', 'medianmarechprebal90',
       'daily_decr30', 'medianamnt_ma_rech30', 'aon', 'last_rech_date_ma',
       'medianamnt_ma_rech90', 'sumamnt_ma_rech90', 'payback30',
       'cnt_ma_rech30', 'maxamnt_loans90', 'cnt_ma_rech90', 'rental30',
       'payback90', 'daily_decr90', 'amnt_loans90', 'label',
       'amnt_loans30', 'rental90', 'Month'], dtype=object)
```

In [190]:
```
feature_importances_fscore = fs_fit_fscore.scores_[fs_indices_fscore]
feature_importances_fscore
```

```
Out[190]: array([2.61945827e+04, 2.52124623e+04, 2.48494998e+04, 2.26565915e+04,
                 1.75839740e+04, 1.56512872e+04, 1.54690558e+04, 1.39139592e+04,
                 1.37270515e+04, 1.29094531e+04, 1.26715262e+04, 1.26698086e+04,
                 1.21728274e+04, 1.17578332e+04, 1.12043071e+04, 1.06087038e+04,
                 1.04852870e+04, 9.85038855e+03, 8.55000704e+03, 4.09851589e+03,
                 3.93303168e+03, 3.40373520e+03, 1.75499612e+03, 1.58444713e+03,
                 9.97689354e+02, 6.72953959e+02, 4.91179019e+00])
```

```
In [191]: import matplotlib.pyplot as plt
          %matplotlib inline
          %config InlineBackend.figure_format = 'retina'
          plt.style.use("ggplot")

          def plot_imp(best_features, scores, method_name):
              plt.barh(best_features, scores)
              plt.title(method_name + ' Feature Importances')
              plt.xlabel("Importance")
              plt.ylabel("Features")
              plt.show()
```

```
In [192]: plot_imp(best_features_fscore, feature_importances_fscore, 'F-Score')
```



The most related and important feature is fr_ma_rech90 which is F-Score

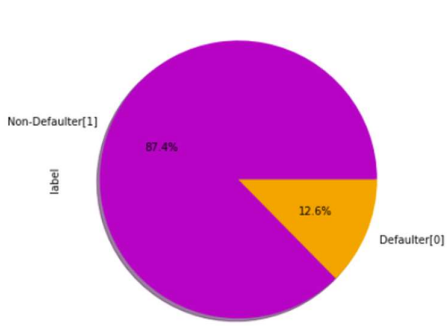
Visualizations

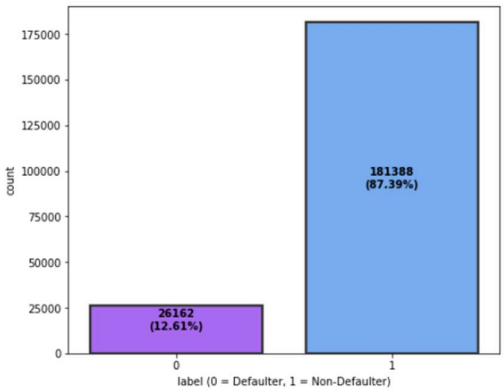
Univariate Analysis

```
In [270]: # Checking for total defaulters as it is imbalanced data.
```

```
In [271]: Rows = df.shape[0]
          Non_Defaulter = df[df["label"] == 1].shape[0]
          Defaulter = df[df["label"] == 0].shape[0]
          print("Total records = ",Rows)
          print("Non_Defaulter = ",Non_Defaulter)
          print("Defaulter     = ",Defaulter)
```

```
Total records =  207550
Non_Defaulter =  181388
Defaulter     =  26162
```





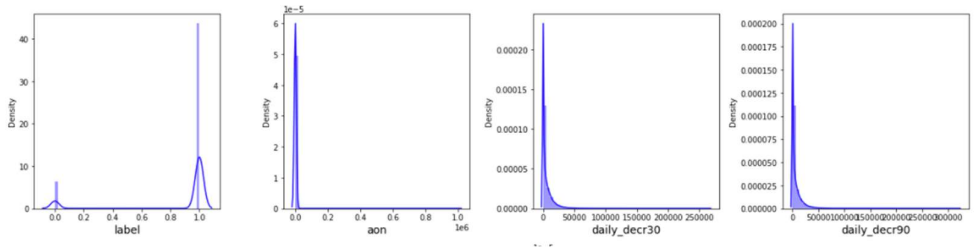
Observations:

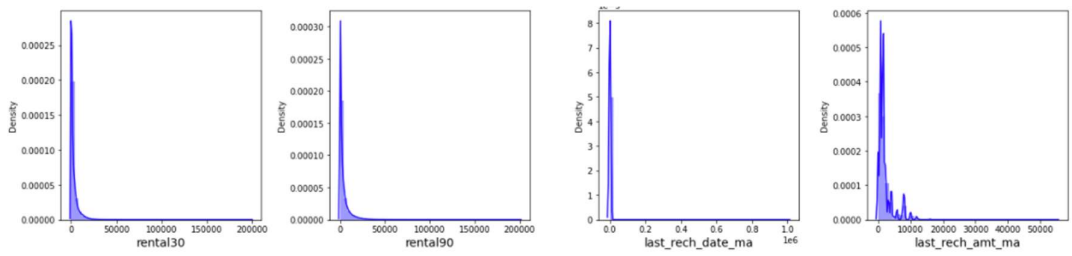
From the above plots we can observe around 87% of the loan has been paid by the user and only 12% of the loan failed to pay. Also the dataset is highly imbalanced, so we need to work on that or else our model will be more biased towards success and make false interpretation.

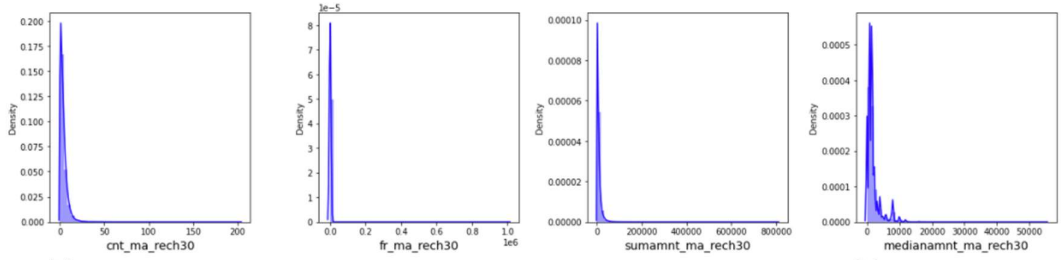
There is a data imbalancing issue so we have to treat this by using oversampling or undersampling.

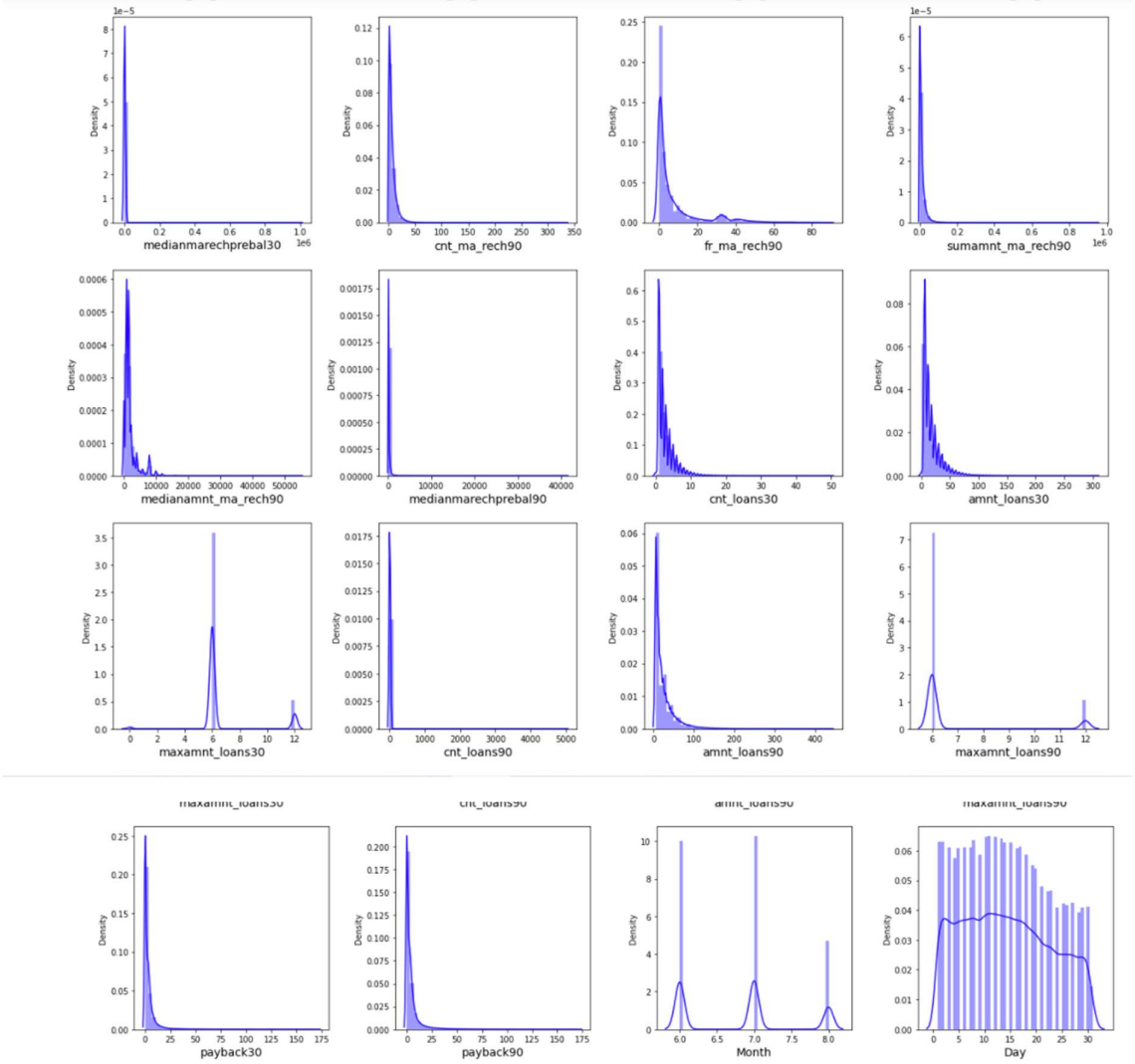
Distribution of skewness

In [273]:
```
# Checking how the data has been distriubted in each column

plt.figure(figsize=(18,30),facecolor='white')
plotnumber=1
for column in df:
    if plotnumber<=28:
        ax=plt.subplot(7,4,plotnumber)
        sns.distplot(df[column],color="blue")
        plt.xlabel(column,fontsize=14)
    plotnumber+=1
plt.tight_layout()
```









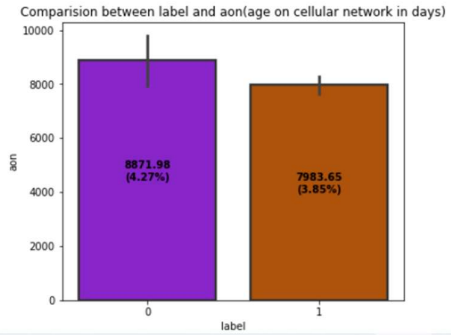
Observations:

From the above distribution plot, I can observe most of the columns are not normally distributed only Day column somewhat distributed normally. All the columns have skewness and are skewed to right since the mean is greater than the median in these columns. We need to remove this skewness before building our machine learning models.Data is highly spreaded and positively skewed which needs to be treated accordingly.

Bivariate Analysis

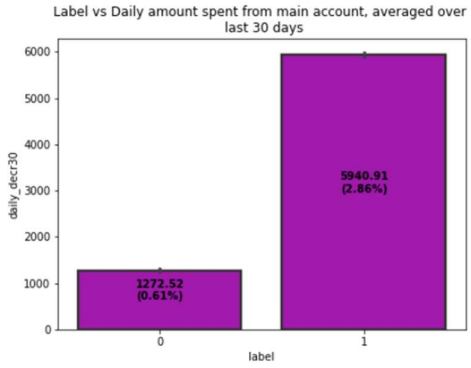
Compare independent features with Target(Label)

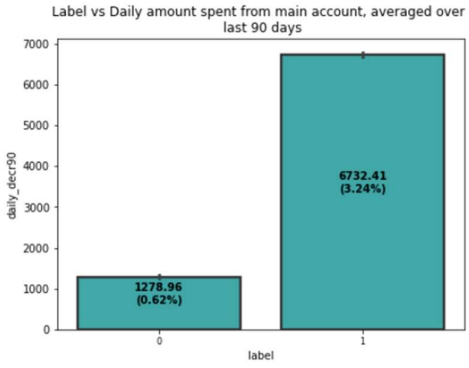
In [274]:
```
# Visualizing the age on cellular network in days and whether the user paid back the credit amount within 5 days of issuing the l
plt.figure(figsize=(6,5))
ax = sns.barplot(df['label'],df['aon'], data=df,palette="gnuplot",linewidth=2.3, edgecolor=".2");
index=0

for i in ax.patches:
    height = round(i.get_height(),2)
    total = len(df["aon"])
    percentage = f"{height}\n({round(height*100/total,2)}%)"
    plt.text(index,height/2,percentage,ha="center",fontweight="bold")
    index += 1
plt.title('Comparision between label and aon(age on cellular network in days)')
plt.show()
```



Observations:

From the above bar plot we can observe that the defaulter rate is higher where the user age on cellular network in days is high which has around 8871 counts (in days). Customers with high value of Age on cellular network in days(aon) are maximum defaulters(who have not paid there loan amount-0).

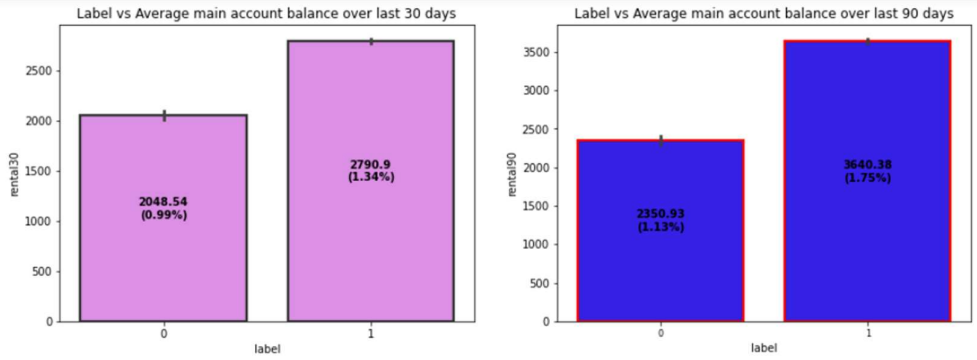




Observations:

Customers with high value of Daily amount spent from main account, averaged over last 30 days (in Indonesian Rupiah)(daily_decr30) are maximum Non-defaulters(who have paid there loan amount-1). Customers with high value of Daily amount spent from main account, averaged over last 90 days (in Indonesian Rupiah)(daily_decr90) are maximum Non-defaulters(who have paid there loan amount-1).

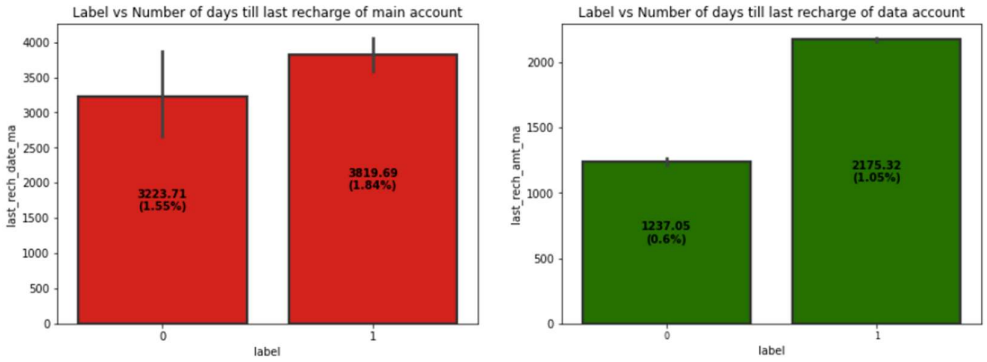
Hence,Most of the users who have paid back the credit amount within 5 days of issuing loan, they have high rate of daily amount spent from the account over last 30 days and 90 days which have the count around 5940 and 6732 respectively. The users who have spent daily amount from main account over last 30 days and 90 days have always paid back the loan amount within 5 days. Around 0.6% of the users failed to pay back the loan within due date.



Observations:

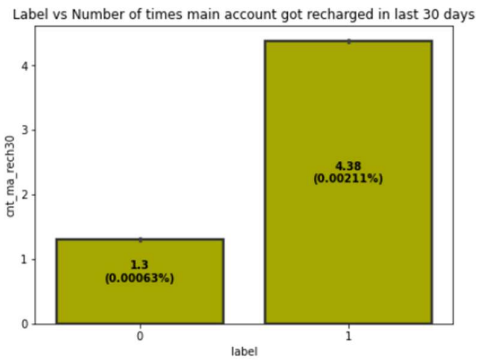
Customers with high value of Average main account balance over last 30 days(rental30) are maximum Non-defaulters(who have paid there loan amount-1).
Customers with high value of Average main account balance over last 90 days(rental90) are maximum Non-defaulters(who have paid there loan amount-1).

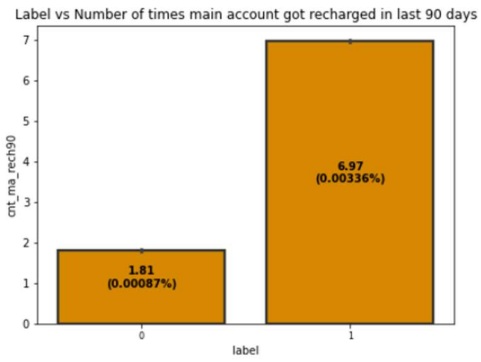
Hence,Non defaulter users have average main account balance over last 30 days and 90 days which have count around 2790 & 3640 compared to defaulter. That means the users who have average main account balance always pays back the credit amounts within 5 days. And around 1% of the users either failed to payback the loan amount within the due date or they are not paying the loan.



Observations: Customers with high Number of days till last recharge of main account(last_rech_date_ma) are maximum Non-defaulters(who have paid there loan amount-1). Customers with high value of Amount of last recharge of main account (in Indonesian Rupiah)(last_rech_amt_ma) are maximum Non-defaulters(who have paid there loan amount-1).

Hence,The users who have recharged their main account on time are most likely to pay back their loan amount within 5 days. Also some of the users who have not paid back their loan within 5 days they also recharged their main account on time. Looking at above plot of last_rech_amt_ma, we can say that if the amount of last recharge of main account is around 2000 then a greater number of people will pay back the loan amount.

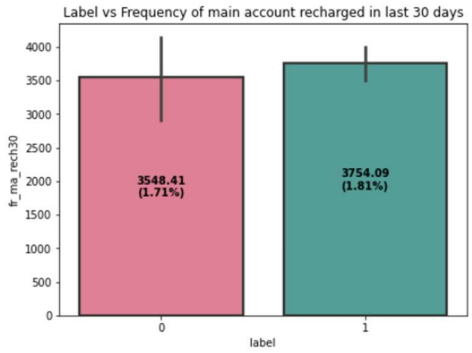


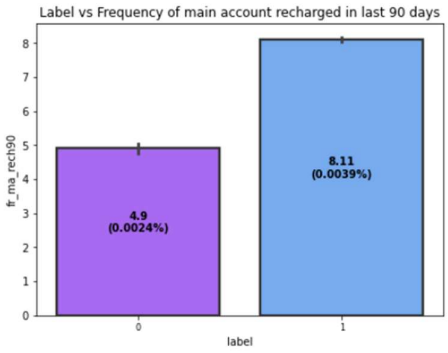


Observations:

Customers with high value of Number of times main account got recharged in last 30 days(cnt_ma_rech30) and in last 90 days(cnt_ma_rech90) are maximum Non-defaulters(who have paid there loan amount-1).

Hence,The non defaulters got recharged their main account more than 4 times in last 30 days and defaulters used to recharge their main account 1 time. The users who have paid back their loan within 5 days have got recharged their main account upto 7 times in last 90 days and the users who have not been paid loan within due date, they have got recharged their main account twice in last 90 days. From both the plots we can say that the users who got recharged their main account maximum times, they are able to pay back their loan amount within 5 days compared to the users who got their main account recharged less than 2 times.

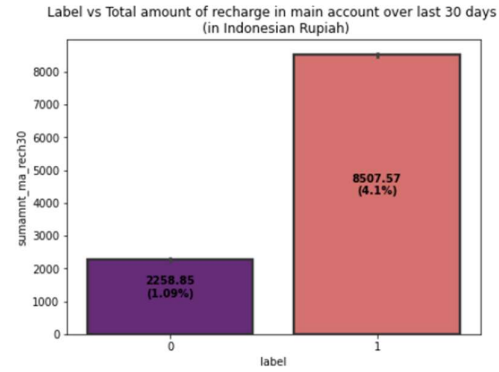


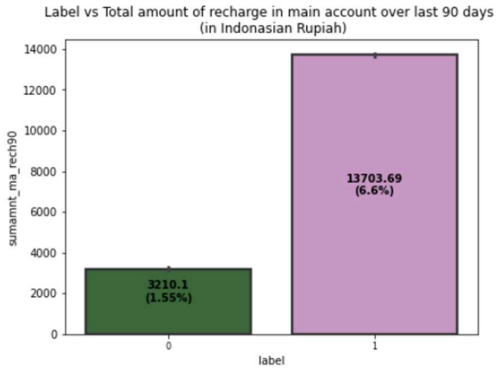


Observations:

Customers with high value of Frequency of main account recharged in last 30 days(fr_ma_rech30) and in last 90 days(fr_ma_rech90)are maximum Non-defaulters(who have paid there loan amount-1) and also the count is high for defaulters comparitively Non-defaulters are more in number.

The count of defaulters and non-defaulters is almost similar for the frequency of main account recharged in last 30 days. They didn't pay back the loan within 5 days. Which means there it is not contributing more for prediction. The frequency of main account recharged in last 90 days is increased for non-defaulters compared to defaulters. From the frequency of main account recharged in last 30 days & 90 days we have seen the users with low frequency are causing huge losses, company should implement some kind of strategies to reduce that like send SMS alerts for notification.

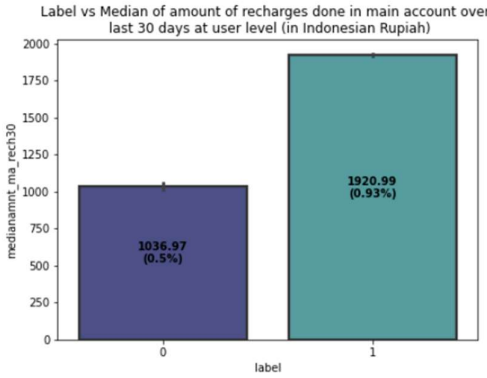


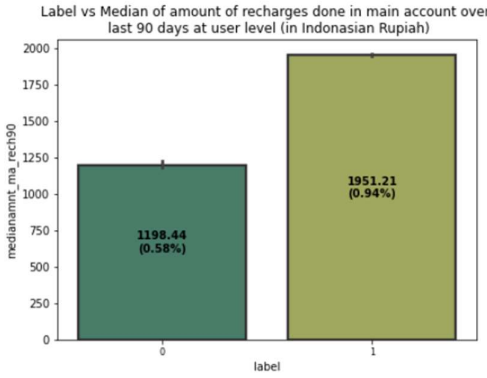


Observations:

Customers with high value of Total amount of recharge in main account over last 30 days (in Indonesian Rupiah)(sumamnt_ma_rech30) and in last 90 days (in Indonesian Rupiah) (sumamnt_ma_rech90) are maximum Non-defaulters(who have paid there loan amount-1).

Hence,The users who failed to pay back the loan within 5 days have less amount of recharge in their main account over last 30 days which is around 2000-2400 (in Indonesian Rupiah). And the users who paid back their loan within 5 days, they are recharging their main account more than 8000 (in Indonesian Rupiah) in last 30 days. The users who have paid their loan amount within 5 days have the total amount of recharge in their main account around 13700 (Indonesian Rupiah) in last 90 days while the defaulters have their total amount of recharge around 3200 (Indonesian Rupiah) over last 90 days.

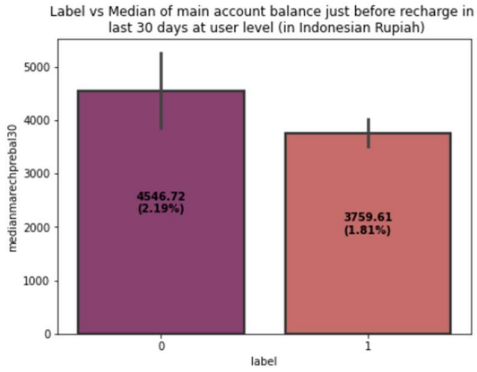


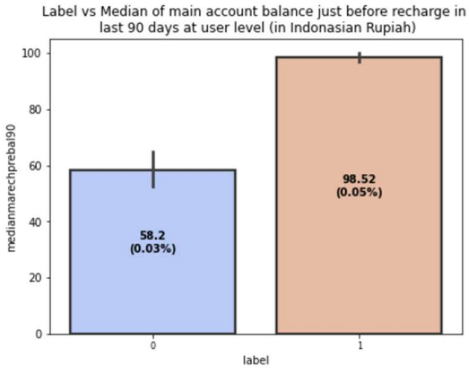


Observations:

Customers with high value of Median of amount of recharges done in main account over last 30 days at user level (in Indonesian Rupiah) (medianamnt_ma_rech30) and in last 90 days at user level (in Indonesian Rupiah)(medianamnt_ma_rech90) are maximum Non-defaulters(who have paid there loan amount-1).

Hence,The users who have done their median amount of recharge of 1920 (Indonesian Rupiah) in main account over last 30 days have successfully paid their credit amount within 5 days of issuing loan while the users who have done amount recharge of 1036 have failed to pay back the loan within due date. Similar to 30 days data, here also the users who have done their median amount recharge of 1950 in their main account over last 90 days they have paid back their credit amount within 5 days while the users having their median amount 1198 have not paid the loan within 5 days.

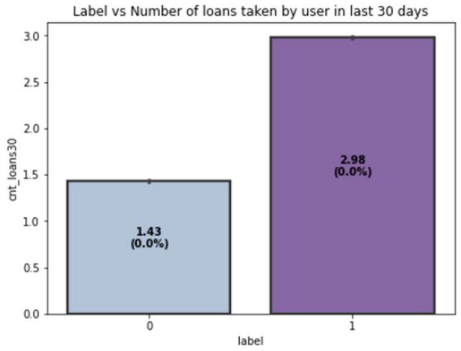


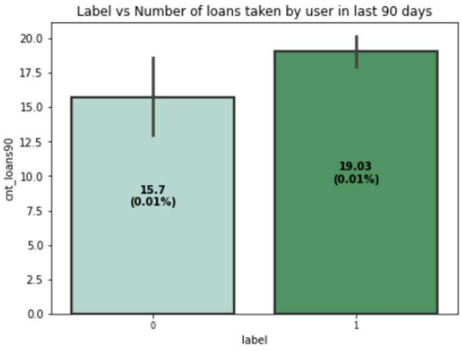


Observations:

Customers with high value of Median of main account balance just before recharge in last 30 days at user level (in Indonesian Rupiah) (medianmarechprebal30) and in last 90 days at user level (in Indonesian Rupiah)(medianmarechprebal90) are maximum defaulters(who have not paid there loan amount-0).

In 30 days data, the median of main account balance for defaulters are around 4500 (Indonesian Rupiah) which is high compared to non-defaulters. Which means increasing median of main account balance just before recharge in last 30 days at user level, increasing the probability to being defaulter. In last 90 days data, the median of main account balance for non-defaulters are around 100 (Indonesian Rupiah) which is high compared to defaulters. Which means increasing median of main account balance just before recharge in last 90 days at user level, increasing the probability of being non-defaulters.

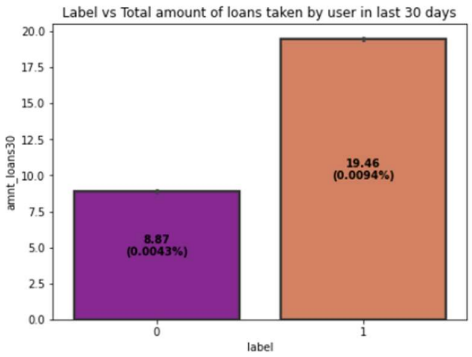


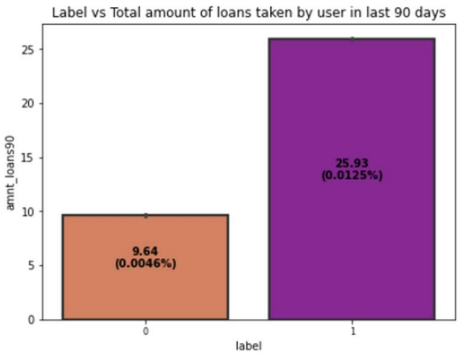


Observations:

Customers with high value of Number of loans taken by user in last 30 days(cnt_loans30) and in last 90 days(cnt_loans90) are maximum Non-defaulters(who have paid there loan amount-1).

And Defaulters have taken 1 loan in last 30 days that is when a person takes loan amount for 1 time in last 30 days the chances of not paying back the credit amount are higher. And the users who have paid back the loan, they have taken maximum number of 3 loans in last 30 days data. In 90 days data, the number of loans taken by the defaulters are highly increasing also increasing the probability to being defaulter. Also, the number of loans taken by non-defaulters being decreased in last 90 days when compared to 30 days data.

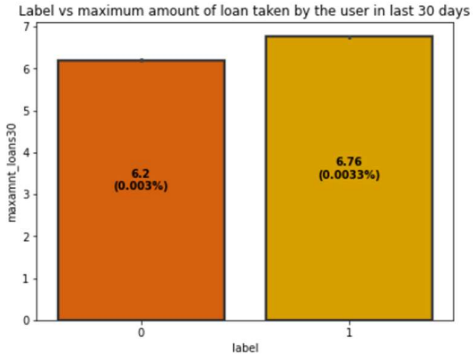


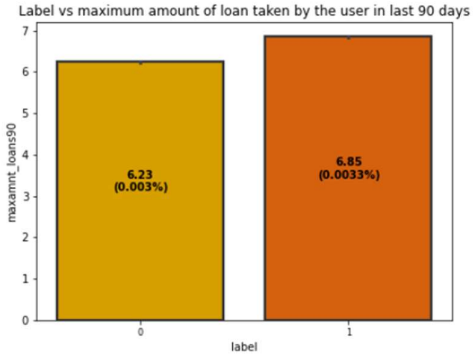


Observations:

Customers with high value of Total amount of loans taken by user in last 30 days(amnt_loans30) and in last 90 days(amnt_loans90)are maximum Non-defaulters(who have paid there loan amount-1).

The total amount of loans taken by the defaulters in last 30 days are in the range of 7.5-10 while the non-defaulters have taken upto 20 loans in last 30 days. The total amount of loans taken by the defaulters in last 90 days are upto 10 and the non- defaulters have taken total amount of loans upto 26 in last 90 days. So, from the above plot we can conclude that when the total number of loans taken by the users in last 90 days is below 10, then the chances of not paying back the loan amount are high.

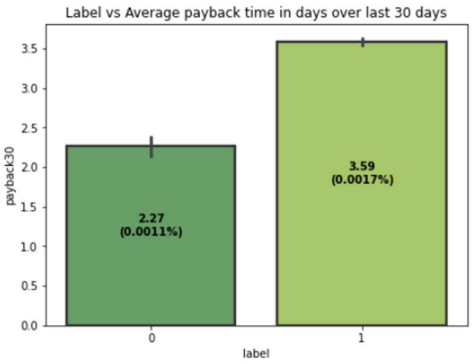


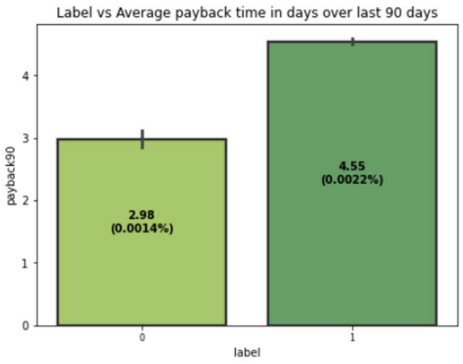


Observations:

Customers with high value of maximum amount of loan taken by the user in last 30 days(maxamnt_loans30) and in last 90 days(maxamnt_loans90)are maximum Non-defaulters(who have paid there loan amount-1).

The maximum amount of loan taken by the user in last 30 days and 90 days are almost same. The maximum amount of loan taken by the defaulters and non-defaulters are upto 6 and 7 respectively in last 30 and 90 days. So from the plot we can say that whenever the user takes the maximum loan amount of 6, then only some users may not pay back the loan amount.

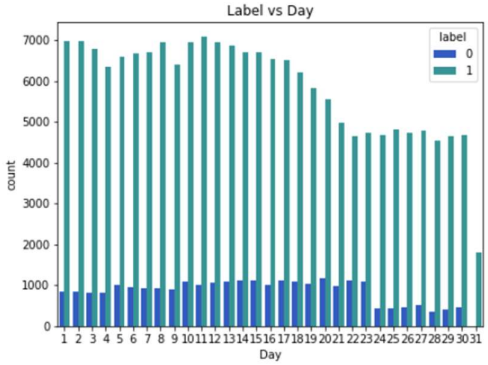


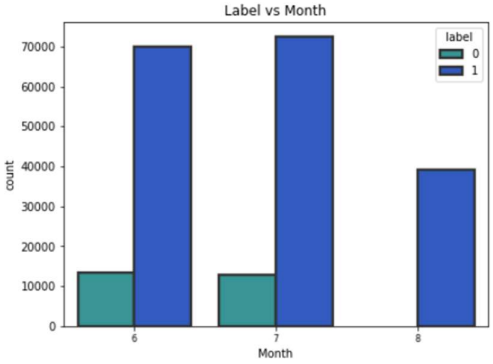


Observations:

Customers with high value of Average payback time in days over last 30 days(payback30) and in last 90 days(payback90) are maximum Non-defaulters(who have paid there loan amount-1).

The defaulters are paying back their loan in an average of 2-2.5 days and the non-defaulters are paying back their loan in an average of 3 days over last 30 days. The defaulters in last 90 days, are paying back their loan in an average of 3 days and non-defaulters are paying back their loan in 4-5 days over last 90 days. It is seen from the plot that when an average payback time is below 3 days over last 30 & 90 days, then defaulters' rate is high.

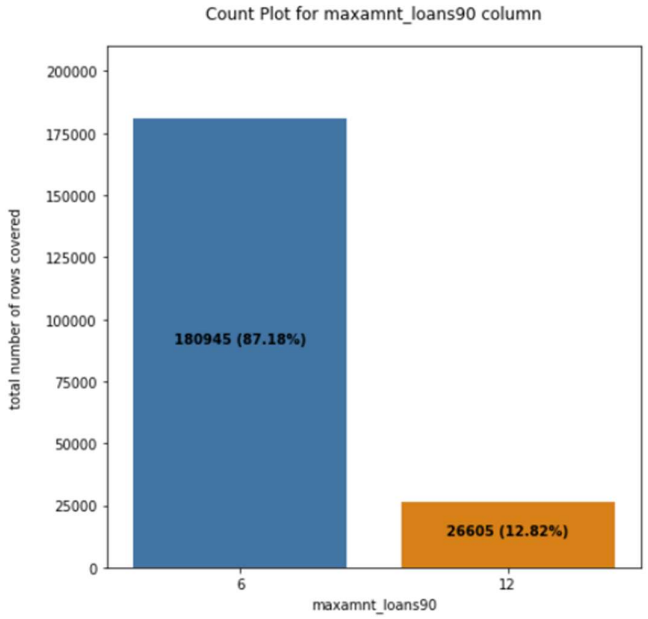




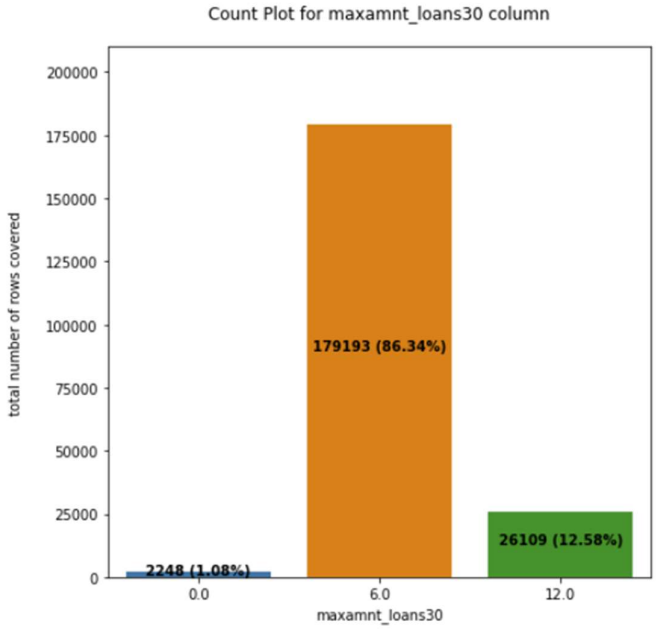
Observations:

In between 6th and 7th month maximum customers both defualters and Non-defaulters have paid there loan amount. The users who have taken loans in the month of august, they seem paying back their loan within 5 days.

Below 14th of each month all the customers have paid there loan amount.



Observation: This shows that,In 90 days maximum amount of loan are taken for 5 rupees and payed back as 6 rupees.



Observation: This shows that,In 30 days maximum amount of loan are taken for 5 rupees and payed back as 6 rupees.

Interpretation of the Results

CONCLUSION

OUTPUT

In [180]:
```
import numpy as np
a=np.array(y_test)
predicted=np.array(cat.predict(new_xtest))
df_con=pd.DataFrame({"Original":a,"Predicted":predicted},index=range(len(a)))
df_con
```

Out[180]:

	Original	Predicted
0	1	1
1	1	1
2	1	1
3	1	1
4	1	1
...	...	...
41505	1	1
41506	1	1
41507	0	1
41508	0	0
41509	0	0

41510 rows × 2 columns

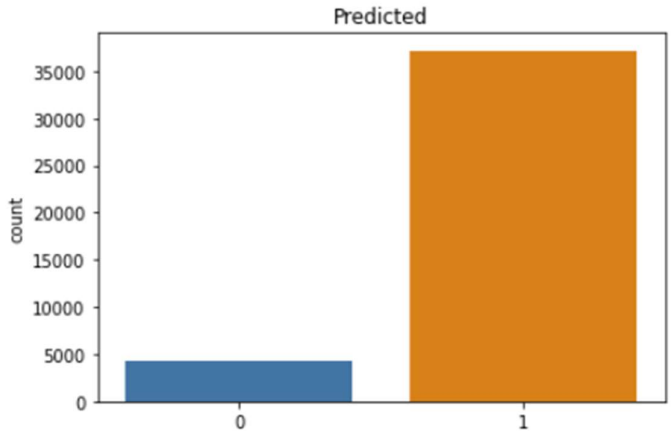
In [179]:
```
import pandas as pd
Model_scores=pd.DataFrame({})
Model_scores['Nos']=Nos
Model_scores['Model Names']=models
Model_scores['Scores']=scores
Model_scores.sort_values(by='Scores', ascending=False).style.hide_index()
```

Out[179]:

Nos	Model Names	Scores
7	CatBoostClassifier	91.970609
8	XGBoost	91.638159
6	Light Gradient Boosting Classifier	91.002168
3	DecisionTreeClassifier	87.350036
5	Gradient Boosting Classifier	87.024813
2	RandomForestClassifier	76.135871
1	LogisticRegression	75.834739
4	GaussianNB	73.784630

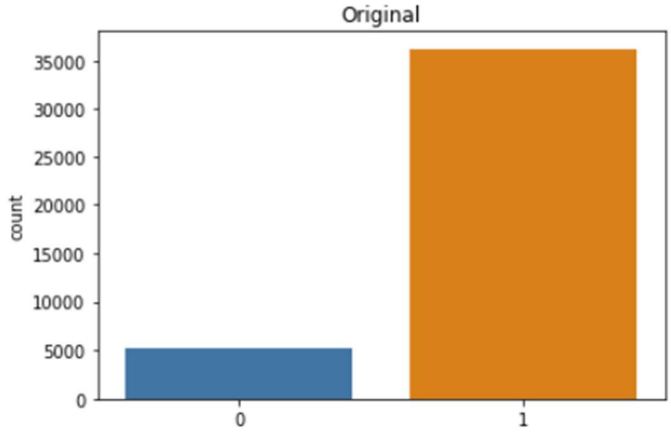
In [181]:
```
sns.countplot(x=predicted, data=df_con)
plt.title("Predicted")
```

Out[181]:
```
Text(0.5, 1.0, 'Predicted')
```



In [182]:
```
ax = sns.countplot(x=a, data=df)
plt.title("Original")
```

Out[182]:
```
Text(0.5, 1.0, 'Original')
```



MODEL SAVING:

SAVE MODEL

In [184]:
```
import pickle
filename='Micro-Credit Defaulter_Prediction _cat.pkl'
pickle.dump(cat,open(filename,'wb'))
```

In [185]:
```
df_con.to_csv("Micro-Credit Defaulter_Prediction _cat.csv",sep='\t')
```

Inferences:

The Defaulter cases of Micro card credit company can be monitored and can improve them as Non Defaulters status by concentrating and changing the following factors mainly,

cnt_ma_rech30 ----Number of times main account got recharged in last 30 days

cnt_ma_rech90 ----Number of times main account got recharged in last 90 days

sumamnt_ma_rech90 ----Total amount of recharge in main account over last 90 days (in Indonasian Rupiah)

sumamnt_ma_rech30 ---Total amount of recharge in main account over last 30 days (in Indonasian Rupiah)

amnt_loans90 ----Total amount of loans taken by user in last 90 days

amnt_loans30 ----Total amount of loans taken by user in last 30 days

cnt_loans30 ----Number of loans taken by user in last 30 days

The following cases is more related with defaulter cases.Deep Analysis can be made and Improve the results on by considering the following factors too:

cnt_loans90 ----Number of loans taken by user in last 90 days

rental30 and rental90------ Average main account balance over last 30 and 90 days

last_rech_amnt_ma------ Amount of last recharge of main account (in Indonesian Rupiah)

fr_amnt_rech_30 and fr_amnt_rech_90----- Frequency of main account recharged in last 30 and 90 days

median_amnt_loans_30 and median_amnt_loans_90-------- Median of amounts of loan taken by the user in last 30 and 90 days

In aug why payback is low than other two months?

Why after calendar day 14, we foresee less payback?

Why all take for 5 Rupiah pay back loan (6 Rupiah).

The company's customer retention and growth can also be improved mainly by considering the factors

daily_decr90 ------ Daily amount spent from main account, averaged over last 90 days (in Indonesian Rupiah)

fr_ma_rech90------ Frequency of main account recharged in last 90 days

CONCLUSION

Key Findings and Conclusions of the Study

Customer selection & Improvement can be clearly understood from outcome of the model.
Key factors which needs to be overlooked to define a customer turning defaulter or non-defaulter, is listed as below,

1. Number of loans taken in last 90 days
2. Rental amount spent across last 30 & 90 days.
3. Last recharge amount
4. Frequency of recharge done in last 30 & 90 days.
5. Median amount of loans taken in last 30 & 90 days.

Critical elements to be better understood & suitable solution to be developed for improvement of credit scheme is as below,

Reason for

- Less customer payback post 14th calendar day of every month?
- Less customer payback frequency in Aug month?
- Frequency of more people opting for 5 Rupiah credit over 10 Rupiah credit plan.

Learning Outcomes of the Study in respect of Data Science

From the above models CatBoostClassifier performs well. Because,CatBoost is the only boosting algorithm with very less prediction time. It is comparatively 8x faster than XGBoost while predicting.It uses symmetric trees, that makes it to have a fast inference. Its boosting schemes helps to reduce over fitting and improves quality of the model. It supports sophisticated categorical features. So we save this model for prediction

We faced multicollinearity,variable dependencies and unrealistic values in data set issue because too many features in the dataset.Eventhough the number of features after reduced from 37 to 28.we still face multicollinearity issue.So we dropped unwanted and redundant columns to overcome this.We handled unrealistic values in dataset by dropping zero value rows and making absolute value of the data values etc., And also Cat Boost Classifier because it handles the problem of multicollinearity very well.

CatBoost has some useful benefits, with easy implementation. Some of the main features of this competitive library are that even without parameter tuning the default parameters provide

for great results, categorical features do not need preprocessing, quick computation, increase in accuracy with less overfitting, and lastly, efficient predictions.

CatBoost is that it is easy to use, efficient, and works especially well with categorical variables. As the name implies, CatBoost means '***categorical***' boosting. It is quicker to use than, say, XGBoost, because it does not require the use of pre-processing your data, which can take the most amount of time in model building process.

In classification, a permutation is performed randomly, then a calculation is performed from a standard formula unique to CatBoost (***ordered target encoding***):

```
target_average = countInClass + prior / totalCount + 1
```

CatBoost has some useful benefits, with easy implementation. Some of the main features of this competitive library are that even without parameter tuning the default parameters provide for great results, categorical features do not need preprocessing, quick computation, increase in accuracy with less overfitting, and lastly, efficient predictions.

Thus this Cat Boost Classifier Model performs well for this dataset.so we saved this model.

Limitations of this work and Scope for Future Work

- It needs to build deep decision trees to recover dependencies in data in case of features with high cardinality. ...
- Doesn't work for unknown category values, i.e., the values that don't exist in the learn dataset.
- Understanding in depth of variable dependencies and handling can solve this problem.

For Implementation Code check my [srividya89/Micro-Credit-Defaulter-Project---Internship-19 (github.com)]