
# MICRO CREDIT PROJECT

Submitted by:

**PROMISE AZOM**

# ACKNOWLEDGMENT

I would like to express my appreciation to Datatrained Education and Fliprobo technologies for their support in bringing me this far in my Data Science Journey. Your lectures and videos made the difference.

I also want to thank my wife and children for their patience.

# INTRODUCTION

- **Business Problem Framing**

A Microfinance Institution (MFI) is an organization that offers financial services to low income populations. MFS becomes very useful when targeting especially the unbanked poor families living in remote areas with not much sources of income. The Microfinance services (MFS) provided by MFI are Group Loans, Agricultural Loans, Individual Business Loans and so on.

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

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

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.

Exercise:

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.

Points to Remember:

- 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. You may have to observe them and treat them with a suitable explanation.
- You might come across outliers in some features which you need to handle as per your understanding. Keep in mind that data is expensive and we cannot lose more than 7-8% of the data.

  Find Enclosed the Data Description File and The Sample Data for the Modeling Exercise.

- ## Conceptual Background of the Domain Problem

  This is critical stage in any machine learning process. It involves brainstorming and coming up with as many hypotheses as possible about what could affect the target variable. It facilitates in exploring the data at hand more efficiently and effectively. Domain Knowledge should be done before seeing the data or else we will end up with biased hypotheses. Below are some anticipated assertions on the problem statement.

Lenders don't want to make loans to people who can't pay them back. Hence, there are certain factors that can affect micro credit Loan:.

- Inflation
- Market condition
- Demand and supply
- Business Environment
- consumer behaviour

- **Review of Literature**

  The below key operations will be adopted:

  - Exploratory Data Analysis (EDA)
  - Data Preprocessing (Univariate,Bivariate,Multivariate)
  - Feature Selection
  - Metrics Measurement
  - Model Execution
  - Hyperparameter Tunning
  - Model Saving

- **Motivation for the Problem Undertaken**

  Our objective is 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.
  yield high returns. Further, the model will be a good way for the management to understand the pricing dynamics of a new market.

# Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem

  Mathematical Modelling:
  - Update of null values
  - Encoding

  Statistical Modelling:
  - Variance Inflation factor
  - SelectKBest: fscore, ANOVA

  Analytical Modelling:
  - Visualization Techniques: Density Plot, Scatter Plot, Count Plot, Boxplot

- **Data Sources and their formats**
  - One dataset was provided.
  - Train Data contains 209,593 entries each having 37 variables (Target variable inclusive).
  - Data contains no Null values.
  - Data contains numerical as well as categorical variable.

**See below table showing summary of the Variables:**

| S/N | Variable | Variable Type | Data Type | Data Form | Null | Variable Attributes |
|---|---|---|---|---|---|---|
| **1** | Unnamed: 0 | Continous | int64 | Numerical | 0 | Unanmed |
| **2** | label | Categorical | int64 | Numerical | 0 | Flag indicating whether the user paid back the credit amount within 5 days of issuing the loan{1:success, 0:failure} |
| **3** | msisdn | Categorical | object | Non-Numerical | 0 | mobile number of user |
| **4** | aon | Continous | float64 | Numerical | 0 | age on cellular network in days |
| **5** | daily_decr30 | Continous | float64 | Numerical | 0 | Daily amount spent from main account, averaged over last 30 days (in Indonesian Rupiah) |

| | | | | | | |
|---|---|---|---|---|---|---|
| **6** | daily_decr90 | Continous | float64 | Numerical | 0 | Daily amount spent from main account, averaged over last 90 days (in Indonesian Rupiah) |
| **7** | rental30 | Continous | float64 | Numerical | 0 | Average main account balance over last 30 days |
| **8** | rental90 | Continous | float64 | Numerical | 0 | Average main account balance over last 90 days |
| **9** | last_rech_date_ma | Continous | float64 | Numerical | 0 | Number of days till last recharge of main account |
| **10** | last_rech_date_da | Continous | float64 | Numerical | 0 | Number of days till last recharge of data account |
| **11** | last_rech_amt_ma | Continous | int64 | Numerical | 0 | Amount of last recharge of main account (in Indonesian Rupiah) |
| **12** | cnt_ma_rech30 | Continous | int64 | Numerical | 0 | Number of times main account got recharged in last 30 days |
| **13** | fr_ma_rech30 | Continous | float64 | Numerical | 0 | Frequency of main account recharged in last 30 days |
| **14** | sumamnt_ma_rech30 | Continous | float64 | Numerical | 0 | Total amount of recharge in main account over last 30 days (in Indonesian Rupiah) |
| **15** | medianamnt_ma_rech30 | Continous | float64 | Numerical | 0 | Median of amount of recharges done in main account over last 30 days at user level (in Indonesian Rupiah) |
| **16** | medianmarechprebal30 | Continous | float64 | Numerical | 0 | Median of main account balance just before recharge in last 30 days at user level (in Indonesian Rupiah) |
| **17** | cnt_ma_rech90 | Continous | int64 | Numerical | 0 | Number of times main account got recharged in last 90 days |
| **18** | fr_ma_rech90 | Continous | int64 | Numerical | 0 | Frequency of main account recharged in last 90 days |
| **19** | sumamnt_ma_rech90 | Continous | int64 | Numerical | 0 | Total amount of recharge in main account over last 90 days (in Indonasian Rupiah) |
| **20** | medianamnt_ma_rech90 | Continous | float64 | Numerical | 0 | Median of amount of recharges done in main account over last 90 days at user level (in Indonasian Rupiah) |

| | | | | | | |
|---|---|---|---|---|---|---|
| **21** | medianmarechprebal90 | Continous | float64 | Numerical | 0 | Median of main account balance just before recharge in last 90 days at user level (in Indonasian Rupiah) |
| **22** | cnt_da_rech30 | Continous | float64 | Numerical | 0 | Number of times data account got recharged in last 30 days |
| **23** | fr_da_rech30 | Continous | float64 | Numerical | 0 | Frequency of data account recharged in last 30 days |
| **24** | cnt_da_rech90 | Continous | int64 | Numerical | 0 | Number of times data account got recharged in last 90 days |
| **25** | fr_da_rech90 | Continous | int64 | Numerical | 0 | Frequency of data account recharged in last 90 days |
| **26** | cnt_loans30 | Continous | int64 | Numerical | 0 | Number of loans taken by user in last 30 days |
| **27** | amnt_loans30 | Continous | int64 | Numerical | 0 | Total amount of loans taken by user in last 30 days |
| **28** | maxamnt_loans30 | Continous | float64 | Numerical | 0 | maximum amount of loan taken by the user in last 30 days |
| **29** | medianamnt_loans30 | Continous | float64 | Numerical | 0 | Median of amounts of loan taken by the user in last 30 days |
| **30** | cnt_loans90 | Continous | float64 | Numerical | 0 | Number of loans taken by user in last 90 days |
| **31** | amnt_loans90 | Continous | int64 | Numerical | 0 | Total amount of loans taken by user in last 90 days |
| **32** | maxamnt_loans90 | Continous | int64 | Numerical | 0 | maximum amount of loan taken by the user in last 90 days |
| **33** | medianamnt_loans90 | Continous | float64 | Numerical | 0 | Median of amounts of loan taken by the user in last 90 days |
| **34** | payback30 | Continous | float64 | Numerical | 0 | Average payback time in days over last 30 days |
| **35** | payback90 | Continous | float64 | Numerical | 0 | Average payback time in days over last 90 days |
| **36** | pcircle | Categorical | object | Non-Numerical | 0 | telecom circle |
| **37** | pdate | Categorical | object | Non-Numerical | | date |

**NB:** The label is the Dependent Variable (Target/Label) while the rest are the independent variables (Features).

Also see snap shots of data…

## Full view of Data

| | Unnamed: 0 | label | msisdn | aon | daily_decr30 | daily_decr90 | rental30 | rental90 | last_rech_date_ma | last_rech_date_da | ... | maxamnt_loans30 | mec |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **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 | |
| **209589** | 209590 | 1 | 95583I84455 | 1075.0 | 36.936000 | 36.936000 | 1728.36 | 1728.36 | 4.0 | 0.0 | ... | 6.0 | |
| **209590** | 209591 | 1 | 28556I85350 | 1013.0 | 11843.111670 | 11904.350000 | 5861.83 | 8893.20 | 3.0 | 0.0 | ... | 12.0 | |
| **209591** | 209592 | 1 | 59712I82733 | 1732.0 | 12488.228330 | 12574.370000 | 411.83 | 984.58 | 2.0 | 38.0 | ... | 12.0 | |
| **209592** | 209593 | 1 | 65061I85339 | 1581.0 | 4489.362000 | 4534.820000 | 483.92 | 631.20 | 13.0 | 0.0 | ... | 12.0 | |

209593 rows × 37 columns

## First Five rows

| | Unnamed: 0 | label | msisdn | aon | daily_decr30 | daily_decr90 | rental30 | rental90 | last_rech_date_ma | last_rech_date_da | ... | maxamnt_loans30 | medianam |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **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 | |

5 rows × 37 columns

## Last Five rows

| | Unnamed: 0 | label | msisdn | aon | daily_decr30 | daily_decr90 | rental30 | rental90 | last_rech_date_ma | last_rech_date_da | ... | maxamnt_loans30 | mec |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **209588** | 209589 | 1 | 22758I85348 | 404.0 | 151.872333 | 151.872333 | 1089.19 | 1089.19 | 1.0 | 0.0 | ... | 6.0 | |
| **209589** | 209590 | 1 | 95583I84455 | 1075.0 | 36.936000 | 36.936000 | 1728.36 | 1728.36 | 4.0 | 0.0 | ... | 6.0 | |
| **209590** | 209591 | 1 | 28556I85350 | 1013.0 | 11843.111670 | 11904.350000 | 5861.83 | 8893.20 | 3.0 | 0.0 | ... | 12.0 | |
| **209591** | 209592 | 1 | 59712I82733 | 1732.0 | 12488.228330 | 12574.370000 | 411.83 | 984.58 | 2.0 | 38.0 | ... | 12.0 | |
| **209592** | 209593 | 1 | 65061I85339 | 1581.0 | 4489.362000 | 4534.820000 | 483.92 | 631.20 | 13.0 | 0.0 | ... | 12.0 | |

5 rows × 37 columns

## Random Four samples

| | Unnamed: 0 | label | msisdn | aon | daily_decr30 | daily_decr90 | rental30 | rental90 | last_rech_date_ma | last_rech_date_da | ... | maxamnt_loans30 | medi |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **104141** | 104142 | 1 | 04235I84459 | 1836.0 | 4.560000 | 4.56 | 1583.46 | 1583.46 | 4.0 | 0.0 | ... | 6.0 | |
| **200167** | 200168 | 1 | 07747I88648 | 322.0 | 6197.256667 | 6211.77 | 7049.75 | 7483.19 | 11.0 | 10.0 | ... | 6.0 | |
| **148069** | 148070 | 1 | 52059I88688 | 964.0 | 8693.000000 | 13904.31 | 599.40 | 1438.80 | 16.0 | 0.0 | ... | 6.0 | |
| **100591** | 100592 | 1 | 95714I90584 | 591.0 | 6274.519000 | 6317.17 | 2227.26 | 3096.03 | 1.0 | 0.0 | ... | 6.0 | |

4 rows × 37 columns

Data Description(Continous Data)

| | Unnamed: 0 | label | aon | daily_decr30 | daily_decr90 | rental30 | rental90 | last_rech_date_ma | last_rech_date_da | last_r |
|---|---|---|---|---|---|---|---|---|---|---|
| count | 209593.000000 | 209593.000000 | 209593.000000 | 209593.000000 | 209593.000000 | 209593.000000 | 209593.000000 | 209593.00000 | 209593.000000 | 20 |
| mean | 104797.000000 | 0.875177 | 8112.343445 | 5381.402289 | 6082.515068 | 2692.581910 | 3483.406534 | 3755.84780 | 3712.202921 | |
| std | 60504.431823 | 0.330519 | 75696.082531 | 9220.623400 | 10918.812767 | 4308.586781 | 5770.461279 | 53905.89223 | 53374.833430 | |
| min | 1.000000 | 0.000000 | -48.000000 | -93.012667 | -93.012667 | -23737.140000 | -24720.580000 | -29.00000 | -29.000000 | |
| 25% | 52399.000000 | 1.000000 | 246.000000 | 42.440000 | 42.692000 | 280.420000 | 300.260000 | 1.00000 | 0.000000 | |
| 50% | 104797.000000 | 1.000000 | 527.000000 | 1469.175667 | 1500.000000 | 1083.570000 | 1334.000000 | 3.00000 | 0.000000 | |
| 75% | 157195.000000 | 1.000000 | 982.000000 | 7244.000000 | 7802.790000 | 3356.940000 | 4201.790000 | 7.00000 | 0.000000 | |
| max | 209593.000000 | 1.000000 | 999860.755200 | 265926.000000 | 320630.000000 | 198926.110000 | 200148.110000 | 998650.37770 | 999171.809400 | 5 |

8 rows × 34 columns

Data Description(Categorical Data)

| | msisdn | pcircle | pdate |
|---|---|---|---|
| count | 209593 | 209593 | 209593 |
| unique | 186243 | 1 | 82 |
| top | 04581I85330 | UPW | 7/4/2016 |
| freq | 7 | 209593 | 3150 |

---

- **Data Pre-processing Done**
  **Observations and Assumptions on Data Cleaning**
  Observations

  - There are no null values

  - There is an unwanted column named 'Unnamed: 0'

  - There is no mismatched datatype

  - There will be need for datatype conversion

  **Assumptions**

  1. We shall drop the unwanted columns named 'Unnamed: 0'

  2. The fillna method will not be adopted since there are null values

3. There will be need for datatype conversion

## Dropping of unwanted column – Unnamed: 0

| | label | msisdn | aon | daily_decr30 | daily_decr90 | rental30 | rental90 | last_rech_date_ma | last_rech_date_da | last_rech_amt_ma | ... | maxamnt_loans |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 21408I70789 | 272.0 | 3055.050000 | 3065.150000 | 220.13 | 260.13 | 2.0 | 0.0 | 1539 | ... | 6 |
| **1** | 1 | 76462I70374 | 712.0 | 12122.000000 | 12124.750000 | 3691.26 | 3691.26 | 20.0 | 0.0 | 5787 | ... | 12 |
| **2** | 1 | 17943I70372 | 535.0 | 1398.000000 | 1398.000000 | 900.13 | 900.13 | 3.0 | 0.0 | 1539 | ... | 6 |
| **3** | 1 | 55773I70781 | 241.0 | 21.228000 | 21.228000 | 159.42 | 159.42 | 41.0 | 0.0 | 947 | ... | 6 |
| **4** | 1 | 03813I82730 | 947.0 | 150.619333 | 150.619333 | 1098.90 | 1098.90 | 4.0 | 0.0 | 2309 | ... | 6 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **209588** | 1 | 22758I85348 | 404.0 | 151.872333 | 151.872333 | 1089.19 | 1089.19 | 1.0 | 0.0 | 4048 | ... | 6 |
| **209589** | 1 | 95583I84455 | 1075.0 | 36.936000 | 36.936000 | 1728.36 | 1728.36 | 4.0 | 0.0 | 773 | ... | 6 |
| **209590** | 1 | 28556I85350 | 1013.0 | 11843.111670 | 11904.350000 | 5861.83 | 8893.20 | 3.0 | 0.0 | 1539 | ... | 12 |
| **209591** | 1 | 59712I82733 | 1732.0 | 12488.228330 | 12574.370000 | 411.83 | 984.58 | 2.0 | 38.0 | 773 | ... | 12 |
| **209592** | 1 | 65061I85339 | 1581.0 | 4489.362000 | 4534.820000 | 483.92 | 631.20 | 13.0 | 0.0 | 7526 | ... | 12 |

209593 rows × 36 columns

**4**. Encoding

- We are fully aware we cannot run an exhaustive EDA on non-numerical data.

- This makes it necessary for us to convert all non-numerical data into numerical ones.

- Hence Label encoding was used on all the object columns.

## Encoded data

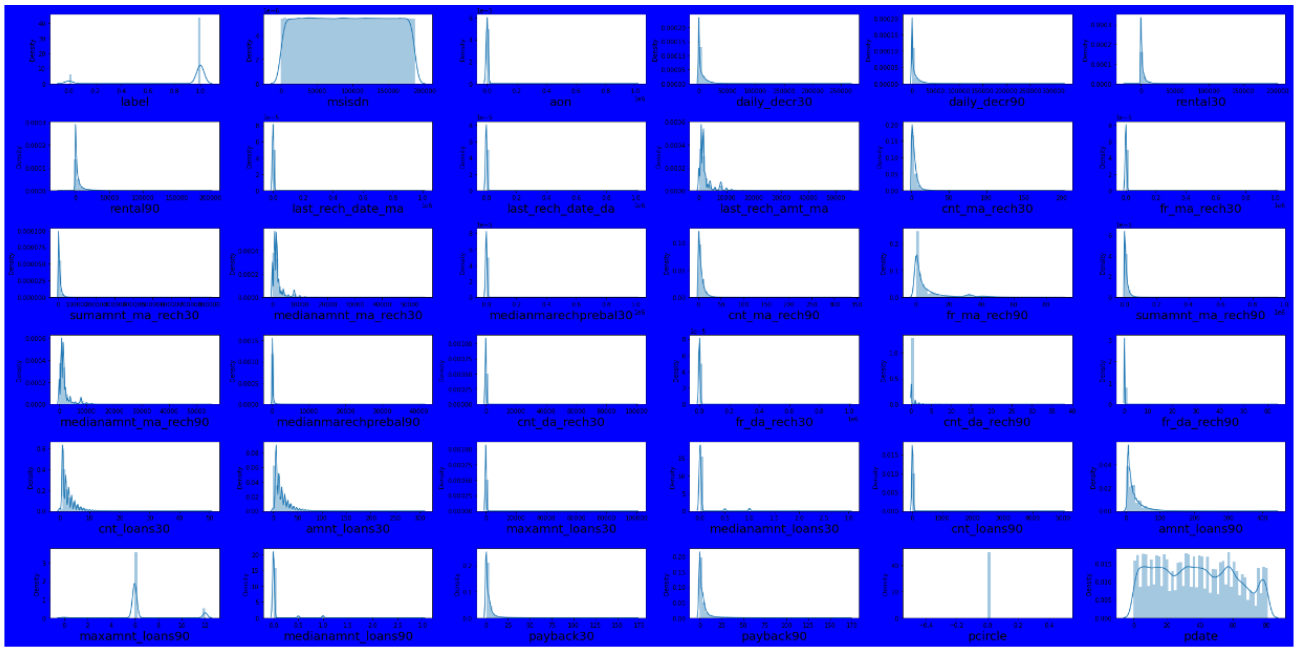
| | label | msisdn | aon | daily_decr30 | daily_decr90 | rental30 | rental90 | last_rech_date_ma | last_rech_date_da | last_rech_amt_ma | ... | maxamnt_loans30 | r |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 40191 | 272.0 | 3055.050000 | 3065.150000 | 220.13 | 260.13 | 2.0 | 0.0 | 1539 | ... | 6.0 | |
| **1** | 1 | 142291 | 712.0 | 12122.000000 | 12124.750000 | 3691.26 | 3691.26 | 20.0 | 0.0 | 5787 | ... | 12.0 | |
| **2** | 1 | 33594 | 535.0 | 1398.000000 | 1398.000000 | 900.13 | 900.13 | 3.0 | 0.0 | 1539 | ... | 6.0 | |
| **3** | 1 | 104157 | 241.0 | 21.228000 | 21.228000 | 159.42 | 159.42 | 41.0 | 0.0 | 947 | ... | 6.0 | |
| **4** | 1 | 6910 | 947.0 | 150.619333 | 150.619333 | 1098.90 | 1098.90 | 4.0 | 0.0 | 2309 | ... | 6.0 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **209588** | 1 | 42866 | 404.0 | 151.872333 | 151.872333 | 1089.19 | 1089.19 | 1.0 | 0.0 | 4048 | ... | 6.0 | |
| **209589** | 1 | 178248 | 1075.0 | 36.936000 | 36.936000 | 1728.36 | 1728.36 | 4.0 | 0.0 | 773 | ... | 6.0 | |
| **209590** | 1 | 53995 | 1013.0 | 11843.111670 | 11904.350000 | 5861.83 | 8893.20 | 3.0 | 0.0 | 1539 | ... | 12.0 | |
| **209591** | 1 | 111388 | 1732.0 | 12488.228330 | 12574.370000 | 411.83 | 984.58 | 2.0 | 38.0 | 773 | ... | 12.0 | |
| **209592** | 1 | 121263 | 1581.0 | 4489.362000 | 4534.820000 | 483.92 | 631.20 | 13.0 | 0.0 | 7526 | ... | 12.0 | |

209593 rows × 36 columns

- **Data Inputs- Logic- Output Relationships**

  Normal Distribution Check(Univariate Analysis)

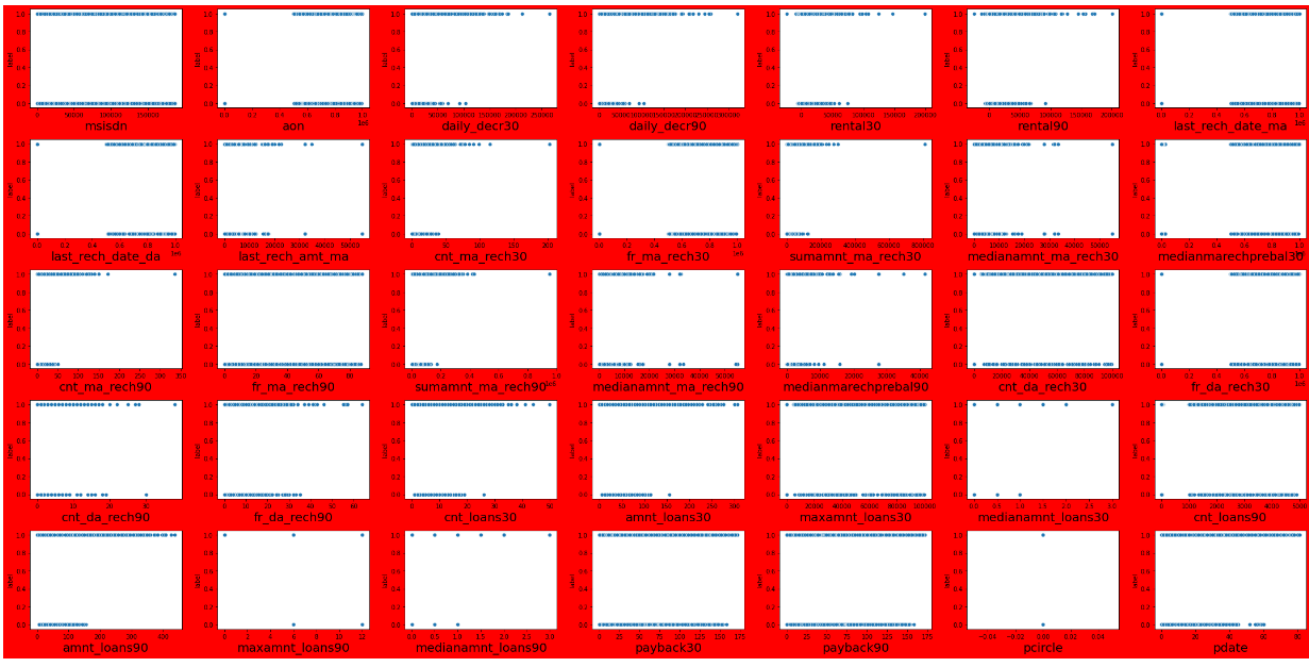
  From a density plot standpoint:




  - All the features does not obey a normal distribution, the building blocks is not in tandem with a normalized curve.
  - The normal distribution of the sales columns also has no contribution to our Model Building since its the Target variable

## Scatter Plot Check(Bivariate Analysis)

From a Scatter plot standpoint:




From the above scatter plot we can see a strong relationship between some of the features and the Target (label):

## Correlation Check (Collinearity and Multicollinearity)- Multivariate Analysis;

| | label | msisdn | aon | daily_decr30 | daily_decr90 | rental30 | rental90 | last_rech_date_ma | last_rech_date_da | last_rech_amt_m |
|---|---|---|---|---|---|---|---|---|---|---|
| **label** | 1.000000 | 0.001976 | -0.003785 | 0.168298 | 0.166150 | 0.058085 | 0.075521 | 0.003728 | 0.001711 | 0.1318( |
| **msisdn** | 0.001976 | 1.000000 | 0.000566 | 0.000717 | 0.000950 | -0.001404 | -0.000691 | -0.000928 | 0.001400 | -0.0023( |
| **aon** | -0.003785 | 0.000566 | 1.000000 | 0.001104 | 0.000374 | -0.000960 | -0.000790 | 0.001692 | -0.001693 | 0.0042( |
| **daily_decr30** | 0.168298 | 0.000717 | 0.001104 | 1.000000 | 0.977704 | 0.442066 | 0.458977 | 0.000487 | -0.001636 | 0.2758( |
| **daily_decr90** | 0.166150 | 0.000950 | 0.000374 | 0.977704 | 1.000000 | 0.434685 | 0.471730 | 0.000908 | -0.001886 | 0.2641( |
| **rental30** | 0.058085 | -0.001404 | -0.000960 | 0.442066 | 0.434685 | 1.000000 | 0.955237 | -0.001095 | 0.003261 | 0.1272( |
| **rental90** | 0.075521 | -0.000691 | -0.000790 | 0.458977 | 0.471730 | 0.955237 | 1.000000 | -0.001688 | 0.002794 | 0.1214( |
| **last_rech_date_ma** | 0.003728 | -0.000928 | 0.001692 | 0.000487 | 0.000908 | -0.001095 | -0.001688 | 1.000000 | 0.001790 | -0.0001( |
| **last_rech_date_da** | 0.001711 | 0.001400 | -0.001693 | -0.001636 | -0.001886 | 0.003261 | 0.002794 | 0.001790 | 1.000000 | -0.0001( |
| **last_rech_amt_ma** | 0.131804 | -0.002368 | 0.004256 | 0.275837 | 0.264131 | 0.127271 | 0.121416 | -0.000147 | -0.000149 | 1.0000( |
| **cnt_ma_rech30** | 0.237331 | 0.000617 | -0.003148 | 0.451385 | 0.426707 | 0.233343 | 0.230260 | 0.004311 | 0.001549 | -0.0026( |
| **fr_ma_rech30** | 0.001330 | -0.001804 | -0.001163 | -0.000577 | -0.000343 | -0.001219 | -0.000503 | -0.001629 | 0.001158 | 0.0028( |
| **sumamnt_ma_rech30** | 0.202828 | 0.001094 | 0.000707 | 0.636536 | 0.603886 | 0.272649 | 0.259709 | 0.002105 | 0.000046 | 0.4408( |
| **medianamnt_ma_rech30** | 0.141490 | 0.000668 | 0.004306 | 0.295356 | 0.282960 | 0.129853 | 0.120242 | -0.001358 | 0.001037 | 0.7946( |
| **medianmarechprebal30** | -0.004829 | -0.000238 | 0.003930 | -0.001153 | -0.000746 | -0.001415 | -0.001237 | 0.004071 | 0.002849 | -0.0023( |
| **cnt_ma_rech90** | 0.236392 | -0.001859 | -0.002725 | 0.587338 | 0.593069 | 0.312118 | 0.345293 | 0.004263 | 0.001272 | 0.0167( |
| **fr_ma_rech90** | 0.084385 | -0.004413 | 0.004401 | -0.078299 | -0.079530 | -0.033530 | -0.036524 | 0.001414 | 0.000798 | 0.1062( |
| **sumamnt_ma_rech90** | 0.205793 | -0.000936 | 0.001011 | 0.762981 | 0.768817 | 0.342306 | 0.360601 | 0.002243 | -0.000414 | 0.4187( |
| **medianamnt_ma_rech90** | 0.120855 | -0.000625 | 0.004909 | 0.257847 | 0.250518 | 0.110356 | 0.103151 | -0.000726 | 0.000219 | 0.8187( |
| **medianmarechprebal90** | 0.039300 | -0.000615 | -0.000859 | 0.037495 | 0.036382 | 0.027170 | 0.029547 | -0.001086 | 0.004158 | 0.1246( |
| **cnt_da_rech30** | 0.003827 | -0.001720 | 0.001564 | 0.000700 | 0.000661 | -0.001105 | -0.000548 | -0.003467 | -0.003628 | -0.0018( |
| **fr_da_rech30** | -0.000027 | -0.000241 | 0.000892 | -0.001499 | -0.001570 | -0.002558 | -0.002345 | -0.003626 | -0.000074 | -0.0032( |

From the above correlation statistics;

Collinearity:

- From the above we can see that most of the features have correlation with the target

Multicollinearity:

- From the heatmap we can see that the some pairs of features have noticeable correlation between them

Skewness Check

We assumed a Skewness threshold is taken as +/-0.50. Meaning any value outside +/-0.50 contains skewness. Hence all features are having skewness except:

- pcircle 0.000000
- pdate 0.138073
- msisdn 0.000719

- **State the set of assumptions (if any) related to the problem under consideration**

Assumption on Feature Selection

The selectKBest feature was used to pick out 30 features considering their fscores in descending order

| | Feature_Name | Score |
|---|---|---|
| 9 | cnt_ma_rech30 | 12509.987949 |
| 14 | cnt_ma_rech90 | 12405.449295 |
| 16 | sumamnt_ma_rech90 | 9268.915730 |
| 11 | sumamnt_ma_rech30 | 8992.305501 |
| 28 | amnt_loans90 | 8713.675655 |
| 24 | amnt_loans30 | 8486.686219 |
| 23 | cnt_loans30 | 8398.407875 |
| 2 | daily_decr30 | 6109.551285 |
| 3 | daily_decr90 | 5950.205909 |
| 34 | pdate | 5118.593612 |
| 12 | medianamnt_ma_rech30 | 4281.620838 |
| 8 | last_rech_amt_ma | 3705.466816 |
| 17 | medianamnt_ma_rech90 | 3106.655486 |
| 15 | fr_ma_rech90 | 1503.180717 |
| 29 | maxamnt_loans90 | 1494.540036 |
| 5 | rental90 | 1202.226259 |

| | | |
|---|---|---|
| 4 | rental30 | 709.505994 |
| 32 | payback90 | 508.116565 |
| 31 | payback30 | 490.705161 |
| 26 | medianamnt_loans30 | 417.544932 |
| 18 | medianmarechprebal90 | 324.207470 |
| 30 | medianamnt_loans90 | 268.169218 |
| 22 | fr_da_rech90 | 6.152145 |
| 13 | medianmarechprebal30 | 4.887167 |
| 27 | cnt_loans90 | 4.695777 |
| 19 | cnt_da_rech30 | 3.070296 |
| 1 | aon | 3.002461 |
| 6 | last_rech_date_ma | 2.913453 |
| 21 | cnt_da_rech90 | 1.884903 |
| 0 | msisdn | 0.816760 |
| 7 | last_rech_date_da | 0.613697 |
| 10 | fr_ma_rech30 | 0.370925 |
| 25 | maxamnt_loans30 | 0.012877 |
| 20 | fr_da_rech30 | 0.000146 |
| 33 | pcircle | NaN |

Assumption on Variance Inflation Factor

We used a variance inflation Factor of 7. Meaning any feature with Variance Inflation Factor greater than 7 is assumed to have a multicollinearity problem. It is not standard. The dataset demands.

In lieu of the above assumption, we will further drop the following:

a. cnt_loans30
b. amnt_loans30
c. amnt_loans90
d. sumamnt_ma_rech90
e. cnt_ma_rech90
f. sumamnt_ma_rech30
g. cnt_ma_rech30
h. rental90
i. rental30
j. daily_decr90
k. daily_decr30

- **Hardware and Software Requirements and Tools Used**

Hardware Requirments

- Device name: DESKTOP
- ProcessorIntel(R) Core(TM) i7-7500U CPU @ 2.70GHz, 2.90GHz
- Installed RAM: 8.00 GB
- System type: 64-bit operating system, x64-based processor

Software Requirments

- Jupyter Notebook
- Python Programming Language
- Libraries
    i. Pandas: Used loading the dataset
    ii. Numpy: Used for rounding up numbers
    iii. sklearn.linear_model: Used for initializing the Logistic Regression Model
    iv. sklearn.neighbors: Used for initializing the KNeighbours Classification Model
    v. sklearn.tree: Used for initializing the DecisionTree Classification Model
    vi. sklearn.ensemble: Used for initializing the ensemble Techniques/Models - RandomForest Classifier,AdaBoost Classifier, GradientBoosting Classifier, ExtraTrees Classifier
    vii. sklearn.preprocessing: Used Scaling, Power Transformation and Encoding of data
    viii. sklearn.feature_selection: Used for feature selection using SelectKBest
    ix. sklearn.model_selection: Used Data split into test and train. Also used for Initializing GridsearchCV during hyperparameter tunning

x. sklearn.metrics: Used for metrics measurment
xi. xgboost: Used for initializing the XGBoost Model
xii. statsmodels: Used to solve for multicollinearity via Variance inflation Factor
xiii. Scipy.stats: Used to remove outliers via zscore
xiv. Seaborn: Used for visualization during variate analysis
xv. Matplotlib: Also used for visualization during variate analysis

## Model/s Development and Evaluation

- Testing of Identified Approaches (Algorithms)
  The algorithm used were:
  - Standard Scaler
  - train_test_split
  - fit(x_train,y_train)

- Run and evaluate selected models
  Eight Models were used:
    - Logistic Regression

```
lr=LogisticRegression()
lr.fit(x_train,y_train)
pred_test=lr.predict(x_test)
pred_train=lr.predict(x_train)
Test_Accuracy_lr= (accuracy_score(y_test,pred_test))
Train_Accuracy_lr= (accuracy_score(y_train,pred_train))
print("Test_Accuracy ",round(accuracy_score(y_test,pred_test)*100,2))#testing score
print("Confusion Matrix ", confusion_matrix(y_test,pred_test))
print("Classification Report ", classification_report(y_test,pred_test))
```

```
Test_Accuracy  87.96
Confusion Matrix  [[    1  5048]
 [    1 36869]]
Classification Report                precision    recall  f1-score   support

           0       0.50      0.00      0.00      5049
           1       0.88      1.00      0.94     36870

    accuracy                           0.88     41919
   macro avg       0.69      0.50      0.47     41919
weighted avg       0.83      0.88      0.82     41919
```

LogisticRegression is producing good accuracy 88%. Now we will check Cross Validation score as well for overfitting(if exists).

    - KNeighbors Classifier

```
knn=KNeighborsClassifier()
knn.fit(x_train,y_train)
pred_test=knn.predict(x_test)
pred_train=knn.predict(x_train)
Test_Accuracy_knn= (accuracy_score(y_test,pred_test))
Train_Accuracy_knn= (accuracy_score(y_train,pred_train))
print("Test_Accuracy ",round(accuracy_score(y_test,pred_test)*100,2))#testing score
print("Confusion Matrix ", confusion_matrix(y_test,pred_test))
print("Classification Report ", classification_report(y_test,pred_test))
```

```
Test_Accuracy  88.35
Confusion Matrix  [[ 1780  3269]
 [ 1615 35255]]
Classification Report                precision    recall  f1-score   support

           0       0.52      0.35      0.42      5049
           1       0.92      0.96      0.94     36870

    accuracy                           0.88     41919
   macro avg       0.72      0.65      0.68     41919
weighted avg       0.87      0.88      0.87     41919
```

KNeighbors is producing good accuracy 88%. Now we will check Cross Validation score as well for overfitting(if exists).

- Decision Tree Classifier

```
dt=DecisionTreeClassifier()
dt.fit(x_train,y_train)
pred_test=dt.predict(x_test)
pred_train=dt.predict(x_train)
Test_Accuracy_dt= (accuracy_score(y_test,pred_test))
Train_Accuracy_dt= (accuracy_score(y_train,pred_train))
print("Test_Accuracy ",round(accuracy_score(y_test,pred_test)*100,2))#testing score
print("Confusion Matrix ", confusion_matrix(y_test,pred_test))
print("Classification Report ", classification_report(y_test,pred_test))
```

```
Test_Accuracy  84.6
Confusion Matrix  [[ 2149  2955]
 [ 3499 33316]]
Classification Report                precision    recall  f1-score   support

           0       0.38      0.42      0.40      5104
           1       0.92      0.90      0.91     36815

    accuracy                           0.85     41919
   macro avg       0.65      0.66      0.66     41919
weighted avg       0.85      0.85      0.85     41919
```

Decision Tree Model is producing good accuracy; 85%!. Now we will check Cross Validation score as well for overfitting(if exists).

- RandomForest Classifier

```
rf=RandomForestClassifier()
rf.fit(x_train,y_train)
pred_test=rf.predict(x_test)
pred_train=rf.predict(x_train)
Test_Accuracy_rf= (accuracy_score(y_test,pred_test))
Train_Accuracy_rf= (accuracy_score(y_train,pred_train))
print("Test_Accuracy ",round(accuracy_score(y_test,pred_test)*100,2))#testing score
print("Confusion Matrix ", confusion_matrix(y_test,pred_test))
print("Classification Report ", classification_report(y_test,pred_test))
```

```
Test_Accuracy  88.89
Confusion Matrix  [[ 1628  3421]
 [ 1238 35632]]
Classification Report                precision    recall  f1-score   support

           0       0.57      0.32      0.41      5049
           1       0.91      0.97      0.94     36870

    accuracy                           0.89     41919
   macro avg       0.74      0.64      0.68     41919
weighted avg       0.87      0.89      0.88     41919
```

RandomForest Classifier is producing very good accuracy = 89%!. Now we will check Cross Validation score as well for overfitting(if exists).

- AdaBoost Classifier

```
ada=AdaBoostClassifier()
ada.fit(x_train,y_train)
pred_test=ada.predict(x_test)
pred_train=ada.predict(x_train)
Test_Accuracy_ada= (accuracy_score(y_test,pred_test))
Train_Accuracy_ada= (accuracy_score(y_train,pred_train))
print("Test_Accuracy ",round(accuracy_score(y_test,pred_test)*100,2))#testing score
print("Confusion Matrix ", confusion_matrix(y_test,pred_test))
print("Classification Report ", classification_report(y_test,pred_test))
```

```
Test_Accuracy  89.32
Confusion Matrix  [[ 1509  3614]
 [  861 35935]]
Classification Report                precision    recall  f1-score   support

           0       0.64      0.29      0.40      5123
           1       0.91      0.98      0.94     36796

    accuracy                           0.89     41919
   macro avg       0.77      0.64      0.67     41919
weighted avg       0.88      0.89      0.88     41919
```

AdaBoostClassifier is producing good accuracy = 89%!. Now we will check Cross Validation score as well for overfitting(if exists).

- GradientBoosting Classifier

```
gb=GradientBoostingClassifier()
gb.fit(x_train,y_train)
pred_test=gb.predict(x_test)
pred_train=gb.predict(x_train)
Test_Accuracy_gb= (accuracy_score(y_test,pred_test))
Train_Accuracy_gb= (accuracy_score(y_train,pred_train))
print("Test_Accuracy ",round(accuracy_score(y_test,pred_test)*100,2))#testing score
print("Confusion Matrix ", confusion_matrix(y_test,pred_test))
print("Classification Report ", classification_report(y_test,pred_test))
```

```
Test_Accuracy  89.57
Confusion Matrix  [[ 1568  3481]
 [  892 35978]]
Classification Report                precision    recall  f1-score   support

           0       0.64      0.31      0.42      5049
           1       0.91      0.98      0.94     36870

    accuracy                           0.90     41919
   macro avg       0.77      0.64      0.68     41919
weighted avg       0.88      0.90      0.88     41919
```

GradientBoosting Classifier is producing good accuracy = 90%. Now we will check Cross Validation score as well for overfitting(if exists).

- XGBoost Classifier

```
xgb=XGBClassifier()
xgb.fit(x_train,y_train)
pred_test=xgb.predict(x_test)
pred_train=xgb.predict(x_train)
Test_Accuracy_xgb= (accuracy_score(y_test,pred_test))
Train_Accuracy_xgb= (accuracy_score(y_train,pred_train))
print("Test_Accuracy ",round(accuracy_score(y_test,pred_test)*100,2))#testing score
print("Confusion Matrix ", confusion_matrix(y_test,pred_test))
print("Classification Report ", classification_report(y_test,pred_test))
```

```
Test_Accuracy  89.73
Confusion Matrix  [[ 2033  3022]
 [ 1283 35581]]
Classification Report                precision    recall  f1-score   support

           0       0.61      0.40      0.49      5055
           1       0.92      0.97      0.94     36864

    accuracy                           0.90     41919
   macro avg       0.77      0.68      0.71     41919
weighted avg       0.88      0.90      0.89     41919
```

XGBoost Classifier is producing good accuracy = 90%. Now we will check Cross Validation score as well for overfitting(if exists).

- ExtraTrees Classifier

```
ex=XGBClassifier()
ex.fit(x_train,y_train)
pred_test=ex.predict(x_test)
pred_train=ex.predict(x_train)
Test_Accuracy_ex= (accuracy_score(y_test,pred_test))
Train_Accuracy_ex= (accuracy_score(y_train,pred_train))
print("Test_Accuracy ",round(accuracy_score(y_test,pred_test)*100,2))#testing score
print("Confusion Matrix ", confusion_matrix(y_test,pred_test))
print("Classification Report ", classification_report(y_test,pred_test))
```

```
Test_Accuracy  89.61
Confusion Matrix  [[ 1954  3095]
 [ 1259 35611]]
Classification Report                precision    recall  f1-score   support

           0       0.61      0.39      0.47      5049
           1       0.92      0.97      0.94     36870

    accuracy                           0.90     41919
   macro avg       0.76      0.68      0.71     41919
weighted avg       0.88      0.90      0.89     41919
```

Extratrees Classifier is producing good accuracy = 90%. Now we will check Cross Validation score as well for overfitting(if exists).

- MLP Classifier

```
mlp=MLPClassifier()
mlp.fit(x_train,y_train)
pred_test=mlp.predict(x_test)
pred_train=mlp.predict(x_train)
Test_Accuracy_mlp= (accuracy_score(y_test,pred_test))
Train_Accuracy_mlp= (accuracy_score(y_train,pred_train))
print("Test_Accuracy ",round(accuracy_score(y_test,pred_test)*100,2))#testing score
print("Confusion Matrix ", confusion_matrix(y_test,pred_test))
print("Classification Report ", classification_report(y_test,pred_test))
```

```
Test_Accuracy  89.12
Confusion Matrix  [[ 1727  3322]
 [ 1239 35631]]
Classification Report                precision    recall  f1-score   support

           0       0.58      0.34      0.43      5049
           1       0.91      0.97      0.94     36870

    accuracy                           0.89     41919
   macro avg       0.75      0.65      0.69     41919
weighted avg       0.87      0.89      0.88     41919
```

MLP Classifier is producing good accuracy = 89%. Now we will check Cross Validation score as well for overfitting(if exists).

- Guassian NB Classifier

```
nb=GaussianNB()
nb.fit(x_train,y_train)
pred_test=nb.predict(x_test)
pred_train=nb.predict(x_train)
Test_Accuracy_nb= (accuracy_score(y_test,pred_test))
Train_Accuracy_nb= (accuracy_score(y_train,pred_train))
print("Test_Accuracy ",round(accuracy_score(y_test,pred_test)*100,2))#testing score
print("Confusion Matrix ", confusion_matrix(y_test,pred_test))
print("Classification Report ", classification_report(y_test,pred_test))
```

```
Test_Accuracy  75.87
Confusion Matrix  [[ 2668  2381]
 [ 7736 29134]]
Classification Report                precision    recall  f1-score   support

           0       0.26      0.53      0.35      5049
           1       0.92      0.79      0.85     36870

    accuracy                           0.76     41919
   macro avg       0.59      0.66      0.60     41919
weighted avg       0.84      0.76      0.79     41919
```

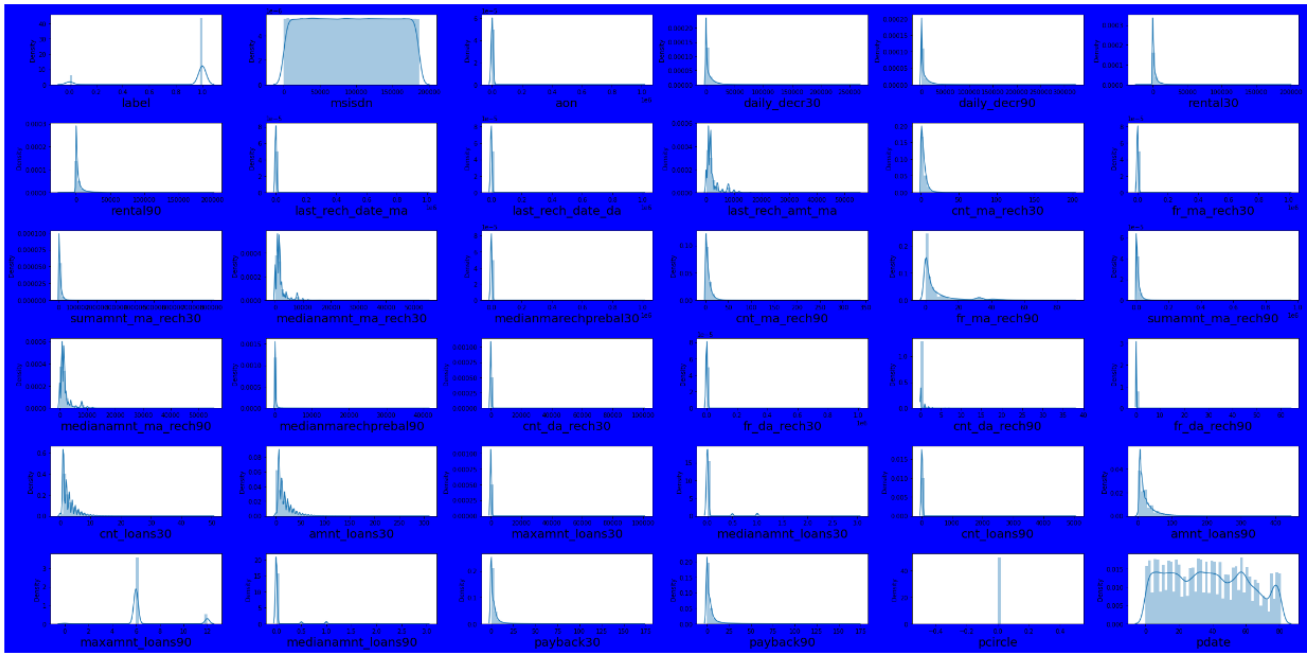
GaussianNB is producing good accuracy 76%. Now we will check Cross Validation score as well for overfitting(if exists).

- **Key Metrics for success in solving problem under consideration**
- Accuracy Score
- Cross validation
- Confusion Matrix
- ROC AUC Curve.

The above metrics were used since it's a Classification Problem

- **Visualizations**

  **<u>Normal Distribution Check(Univariate Analysis)</u>**



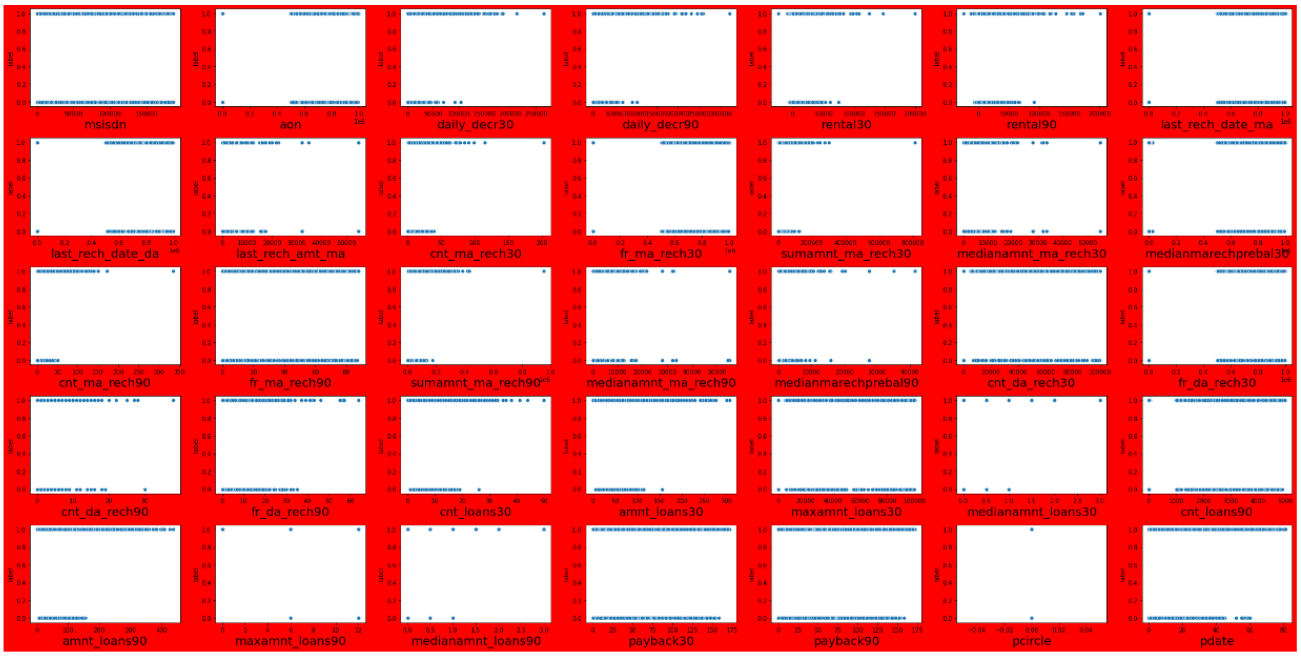

<u>Observations on Normal Distribution Check</u>

From the above density plot

- We observed all the features does not obey a normal distribution, the building blocks is not in tandem with a normalized curve.

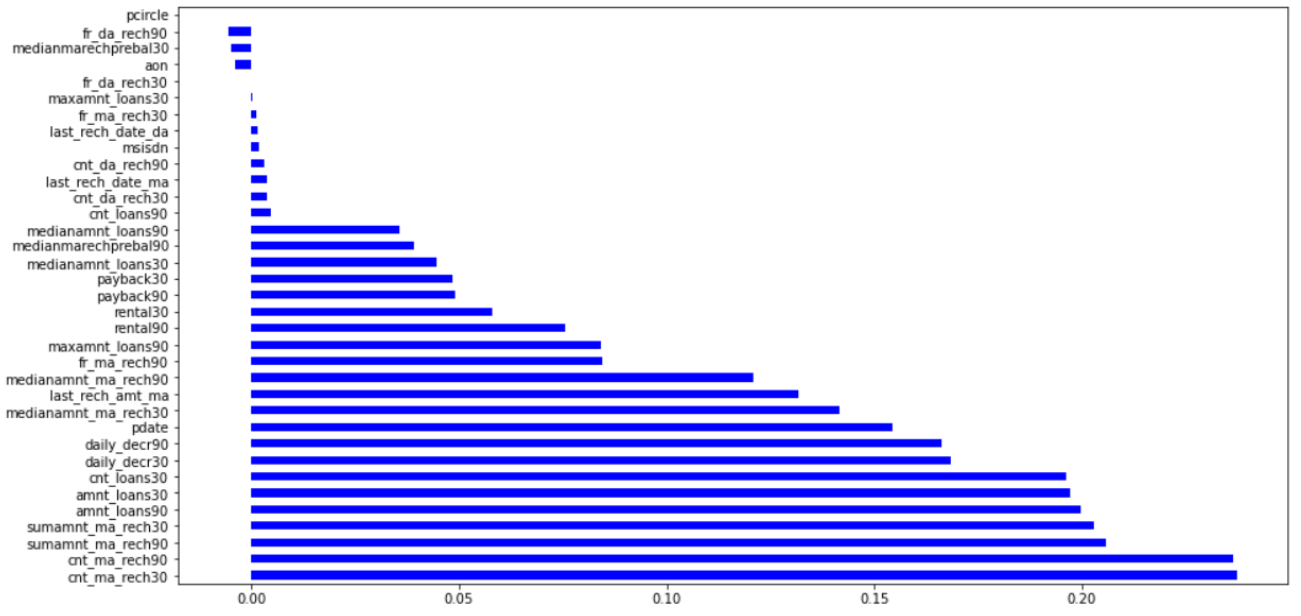
- The normal distribution of the sales columns also has no contribution to our Model Building since its the Target variable

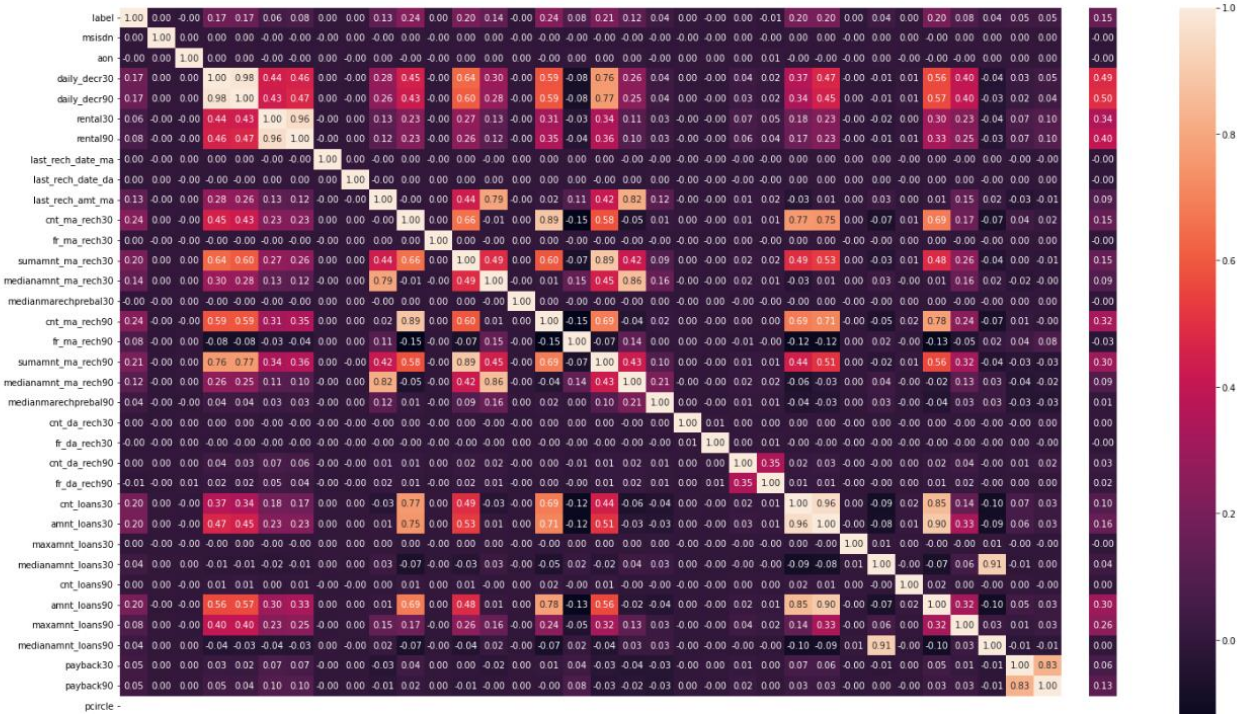
## Scatter Plot Check(Bivariate Analysis)




From the above scatter plot we can see a strong relationship between some of the features and the Target (label):

## Correlation Check(Collinearity and Multicollinearity)- Multivariate Analysis





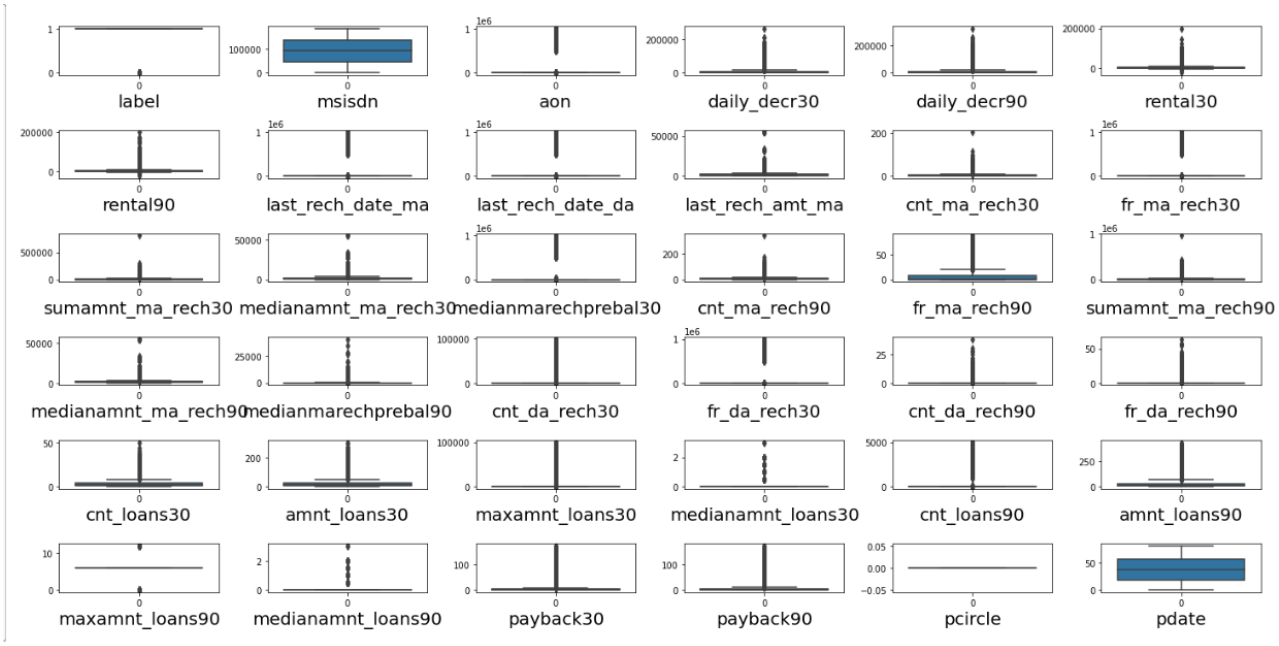

## Observations

From the above correlation statistics;

Collinearity: From the above we can see that most of the features have correlation with the target

Multicollinearity: From the heatmap we can see that the some pairs of features have noticeable correlation between them

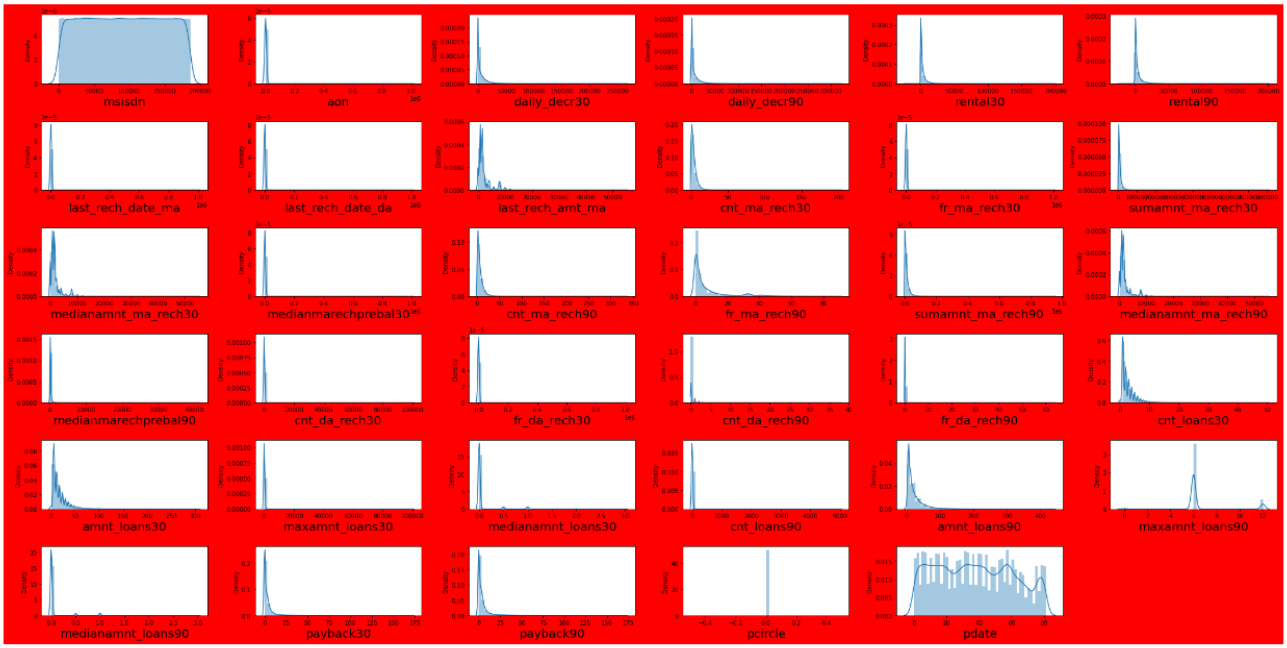
## Outlier Check




## Observations on Outlier Check

From the above visualization plot its evident the most features possess outliers,

## Skewness Check



**Observations on Skewness Check:**

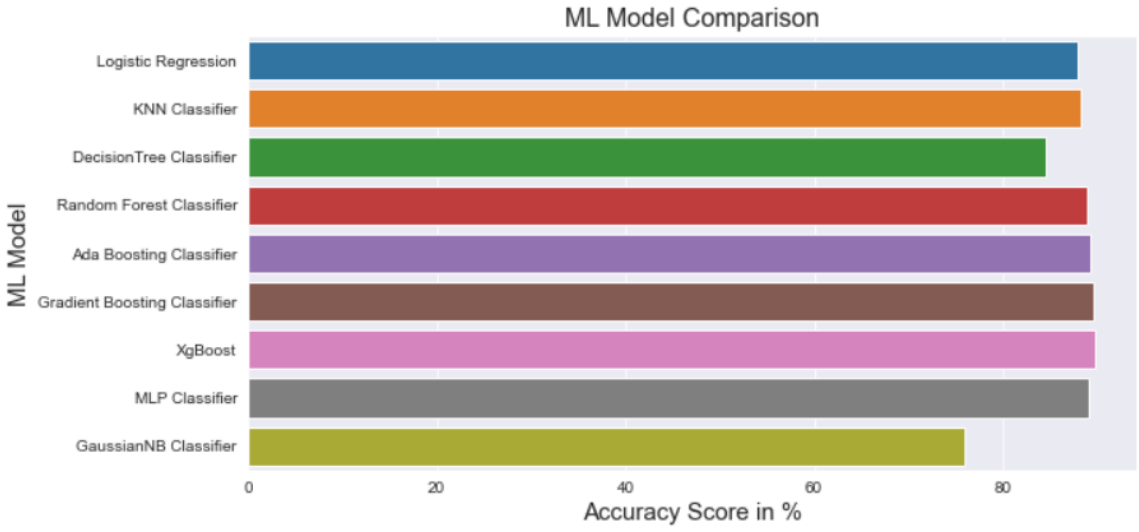
We assumed a Skewness threshold of +/-0.50. Meaning any value outside +/-0.50 contains skewness. Hence majority of the features are having a skewness.

## CONCLUSION

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

COMPARING ALL EIGHT MACHINE LEARNING MODELS

| | Model | Accuracy_Score | Cross_Validation_Score | Accuracy_VS_CVScore |
|---|---|---|---|---|
| 6 | XgBoost | 89.73 | 89.36 | 0.003680 |
| 5 | Gradient Boosting Classifier | 89.57 | 89.23 | 0.003384 |
| 4 | Ada Boosting Classifier | 89.32 | 88.85 | 0.004730 |
| 7 | MLP Classifier | 89.12 | 87.86 | 0.012636 |
| 3 | Random Forest Classifier | 88.89 | 88.49 | 0.003933 |
| 1 | KNN Classifier | 88.35 | 87.34 | 0.010112 |
| 0 | Logistic Regression | 87.96 | 87.51 | 0.004468 |
| 2 | DecisionTree Classifier | 84.60 | 84.22 | 0.003819 |
| 8 | GaussianNB Classifier | 75.87 | 85.52 | 0.096579 |

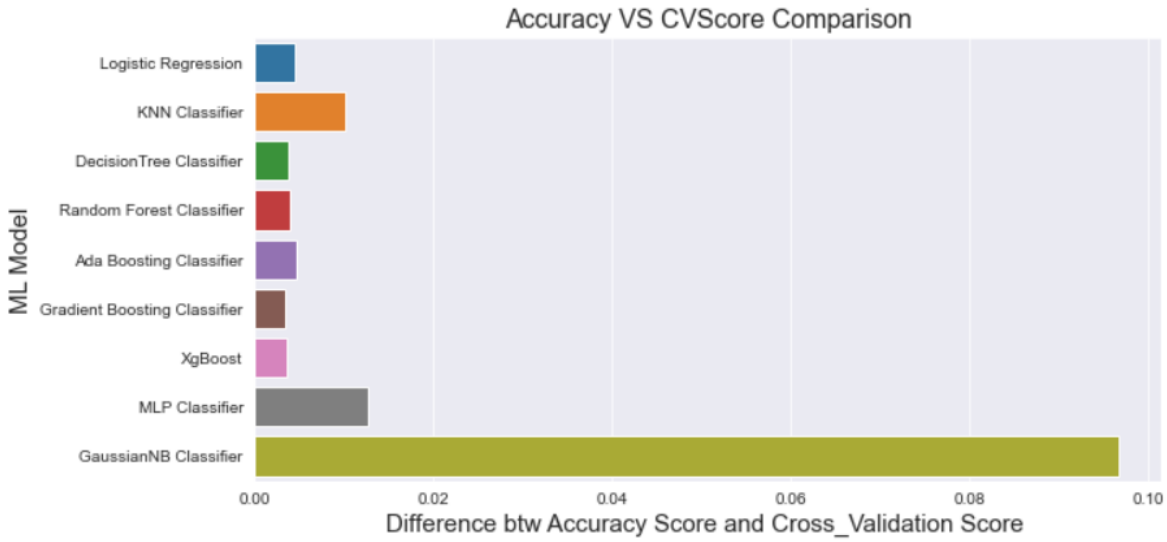



Now from the above diagram it seems that XGBOOST Classifier(89.73%) has the highest Accuracy, However, our aim is to find the BEST MODEL, so if we consider the

difference Between Accuracy_Score and Cross_Validation_Score....

## Comparing Differences between Accuracy and Cross  Validation Scores...

| | Model | Accuracy_Score | Cross_Validation_Score | Accuracy_VS_CVScore |
|---|---|---|---|---|
| 5 | Gradient Boosting Classifier | 89.57 | 89.23 | 0.003384 |
| 6 | XgBoost | 89.73 | 89.36 | 0.003680 |
| 2 | DecisionTree Classifier | 84.60 | 84.22 | 0.003819 |
| 3 | Random Forest Classifier | 88.89 | 88.49 | 0.003933 |
| 0 | Logistic Regression | 87.96 | 87.51 | 0.004468 |
| 4 | Ada Boosting Classifier | 89.32 | 88.85 | 0.004730 |
| 1 | KNN Classifier | 88.35 | 87.34 | 0.010112 |
| 7 | MLP Classifier | 89.12 | 87.86 | 0.012636 |
| 8 | GaussianNB Classifier | 75.87 | 85.52 | 0.096579 |




From the above we can see the Model with least difference is GRADIENT BOOSTING REGRESSOR!

## Conclusion on Best Choice of Model

From the above we can see:

- The least difference is 0.003(very neglible)!

- The Model with least difference is Gradient Boosting Classifier!

- Accuracy is 89%

- We also went further to engage in **Hyperparamter Tunning** using GridSearchCV and arrived at a Final Accuracy of 90%

- Learning Outcomes of the Study in respect of Data Science
  - The SVM and MLP were not suitable models as they produced lower accuracy.
  - Dropping of features with VIF greater than 7 was very key in feature selection
  - SelectKBest was also instrumental in feature selection since we had a lot of features.
- **Limitations of this work and Scope for Future Work**
  The limitation was basically Time