MICRO CREDIT DEFAULTER PROJECT

Submitted by:

Khanin Deka

ACKNOWLEDGMENT

- https://datascience.stackexchange.com/
- https://stackoverflow.com/
- FlipRobo Technologies
- https://scikit-learn.org/

INTRODUCTION

- ## Business Problem Framing

A Microfinance Institution (MFI) is an organization that offers financial services to low income populations. Micro-finance services have become very useful when targeting especially the unbanked poor families living in remote areas with not much sources of income. 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. However, the implementation of MFS has been uneven with both significant challenges and successes. Let's look at one such case study from machine learning perspective.

A telecommunications network provider in collaboration with a MFI is providing 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. In order to improve the selection of customers for the credit, we will attempt to build a machine learning model that could help in making some predictions to facilitate further investment and improvement in selection of customers.

- ## Conceptual Background of the problem

We will attempt to build a machine learning model to predict whether a consumer will repay the loaned amount within the stipulated time or not. Since it is a classification task, hence we will use classification algorithms. We have a dataset of 209593 entries and 37 columns which we will be using for building our model.

Analytical Problem Framing

- About the Dataset
    1. The dataset contains 209593 rows and 37 columns.
    2. Out of the 37 columns, 21 are of float data-type, 13 of int and 3 of object data-type.
    3. Here is a snapshot of the dataset:

	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	

    4. Some of the columns have data in text format. A machine learning model doesn't understand text data. Hence, we have to encode these text data into numeric format.
    5. No missing values were found in the dataset.

- Tools and Libraries used
    1. This project is built using Jupyter-Notebook of Anaconda Navigator.
    2. python language version 3.9.12 is used.
    3. List of libraries include pandas, numpy and sklearn.
    4. For data visualization matplotlib and seaborn is used.

- Data Preprocessing
  - *'Unnamed: 0'* column is deleted as it is used for numbering the entries in the dataset.
  - *'msisdn'* column is deleted as it contains the mobile numbers of the customers which is not an useful data for us.
  - *'pcircle'* column is deleted as it contains only one unique value for the entire column.
  - 31 duplicate entries found which were removed.
  - On *'pdate'* column pandas *to_datetime* method is applied and the day, month and year data is extracted into separate columns. The *'pdate'* column is then removed.

```
df['pdate']= pd.to_datetime(df['pdate'],format='%Y-%m-%d',dayfirst=True)
df['day']= df['pdate'].dt.day
df['month']= df['pdate'].dt.month
df['year']= df['pdate'].dt.year
df.drop(columns=['pdate'],inplace=True)
```

  - *'year'* column is deleted as all the entries correspond to a single year i.e. 2016
  - The 'pearson' correlation of the features with the label is calculated. All the features with correlation less than 0.05 with the label are dropped as the correlation is considered statistically insignificant.

Absolute values of correlation with label in descending Below shown list is not the complete list.

```
# Let us see the correlation of various features with label.
corr_mat=df.corr()
corr_with_target= np.abs(corr_mat["label"]).sort_values(ascending= False)
corr_with_target
```

```
label                   1.000000
cnt_ma_rech30           0.237120
cnt_ma_rech90           0.236200
sumamnt_ma_rech90       0.205634
sumamnt_ma_rech30       0.202658
amnt_loans90            0.199649
amnt_loans30            0.197123
cnt_loans30             0.196133
daily_decr30            0.168174
daily_decr90            0.166034
month                   0.154707
medianamnt_ma_rech30    0.141248
last_rech_amt_ma        0.131558
medianamnt_ma_rech90    0.120594
fr_ma_rech90            0.084205
maxamnt_loans90         0.084064
rental90                0.075339
rental30                0.057886
payback90               0.049050
payback30               0.048213
medianamnt_loans30      0.044521
medianmarechprebal90    0.039228
medianamnt_loans90      0.035681
day                     0.006630
fr_da_rech90            0.005438
medianmarechprebal30    0.004857
```

List of features with correlation <0.05:

```
# Get the list of columns having correlation <0.05 with 'label'
drop_list=[]
for i in range(len(corr_with_target)):
    if corr_with_target[i]<0.05:
        drop_list.append(corr_with_target.index[i])
drop_list
```

```
['payback90',
 'payback30',
 'medianamnt_loans30',
 'medianmarechprebal90',
 'medianamnt_loans90',
 'day',
 'fr_da_rech90',
 'medianmarechprebal30',
 'cnt_loans90',
 'aon',
 'cnt_da_rech30',
 'last_rech_date_ma',
 'cnt_da_rech90',
 'last_rech_date_da',
 'fr_ma_rech30',
 'maxamnt_loans30',
 'fr_da_rech30']
```

- If we carefully see the features then we will find that a lot of them have data for last 30 and 90 days of same feature like '*daily_decr30*' and '*daily_decr90*'. Both represent the averaged daily amount spent from main account, but over the duration of 30 days and 90 days. To check for muticollinearity among them correlation heat-map and variance inflation factor (VIF) is used. All the feature-pairs with correlation above 0.85 are checked for VIF scores. VIF score of above 5 is considered to be a multicollinearity issue. Hence out of the two features which are multicollinear, one is dropped. Whichever feature among the two have lower correlation with label is dropped.

  For correlation heat-map please refer to the visualization section.

  Features and their VIF scores:

	vif	Features
0	29.230267	daily_decr30
1	32.055149	daily_decr90
2	13.218987	rental30
3	14.106014	rental90
4	3.411403	last_rech_amt_ma
5	14.977954	cnt_ma_rech30
6	12.727251	sumamnt_ma_rech30
7	5.066614	medianamnt_ma_rech30
8	16.378333	cnt_ma_rech90
9	1.059913	fr_ma_rech90
10	15.165421	sumamnt_ma_rech90
11	5.482834	medianamnt_ma_rech90
12	23.259967	cnt_loans30
13	29.308025	amnt_loans30
14	10.815912	amnt_loans90
15	1.952122	maxamnt_loans90
16	1.733771	month

  Features that are deleted:

  'daily_decr90','rental30','cnt_ma_rech90','sumamnt_ma_rech30','medianamnt_ma_rech90', 'amnt_loans30' and 'cnt_loans30'

Final list of features and their VIF scores:

	vif	Features
0	3.301530	daily_decr30
1	1.367982	rental90
2	2.779258	last_rech_amt_ma
3	2.377434	cnt_ma_rech30
4	3.028557	medianamnt_ma_rech30
5	1.054324	fr_ma_rech90
6	3.891353	sumamnt_ma_rech90
7	2.374933	amnt_loans90
8	1.245317	maxamnt_loans90
9	1.529222	month

- The skewness of the continuous data features is checked. All of them have very highly skewed data. Below is the list:

```
df[cont_features].skew()
```

```
daily_decr30            3.945997
rental90                4.437471
last_rech_amt_ma        3.781206
cnt_ma_rech30           3.283935
medianamnt_ma_rech30    3.512361
fr_ma_rech90            2.285206
sumamnt_ma_rech90       4.897896
amnt_loans90            3.149830
dtype: float64
```

Cube-root and power transformations are applied to bring down the skewness:

```
from sklearn.preprocessing import PowerTransformer
pt= PowerTransformer(method="yeo-johnson")

def power_transform(column):
    df[column]= pt.fit_transform(df[[column]])

def cube_root_transform(column):
    df[column]= np.cbrt(df[column])

for i in power_transform_columns:
    power_transform(i)

for i in cbrt_transform_columns:
    cube_root_transform(i)

df[cont_features].skew()
```

```
daily_decr30            0.538027
rental90                0.173255
last_rech_amt_ma       -0.060722
cnt_ma_rech30          -0.000146
medianamnt_ma_rech30   -0.189169
fr_ma_rech90            0.142864
sumamnt_ma_rech90      -0.016310
amnt_loans90           -0.008978
dtype: float64
```

Features on which power transformation is applied are: *'last_rech_amt_ma','cnt_ma_rech30', 'amnt_loans90' 'medianamnt_ma_rech30'* and *'fr_ma_rech90'*
Features on which cube-root transformation is applied are: *'daily_decr30','rental90'* and *'sumamnt_ma_rech90'.*

- The features are then scaled using 'StandardScaler' of scikit-learn.

Model/s Development and Evaluation

- Identification of possible problem-solving approaches

1) Our task is to predict whether a customer is likely to return the loan amount within stipulated time. For this multiple classification algorithms are tried and their performances are evaluated based on accuracy, precision, recall and ROC-AUC score.

2) **Logistic Regression**: It estimates the probability of an event occurring based on a given dataset of independent variables. Since the outcome is a probability, the dependent variable is bounded between 0 and 1. In logistic regression, a logit transformation is applied on the odds i.e., the probability of success divided by the probability of failure. This is also commonly known as the log odds, or the natural logarithm of odds.

3) **Random Forests Classification**: It is an ensemble learning method that operates by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees.

4) **Ada-boost Classification**: Ada-boost or Adaptive Boosting is an ensemble boosting classifier. This classifier builds a strong classifier by combining multiple poorly performing classifiers so that we get a high accuracy strong classifier.

5) **XGBoost Classification:** XGBoost stands for Extreme Gradient Boosting. In this algorithm, decision trees are created in sequential form. Weights play an important role in this algorithm. Weights are assigned to all the independent variables which are then fed into the decision tree which predicts results. The weight of variables predicted wrong by the tree is increased and these variables are then fed to the second decision tree. These individual classifiers/predictors then ensemble to give a strong and more precise model.

6) **K-Neighbors Classification**: In this classification algorithm, the output is a class membership. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its *k* nearest neighbors (*k* is a positive integer, typically small).

- Performance Evaluation

To evaluate the classification models the following metrics are used:

1. Accuracy: It is the number of correct predictions divided by the total number of predictions.
2. Precision: It is proportion of positive predictions that are actually correct. In our case, precision with respect to defaulters is the proportion of positive-defaulter predictions that are actually defaulters.
3. Recall: It is the proportion of actual positives that predicted correctly by the model. In our case, recall with respect to defaulters is the proportion of actual defaulters that are predicted correctly by the model.
4. ROC-AUC: ROC is a probability curve and AUC represents the degree or measure of separability. It tells how much the model is capable of distinguishing between classes. Higher the AUC, the better the model is at predicting 0 classes as 0 and 1 classes as 1.

The performance of different algorithms on these metrics are summarized in a dataframe. The performance is tested for two different train-test splits.

Train-test split 1 results:

	Logistic Regression	Random-Forests Classifier	Adaboost Classifier	XGBoost Classifier	K-Neighbors Classifier
Accuracy	0.82	0.90	0.87	0.90	0.84
Precision	0.37	0.61	0.48	0.61	0.42
Recall	0.61	0.55	0.67	0.63	0.66
ROC-AUC	0.73	0.75	0.79	0.78	0.76

Train-test split 2 results:

	Logistic Regression	Random-Forests Classifier	Adaboost Classifier	XGBoost Classifier	K-Neighbors Classifier
Accuracy	0.82	0.90	0.87	0.91	0.85
Precision	0.37	0.61	0.48	0.62	0.42
Recall	0.61	0.55	0.68	0.62	0.66
ROC-AUC	0.73	0.75	0.79	0.78	0.76

It can be seen that XGBoost Classifier is giving the best overall performance if we consider all metrics. Random-Forests have a good accuracy but the recall score is not good enough. Adaboost has better recall than XGBoost but the precision is not good enough.

Let's move ahead with XGBoost Classifier and try to tune it further.

- Hyper-parameter Tuning

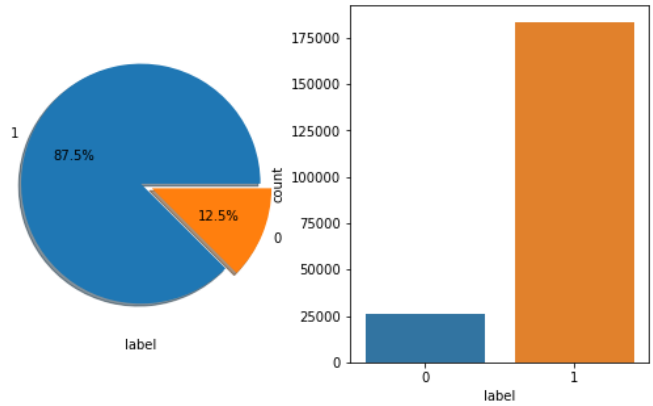
```
#Hyperparameter tuning of XGBoost Classifier model using GridSearchCV.
x_train,x_test,y_train,y_test = train_test_split(X_scaled,y,test_size=0.25,random_state=2,stratify=y)
params= {"n_estimators": [100,250,500,1000],
         "max_depth":[5,7,9,13,15],
         "learning_rate":[0.01,0.1,1]
         }
grd_xgb= GridSearchCV(xgb_clf, param_grid=params,cv=5,n_jobs=-1,scoring=make_scorer(recall_score))
grd_xgb.fit(x_train,y_train)
print("Best Parameters:",grd_xgb.best_params_)
```

```
Best Parameters: {'learning_rate': 0.1, 'max_depth': 7, 'n_estimators': 250}
```

3 hyper-parameters of XGBoost are tuned: 'n_estimators', 'max_depth' and 'learning_rate'. The best parameters are found to be 0.1, 7 and 250 respectively.

- Visualizations

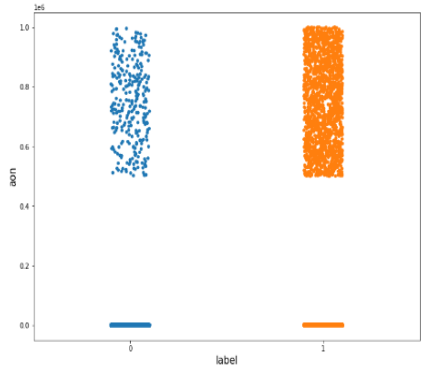
First let's see the distribution in target variable:

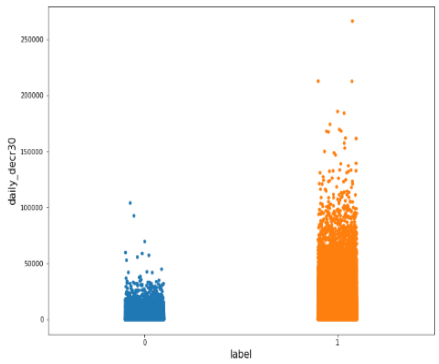


The target column is imbalanced.

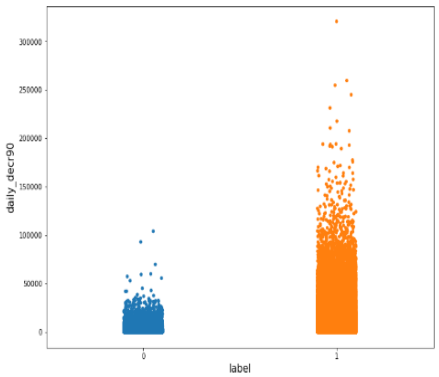
To analyze the relation between label and the features, various plots are made. Let's see them.

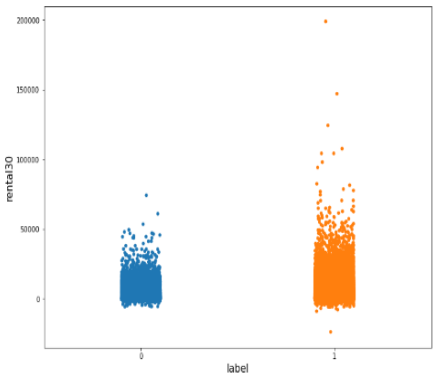
Strip-plots of 'aon' and 'daily_decr30' with the label:



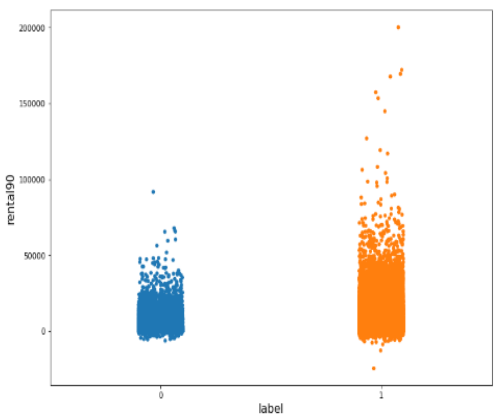


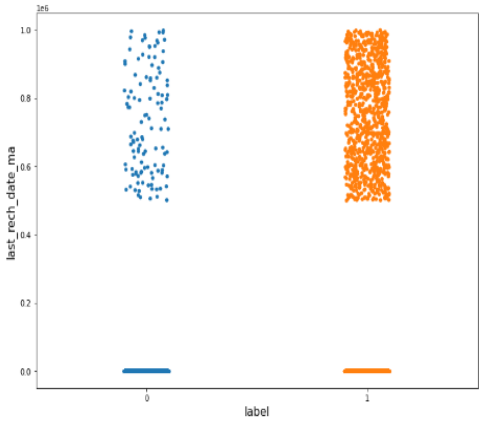
Strip-plots of 'daily_decr90' and 'rental30' with the label:



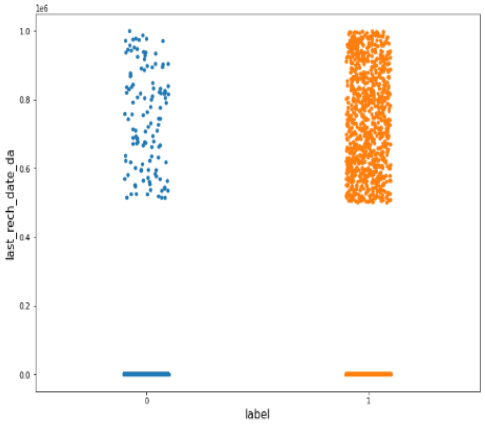


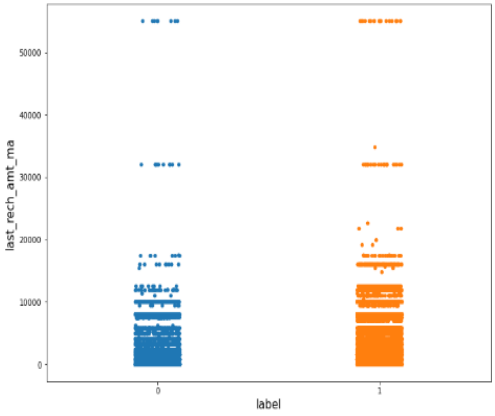
Strip-plots of ‘rental90’ and ‘last_rech_date_ma’ with the label:



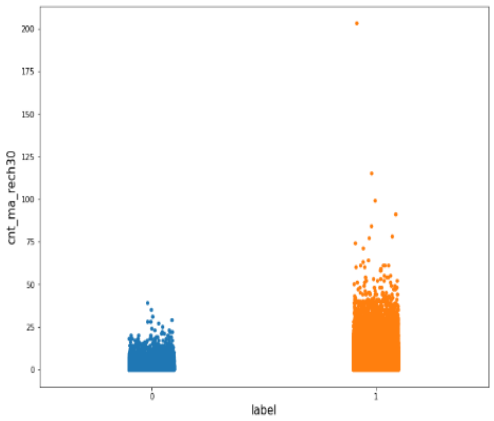


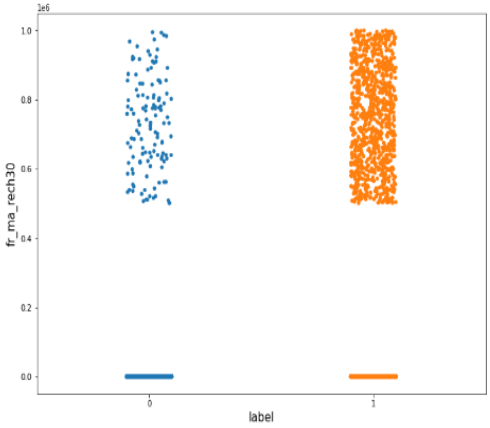
Strip-plots of ‘last_rech_date_da’ and ‘last_rech_amt_ma’ with the label:



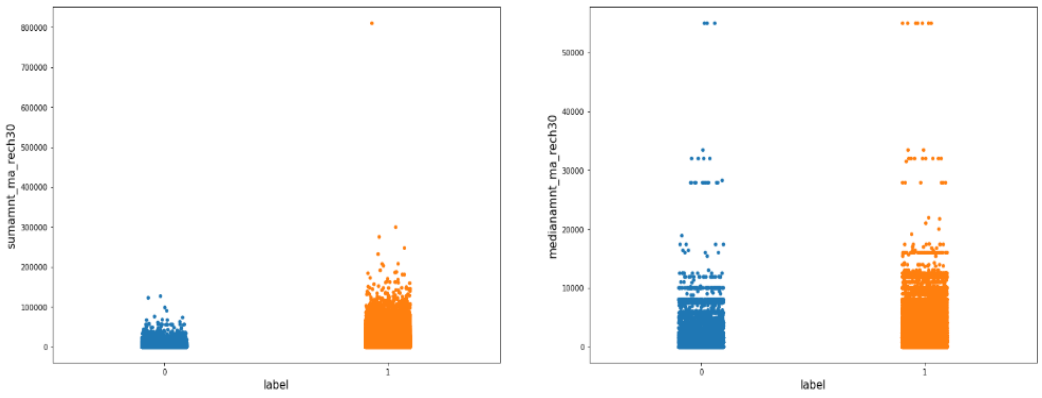


Strip-plots of ‘cnt_ma_rech30’ and ‘fr_ma_rech30’ with the label:

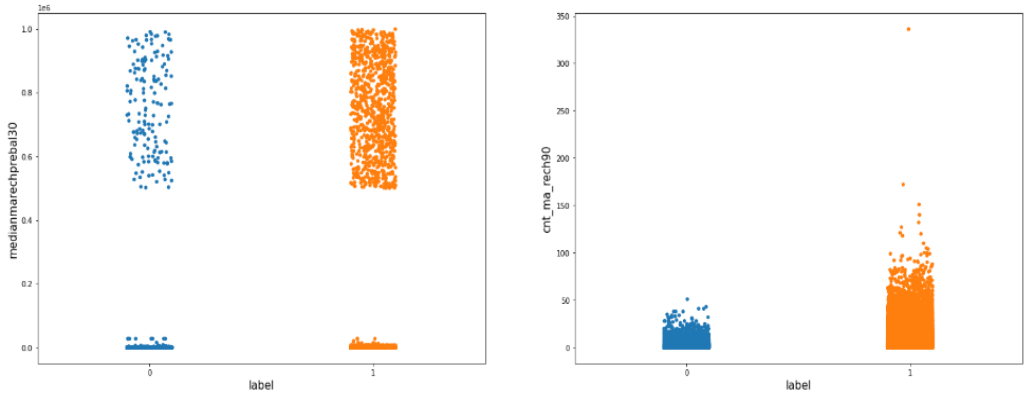




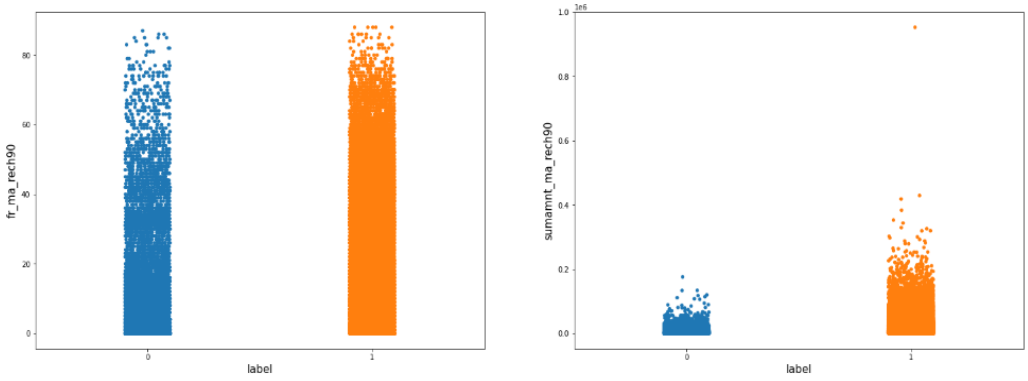
Strip-plots of ‘sumamnt_ma_rech30’ and ‘medianamnt_ma_rech30’ with the label:



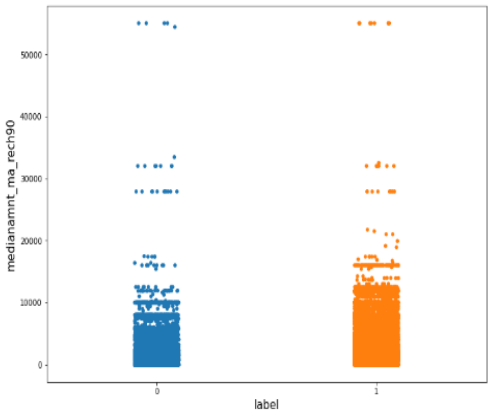
Strip-plots of ‘medianmarechprebal30’ and ‘cnt_ma_rech90’ with the label:

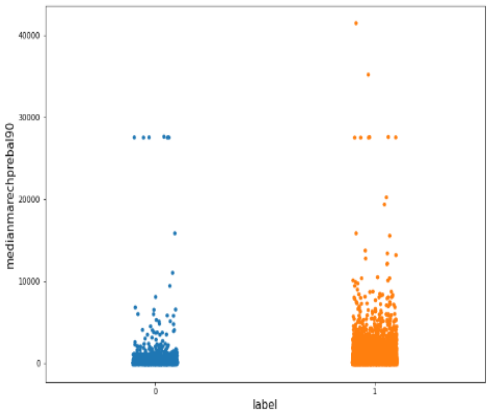


Strip-plots of ‘fr_ma_rech90’ and ‘sumamnt_ma_rech90’ with the label:

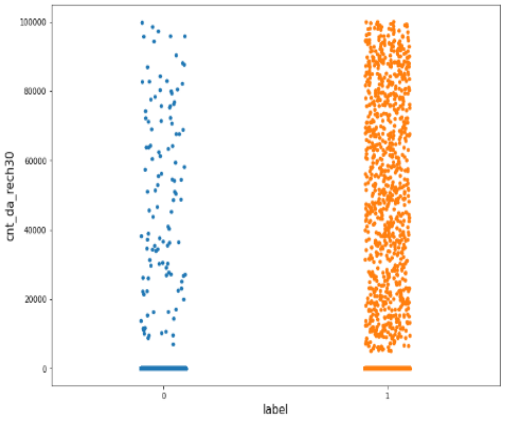


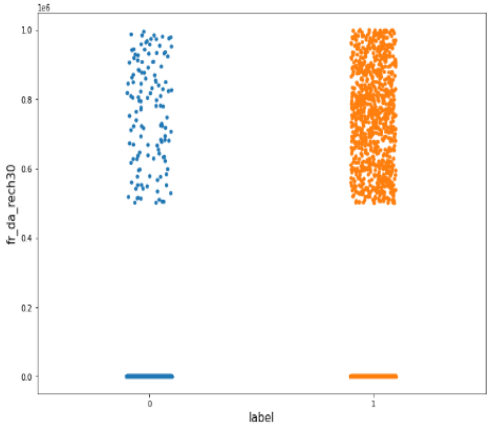
Strip-plots of ‘medianamnt_ma_rech90’ and ‘medianmarechprebal90’ with the label:



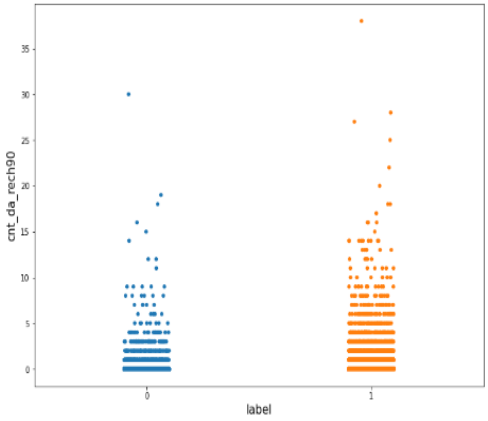


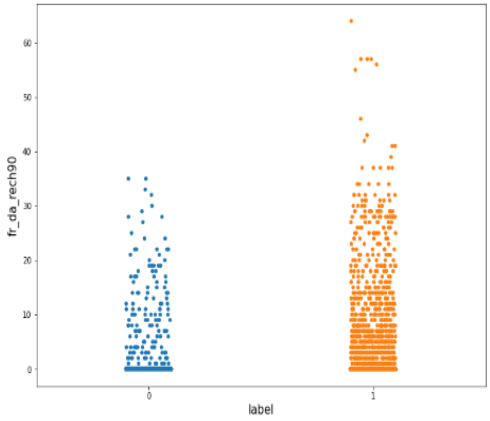
Strip-plots of ‘cnt_da_rech30’ and ‘fr_da_rech30’ with the label:



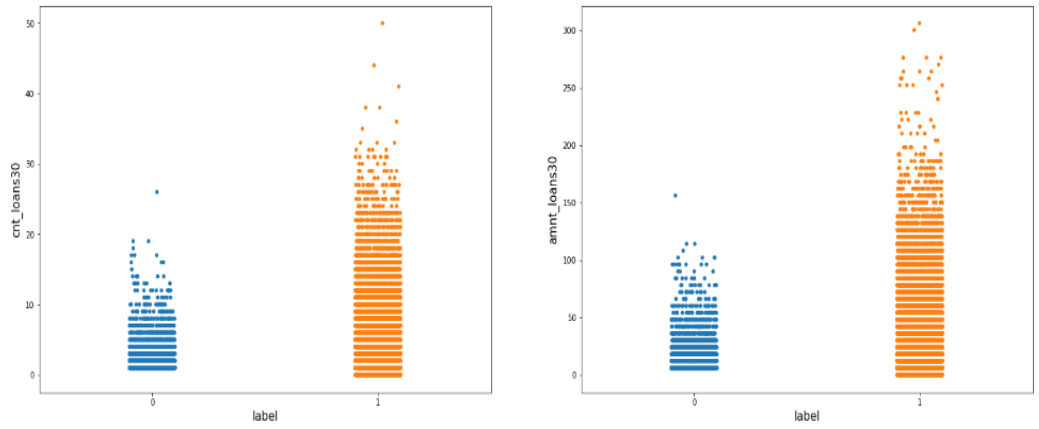


Strip-plots of ‘cnt_da_rech90’ and ‘fr_da_rech90’ with the label:

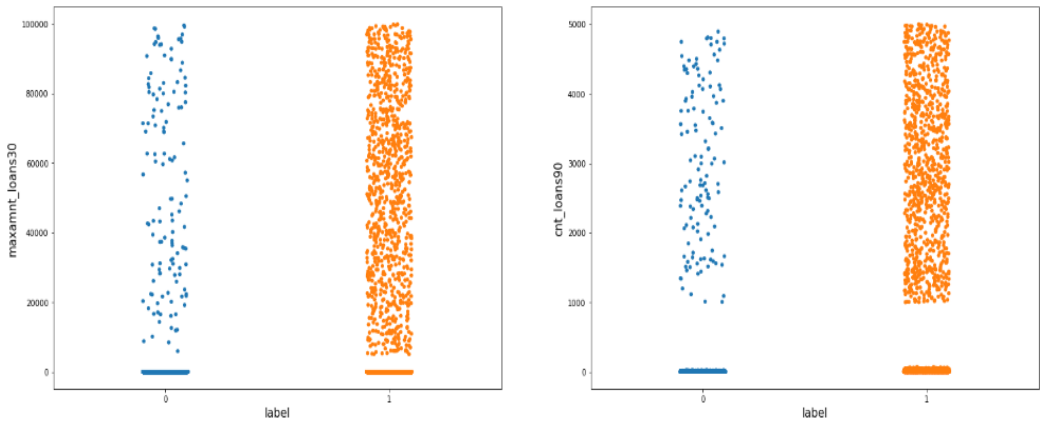




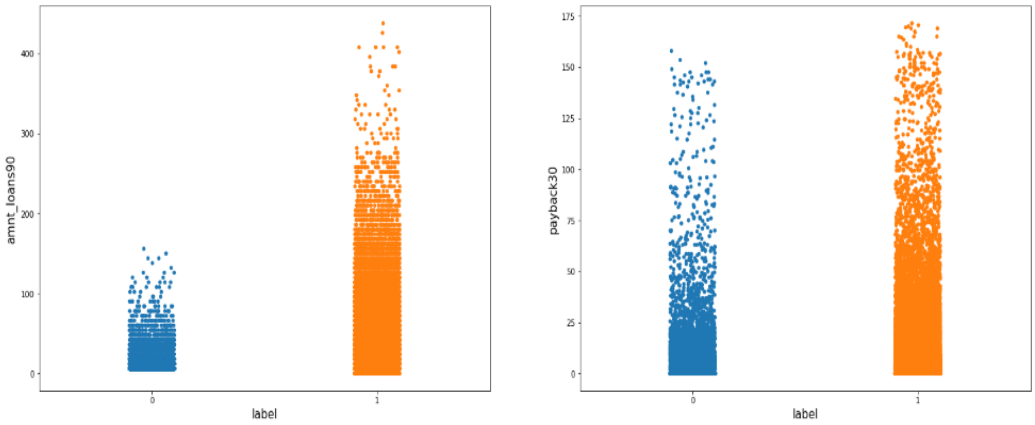
Strip-plots of ‘cnt_loans30’ and ‘amnt_loans30’ with the label:



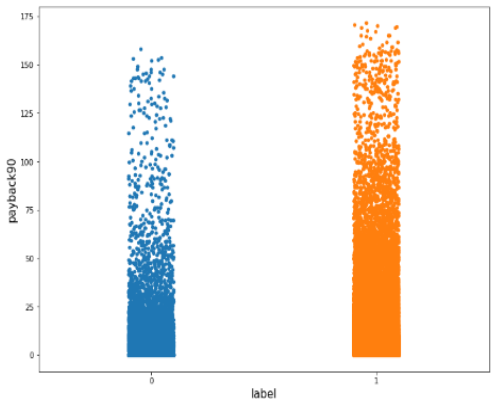
Strip-plots of ‘maxamnt_loans30’ and ‘cnt_loans90’ with the label:



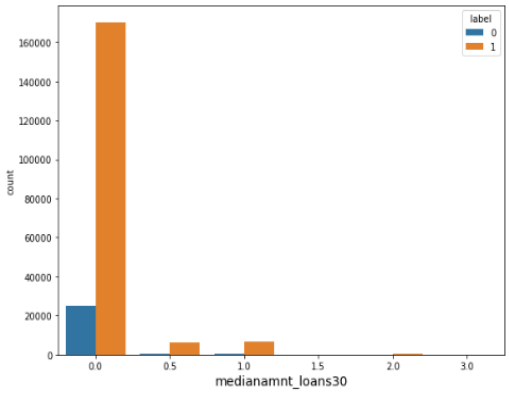
Strip-plots of ‘amnt_loans90’ and ‘payback30’ with the label:

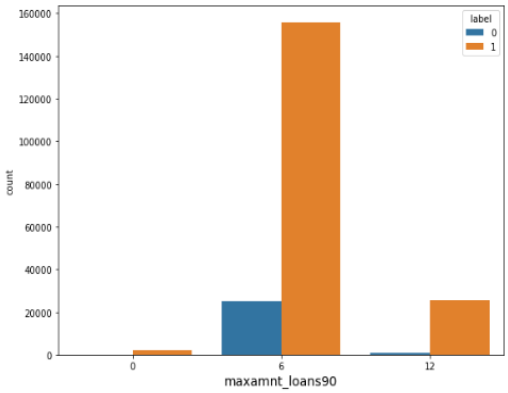


Strip-plot of ‘payback90’ with the label:

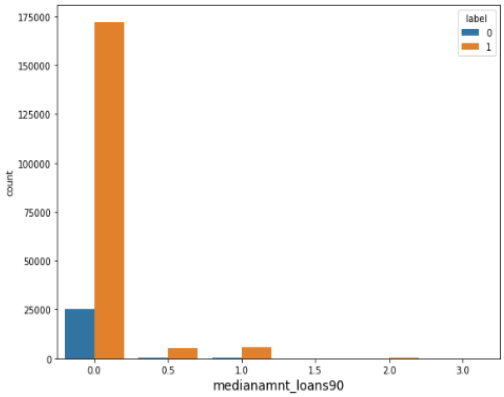


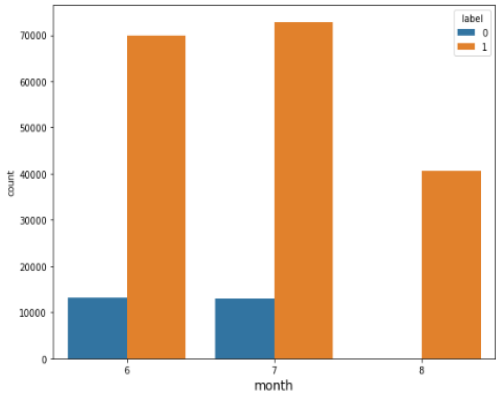
Count-plots of ‘medianamnt_loans30’ and ‘maxamnt_loans90’ with color coding of the label:



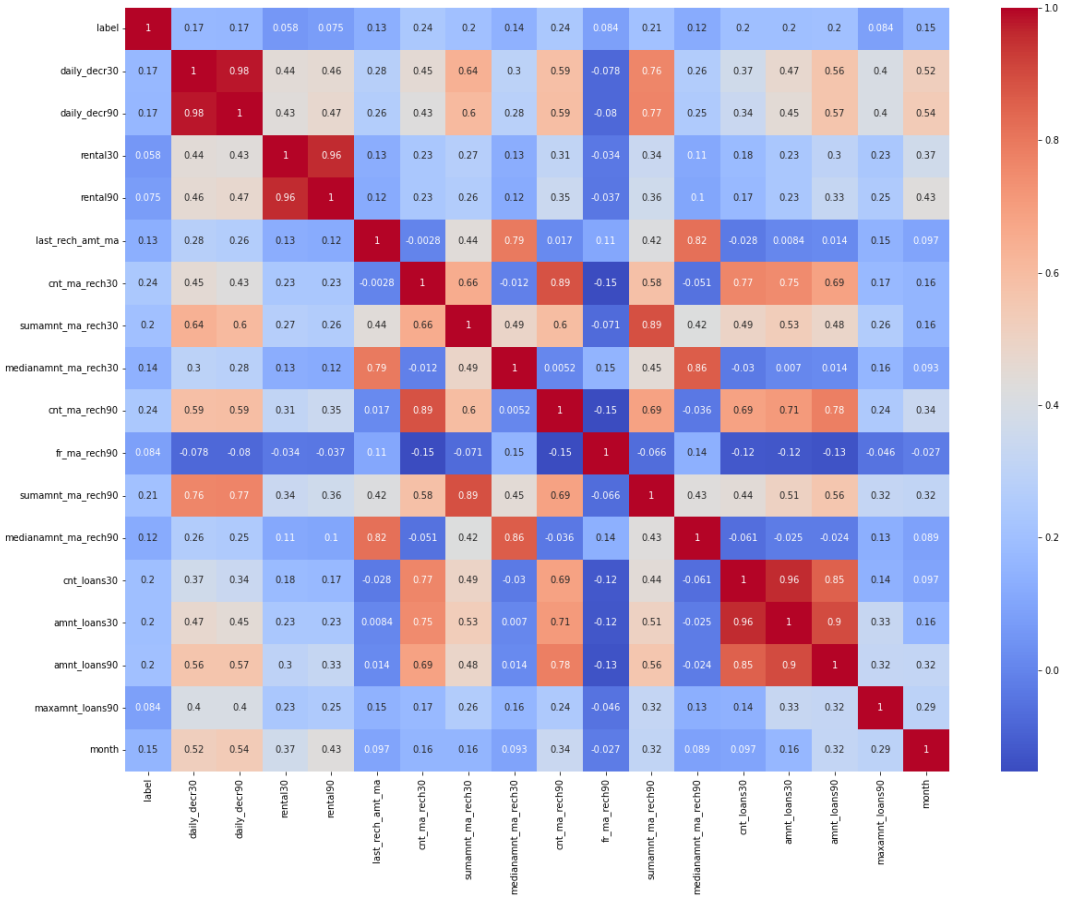


Count-plots of ‘medianamnt_loans90’ and ‘month’ with color coding of the label:





Correlation Heat-map (only those features having correlation>0.05 with label:



Observations:

1. 'daily_decr30' and 'daily_decr90' have multicollinearity issue.
2. 'rental30' and 'rental90' have multicollinearity issue.
3. 'amnt_loans30' and 'amnt_loans90' have multicollinearity issue.
4. 'amnt_loans30' and 'cnt_loans90' have multicollinearity issue.
5. 'cnt_ma_rech30' and 'cnt_ma_rech90' might have multicollinearity issue.
6. 'medianamnt_ma_rech30' and 'medianamnt_ma_rech90' might have multicollinearity issue.

CONCLUSION

- Key Findings and Conclusions of the Study
  1) The top 17 features affecting whether a customer is likely to be a defaulter are shown below in a list along with the Pearson standard correlation coefficient.

```
label                   1.000000
cnt_ma_rech30           0.237120
cnt_ma_rech90           0.236200
sumamnt_ma_rech90       0.205634
sumamnt_ma_rech30       0.202658
amnt_loans90            0.199649
amnt_loans30            0.197123
cnt_loans30             0.196133
daily_decr30            0.168174
daily_decr90            0.166034
month                   0.154707
medianamnt_ma_rech30    0.141248
last_rech_amt_ma        0.131558
medianamnt_ma_rech90    0.120594
fr_ma_rech90            0.084205
maxamnt_loans90         0.084064
rental90                0.075339
rental30                0.057886
Name: label, dtype: float64
```

  2) XGBoost Classification Algorithm is giving the best results for prediction of defaulters. It gives accuracy of around 90%. Below is the performance of tuned model on a train-test split.

```
Testing Score: 0.9
Precision for 1: 0.95
Recall for 1: 0.94
Precision for 0: 0.62
Recall for 0: 0.63
++++++++++++++ CLASSIFICATION REPORT +++++++++++++++++

              precision    recall  f1-score   support

           0       0.62      0.63      0.62      6533
           1       0.95      0.94      0.95     45858

    accuracy                           0.90     52391
   macro avg       0.78      0.79      0.78     52391
weighted avg       0.91      0.90      0.91     52391

++++++++++++++ CONFUSION MATRIX +++++++++++++++++

 [[ 4095  2438]
 [ 2547 43311]]
```

- Limitations of this work and Scope for Future Work
    1) The hyper-parameters of our final model can be further tuned.
    2) In this project we worked on only 5 algorithms. There are numerous other classification algorithms with which we can try to make a better model.