Micro-Credit Defaulter Project

Submitted by: Vinita Shinkar

Submitted to: Flip Robo Technologies

ACKNOWLEDGMENT

The dataset for this project was provided by Fliprobo Technologies Pvt. Ltd., Special reference thankful for team Fliprobo for continuous guidance to complete the project.

INTRODUCTION

- Business Problem Framing

  The problem statement is to predict whether the customer will be paying loan amount in 5 days or takes more than 5 days to payback loan.

  The telecom and DTH industry has been lending such short term loans to all the customers in general, with this kind of project business can decide whether it is beneficial to offer such loan, for which we will analyse loan + interest received from this service and loss due to default in payment can be compared and derive a conclusion finally predict future expected losses and future expected additional income to decide loan offer is a feasible decision

- Conceptual Background of the Domain Problem

  The challenge to lenders to predict the future financial status of customer or borrower based on current conditions, but with the help of Machine Learning and handful information one can build a predictive model, which helps the businesses to evaluate the lendable loan amounts, target customers, decision about interest chargeable and so on.

  Data science is the solution for such future accountable predications.

- Review of Literature

  Microfinance, according to our project is an offer provided by a telecommunication network in collaboration with micro financial institution, allows customers to take small loans in a manner that is consistent with ethical lending practices, with low income families and poor customers as their main target.

  The offer(loan) details is that loan borrowed should be paid back in 5 days, failing which Consumer is considered as defaulter, since he is not paid back within given time period of 5 days. For the loan amount of 5 (in Indonesian Rupiah), payback amount is 6 (in Indonesian Rupiah), while, for the loan amount of 10 (in Indonesian Rupiah), the payback amount is 12 (in Indonesian Rupiah).
  The telecom service provider wants to make a decision if the current plan is feasible and understand the future profits out of this offer, hence we have used machine learning as a tool to predict the expected defaulters and non-defaulters on the basis of given attributes as parameters for forecasting.
  Since it is categorical type of prediction (defaulter & non-defaulter) we have used classification algorithms

  This is a comprehensive summary of the research done on the topic. The review should enumerate, describe, summarize, evaluate and clarify the research done.

- Motivation for the Problem Undertaken

  The data shape is 2, 09,593 which is divided into 37 columns, understanding the columns and its impact on the target column was the challenge and motivation to accomplish the goal of predicting the target.

Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem

  1. We have used percentage formula for analysing the data we will have to forego while removing outliers.
  2. Statistical tool – with the help of correlation we divided columns into three category :-

  Highly positive correlation of attributes with label,

  Positively correlated columns

  Negative correlation.

  3. Feature engineering has been used to analyse Pdate column, visualize label column with other attributes & analyse categorical column.
  4. While building the model, we understood few algorithms gave 99.99% accuracy, which could be result of over fitting the model hence cross validated and used grid search CV to extract the best parameter of analysis

- Data Sources and their formats

  The data was provided by Fliprobo Company with a detailed project history which contained

  Detailed description of each column

Dataset

Problem statement of the project.

An over view of data after uploading in notebook

```
In [8]: data.head()
```

Out[8]:

	Unnamed: 0	label	msisdn	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_date_da	...	maxamnt_loans30	medianan
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

```
In [9]: data.tail()
```

Out[9]:

	Unnamed: 0	label	msisdn	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_date_da	...	maxamnt_loans30	me
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

Detailed description of all attributes in our dataset

FEATURES:

label	Flag indicating whether the user paid back the credit amount within 5 days of issuing the loan{1:success, 0:f
msisdn	mobile number of user
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 90 days
last_rech_date_ma	Number of days till last recharge of main account
last_rech_date_da	Number of days till last recharge of data account
last_rech_amt_ma	Amount of last recharge of main account (in Indonesian Rupiah)
cnt_ma_rech30	Number of times main account got recharged in last 30 days
fr_ma_rech30	Frequency of main account recharged in last 30 days
sumamnt_ma_rech30	Total amount of recharge in main account over last 30 days (in Indonesian Rupiah)
medianamnt_ma_rech3(	Median of amount of recharges done in main account over last 30 days at user level (in Indonesian Rupiah)
medianmarechprebal30	Median of main account balance just before recharge in last 30 days at user level (in Indonesian Rupiah)
cnt_ma_rech90	Number of times main account got recharged in last 90 days
fr_ma_rech90	Frequency of main account recharged in last 90 days
sumamnt_ma_rech90	Total amount of recharge in main account over last 90 days (in Indonasian Rupiah)
medianamnt_ma_rech9(	Median of amount of recharges done in main account over last 90 days at user level (in Indonasian Rupiah)
medianmarechprebal90	Median of main account balance just before recharge in last 90 days at user level (in Indonasian Rupiah)
cnt_da_rech30	Number of times data account got recharged in last 30 days
fr_da_rech30	Frequency of data account recharged in last 30 days
cnt_da_rech90	Number of times data account got recharged in last 90 days
fr_da_rech90	Frequency of data account recharged in last 90 days
cnt_loans30	Number of loans taken by user in last 30 days
amnt_loans30	Total amount of loans taken by user in last 30 days
maxamnt_loans30	maximum amount of loan taken by the user in last 30 days
medianamnt_loans30	Median of amounts of loan taken by the user in last 30 days
cnt_loans90	Number of loans taken by user in last 90 days
amnt_loans90	Total amount of loans taken by user in last 90 days
maxamnt_loans90	maximum amount of loan taken by the user in last 90 days
medianamnt_loans90	Median of amounts of loan taken by the user in last 90 days
payback30	Average payback time in days over last 30 days
payback90	Average payback time in days over last 90 days
pcircle	telecom circle
pdate	date

- Data Pre-processing Done

Unnamed column is similar to index hence we cannot make decision with this column so we will drop the column.

Pcircle column has 'UPW' in all the columns hence we cannot make any productive result analyzing this column, so we will drop the column.

Msisdn column again has unique id of the customers hence we remove this column.

Pdate column is in DDMMYYYY format, where for my analysis I need date and month hence I create new columns pdays and pmonth and remove Pdate column.

Converted Pmonth column with one hot encoder for prediction

Understood there is outliers in all columns after visualizing in bar plot and analyzing extent of data being skewed?

Checked outliers removal process and found

Zscore will remove 22.96%

Quantile method will remove 76.095%

Outliers removing process is not recommended as we will lose substantial data.

- Data Inputs- Logic- Output Relationships

  In the process of understanding the data and processing the same we have selected all the columns except: - unnamed: 0, label, msisdn, pcircle & pdate.

  sumamnt_ma_rech30 & 90, cnt_loans30 & 90 is removed because we already have median amount column for the same data.

- Payback90 is removed because we are making analysis of defaulters and non-defaulters based on loan amount paid in 5 days or not paid in 5 days so payback period 90 days is not related to analysis.

- Hardware and Software Requirements and Tools Used

  In case of hardware requirements 4 -8 GB RAM, i5 processor. Whereas software requirements are:-

  Python, Jupyter notebook.

  Libraries :Pandas, Numpy, Scipy, Seaborn, matplotlib.pyplot, zscore, sklearn.model_selection(train_test_split,cross_val_score), sklearn.tree for decision tree classifier, sklearn.neighbors for Kneighbors classifier, sklearn.ensemble for randomforest, Adaboost, Gradient boosting and Bagging classifiers.
  sklearn.metrics (confusion_matrix, accuracy_score, classification_report, roc_curve, auc, recall_score, precision_score), sklearn.externals (joblib)

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

  For understanding the distribution of data and any deviations present, which part of the data contains outliers we used Describe as statistical tool and observed the data on the basis of difference between :- mean and standard deviation, minimum and Q1, Q3 and maximum data. Correlation as used to understand the extent attributes bear impact on our target column 'label'.

  Used sclera to train and test the model and cross_val_score to cross validate the outcome in choose the best, then applied GridsearchCV to choose best parameters for the highest scoring models, metrics and auc_roc curve to derive the best fit model.

- Testing of Identified Approaches (Algorithms)

The classification algorithms used to predict micro credit loan defaulters are:-

Logistic regression

GaussianNB

SVC

Decision tree

Kneighbors

Randomboosting

Adaboosting

Gradientboosting

Bagging

- Run and Evaluate selected models

  Classifiers adopted and results.

  Logistic Regression: It is an appropriate analysis to conduct when the dependent variable is binary type, keeping target column positive heart disease or negative heart disease which is converted to binary type we choose logistic regression for predictive analysis.

  Accuracy score

```
Accuracy Score of  LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
         intercept_scaling=1, max_iter=100, multi_class='warn',
         n_jobs=None, penalty='l2', random_state=None, solver='warn',
         tol=0.0001, verbose=0, warm_start=False) is:
0.8834657315298552
[[  914  4383]
 [  502 36120]]
              precision    recall  f1-score   support

           0       0.65      0.17      0.27      5297
           1       0.89      0.99      0.94     36622

   micro avg       0.88      0.88      0.88     41919
   macro avg       0.77      0.58      0.60     41919
weighted avg       0.86      0.88      0.85     41919

**********************************************************************************************************************
```

Cross validated the result

```
Model: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='warn',
          n_jobs=None, penalty='l2', random_state=None, solver='warn',
          tol=0.0001, verbose=0, warm_start=False)
Score: [0.88312643 0.88397458]
MeanScore: 0.8835505022785919
Standard Deviation: 0.0004240769038196812
********************************************************************************
```

K-Nearest Neighbors: The classifier which derives value of k from root of total number of samples and classification is made by giving majority votes to its neighbors which is more apt for our dataset as we can observe for similar age group, sex, type of chest pain etc., how people have been effected.

Accuracy Score

```
Accuracy Score of  KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
          metric_params=None, n_jobs=None, n_neighbors=5, p=2,
          weights='uniform') is:
0.8873541830673441
[[ 2176  3121]
 [ 1601 35021]]
              precision    recall  f1-score   support

           0       0.58      0.41      0.48      5297
           1       0.92      0.96      0.94     36622

   micro avg       0.89      0.89      0.89     41919
   macro avg       0.75      0.68      0.71     41919
weighted avg       0.87      0.89      0.88     41919

**************************************************************************************************
```

Cross validated the result

```
Model: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
          metric_params=None, n_jobs=None, n_neighbors=5, p=2,
          weights='uniform')
Score: [0.88658072 0.8879442 ]
MeanScore: 0.8872624595369922
Standard Deviation: 0.0006817368063796736
**************************************************************************************************
```

Decision Tree Classifier: It is basically analyzing data in a tree like structure, which is widely used in medical Dataset and best on categorical data.

Accuracy score

```
Accuracy Score of  DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best') is:
0.8808893341921324
[[ 2907  2390]
 [ 2603 34019]]
              precision    recall  f1-score   support

           0       0.53      0.55      0.54      5297
           1       0.93      0.93      0.93     36622

   micro avg       0.88      0.88      0.88     41919
   macro avg       0.73      0.74      0.73     41919
weighted avg       0.88      0.88      0.88     41919

**********************************************************************************************
```

Cross validated the result

```
Model: DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best')
Score: [0.88121797 0.8852628 ]
MeanScore: 0.8832403850434474
Standard Deviation: 0.0020224112464874433
**********************************************************************************
```

GaussianNB Classifier: It is a classification algorithm which adopts Gaussian's probability distribution which means it is a latent variable probabilistically related to observed variables. I prefer using naïve bayes whenever knn is used so that we can observe supervised learning and probabilistic estimation hand in hand.

Accuracy score

```
Accuracy Score of  GaussianNB(priors=None, var_smoothing=1e-09) is:
0.5139435578138791
[[ 4906   391]
 [19984 16638]]
              precision    recall  f1-score   support

           0       0.20      0.93      0.33      5297
           1       0.98      0.45      0.62     36622

   micro avg       0.51      0.51      0.51     41919
   macro avg       0.59      0.69      0.47     41919
weighted avg       0.88      0.51      0.58     41919

**********************************************************************
```

Cross validated score

```
Model: GaussianNB(priors=None, var_smoothing=1e-09)
Score: [0.51213298 0.51497195]
MeanScore: 0.5135524632000223
Standard Deviation: 0.0014194822940804541
**********************************************************
```

Support Vector Classifier: The main objective of support vector classifier is to the data in hyperplane it leads in categorizing the data which helps to make predictions accurately.

Accuracy score

```
Accuracy Score of  SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
  kernel='rbf', max_iter=-1, probability=False, random_state=None,
  shrinking=True, tol=0.001, verbose=False) is:
0.8889047925761587
[[ 1326  3971]
 [  686 35936]]
              precision    recall  f1-score   support

           0       0.66      0.25      0.36      5297
           1       0.90      0.98      0.94     36622

   micro avg       0.89      0.89      0.89     41919
   macro avg       0.78      0.62      0.65     41919
weighted avg       0.87      0.89      0.87     41919

********************************************************************************
```

Cross validated score

```
Model: SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
  kernel='rbf', max_iter=-1, probability=False, random_state=None,
  shrinking=True, tol=0.001, verbose=False)
Score: [0.8883365 0.8883736]
MeanScore: 0.8883550501140397
Standard Deviation: 1.8551931839860814e-05
********************************************************************************
```

AdaBoost Classifier is best used to boost the performance of decision trees on binary classification in our target data we have defaulter and non-defaulter to be predicted which is in binary form and we have used AdaBoost to increase the efficiency of binary classifiers.

Accuracy score

```
Accuracy Score of  AdaBoostClassifier(algorithm='SAMME.R', base_estimator=None,
          learning_rate=1.0, n_estimators=50, random_state=None) is:
0.9085378945108423
[[ 1970  3327]
 [  507 36115]]
              precision    recall  f1-score   support

           0       0.80      0.37      0.51      5297
           1       0.92      0.99      0.95     36622

   micro avg       0.91      0.91      0.91     41919
   macro avg       0.86      0.68      0.73     41919
weighted avg       0.90      0.91      0.89     41919

********************************************************************************
```

Cross validated the result

```
Model: AdaBoostClassifier(algorithm='SAMME.R', base_estimator=None,
          learning_rate=1.0, n_estimators=50, random_state=None)
Score: [0.91036957 0.91077904]
MeanScore: 0.9105743045079409
Standard Deviation: 0.00020473285989758283
************************************************************************
```

Random Forest Classifier Random forest creates decision trees on randomly selected data samples, derives solution or predicts for each tree and finally selects the best solution out of it, so I can be considered as collection of decision trees.

Accuracy score

```
Accuracy Score of  RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=None,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False) is:
0.9148118991388153
[[ 2992  2305]
 [ 1266 35356]]
              precision    recall  f1-score   support

           0       0.70      0.56      0.63      5297
           1       0.94      0.97      0.95     36622

   micro avg       0.91      0.91      0.91     41919
   macro avg       0.82      0.77      0.79     41919
weighted avg       0.91      0.91      0.91     41919

**************************************************************************************************
```

Cross validated score

```
Model: RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators='warn', n_jobs=None,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
Score: [0.91413876 0.91544525]
MeanScore: 0.9147920047579893
Standard Deviation: 0.0006532412437665336
**********************************************************************************
```

Gradient Boosting is a machine learning boosting that fits boosted decision trees by minimizing an error gradient, as the model relies on the intuition that the best model possible next model, when combined with the previous models and hence minimizes the overall prediction error.

Accuracy score

```
Accuracy Score of  GradientBoostingClassifier(criterion='friedman_mse', init=None,
              learning_rate=0.1, loss='deviance', max_depth=3,
              max_features=None, max_leaf_nodes=None,
              min_impurity_decrease=0.0, min_impurity_split=None,
              min_samples_leaf=1, min_samples_split=2,
              min_weight_fraction_leaf=0.0, n_estimators=100,
              n_iter_no_change=None, presort='auto', random_state=None,
              subsample=1.0, tol=0.0001, validation_fraction=0.1,
              verbose=0, warm_start=False) is:
0.9176745628473961
[[ 2652  2645]
 [  806 35816]]
              precision    recall  f1-score   support

           0       0.77      0.50      0.61      5297
           1       0.93      0.98      0.95     36622

   micro avg       0.92      0.92      0.92     41919
   macro avg       0.85      0.74      0.78     41919
weighted avg       0.91      0.92      0.91     41919

**********************************************************************************************
```

Cross validated result

```
Model: GradientBoostingClassifier(criterion='friedman_mse', init=None,
              learning_rate=0.1, loss='deviance', max_depth=3,
              max_features=None, max_leaf_nodes=None,
              min_impurity_decrease=0.0, min_impurity_split=None,
              min_samples_leaf=1, min_samples_split=2,
              min_weight_fraction_leaf=0.0, n_estimators=100,
              n_iter_no_change=None, presort='auto', random_state=None,
              subsample=1.0, tol=0.0001, validation_fraction=0.1,
              verbose=0, warm_start=False)
Score: [0.91874767 0.91910951]
MeanScore: 0.918928591035564
Standard Deviation: 0.00018091696092442655
**********************************************************************************
```

Bagging is one among the best classifiers as it created using multiple estimators which can be trained using different sampling techniques.

Accuracy score

```
Accuracy Score of  BaggingClassifier(base_estimator=None, bootstrap=True,
         bootstrap_features=False, max_features=1.0, max_samples=1.0,
         n_estimators=10, n_jobs=None, oob_score=False, random_state=None,
         verbose=0, warm_start=False) is:
0.9122832128629023
[[ 3106  2191]
 [ 1486 35136]]
              precision    recall  f1-score   support

           0       0.68      0.59      0.63      5297
           1       0.94      0.96      0.95     36622

   micro avg       0.91      0.91      0.91     41919
   macro avg       0.81      0.77      0.79     41919
weighted avg       0.91      0.91      0.91     41919

********************************************************************************
```

Cross validated result

```
Model: BaggingClassifier(base_estimator=None, bootstrap=True,
        bootstrap_features=False, max_features=1.0, max_samples=1.0,
        n_estimators=10, n_jobs=None, oob_score=False, random_state=None,
        verbose=0, warm_start=False)
Score: [0.91128563 0.91243941]
MeanScore: 0.9118625172447975
Standard Deviation: 0.0005768888394042104
********************************************************************************
```

- Key Metrics for success in solving problem under consideration

  After understanding the data, data cleaning, Feature engineering, data visualization and summarising the outcome from it, training model & hyper parameter tuning next step is to apply evaluation metrics to finalize the model, we have used

  Confusion Matrix

  A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known.

  Accuracy

  Accuracy is the proportion of true results among the total number of cases examined.

  $$Accuracy = (TP+TN)/(TP+FP+FN+TN)$$

  Precision

  Precision is the ratio of correctly predicted positive observations to the total predicted positive observations, it is a valid choice of evaluation metric when we want to be very sure of our prediction.

  $$Precision = (TP)/(TP+FP)$$

  Recall

  Recall is a measure that tells us what proportion of customers actually failed to make payment within 5 days by the algorithm as having defaulter.

Auc_roc

The “Area under the Curve” (AUC) of “Receiver Characteristic Operator” (ROC)

The Receiver Operator Characteristic (ROC) curve is an evaluation metric for binary classification problems, it is a probability curve that plots the TPR against FPR at various threshold values and essentially separates the ‘signal’ from the ‘noise’. The Area under the Curve (AUC) is the measure of the ability of a classifier to distinguish between classes and is used as a summary of the ROC curve.

```
']: Model=[]
    score=[]
    cvs=[]
    rocscore=[]
    for name,model in models:
        print('**********',name,'**********')
        print('\n')
        Model.append(name)
        model.fit(x_train,y_train)
        print(model)
        pre=model.predict(x_test)
        print('\n')
        AS=accuracy_score(y_test,pre)
        print('Accuracy_score =',AS)
        score.append(AS*100)
        print('\n')
        sc=cross_val_score(model,x,y,cv=10,scoring='accuracy').mean()
        print('Cross_Val_Score =',sc)
        cvs.append(sc*100)
        print('\n')
        recall = recall_score(y_test,pre, average='binary')
        print('Recall =', recall)
        print('\n')
        precision = precision_score(y_test,pre, average='binary')
        print('Precision: %.3f' % precision)
        print('\n')
        false_positive_rate,true_positive_rate,thresholds=roc_curve(y_test,pre)
        roc_auc=auc(false_positive_rate,true_positive_rate)
        print('roc_auc_score= ',roc_auc)
        rocscore.append(roc_auc*100)
        print('\n')
        print('classification_report\n',classification_report(y_test,pre))
        print('\n')
        cm=confusion_matrix(y_test,pre)
        print(cm)
        print('\n')
        plt.figure(figsize=(10,40))
        plt.subplot(911)
        plt.title(name)
        print(sns.heatmap(cm,annot=False))
        plt.subplot(912)
        plt.title(name)
        plt.plot(false_positive_rate,true_positive_rate,label='AUC=%0.2f'% roc_auc)
        plt.plot([0,1],[0,1],'r--')
        plt.legend(loc='lower right')
        plt.ylabel('Trur Positive Rate')
        plt.xlabel('False Positive Rate')
        print('\n\n')
```

The above picture shows the key metrics passed for the model and their results are as follows:

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=None,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)


Accuracy_score = 0.9161955199312961


Cross_Val_Score = 0.9165716459763722


Recall = 0.966713997050953


Precision: 0.939


roc_auc_score=  0.7668193356974605


classification_report
               precision    recall  f1-score   support

           0       0.71      0.57      0.63      5297
           1       0.94      0.97      0.95     36622

   micro avg       0.92      0.92      0.92     41919
   macro avg       0.83      0.77      0.79     41919
weighted avg       0.91      0.92      0.91     41919


[[ 3003  2294]
 [ 1219 35403]]


AxesSubplot(0.125,0.808774;0.62x0.0712264)
```

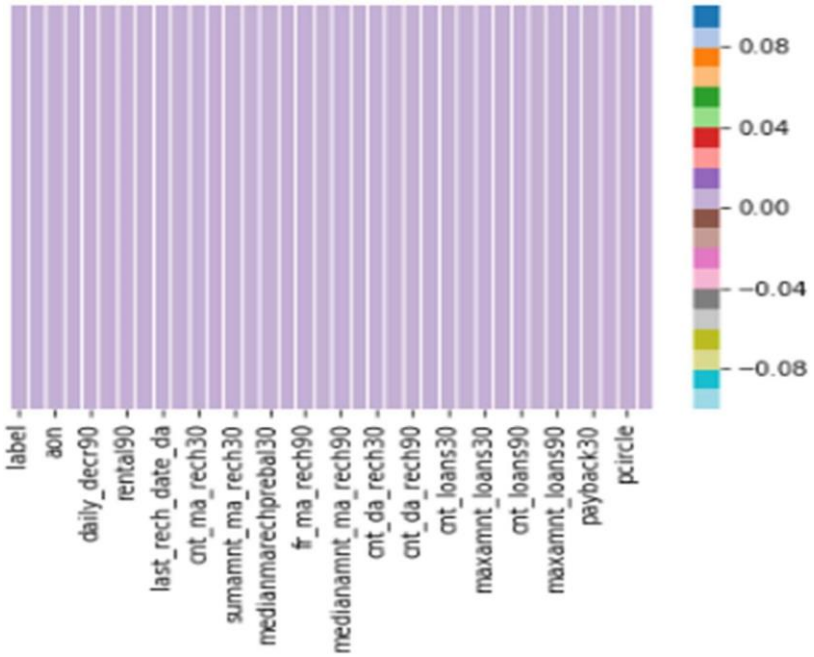
```
********** AdaBoostClassifier **********

AdaBoostClassifier(algorithm='SAMME.R', base_estimator=None,
          learning_rate=1.0, n_estimators=50, random_state=None)

Accuracy_score = 0.9085378945108423

Cross_Val_Score = 0.9101496726294751

Recall = 0.9861558625962537

Precision: 0.916

roc_auc_score=  0.6790322450606339

classification_report
              precision    recall  f1-score   support

           0       0.80      0.37      0.51      5297
           1       0.92      0.99      0.95     36622

   micro avg       0.91      0.91      0.91     41919
   macro avg       0.86      0.68      0.73     41919
weighted avg       0.90      0.91      0.89     41919


[[ 1970  3327]
 [  507 36115]]

AxesSubplot(0.125,0.808774;0.62x0.0712264)
```

```
********** GradientBoostingClassifier **********


GradientBoostingClassifier(criterion='friedman_mse', init=None,
              learning_rate=0.1, loss='deviance', max_depth=3,
              max_features=None, max_leaf_nodes=None,
              min_impurity_decrease=0.0, min_impurity_split=None,
              min_samples_leaf=1, min_samples_split=2,
              min_weight_fraction_leaf=0.0, n_estimators=100,
              n_iter_no_change=None, presort='auto', random_state=None,
              subsample=1.0, tol=0.0001, validation_fraction=0.1,
              verbose=0, warm_start=False)


Accuracy_score = 0.9176745628473961


Cross_Val_Score = 0.9189667424206002


Recall = 0.9779913713068648


Precision: 0.931


roc_auc_score=  0.739326061337782


classification_report
               precision    recall  f1-score   support

           0       0.77      0.50      0.61      5297
           1       0.93      0.98      0.95     36622

   micro avg       0.92      0.92      0.92     41919
   macro avg       0.85      0.74      0.78     41919
weighted avg       0.91      0.92      0.91     41919


[[ 2652  2645]
 [  806 35816]]


AxesSubplot(0.125,0.808774;0.62x0.0712264)
```

```
********** GradientBoostingClassifier **********


GradientBoostingClassifier(criterion='friedman_mse', init=None,
              learning_rate=0.1, loss='deviance', max_depth=3,
              max_features=None, max_leaf_nodes=None,
              min_impurity_decrease=0.0, min_impurity_split=None,
              min_samples_leaf=1, min_samples_split=2,
              min_weight_fraction_leaf=0.0, n_estimators=100,
              n_iter_no_change=None, presort='auto', random_state=None,
              subsample=1.0, tol=0.0001, validation_fraction=0.1,
              verbose=0, warm_start=False)


Accuracy_score = 0.9176745628473961


Cross_Val_Score = 0.9189667424206002


Recall = 0.9779913713068648


Precision: 0.931


roc_auc_score=  0.739326061337782


classification_report
               precision    recall  f1-score   support

           0       0.77      0.50      0.61      5297
           1       0.93      0.98      0.95     36622

   micro avg       0.92      0.92      0.92     41919
   macro avg       0.85      0.74      0.78     41919
weighted avg       0.91      0.92      0.91     41919


[[ 2652  2645]
 [  806 35816]]


AxesSubplot(0.125,0.808774;0.62x0.0712264)
```

- Visualizations

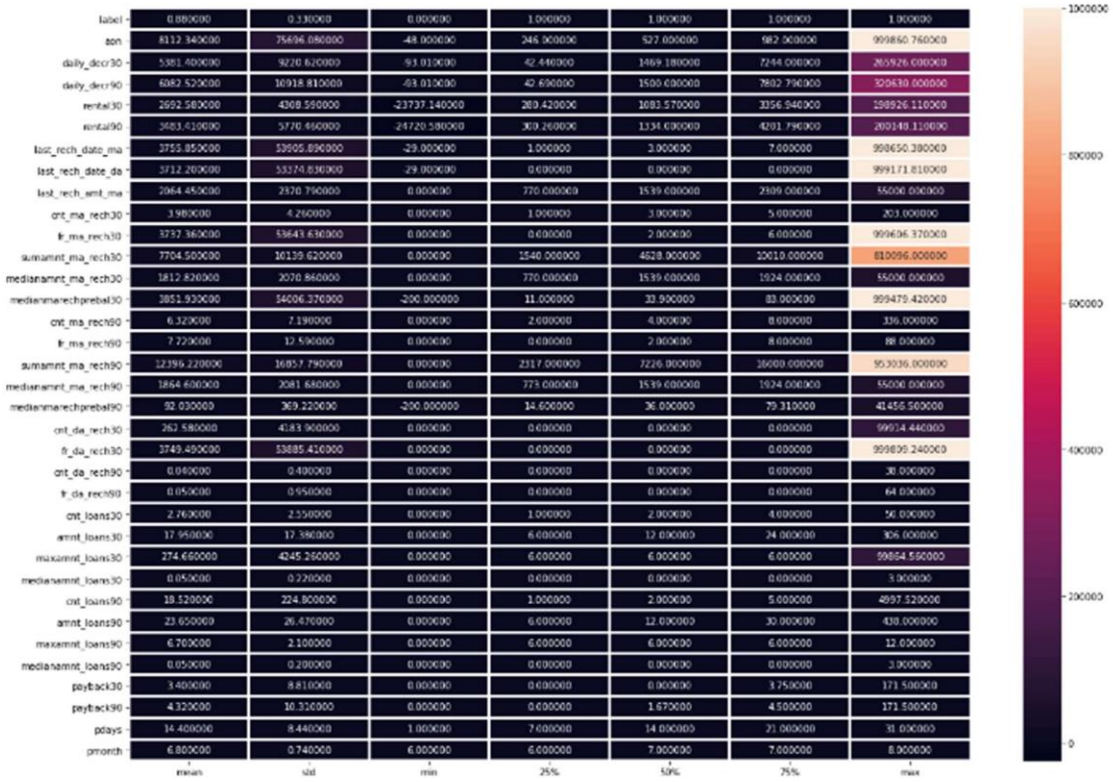
  Chart showing there if there is any null values in the dataset



We can observe all columns are shaded with colour for 0 which means there is no null or Nan values

Chart showing data distribution

```
[17]: plt.figure(figsize=(20,15))
      sns.heatmap(round(data.describe()[1:].transpose(),2),linewidth=2,annot=True,fmt="f")

      plt.show()
```

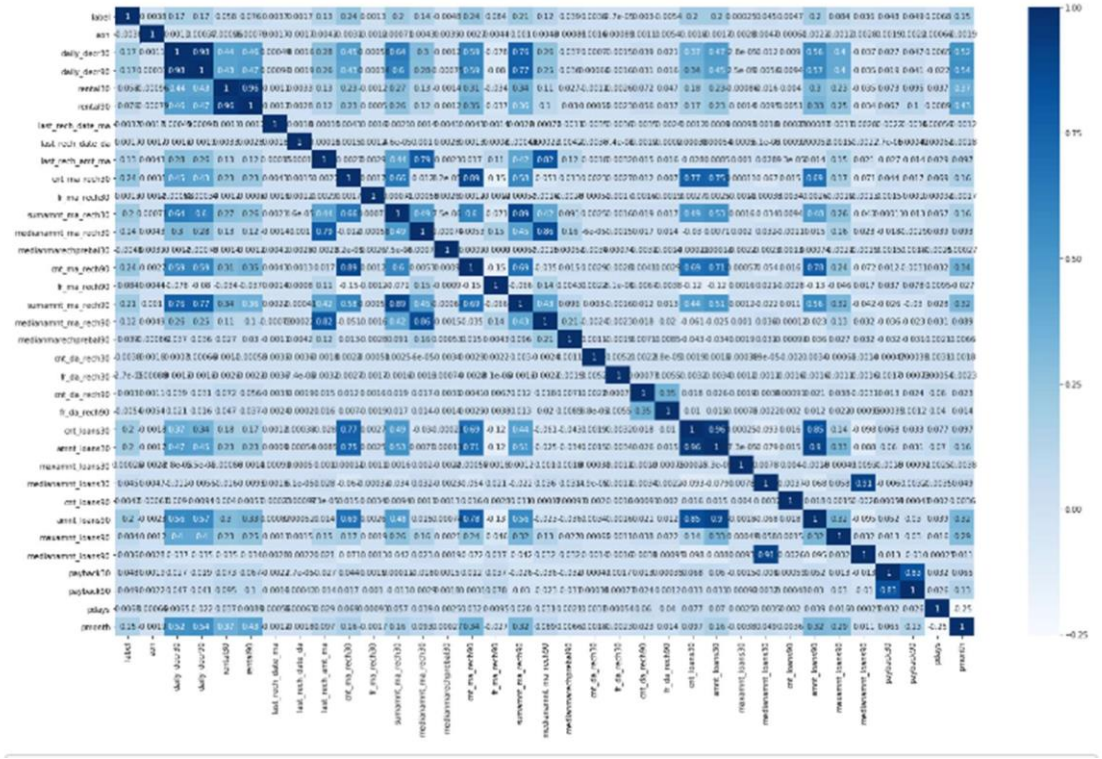


In Descriptive Statistics we found the outliers are high on upper limit i.e. Q3 and maximum variables and the data is dispersed exceedingly in following columns:

- Aon
- last_rech_date_ma
- last_rech_date_da
- Medianmarechprebal30
- sumamnt_ma_rech90
- fr_da_rech30

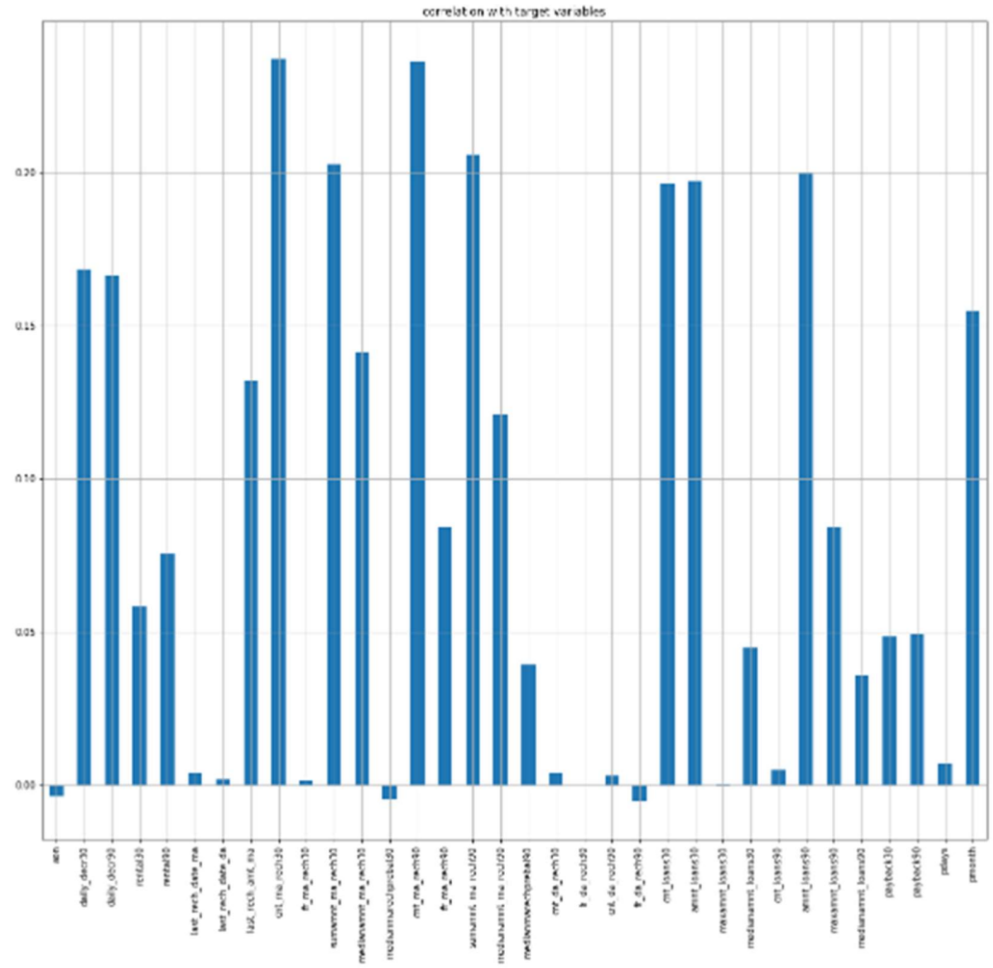
Chart showing correlation of all attributes

```
plt.figure(figsize=(25,15))
sns.heatmap(datacor,cmap='Blues',annot=True)
plt.show()
```



As we have correlation results for all 37 columns it would be difficult to derive any conclusion so next we will be observing correlation all attributes with ‘label’ column.

Chart showing correlation of all variables with column ‘label’.



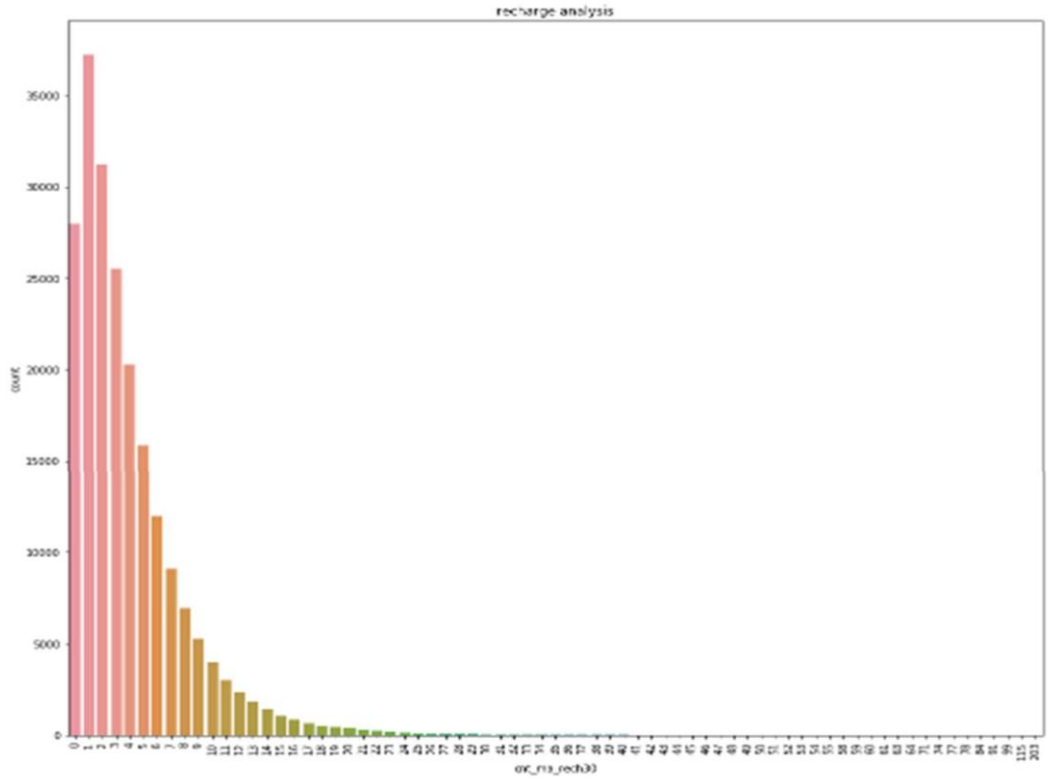
From the above chart we can understand

- Number of times main account was recharged in last 30 and 90 days
- Total amount of recharge in main account over last 30 days and 90 days
- Number of loans taken by the user in last 30 days
- Total amount of loan taken by users in last 30 days and 90 days.

Are the columns, comparatively highly positive correlated with label column.

Chart showing count plot

```
23]: plt.subplots(figsize=(16,14))
     sns.countplot(x='cnt_ma_rech30',data=data)
     plt.title("recharge analysis")
     plt.xlabel('cnt_ma_rech30')
     plt.ylabel('count')
     plt.xticks(rotation=90)

     plt.show()
```



Recharge made once by customers is highest for about 40,000 customers, followed by twice recharge in last 30 days is 30,000 and then no recharge is about 27,000 customers.

We have also used count plot to analyse number of defaulters and non-defaulters in our population and found there are more number of non-defaulters hence population is biased.

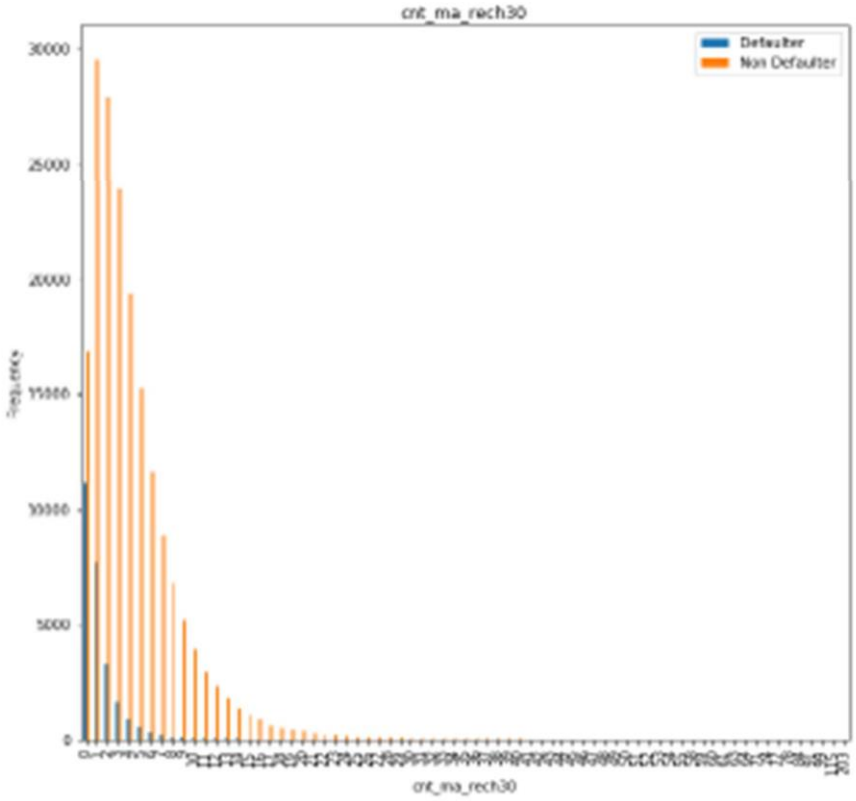
, therefore we are not making analysis based on count plot as it will show more number of non-defaulters in all the columns due to

uneven data distribution. We w

```
In [17]: df = data.copy()

def chng1(prob):
    if prob == 0:
        return 'Defaulter'
    else:
        return 'Non-Defaulter'
df['label'] = df['label'].apply(chng1)
```

In order to analyse each column on basis number of defaulter and non - defaulter we are using define operand on label column

```
In [25]: pd.crosstab(df.cnt_ma_rech30,df.label).plot(kind="bar",figsize=(10,10))
plt.title('cnt_ma_rech30')
plt.xlabel('cnt_ma_rech30')
plt.xticks(rotation=90)
plt.legend(["Defaulter", "Non Defaulter"])
plt.ylabel('Frequency')
plt.show()
```



We have used count plot for the same column but with bifurcation based on defaulters and non-defaulters and found in case of defaulters 12,000 customers have not made any recharge in last 30 days, 7,000 customers have made recharge once in last 30 days followed by 3,000 customers have made recharge twice in last 30 days.

In case of non-defaulters 30,000 customers have made once, 27,000 customers twice and 23,000 customers thrice in last 30 days.

Here important point to observe is that we cannot compare whether defaulters have made higher number of recharge or non- defaulters have made higher number of recharge, because the data is biased in all cases since non – defaulters count is highest we will find non defaulters always on higher side, for re confirmation I have attached another analysis based on count plot, due to this result expected I have choose scatterplot for

Bi-variate analysis.

```
pd.crosstab(df.cnt_loans30,df.label).plot(kind="bar",figsize=(15,10),color=['#FFC300','#581845' ])
plt.title('cnt_loans30')
plt.xlabel('cnt_loans30')
plt.xticks(rotation=90)
plt.legend(["Defaulter", "Non Defaulter"])
plt.ylabel('Frequency')
plt.show()
```

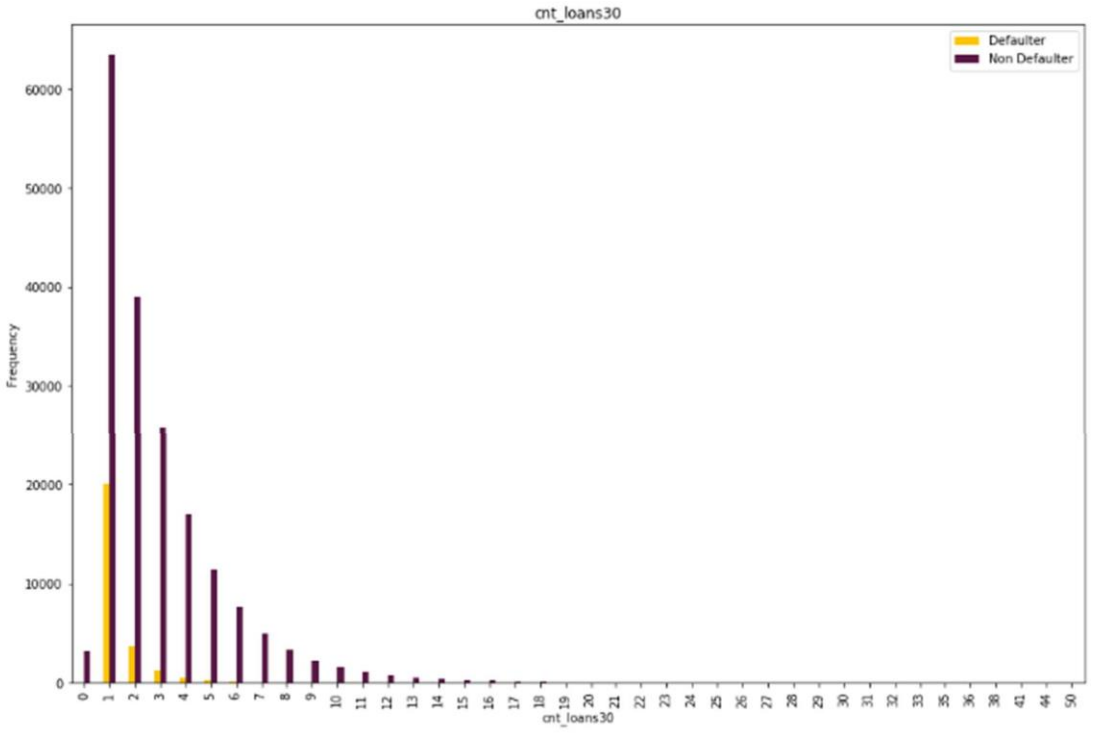
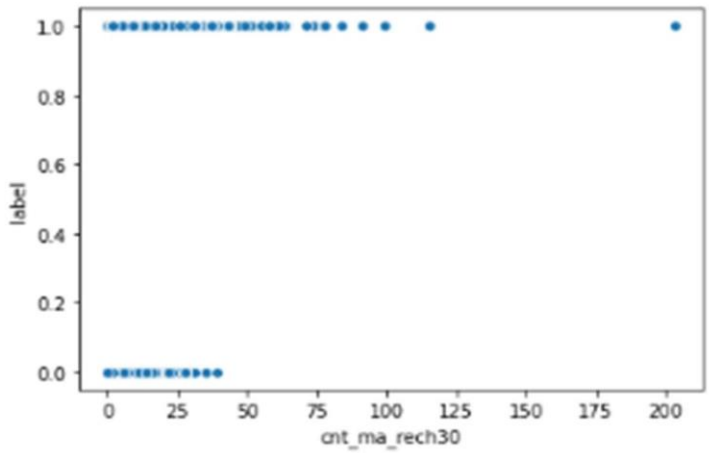


Diagram showing comparative analysis with the help of scatter plot.

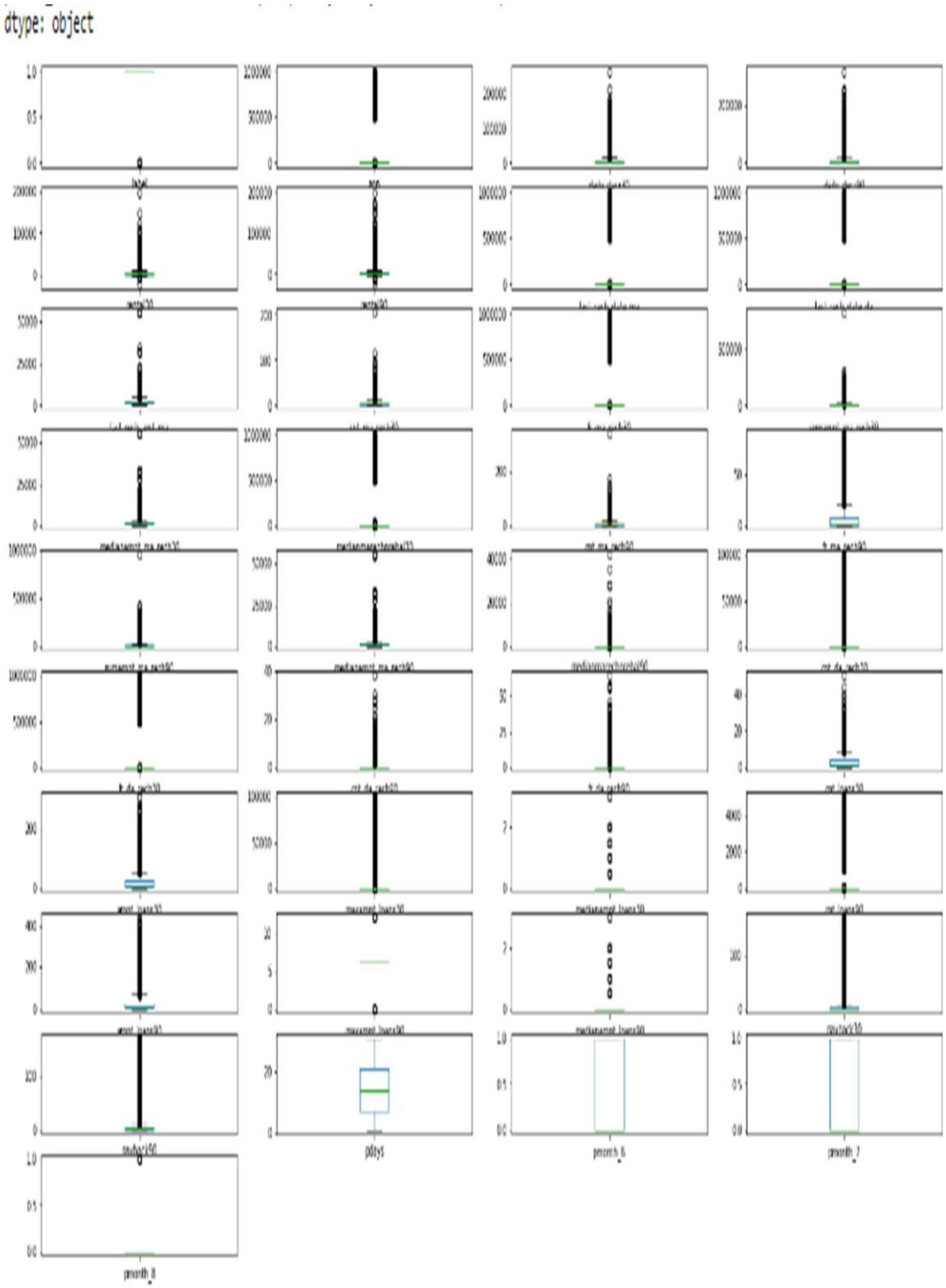
```
21]: sns.scatterplot(x="cnt_ma_rech30",y="label",data=data)
21]: <matplotlib.axes._subplots.AxesSubplot at 0x2fab8391520>
```



Number of times main account got recharged in last 30days is more in number by the non-defaulters ranging from 0-100 compared to defaulters ranging 0-40.

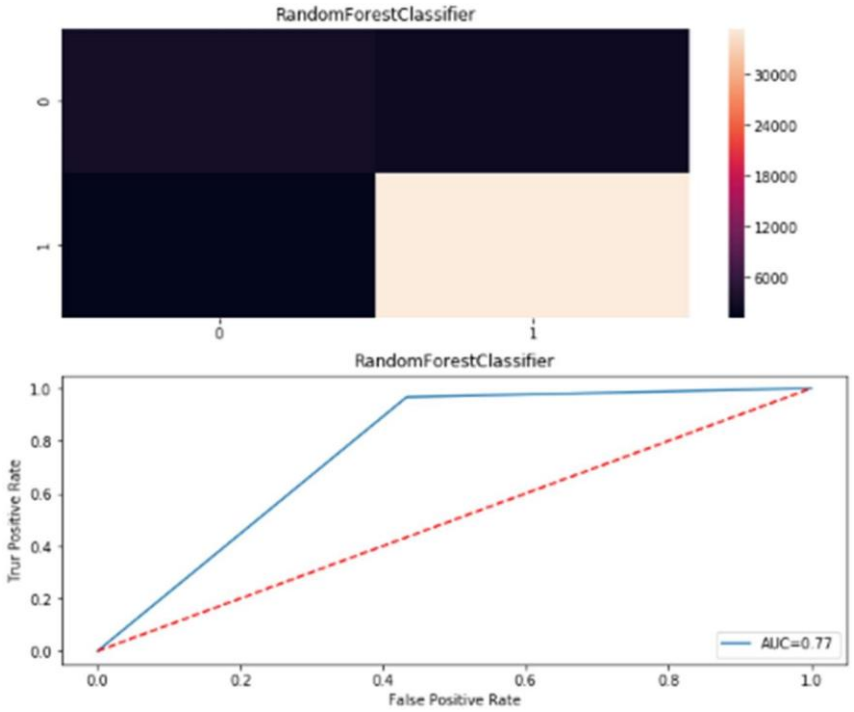
.

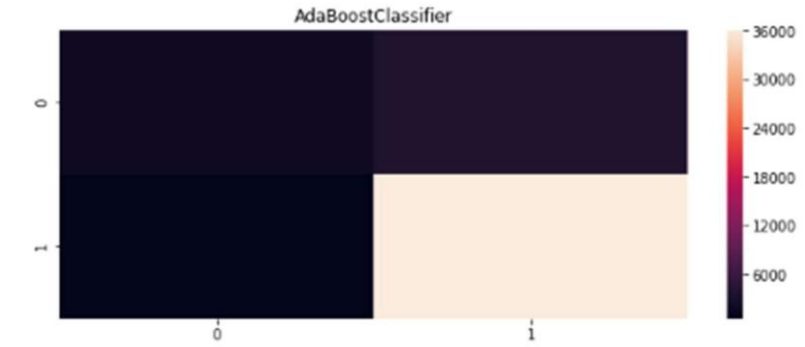
Chart showing outliers of all columns

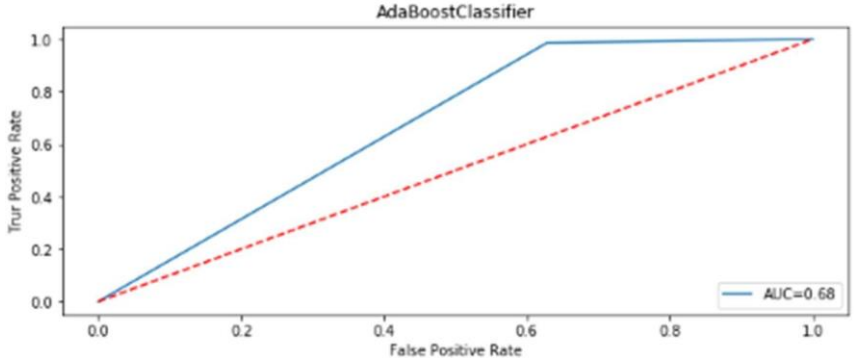


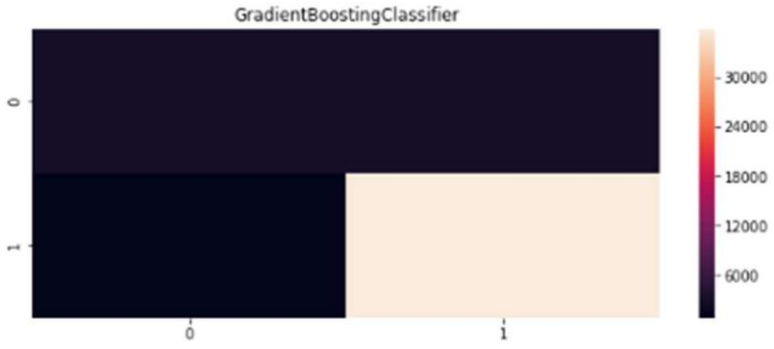
We can observe huge outliers are found in all the columns.

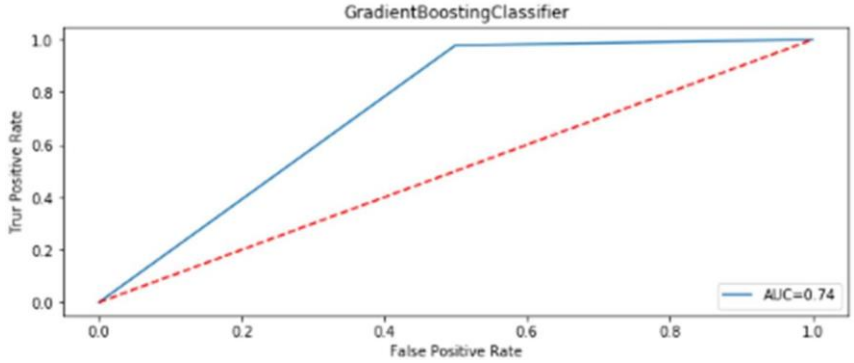
Diagrams showing auc_roc curve of highest accuracy derived algorithms.

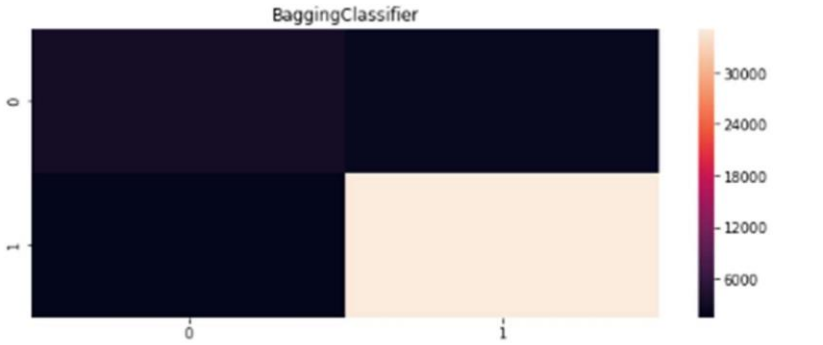


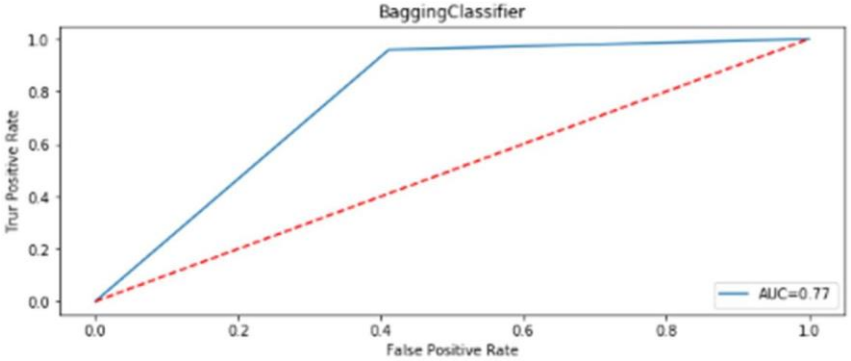

AdaBoostClassifier
0
1
0
1
36000
30000
24000
18000
12000
6000


AdaBoostClassifier
Trur Positive Rate
False Positive Rate
AUC=0.68
0.0
0.2
0.4
0.6
0.8
1.0


GradientBoostingClassifier
0
1
0
1
30000
24000
18000
12000
6000


GradientBoostingClassifier
Trur Positive Rate
False Positive Rate
AUC=0.74
0.0
0.2
0.4
0.6
0.8
1.0


BaggingClassifier
0
1
0
1
30000
24000
18000
12000
6000


BaggingClassifier
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
AUC=0.77

- Interpretation ion of the Results

  The best results interpreted after pre-processing and model building are as follows

t[48]:

	Model	Accuracy_score	Cross_val_score	Roc_cuc_curve	Recall	precision
0	RandomForestClassifier	91.619552	91.657165	76.681934	0.959451	0.941607
1	AdaBoostClassifier	90.853789	91.014967	67.903225	0.959451	0.941607
2	GradientBoostingClassifier	91.767456	91.896674	73.932606	0.959451	0.941607
3	BaggingClassifier	91.259333	91.311256	77.404284	0.959451	0.941607

We can conclude Gradient Boosting Classifier has the best predictable accuracy score hence we have dumped Gradient Boosting Classifier in the pickle file.

CONCLUSION

- Key Findings and Conclusions of the Study

  We understood Pdate column was important but not based on year hence we extracted Pmonth and Pdays from Pdate column and deleted Pdate.

  We could observe from 37 columns there were few columns repeated like for example data column, data with minimum values, data with maximum values and data with median column such information has deleted and reserved only the apt column for analysis.

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

  With the help of exploratory data analysis we could visualise the large data and make interpretation of it and also summary statistics helped us understand how data is spread, where outliers are present , to what extent deviation is present and understanding the relationship between target column and other attributes, repeated columns was eliminated to making right prediction.

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

  Limitation of the study is, we could have made better data analysis if the population had equal number defaulters and non – defaulters.

  Once we make analysis based on above specification we can understand columns better in order to delete or divide the important columns categorically for better analysis.