FLIP ROBO

MICRO CREDT PROJECT

Submitted by:

MANOJ.I.V

ACKNOWLEDGMENT

This includes mentioning of all the references, research papers, data sources, professionals and other resources that helped you and guided you in completion of the project.

INTRODUCTION

- ## Business Problem Framing

  Describe the business problem and how this problem can be related to the real world.

  Answer: 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. In, real world also, there are many financial corporations and institutions that wants to help the local businesses to grow.

- ## Conceptual Background of the Domain Problem

  Describe the domain related concepts that you think will be useful for better understanding of the project.

  Answer: Bank account details, transaction details, total amount of profit for a business, previous loans, fine paid for missing the instalments, total instalments missing, time period and interest of the loan, loan amount, field/domain of business etc.

- ## Review of Literature

  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

  Describe your objective behind to make this project, this domain and what is the motivation behind.
  Answer: The objective behind the project is to earn money helping the unbanked poor families living in remote areas with

not much sources of income. It is type of win-win situation as it provides an opportunity to small vendors to prosper and at the same time the credit company also earns money. So a model have to be build such that it can predict whether the vendor or the party going for the credit can give back the money or not.

Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem

  Describe the mathematical, statistical and analytics modelling done during this project along with the proper justification.

- Data Sources and their formats

  What are the data sources, their origins, their formats and other details that you find necessary? They can be described here. Provide a proper data description. You can also add a snapshot of the data.

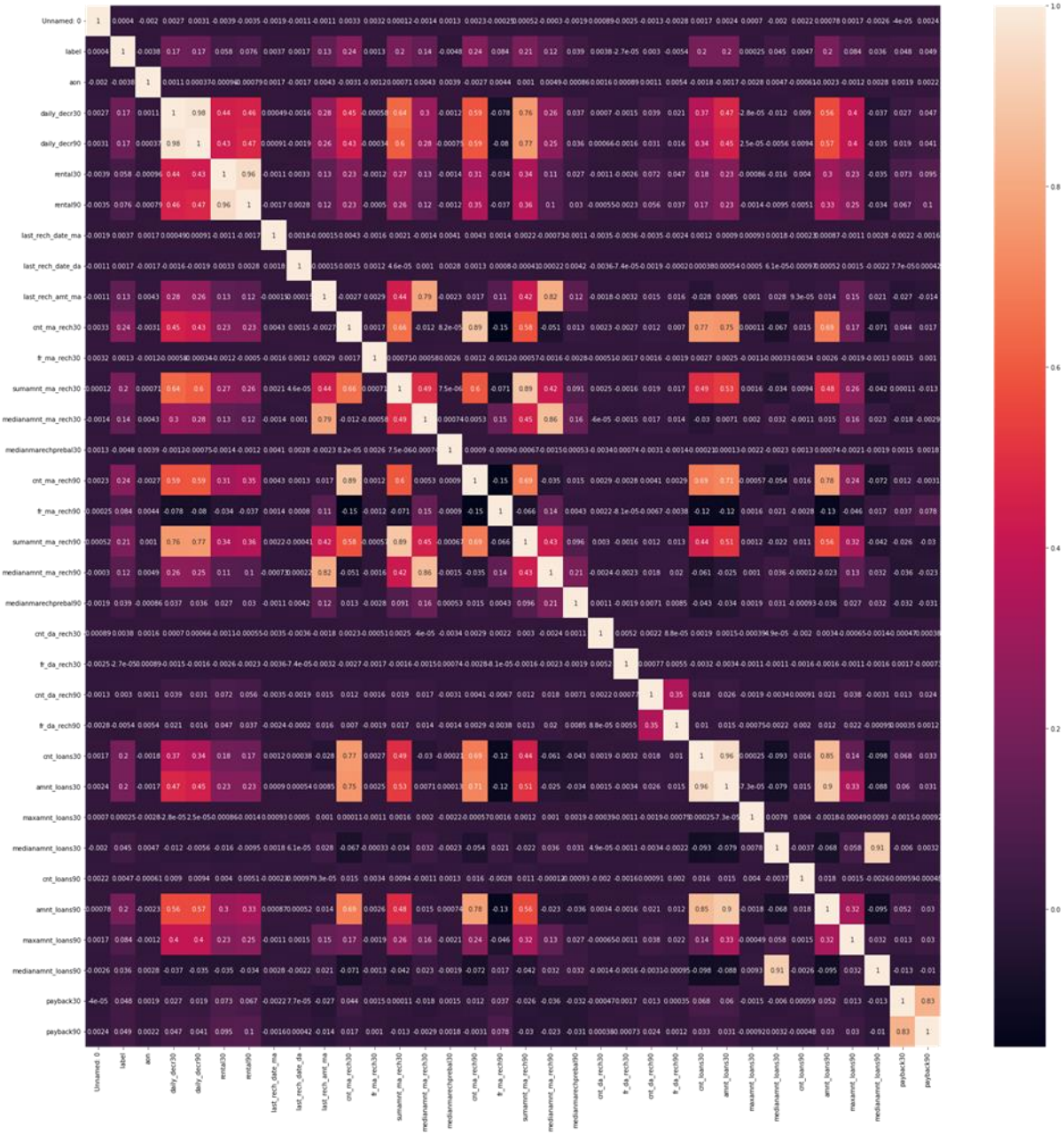
  Answer: The data can be taken by a survey conducted by micro credit company, open source websites like Kaggel etc. The data is in the form of .csv file it may also be in .json or Excel files. Currently the data was provided in terms of .csv files. There is training file with 1168 columns and 81 rows. There is a test file with 37 columns and 209593 rows. The below shows the column list.

  Data columns (total 37 columns):

#	Column	Non-Null Count	Dtype
0	Unnamed: 0	209593 non-null	int64
1	label	209593 non-null	int64
2	msisdn	209593 non-null	object
3	aon	209593 non-null	float64
4	daily_decr30	209593 non-null	float64
5	daily_decr90	209593 non-null	float64
6	rental30	209593 non-null	float64
7	rental90	209593 non-null	float64

```
 8   last_rech_date_ma     209593 non-null  float64
 9   last_rech_date_da     209593 non-null  float64
 10  last_rech_amt_ma      209593 non-null  int64
 11  cnt_ma_rech30         209593 non-null  int64
 12  fr_ma_rech30          209593 non-null  float64
 13  sumamnt_ma_rech30     209593 non-null  float64
 14  medianamnt_ma_rech30  209593 non-null  float64
 15  medianmarechprebal30  209593 non-null  float64
 16  cnt_ma_rech90         209593 non-null  int64
 17  fr_ma_rech90          209593 non-null  int64
 18  sumamnt_ma_rech90     209593 non-null  int64
 19  medianamnt_ma_rech90  209593 non-null  float64
 20  medianmarechprebal90  209593 non-null  float64
 21  cnt_da_rech30         209593 non-null  float64
 22  fr_da_rech30          209593 non-null  float64
 23  cnt_da_rech90         209593 non-null  int64
 24  fr_da_rech90          209593 non-null  int64
 25  cnt_loans30           209593 non-null  int64
 26  amnt_loans30          209593 non-null  int64
 27  maxamnt_loans30       209593 non-null  float64
 28  medianamnt_loans30    209593 non-null  float64
 29  cnt_loans90           209593 non-null  float64
 30  amnt_loans90          209593 non-null  int64
 31  maxamnt_loans90       209593 non-null  int64
 32  medianamnt_loans90    209593 non-null  float64
 33  payback30             209593 non-null  float64
 34  payback90             209593 non-null  float64
 35  pcircle               209593 non-null  object
 36  pdate                 209593 non-null  object
dtypes: float64(21), int64(13), object(3)
memory usage: 59.2+ MB
```

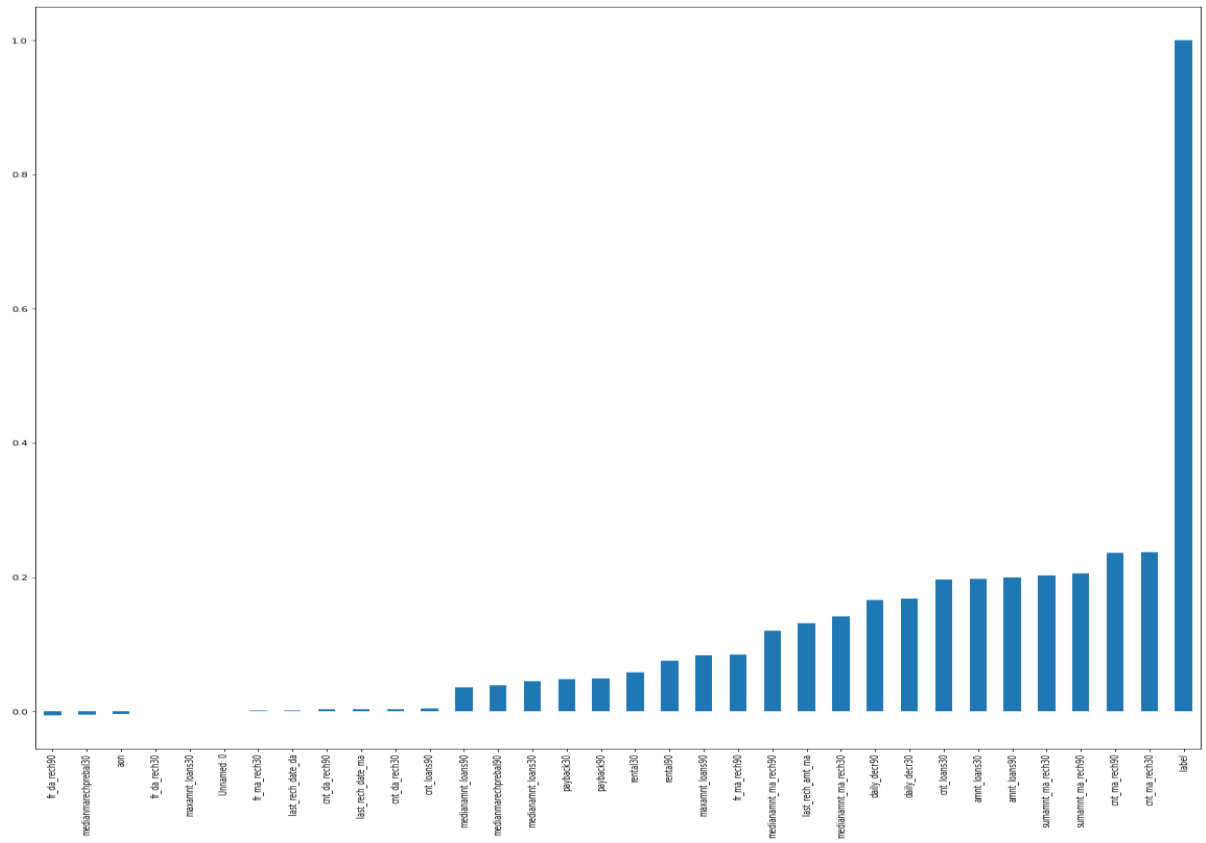
The correlation was performed. It was as shown in the fig.



Form the correlation label was seen:

fr_da_rech90	-0.005418
medianmarechprebal30	-0.004829
aon	-0.003785
fr_da_rech30	-0.000027
maxamnt_loans30	0.000248
Unnamed: 0	0.000403
fr_ma_rech30	0.001330
last_rech_date_da	0.001711
cnt_da_rech90	0.002999
last_rech_date_ma	0.003728

cnt_da_rech30 0.003827
cnt_loans90 0.004733
medianamnt_loans90 0.035747
medianmarechprebal90 0.039300
medianamnt_loans30 0.044589
payback30 0.048336
payback90 0.049183
rental30 0.058085
rental90 0.075521
maxamnt_loans90 0.084144
fr_ma_rech90 0.084385
medianamnt_ma_rech90 0.120855
last_rech_amt_ma 0.131804
medianamnt_ma_rech30 0.141490
daily_decr90 0.166150
daily_decr30 0.168298
cnt_loans30 0.196283
amnt_loans30 0.197272
amnt_loans90 0.199788
sumamnt_ma_rech30 0.202828
sumamnt_ma_rech90 0.205793
cnt_ma_rech90 0.236392
cnt_ma_rech30 0.237331
label 1.000000

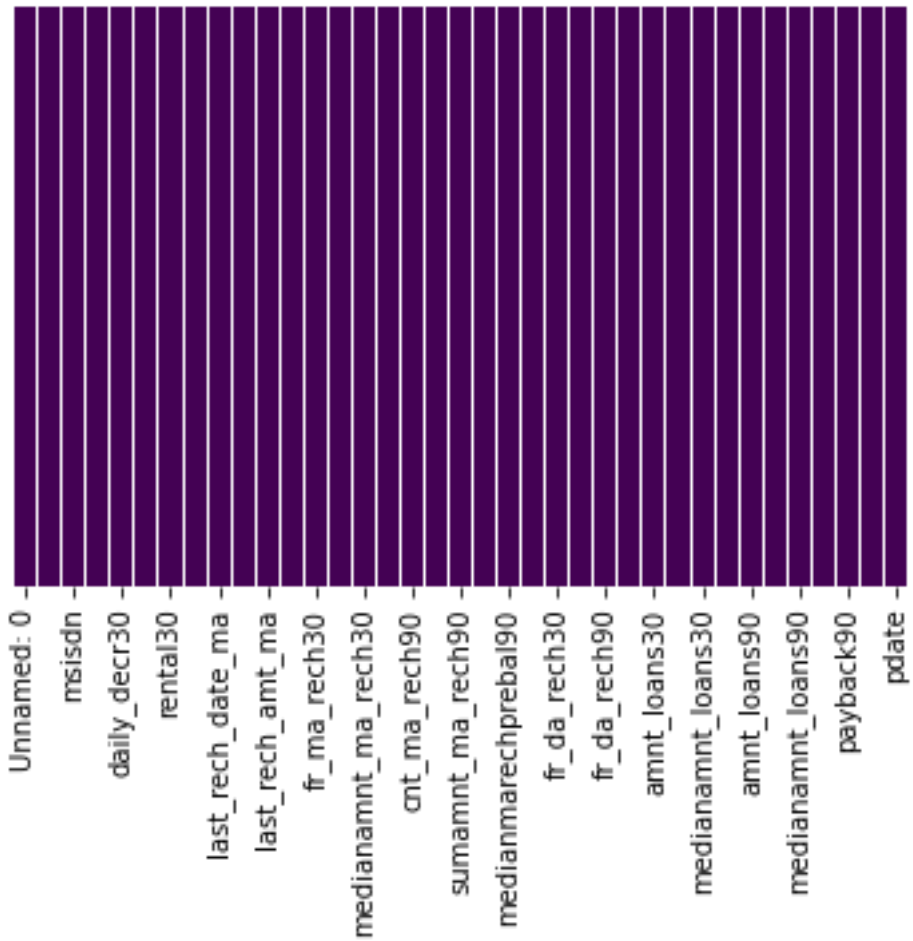


- Data Preprocessing Done

  What were the steps followed for the cleaning of the data? What were the assumptions done and what were the next actions steps over that?

  Answer: As there are no null values there was no imputation done. The assumptions were unnamed 0, pcircle, msisdn, and year was neglected as the data was correctly to the specific year.

  dt.drop(columns=['Unnamed:0','msisdn','pcircle','year'],inplace =True)



- Data Inputs- Logic- Output Relationships

  Describe the relationship behind the data input, its format, the logic in between and the output. Describe how the input affects the output.

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

  Here, you can describe any presumptions taken by you.

- Hardware and Software Requirements and Tools Used

  Listing down the hardware and software requirements along with the tools, libraries and packages used. Describe all the software tools used along with a detailed description of tasks done with those tools.

The different libraries and packages used are:

1. Pandas, 2. Numpy, 3. Matplotlib, 4. Sklearn and 5.Dtale etc.

Pandas: for importing the dataset

Matplotlib and Dtale: For graphing

Sklearn: Modelling

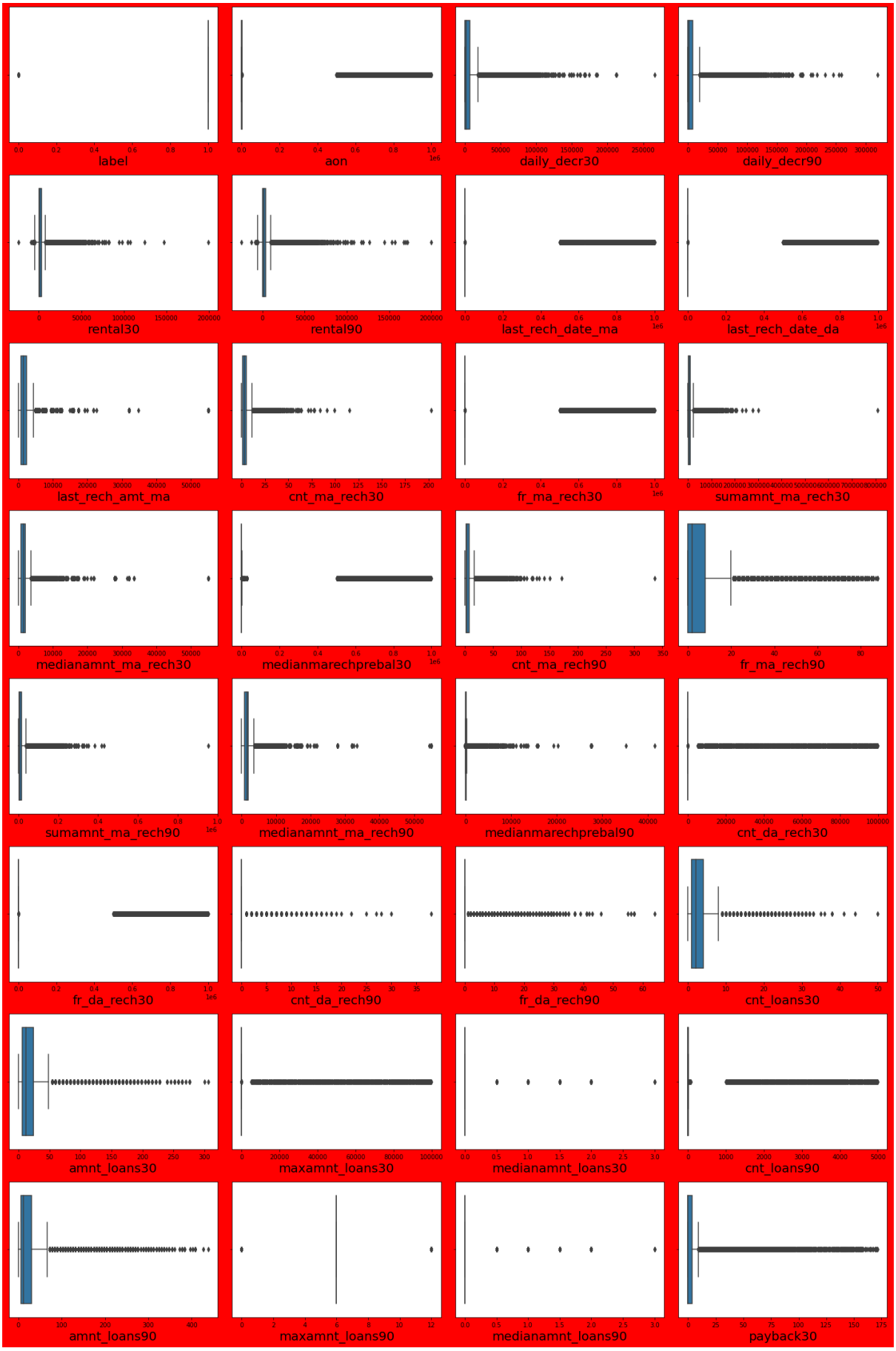
Model/s Development and Evaluation

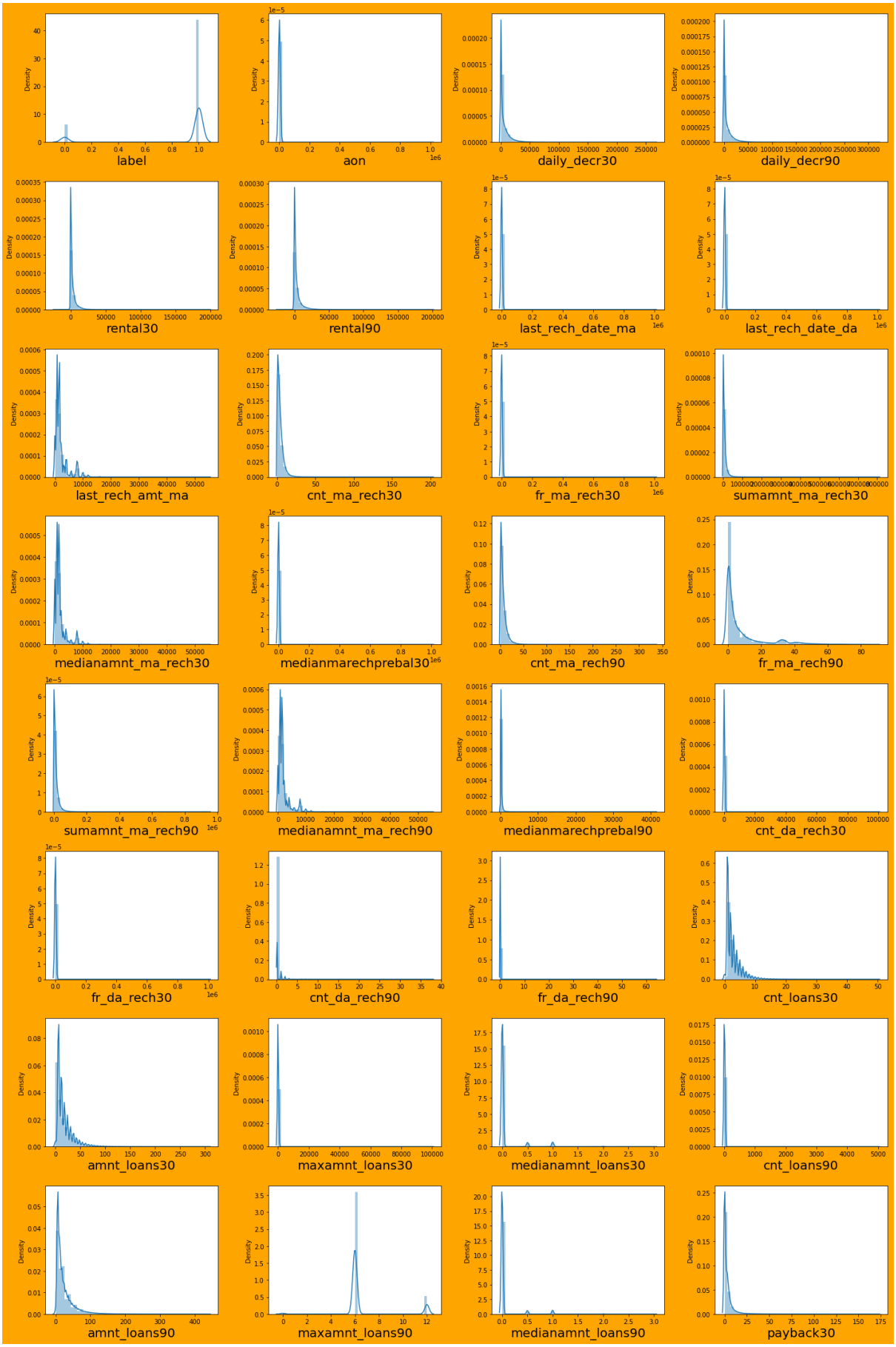
- Identification of possible problem-solving approaches (methods)

  Describe the approaches you followed, both statistical and analytical, for solving of this problem.

  Answer: Both box plot and kde plots were ploted and zscore was applied.

```
[ ] dt.drop_duplicates(inplace=True)

from scipy.stats import zscore
z=np.abs(zscore(dt))
z

array([[2.64935018, 0.10358473, 0.25237   , ..., 2.39398367, 0.66364934,
        0.27324694],
       [0.37745105, 0.09777242, 0.73091668, ..., 0.41926306, 0.52133598,
        1.62198858],
       [0.37745105, 0.10011055, 0.43207267, ..., 0.41926306, 0.54515081,
        1.62198858],
       ...,
       [0.37745105, 0.09379627, 0.70067198, ..., 0.04739711, 1.73013613,
        0.27324694],
       [0.37745105, 0.08429843, 0.77063318, ..., 0.59932627, 1.25614201,
        0.27324694],
       [0.37745105, 0.08629311, 0.09682266, ..., 0.41926306, 0.87683157,
        0.27324694]])

[ ] ab=dt[(z<3).all(axis=1)]

[ ] ab
```


label
aon
daily_decr30
daily_decr90
rental30
rental90
last_rech_date_ma
last_rech_date_da
last_rech_amt_ma
cnt_ma_rech30
fr_ma_rech30
sumamnt_ma_rech30
medianamnt_ma_rech30
medianmarechprebal30
cnt_ma_rech90
fr_ma_rech90
sumamnt_ma_rech90
medianamnt_ma_rech90
medianmarechprebal90
cnt_da_rech30
fr_da_rech30
cnt_da_rech90
fr_da_rech90
cnt_loans30
amnt_loans30
maxamnt_loans30
medianamnt_loans30
cnt_loans90
amnt_loans90
maxamnt_loans90
medianamnt_loans90
payback30


label
aon
daily_decr30
daily_decr90
rental30
rental90
last_rech_date_ma
last_rech_date_da
last_rech_amt_ma
cnt_ma_rech30
fr_ma_rech30
sumamnt_ma_rech30
medianamnt_ma_rech30
medianmarechprebal30
cnt_ma_rech90
fr_ma_rech90
sumamnt_ma_rech90
medianamnt_ma_rech90
medianmarechprebal90
cnt_da_rech30
fr_da_rech30
cnt_da_rech90
fr_da_rech90
cnt_loans30
amnt_loans30
maxamnt_loans30
medianamnt_loans30
cnt_loans90
amnt_loans90
maxamnt_loans90
medianamnt_loans90
payback30

- Testing of Identified Approaches (Algorithms)

  Listing down all the algorithms used for the training and testing.

  Answer: The algorithms used were Logistic regression, KNN, Random forest, decision tree, EGboost, AdaBoost, Gradient Boost, servo vector classification (SVC). The SVC model takes more time for compution so it was neglected in ipython note book.

- Run and Evaluate selected models

  Describe all the algorithms used along with the snapshot of their code and what were the results observed over different evaluation metrics.

  Answer: First the models were imported and random values were found.

```
[ ] # Importing machine learning libraries
    from sklearn.linear_model import LogisticRegression
    from sklearn.linear_model import LinearRegression
    from sklearn.metrics import accuracy_score
    from sklearn.linear_model import LassoCV, RidgeCV
    from sklearn.tree import DecisionTreeClassifier
    from sklearn.ensemble import RandomForestClassifier
    from xgboost import XGBRFClassifier
    from sklearn.ensemble import AdaBoostClassifier, GradientBoostingClassifier
    from sklearn.neighbors import KNeighborsClassifier
    from sklearn.svm import SVC
    from sklearn.metrics import classification_report, confusion_matrix, mean_squared_error,mean_absolute_error, r2_score
```

```
    # Finding the random state
    maxAc=0
    maxrs=0

    for i in range(1,500):
        x_train,x_test,y_train, y_test=train_test_split(prin_comp,y_train_ns,test_size=.20, random_state=i)
        lr=LogisticRegression()
        lr.fit(x_train, y_train)
        pred=lr.predict(x_test)
        acc=accuracy_score(y_test,pred)
        print('accuracy of the model', acc,'random_state', i)

        if acc>maxAc:
            maxAc=acc
            maxrs=i
            print ('accuracy of the optimum model', acc,'random_state', i)
```

```
[ ] print ('Optimum accuracy is obtaned in', maxAc, 'in random state', maxrs)
```

```
[ ] # Finding the random state
    maxAc=0
    maxrs=0

    for i in range(1,500):
        x_train,x_test,y_train, y_test=train_test_split(prin_comp,y_train_ns,test_size=.20, random_state=i)
        kn=KNeighborsClassifier()
        kn.fit(x_train, y_train)
        pred=kn.predict(x_test)
        acc=accuracy_score(y_test,pred)
        print('accuracy of the model', acc,'random_state', i)

        if acc>maxAc:
            maxAc=acc
            maxrs=i
            print ('accuracy of the optimum model', acc,'random_state', i)
```

```
[ ] print ('Optimum accuracy is obtaned in', maxAc, 'in random state', maxrs)
```

```
# Finding the random state
maxAc=0
maxrs=0

for i in range(1,800):
    x_train,x_test,y_train, y_test=train_test_split(prin_comp,y_train_ns,test_size=.20, random_state=i)
    de=DecisionTreeClassifier()
    de.fit(x_train, y_train)
    pred=de.predict(x_test)
    acc=accuracy_score(y_test,pred)
    print('accuracy of the model', acc,'random_state', i)

    if acc>maxAc:
        maxAc=acc
        maxrs=i
        print ('accuracy of the optimum model', acc,'random_state', i)
```

```
[ ] print ('Optimum accuracy is obtaned in', maxAc, 'in random state', maxrs)
```

```
[ ] # Finding the random state
maxAc=0
maxrs=0

for i in range(1,1000):
    x_train,x_test,y_train, y_test=train_test_split(prin_comp,y_train_ns,test_size=.20, random_state=i)
    xe=XGBRFClassifier()
    xe.fit(x_train, y_train)
    pred=xe.predict(x_test)
    acc=accuracy_score(y_test,pred)
    print('accuracy of the model', acc,'random_state', i)

    if acc>maxAc:
        maxAc=acc
        maxrs=i
        print ('accuracy of the optimum model', acc,'random_state', i)
```

```
[ ] print ('Optimum accuracy is obtaned in', maxAc, 'in random state', maxrs)
```

```
# Finding the random state
maxAc=0
maxrs=0

for i in range(1,800):
    x_train,x_test,y_train, y_test=train_test_split(prin_comp,y_train_ns,test_size=.20, random_state=i)
    ad=AdaBoostClassifier()
    ad.fit(x_train, y_train)
    pred=ad.predict(x_test)
    acc=accuracy_score(y_test,pred)
    print('accuracy of the model', acc,'random_state', i)

    if acc>maxAc:
        maxAc=acc
        maxrs=i
        print ('accuracy of the optimum model', acc,'random_state', i)
```

```
[ ] print ('Optimum accuracy is obtaned in', maxAc, 'in random state', maxrs)
```

```
[ ] # Finding the random state
maxAc=0
maxrs=0

for i in range(1,800):
    x_train,x_test,y_train, y_test=train_test_split(prin_comp,y_train_ns,test_size=.20, random_state=i)
    gc=GradientBoostingClassifier()
    gc.fit(x_train, y_train)
    pred=gc.predict(x_test)
    acc=accuracy_score(y_test,pred)
    print('accuracy of the model', acc,'random_state', i)

    if acc>maxAc:
        maxAc=acc
        maxrs=i
        print ('accuracy of the optimum model', acc,'random_state', i)
```

```
[ ] print ('Optimum accuracy is obtaned in', maxAc, 'in random state', maxrs)
```

```
[ ] # Finding the random state
maxAc=0
maxrs=0

for i in range(1,800):
    x_train,x_test,y_train, y_test=train_test_split(prin_comp,y_train_ns,test_size=.20, random_state=i)
    sc=SVC()
    sc.fit(x_train, y_train)
    pred=sc.predict(x_test)
    acc=accuracy_score(y_test,pred)
    print('accuracy of the model', acc,'random_state', i)

    if acc>maxAc:
        maxAc=acc
        maxrs=i
        print ('accuracy of the optimum model', acc,'random_state', i)
```

```
[ ] print ('Optimum accuracy is obtaned in', maxAc, 'in random state', maxrs)
```

```
[ ] # Finding the random state
    maxAc=0
    maxrs=0

    for i in range(1,800):
        x_train,x_test,y_train, y_test=train_test_split(prin_comp,y_train_ns,test_size=.20, random_state=i)
        c=RandomForestClassifier()
        c.fit(x_train, y_train)
        pred=c.predict(x_test)
        acc=accuracy_score(y_test,pred)
        print('accuracy of the model', acc,'random_state', i)

        if acc>maxAc:
            maxAc=acc
            maxrs=i
            print ('accuracy of the optimum model', acc,'random_state', i)
```

```
[ ] print ('Optimum accuracy is obtaned in', maxAc, 'in random state', maxrs)
```

```
[ ] #Ploting ROC and AUC curves
    from sklearn.metrics import roc_curve,auc
    fpr1,tpr1,thresholds=roc_curve(y_test,pred1)
    roc_auc1=auc(fpr1,tpr1)
    fpr2,tpr2,thresholds=roc_curve(y_test,pred2)
    roc_auc2=auc(fpr2,tpr2)
    fpr3,tpr3,thresholds=roc_curve(y_test,pred3)
    roc_auc3=auc(fpr3,tpr3)
    fpr4,tpr4,thresholds=roc_curve(y_test,pred4)
    roc_auc4=auc(fpr4,tpr4)
    fpr5,tpr5,thresholds=roc_curve(y_test,pred5)
    roc_auc5=auc(fpr5,tpr5)
    fpr6,tpr6,thresholds=roc_curve(y_test,pred6)
    roc_auc6=auc(fpr6,tpr6)
    #fpr7,tpr7,thresholds=roc_curve(y_test,pred7)
    #roc_auc7=auc(fpr7,tpr7)
    fpr8,tpr8,thresholds=roc_curve(y_test,pred8)
    roc_auc8=auc(fpr8,tpr8)
```

```
[ ] print(roc_auc1,
          roc_auc2,
          roc_auc3,
          roc_auc4,
          roc_auc5,
          roc_auc6,
          #roc_auc7,
          roc_auc8)
```

```
0.4996523225843945 0.49835020473281155 0.4995474467065849 0.5016725150259062 0.5019237684010774 0.49664047088409913 0.8761554266172742
```

```
[ ] # The Random forest has the highest efficiency of accuracy of the model 0.8765207560154349 random_state 13 with 0.876540219903
```

The Random forest has the highest efficiency of ROC_AUC of the model 0.8765207560154349 random_state 13 with 0.876540219903.

```
[ ] x_train,x_test,y_train, y_test=train_test_split(prin_comp,y_train_ns,test_size=.20, random_state=13)
    u=RandomForestClassifier(max_depth=27, n_estimators=40, n_jobs=2)
    u.fit(x_train, y_train)
    pred8=u.predict(x_test)
    acc=accuracy_score(y_test,pred8)
    print('accuracy of the model', acc,'random_state', 13)
    print(classification_report(y_test,pred8))
```

```
accuracy of the model 0.8744835726932914 random_state 13
              precision    recall  f1-score   support

           0       0.87      0.88      0.87     19857
           1       0.88      0.87      0.88     20807

    accuracy                           0.87     40664
   macro avg       0.87      0.87      0.87     40664
weighted avg       0.87      0.87      0.87     40664
```

- Key Metrics for success in solving problem under consideration

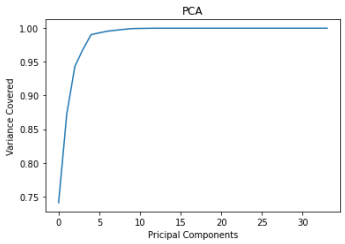
  What were the key metrics used along with justification for using it? You may also include statistical metrics used if any.
  Answer: Principle component analysis was used for dimensionality reduction.

```
[ ] # Importing PCA components
    from sklearn.decomposition import PCA
```

```
pca=PCA()
pca.fit_transform(X_train_ns)
```

```
array([[ 6.19842331e+04,  2.17381168e+03, -3.97393982e+03, ...,
         1.12343355e-02, -2.25147402e-12, -1.39373556e-12],
       [-8.69456479e+03,  1.18621852e+03, -2.43782237e+03, ...,
        -7.08139397e-05, -5.31061760e-13, -7.92857818e-13],
       [-9.32318140e+03,  3.78151972e+02, -1.81880330e+03, ...,
        -5.89310754e-04,  2.05642402e-14,  7.82918987e-14],
       ...,
       [ 2.31403911e+04, -1.28999688e+04,  4.67810194e+03, ...,
        -6.65042682e-04,  2.44406127e-14,  1.07013483e-14],
       [-9.25961779e+03,  5.64612177e+02, -1.44541840e+03, ...,
        -6.58891729e-04,  4.89474495e-15, -3.07669916e-15],
       [-9.09343829e+03,  1.07919080e+03, -4.17454408e+02, ...,
        -6.85821646e-04,  8.62987873e-15, -5.97608283e-16]])
```

```
# Ploting Scree plot to check the best components
plt.figure()
plt.plot(np.cumsum(pca.explained_variance_ratio_))
plt.xlabel('Pricipal Components')
plt.ylabel('Variance Covered')
plt.title('PCA')
plt.show()
```




```
[ ] pca=PCA(n_components=5)
    new_pcomp=pca.fit_transform(X_train_ns)
    prin_comp=pd.DataFrame(new_pcomp, columns=['PC1','PC2','PC3','PC4','PC5'])
```

```
prin_comp
```

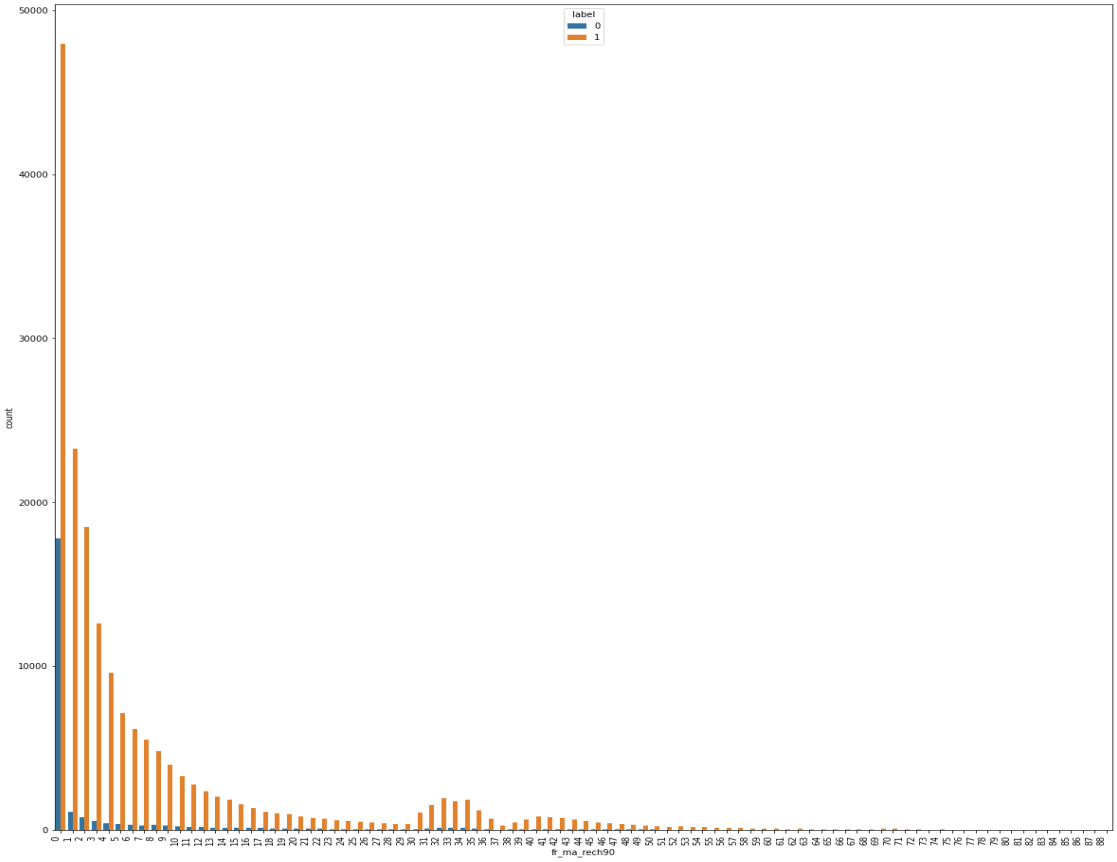
	PC1	PC2	PC3	PC4	PC5
0	61984.233082	2173.811679	-3973.939817	7899.116982	2229.622302
1	-8694.564789	1186.218525	-2437.822372	900.815558	-686.005186
2	-9323.181403	378.151972	-1818.803299	1121.509730	-555.763505
3	17858.044966	3429.590972	-6201.228284	-1104.821886	585.047670
4	1144.261629	565.727464	-2173.541013	347.533757	1422.768619
...	...	...	...	...	...
203315	-2998.850883	-2202.688122	-312.555343	-771.975967	-461.586642
203316	-8072.495371	-623.846429	-1649.180581	-52.062852	158.775069
203317	23140.391147	-12899.968829	4678.101944	-528.788361	-6132.671830
203318	-9259.617787	564.612177	-1445.418400	1110.008429	-544.666664

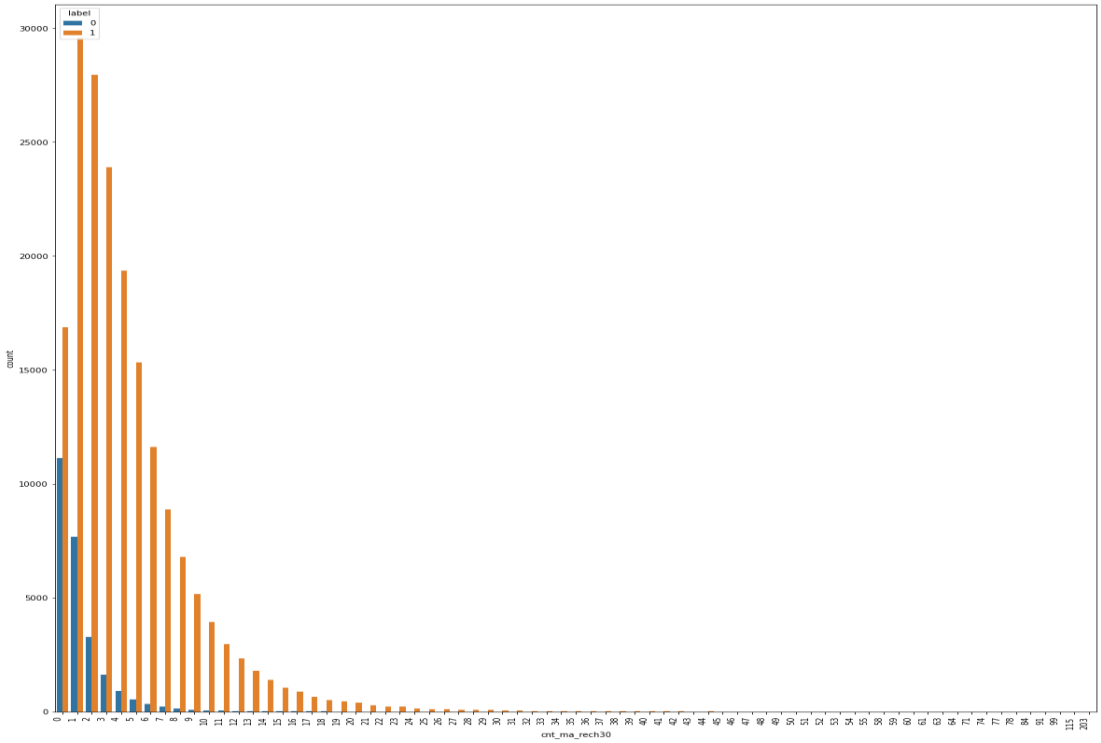
- Visualizations

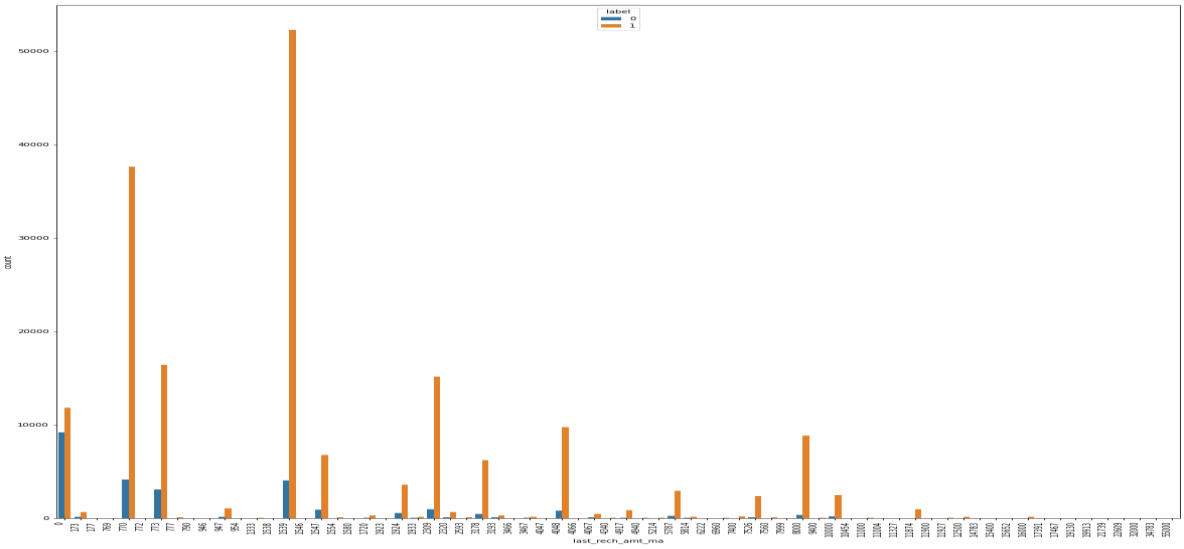
  Mention all the plots made along with their pictures and what were the inferences and observations obtained from those. Describe them in detail.

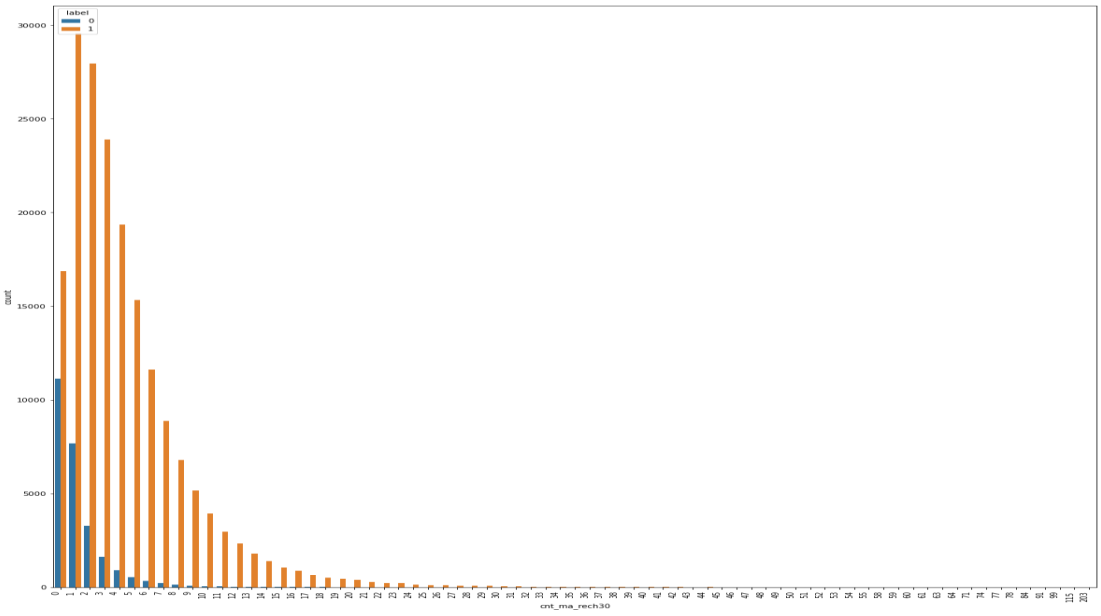
  If different platforms were used, mention that as well.

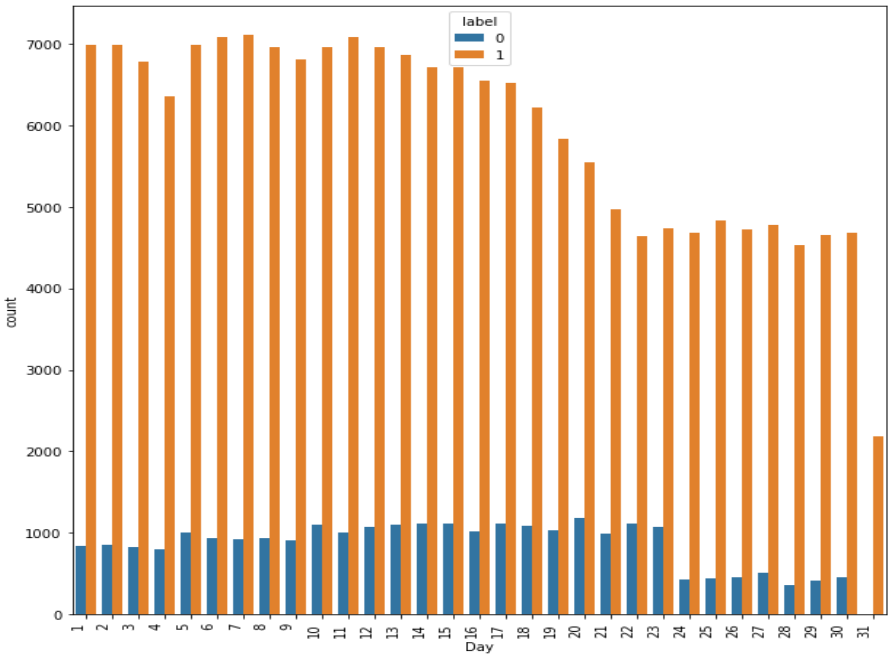
  Answer: Only python was used for visualization:

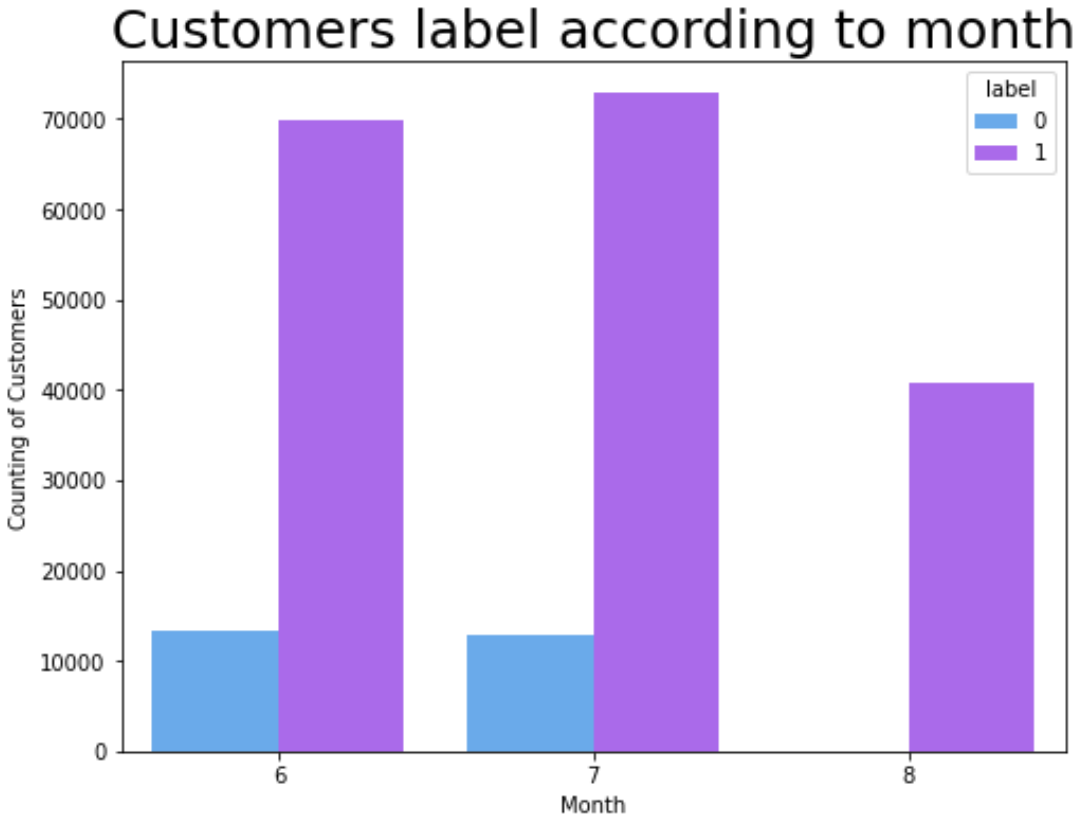

label
0
1
count
fr_ma_rech90

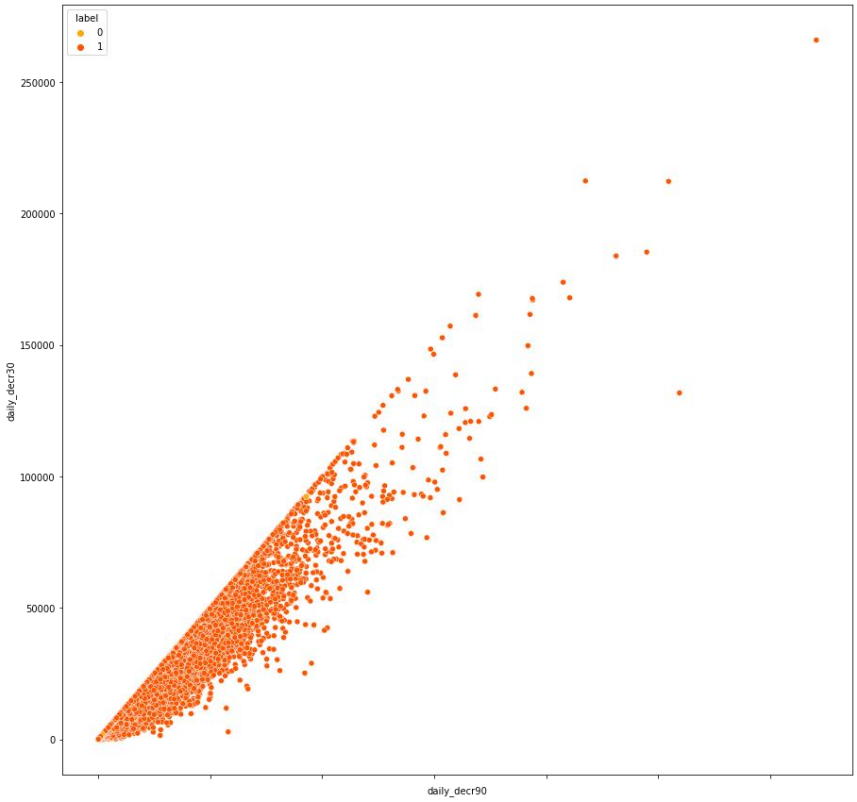

label
0
1
count
cnt_ma_rech30

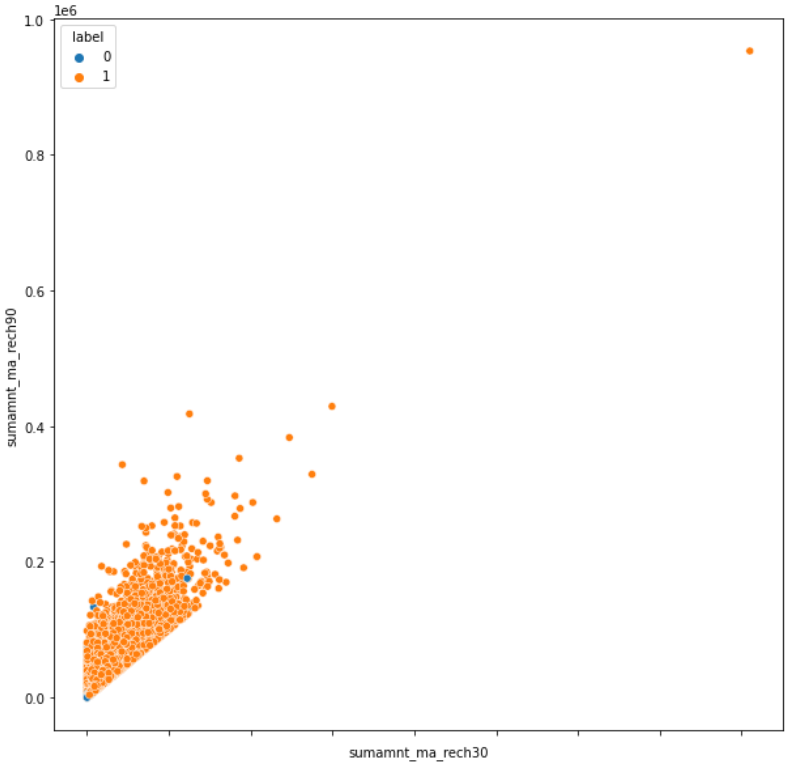

label
0
1
count
last_rech_amt_ma

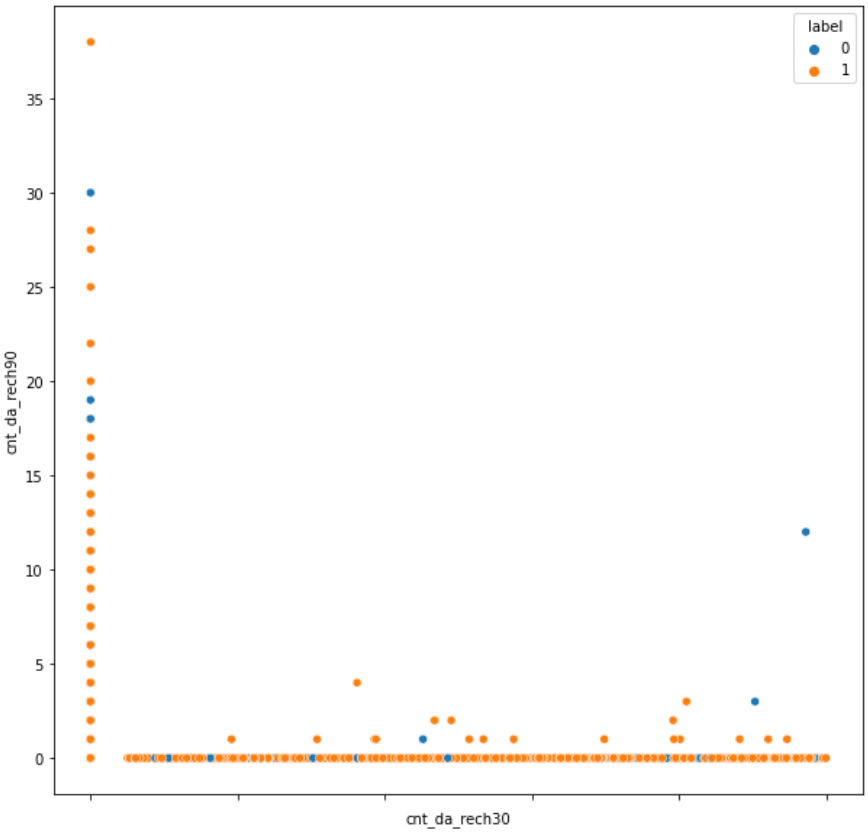

label
0
1
count
cnt_ma_rech30

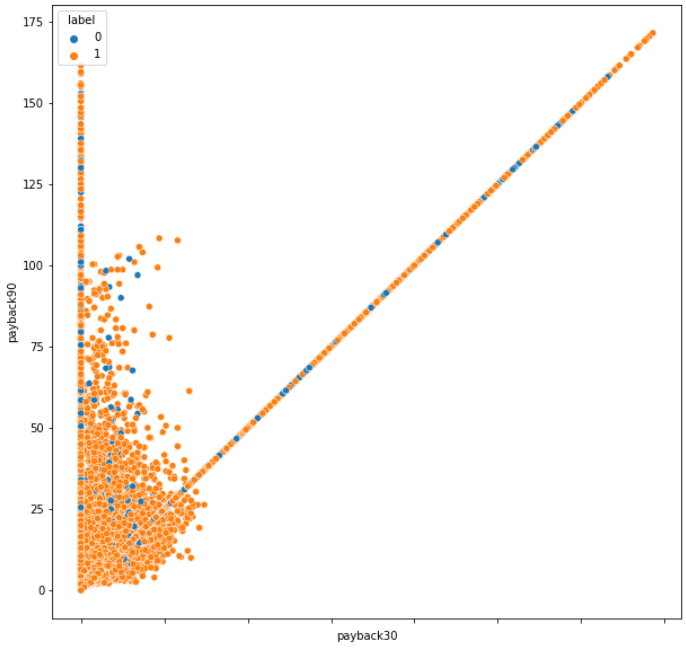

label
0
1
count
Day

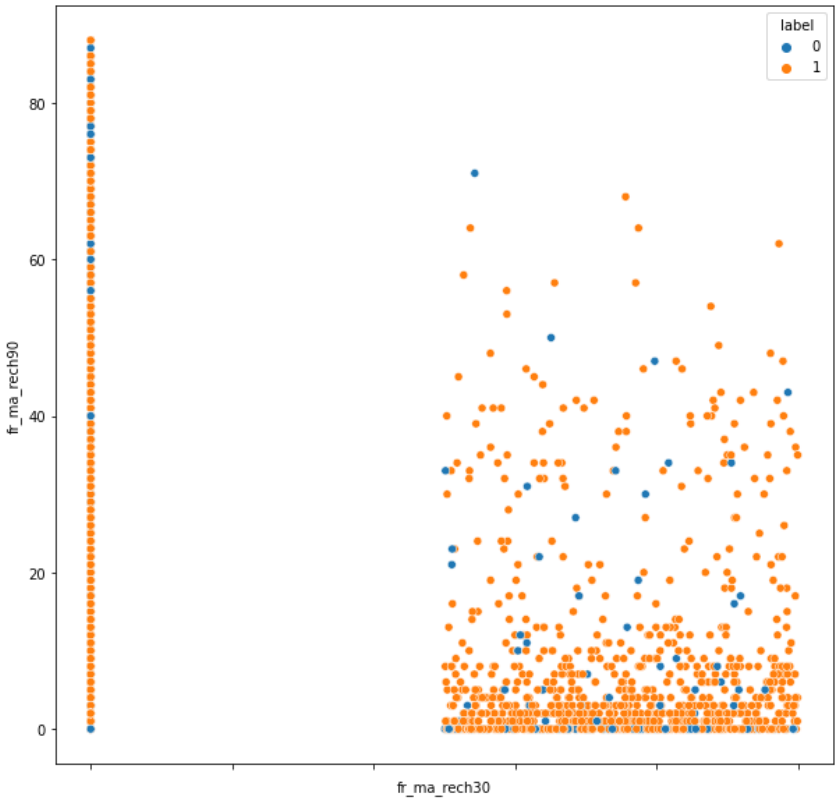

Customers label according to month
label
0
1
Counting of Customers
Month

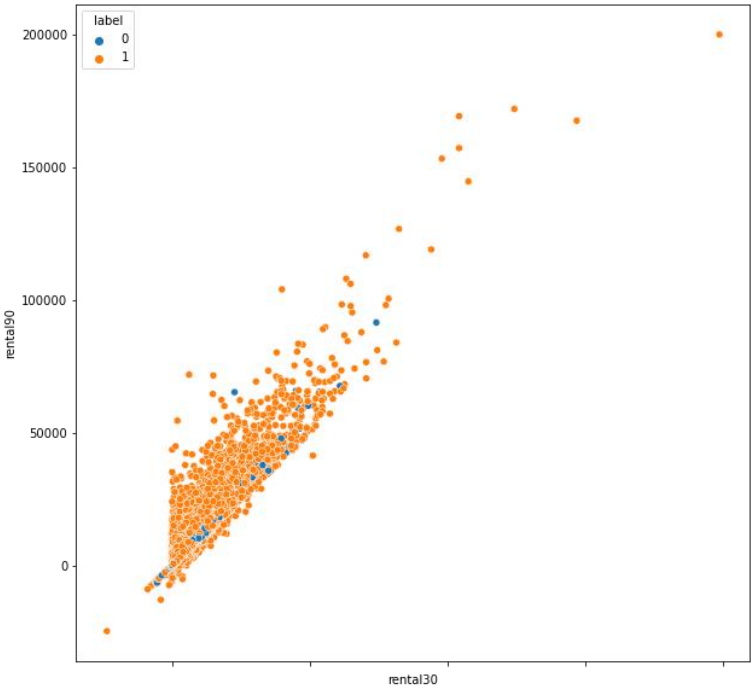

label
0
1
250000
200000
150000
100000
50000
0
daily_decr30
daily_decr90

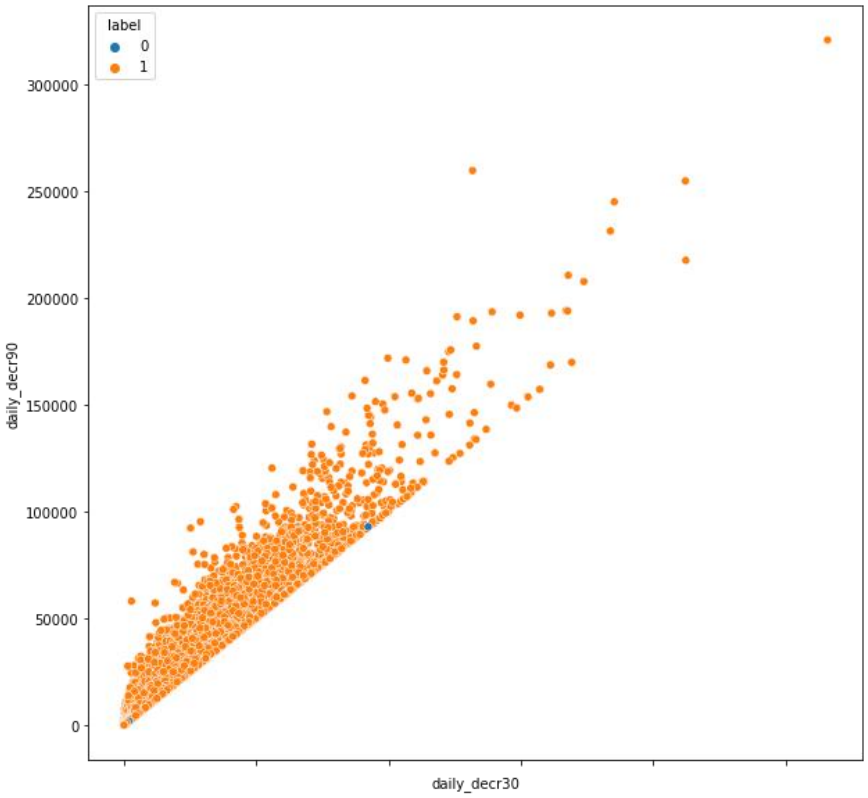

1e6
label
0
1
1.0
0.8
0.6
0.4
0.2
0.0
sumamnt_ma_rech90
sumamnt_ma_rech30

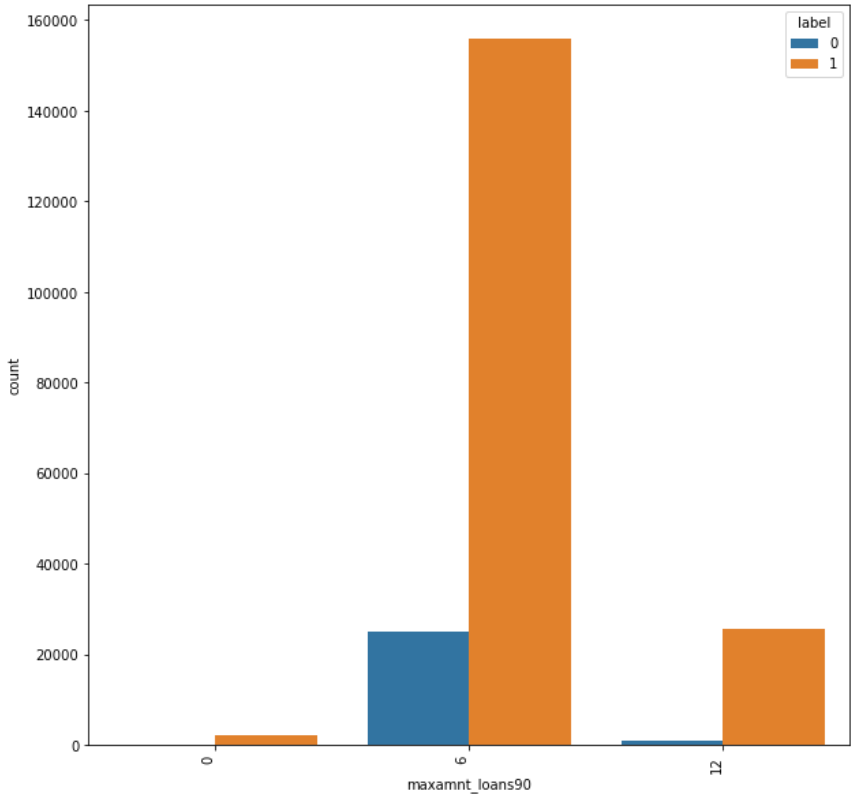

label
0
1
cnt_da_rech90
cnt_da_rech30
0
5
10
15
20
25
30
35

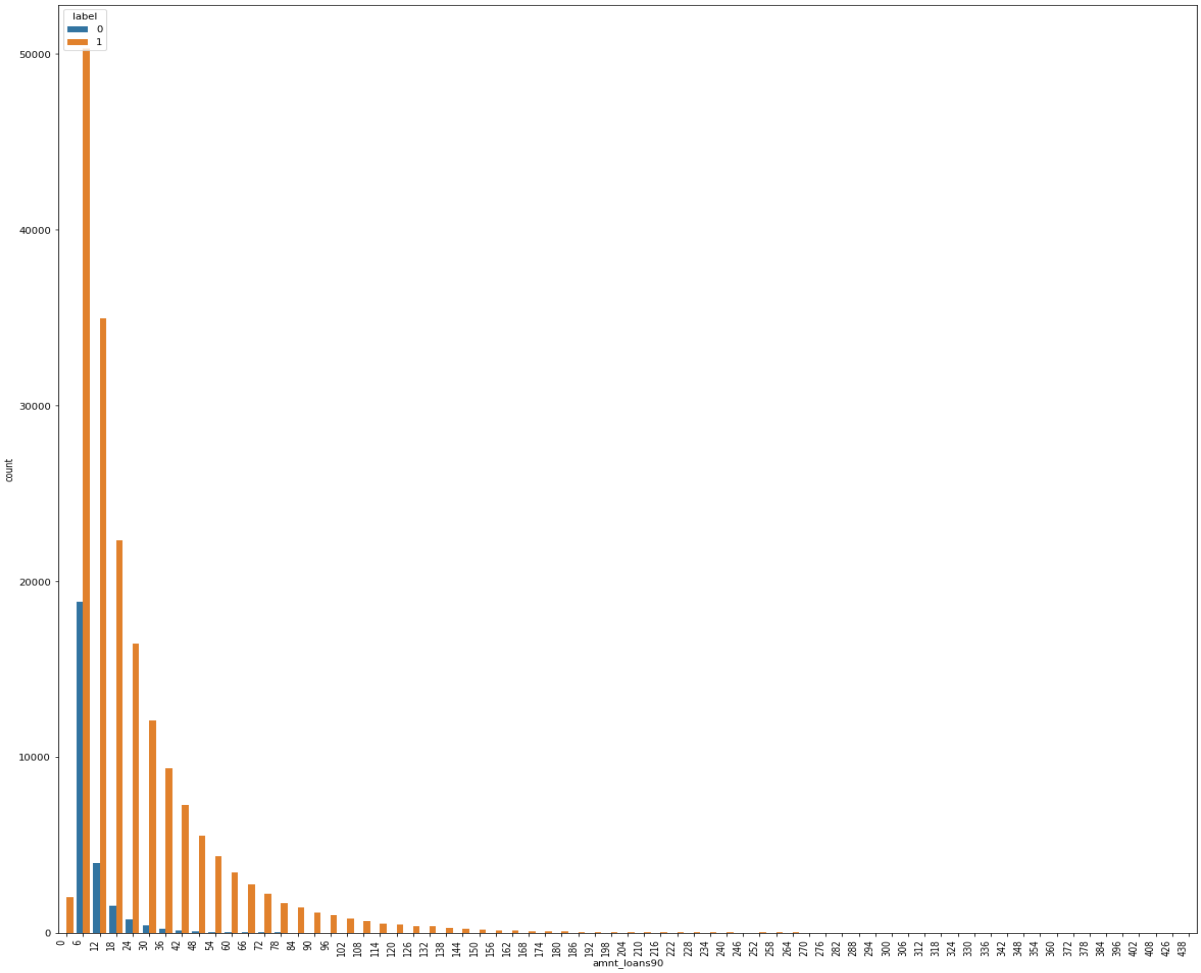

label
0
1
payback90
payback30
0
25
50
75
100
125
150
175

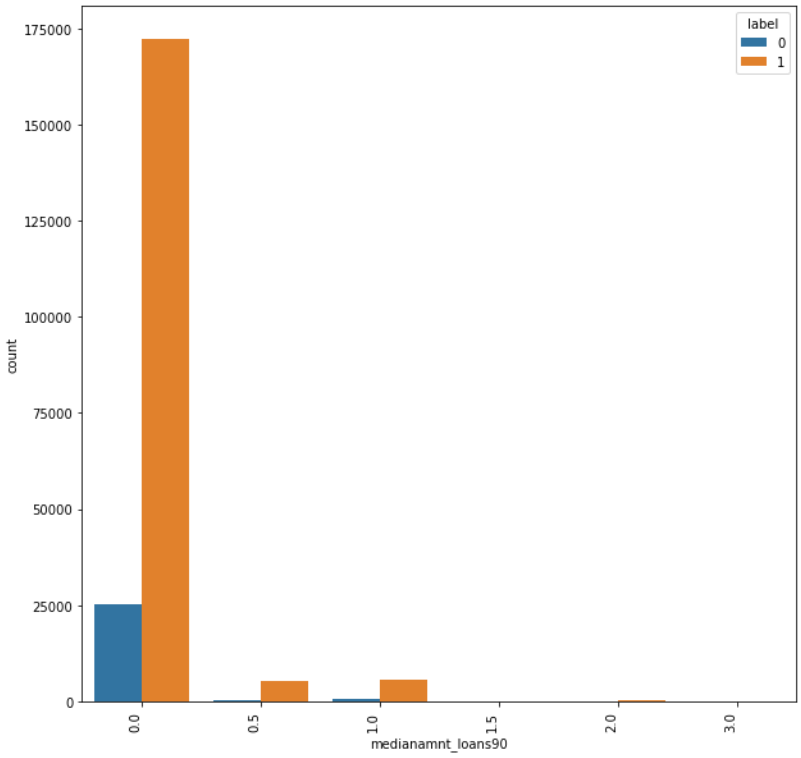

label
0
1
80
60
40
20
0
fr_ma_rech90
fr_ma_rech30

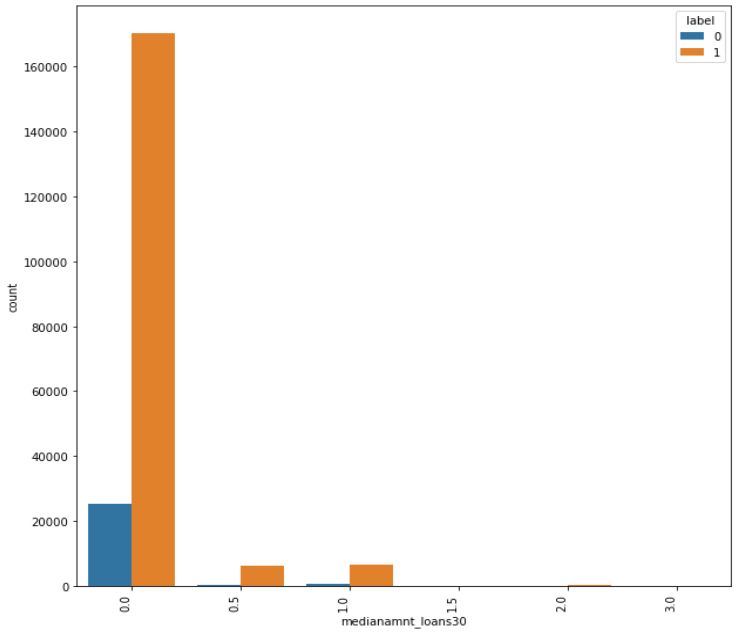

label
0
1
200000
150000
100000
50000
0
rental90
rental30


label
0
1
300000
250000
200000
150000
100000
50000
0
daily_decr90
daily_decr30


label
0
1
160000
140000
120000
100000
80000
60000
40000
20000
0
count
0
6
12
maxamnt_loans90


label
0
1
count
amnt_loans90


label
0
1
count
medianamnt_loans90



- Interpretation of the Results

  Give a summary of what results were interpreted from the visualizations, pre-processing and modelling.

  Answer: In a month the credits taken from 0th -20th days is more and from 20-30 day the credit decreases. The month of july has the highest loans and the month of august has the least loans.

CONCLUSION

- Key Findings and Conclusions of the Study

  Describe the key findings, inferences, observations from the whole problem.

  Answer: In a month the credits taken from 0th -20th days is more and from 20-30 day the credit decreases. The month of july has the highest loans and the month of august has the least loans.

  Learning Outcomes of the Study in respect of Data Science

List down your learnings obtained about the power of visualization, data cleaning and various algorithms used. You can describe which algorithm works best in which situation and what challenges you faced while working on this project and how did you overcome that.

Answer: The column analysis has to be done in details as it involves financial decisions. Every column has to be studied and then dropped if not related.

- Limitations of this work and Scope for Future Work

What are the limitations of this solution provided, the future scope? What all steps/techniques can be followed to further extend this study and improve the results.

Answer: The dataset is imported in the python. Explorative data analysis is performed. The statistical analysis is performed by z-score method. The data loss is tried reduced to be minimal. The SMOTE method is used to reduce the bias in the data. The principle component analysis was also done. All the machine learning models were applied. The SVC model needs more computing time apart from that the random forest is found to have the highest efficiency. The accuracy of the model 0.8748278575644305 random_state 13.The solution provided only has the accuracy of 87 approximately by using neural networks this accuracy can be increased