MICRO CREDIT PREDICTION PROJECT

Submitted by:

JASMINE FERNANDES

ACKNOWLEDGMENT

Following are the acknowledgement sources:

My lecture notes and codes

You tube

INTRODUCTION

Business Problem Framing

A Microfinance Institution (MFI) is an organization that offers financial services to low income populations. currently, many microfinance institutions (MFI), experts and donors are supporting the idea of using mobile financial services (MFS) which offer affordable mobile plans to low income groups.This model provides micro-credit on mobile balances to be paid back in 5 days. The objective is to improve the selection process for the customers applying for the credit through mobile financial services (MFS). The purpose is to reduce credits risks

Conceptual Background of the Domain Problem

It can be difficult to track and predict potential defaulters. The purpose of carrying out machine learning us ti identify defaaulters and theur patterns and to reduc credir risk.

Machine learning can assist lenders in finding similar patterns to predict defaulters based on the past data gathered. The person's' income, previous debt and repayment history can be some of the metrics to consider.

Motivation for the Problem Undertaken

To build a model which can be used to predict the probability for repayment of each loan transaction within 5 days of insurance of loan. In this case, Label '1' indicates that the loan has been payed i.e. Non- defaulter, while, Label '0' indicates that the loan has not been payed i.e. defaulter.

Analytical Problem Framing

Mathematical/ Analytical Modeling of the Problem

Logistic regression modelling is used as we are dealing with a categorical variable. Various techniques to build models are used: Random Forest, Linear regression, Decision Tee, XGB Boost, KNN Beighbors, Gradient Boosting and Adaboost. The main aim of using Logistic Regression model find the best fit such that the error is minimized. Error is the difference between the actual value and Predicted value.

Data Sources and their formats

Following formats are used:

- ✔ Scatter plots
- ✔ Correlation plot
- ✔ Skewness table
- ✔ Used histogram to view distribution of all the numeric variables
- ✔ seaborn as sns
- ✔ Pie chart with percentage
- ✔ Group by plots
- ✔ dist plot
- ✔ Box plot

All data related to 'day and month' is separated to find their correlation with label. All the data is then compared to the target column with bar chart

Corr relation matrix is used to find out multicollinearity and Box plot for fidnong outliers

Data Preprocessing Done

Following steps are used for data cleaning:

Found unique values in object, int64 and float64 data types. Separated these data & found irregularities.

For object data:

```
: df.describe(include=['object','datetime']).transpose()
```

	count	unique	top	freq
msisdn	209593	186243	04581I85330	7
pcircle	209593	1	UPW	209593
pdate	209593	82	2016-07-04	3150

The msisdn:removed this as there were many unique values and they were just mobile numbers
pcircle had just 1 unique data, so dropped thisvalue also
pdate: transformed date into days, month and year using date time function and re-checked for unique values. Later dripped the update values as I already hd days, month and year

```
334]: #for pdate
#Making the new column Day, Month and year from pdate column
df['pDay']=pd.to_datetime(df['pdate'],format='%Y/%m/%d').dt.day
df['pMonth']=pd.to_datetime(df['pdate'],format='%Y/%m/%d').dt.month
df['pYear']=pd.to_datetime(df['pdate'],format='%Y/%m/%d').dt.year
```

For int64 values:

```
: #finding unique values in int64 data types
def explore_object_type(df,feature_name):
    if df[feature_name].dtype == 'int64':
        print(df[feature_name].value_counts())
```

```
: for featureName in df:
    if df[featureName].dtype == 'int64':
        print('\n"' + str(featureName) + '\'s" Values with count are :')
        explore_object_type(df, str(featureName))
```

```
"Unnamed: 0's" Values with count are :
1         1
134331    1
134333    1
134334    1
134336    1
         ..
64547     1
64548     1
64549     1
64550     1
209593    1
Name: Unnamed: 0, Length: 186243, dtype: int64
```

I removed the 'unamed:0' feature column and performed the transpose function. After that i found all unique values for int64

```
#Remove columns where number of unique value is only 1.
unique = df.nunique()
unique = unique[unique.values == 1]
```

```
df.drop(labels = list(unique.index), axis =1, inplace=True)
print("So now we are left with",df.shape ,"rows & columns.")
```

```
So now we are left with (186243, 35) rows & columns.
```

```
#Printing the float and int datatype columns and unique values

#finding unique values in int64 data type
colum_name =[]
unique_value=[]
# Iterate through the columns
for col in df:
    if df[col].dtype == 'int64':
        # If 2 or fewer unique categories
        colum_name.append(str(col))
        unique_value.append(df[col].nunique())
table= pd.DataFrame()
table['Col_name'] = colum_name
table['Value']= unique_value

table=table.sort_values('Value',ascending=False)
table
```

	Col_name	Value
5	sumamnt_ma_rech90	27970
3	cnt_ma_rech90	99
4	fr_ma_rech90	89
1	last_rech_amt_ma	70
10	amnt_loans90	63
2	cnt_ma_rech30	62
7	fr_da_rech90	46
9	amnt_loans30	44
8	cnt_loans30	36
12	pDay	31
6	cnt_da_rech90	27
11	maxamnt_loans90	3
13	pMonth	3

For float64:

We separated the categorical and numeral data, I took their counts

```
#Seprate the categorical columns and Numerical columns
cat_df,num_df=[],[]

for i in df.columns:
    if df[i].dtype==object:
        cat_df.append(i)
    elif (df[i].dtypes=='int64') | (df[i].dtypes=='float64') | (df[i].dtypes=='int32'):
        num_df.append(i)
    else: continue

print('>>> Total Number of Feature::', df.shape[1])
print('>>> Number of categorical features::', len(cat_df))
print('>>> Number of Numerical Feature::', len(num_df))
```

```
>>> Total Number of Feature:: 35
>>> Number of categorical features:: 0
>>> Number of Numerical Feature:: 35
```

I ranked the float64 and int64 as per their unique values

```
0]: #finding unique values in float64 data type
colum_name =[]
unique_value=[]
# Iterate through the columns
for col in df:
    if df[col].dtype == 'float64':
        # If 2 or fewer unique categories
        colum_name.append(str(col))
        unique_value.append(df[col].nunique())
table= pd.DataFrame()
table['Col_name'] = colum_name
table['Value']= unique_value

table=table.sort_values('Value',ascending=False)
table
```

0]:

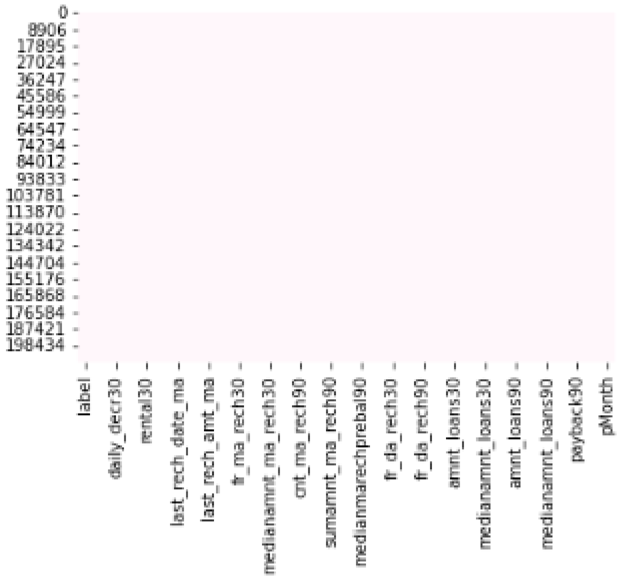
	Col_name	Value
2	daily_decr90	139842
1	daily_decr30	130323
4	rental90	125595
3	rental30	117881
10	medianmarechprebal30	28486
12	medianmarechprebal90	28064
8	sumamnt_ma_rech30	13130
0	aon	4282
20	payback90	2128
19	payback30	1249
6	last_rech_date_da	1061
5	last_rech_date_ma	1061

Following were the observations:
payback30','payback90' has nearly 50% of the values having 0.
Almost 90% of 'last_rech_date_da', 'cnt_da_rech90, 'fr_da_rech90', medianamnt_loans30', and 'medianamnt_loans90' has of values which is 0

Checked for any missing data

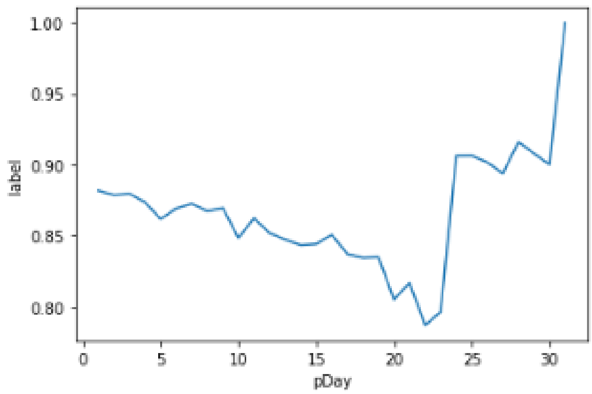
```
#checking for any missing data
# Missing Data Pattern
import seaborn as sns
sns.heatmap(df.isnull(), cbar=False, cmap='PuBu')
```

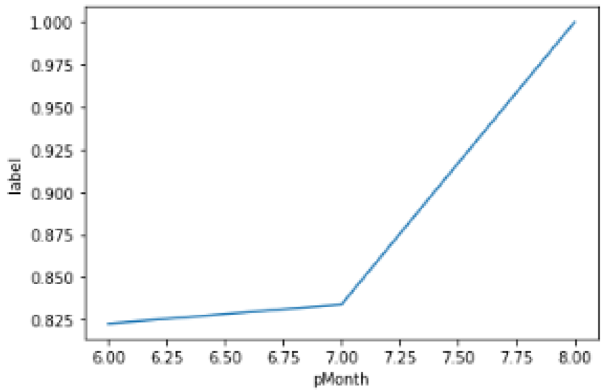
```
<AxesSubplot:>
```



Captured the month and day and compared it with label below:

```
for i in Duration :
    data = df.copy()
    data.groupby(i)['label'].mean().plot()
    plt.xlabel(i)
    plt.ylabel("label")
    plt.show()
```



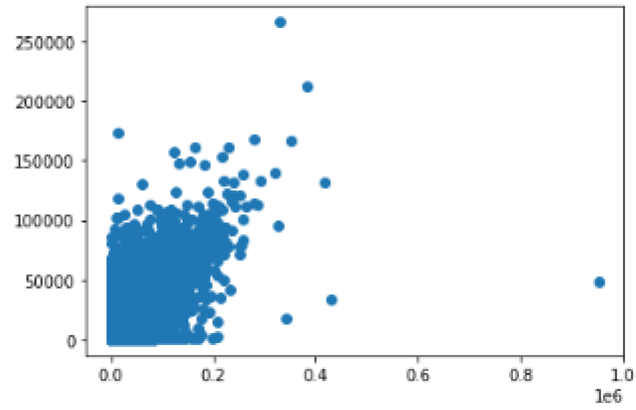


Used the correlation matrix and found the following relations are highly correlated in the matrix:

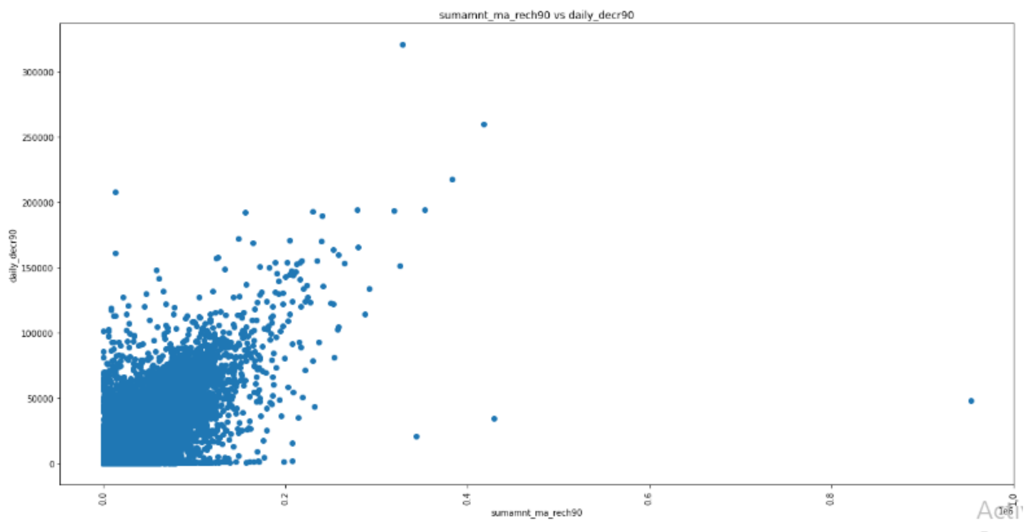
- rental30,rental90
- payback90 vs payback30
- sumamnt_ma_rech90, daily_decr30
- sumamnt_ma_rech30 vs sumamnt_ma_rech90
- sumamnt_ma_rech90 vs daily_decr90
- medianamnt_loans90 vs medianamnt_loans30
- amnt_loans30 vs amnt_loans90
- amnt_loans30 vs cnt_loans90
- cnt_loans30 vs amnt_loans30
- cnt_loans90 vs amnt_loans90
- medianamnt_ma_rech90 vs medianamnt_ma_rech30
- medianamnt_ma_rech90 vs last_rech_amt_ma
- last_rech_amt_ma vs medianmarechprebal30
- cnt_ma_rech90 vs cnt_ma_rech30
- last_rech_amt_ma vs medianamnt_ma_rech30
- last_rech_amt_ma vs medianamnt_ma_rech90

sumamnt_ma_rech90 vs daily_decr30: - Majority have spent below 0.2 rupiah daily over the last 30 days for 100000 Rupiah recharge in main account over last 90 days

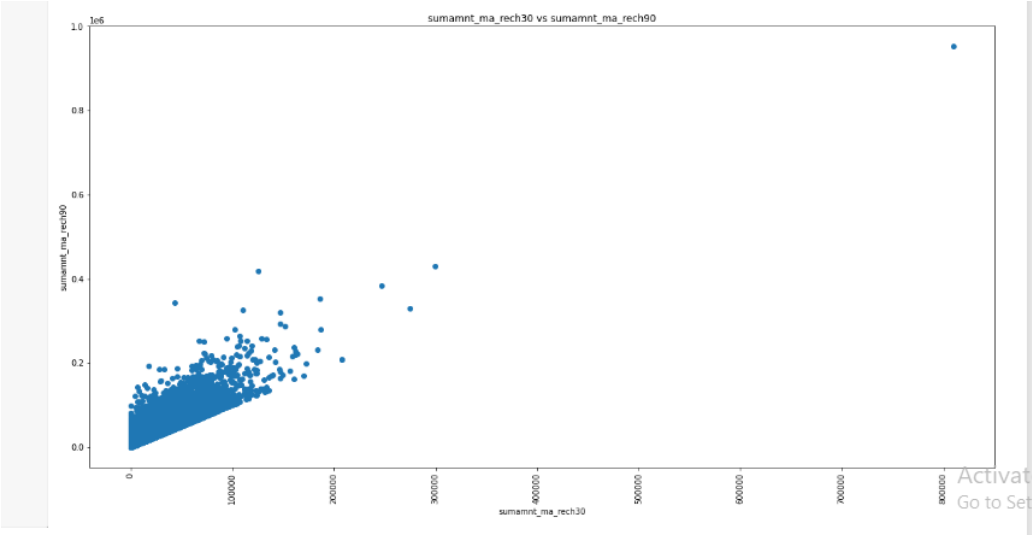
```
#Lets check with scatter plot
plt.scatter(df.sumamnt_ma_rech90, df.daily_decr30)
plt.show()
```



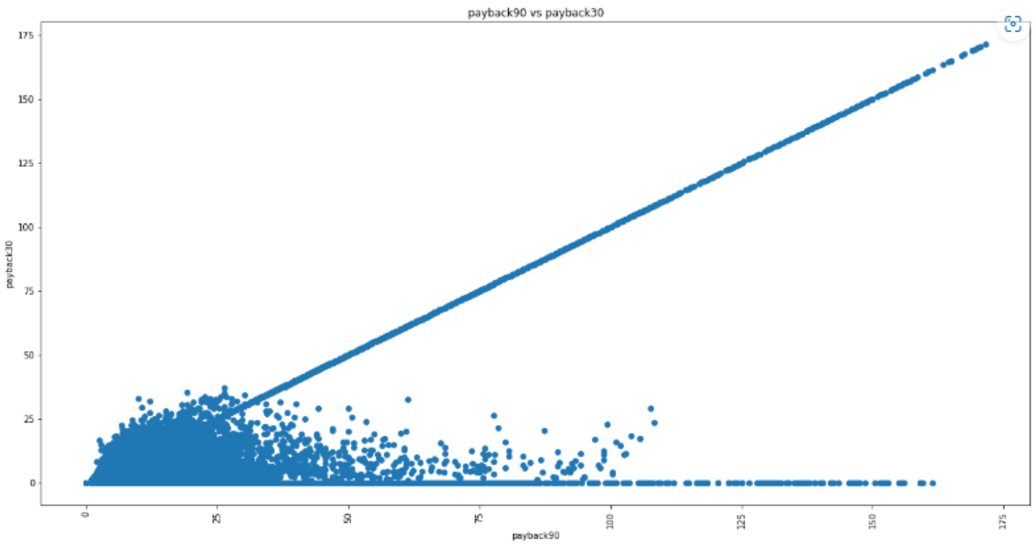
sumamnt_ma_rech30 vs sumamnt_ma_rech90



sumamnt_ma_rech30 vs sumamnt_ma_rech90

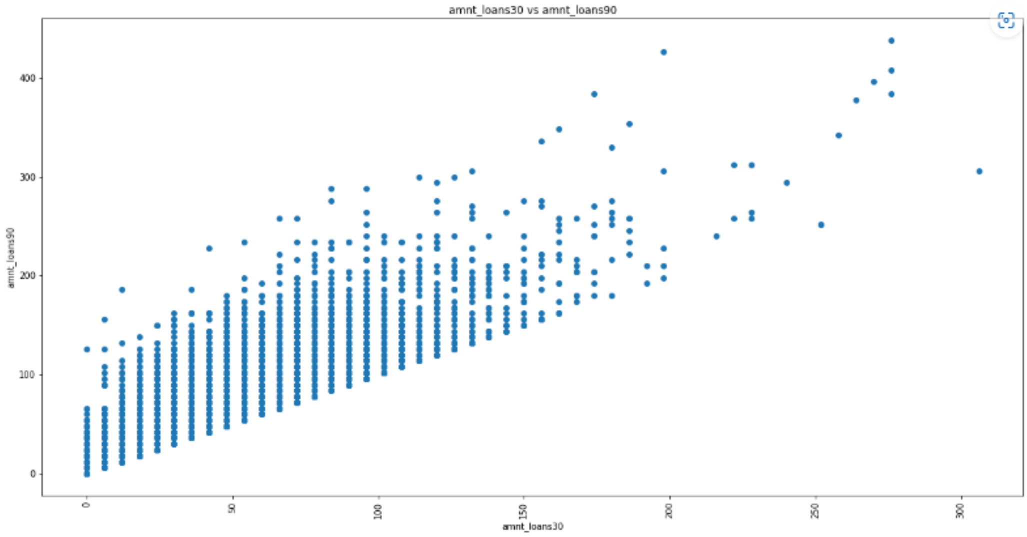


payback90 vs payback30

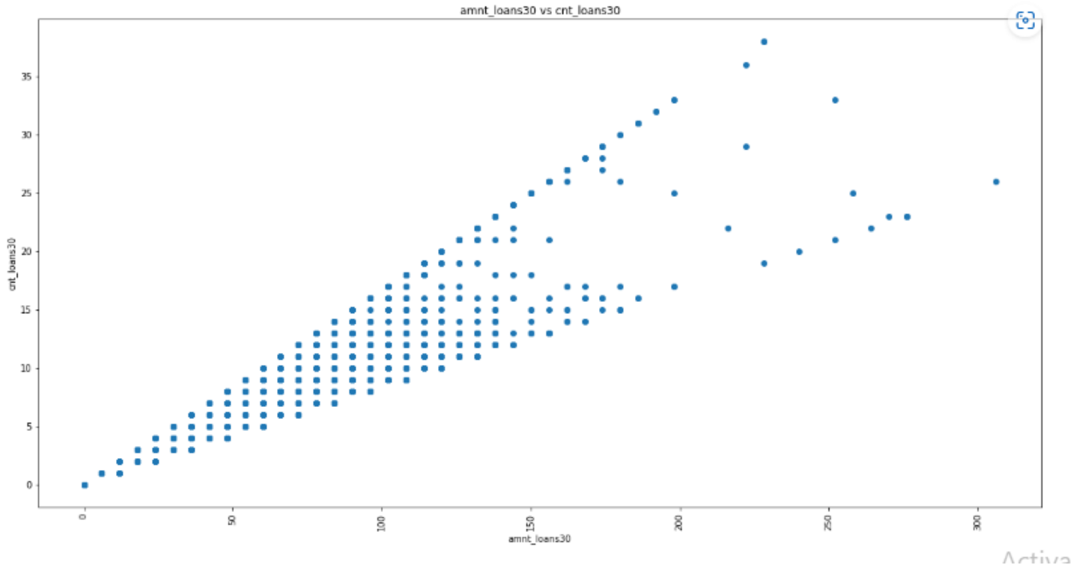


Some

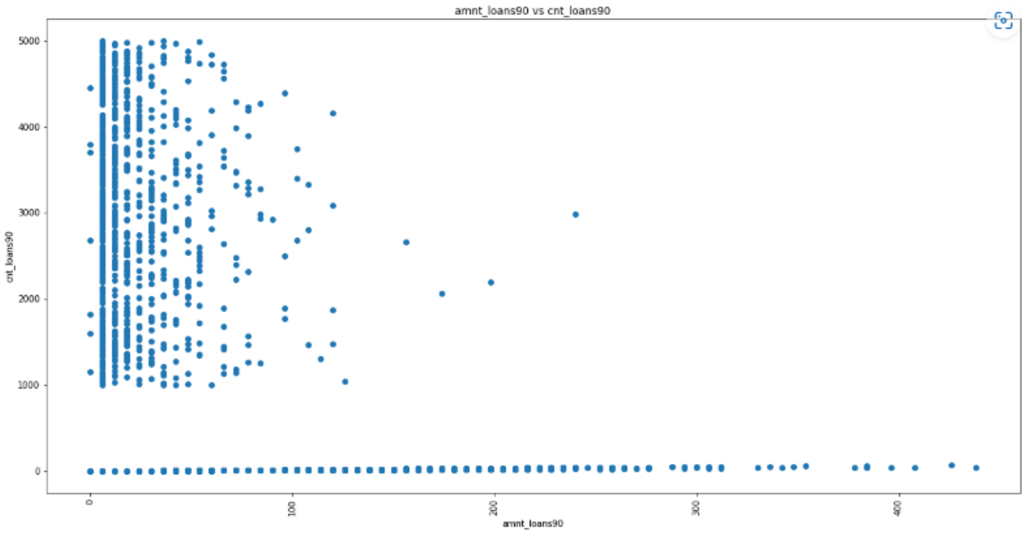
amnt_loans30 vs amnt_loans90



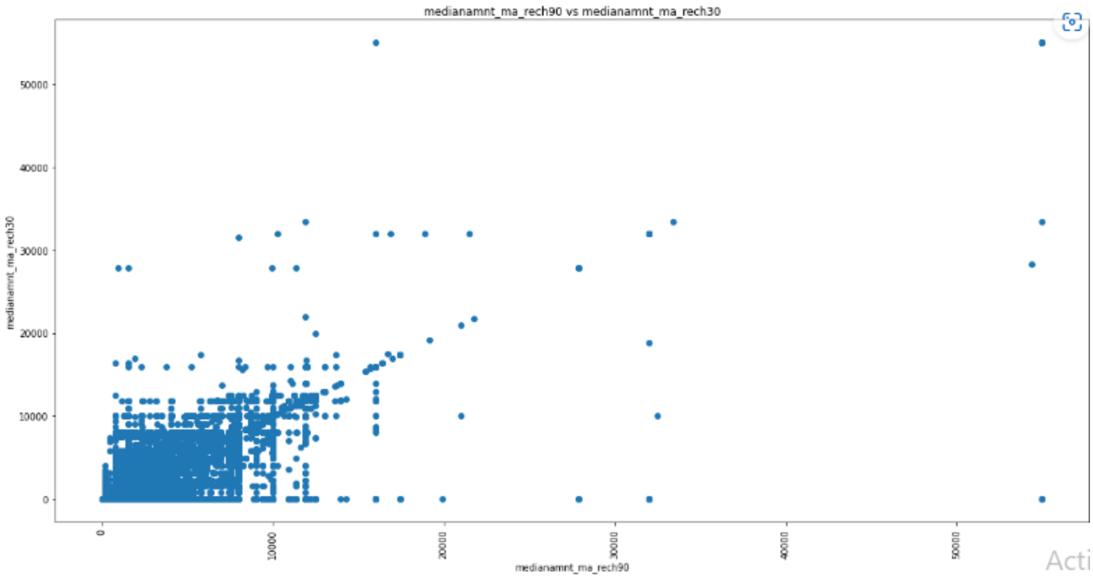
amnt_loans30 vs cnt_loans30



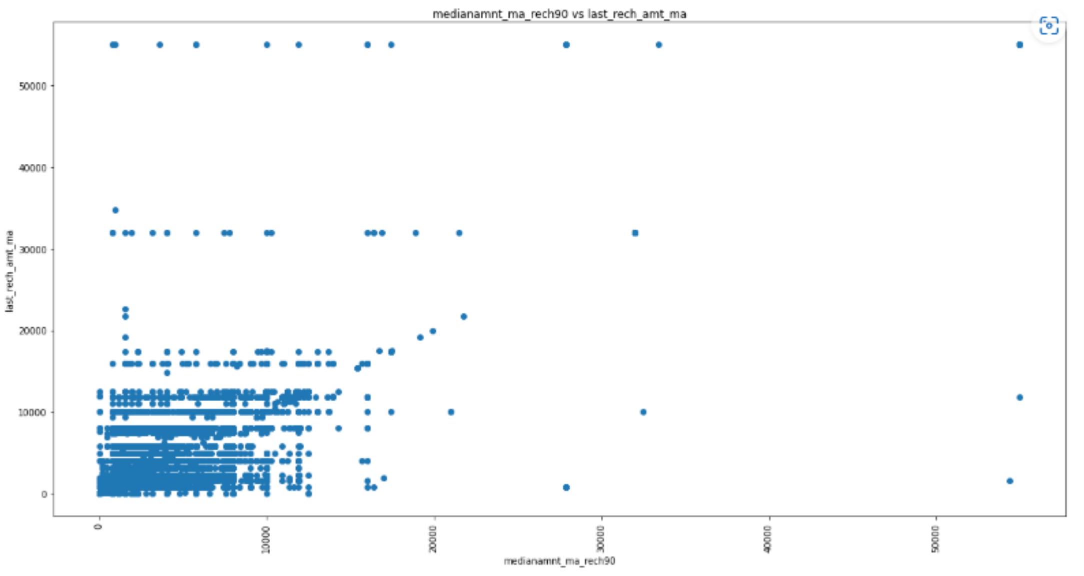
amnt_loans90 vs cnt_loans90



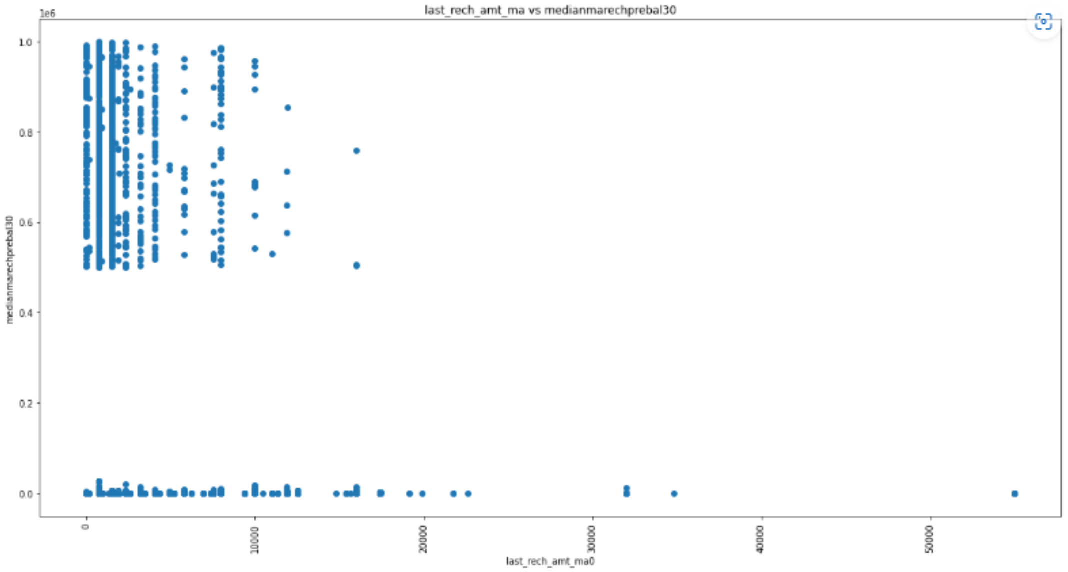
medianamnt_ma_rech90 vs medianamnt_ma_rech30



medianamnt_ma_rech90 vs last_rech_amt_ma



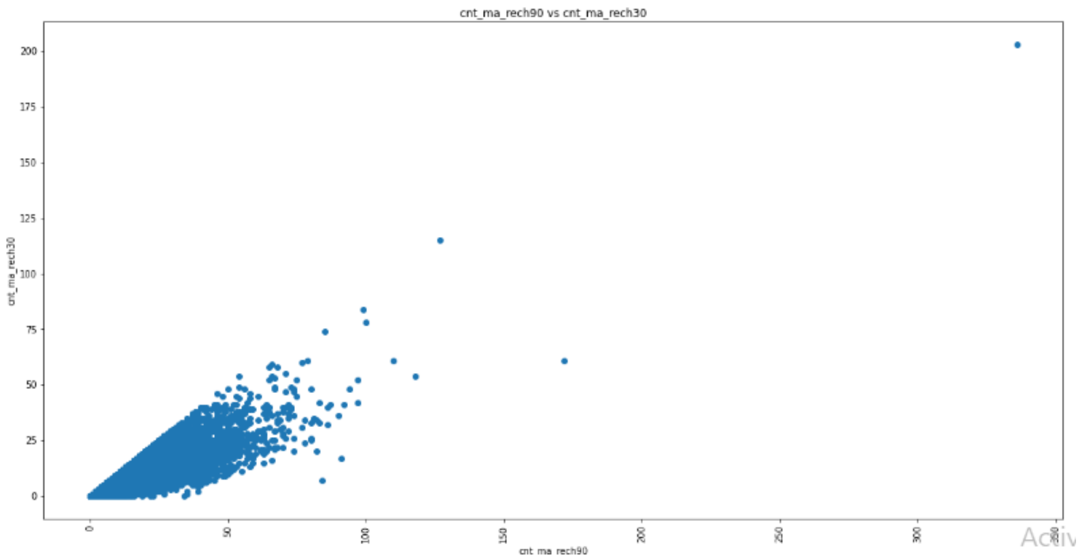
last_rech_amt_ma vs medianmarechprebal30



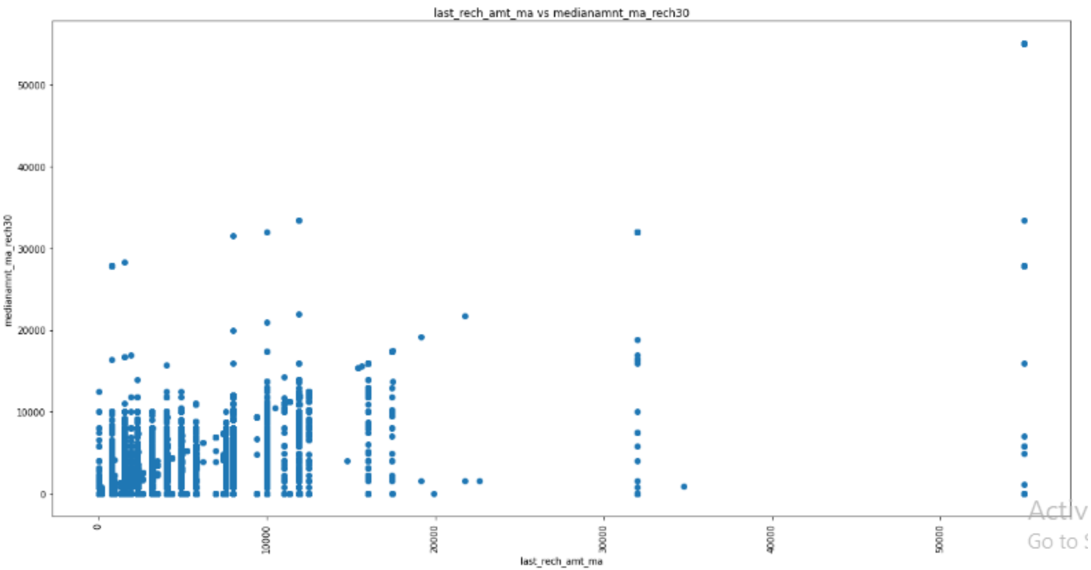
Other relations observed:

cnt_loans30 vs cnt_loans90
last_rech_amt_ma vs medianmarechprebal90
medianmarechprebal30 vs medianmarechprebal90
daily_decr30 vs daily_decr90
sumamnt_ma_rech30 vs daily_decr30
sumamnt_ma_rech30 vs daily_decr90

cnt_ma_rech90 vs cnt_ma_rech30

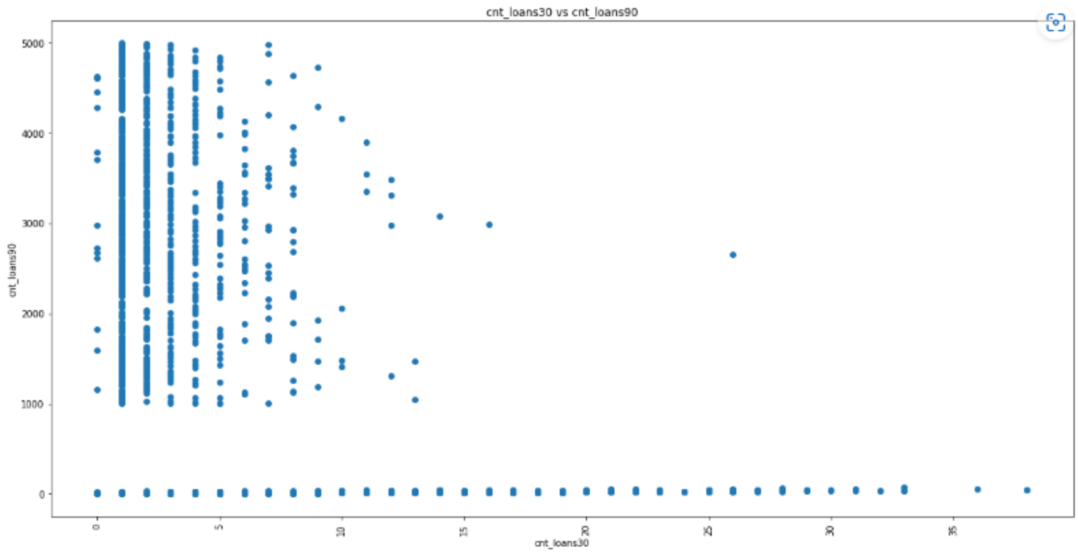


last_rech_amt_ma vs medianamnt_ma_rech30

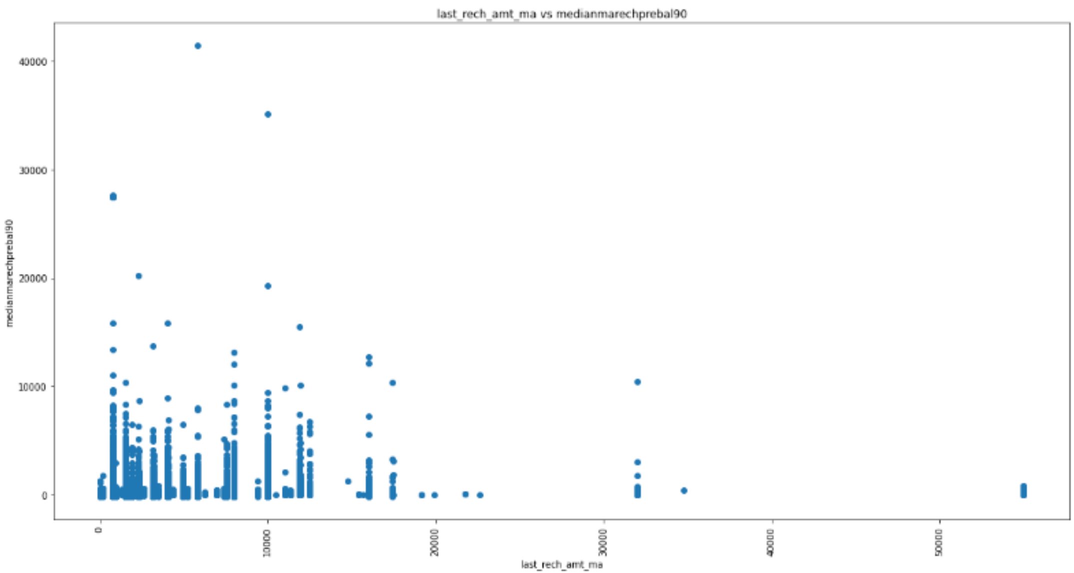


other

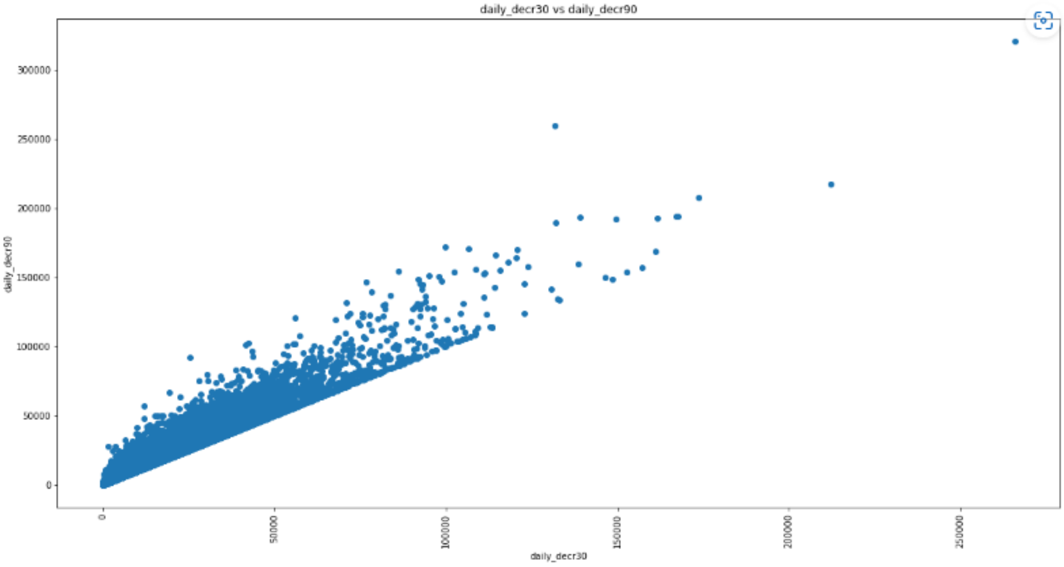
cnt_loans30 vs cnt_loans90



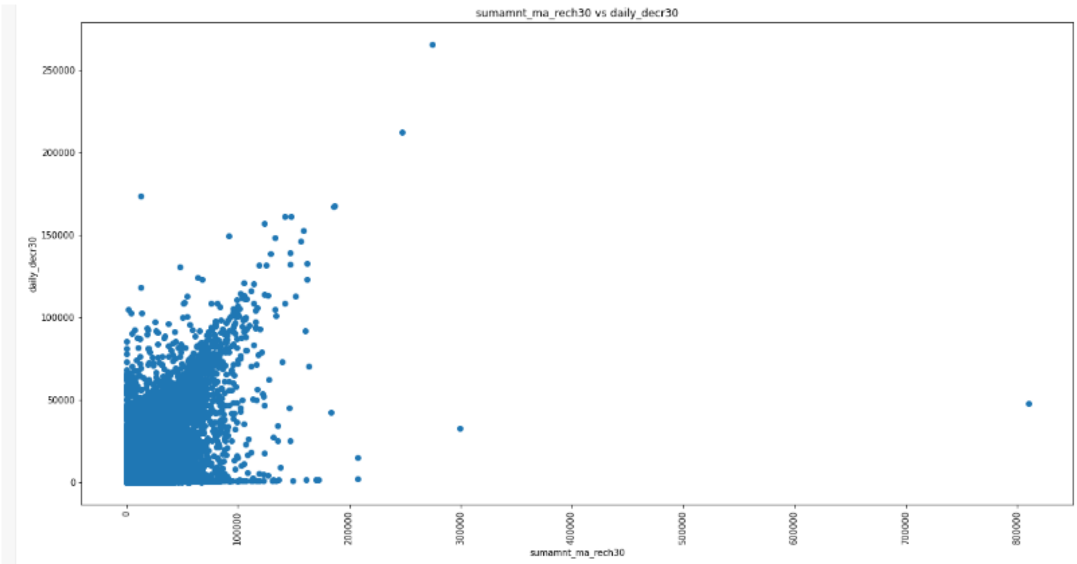
last_rech_amt_ma vs medianmarechprebal90



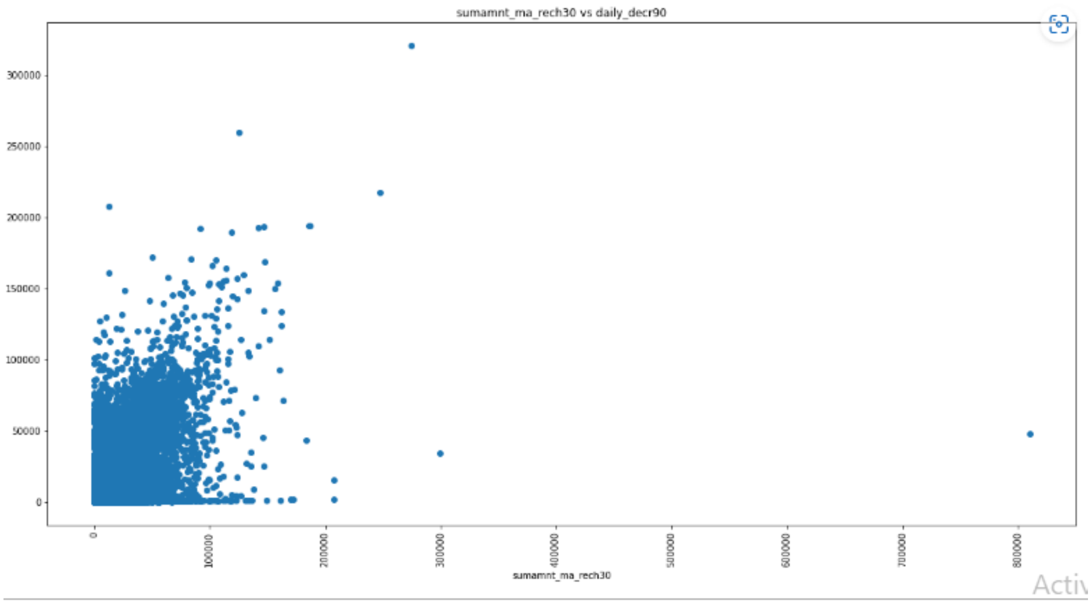
daily_decr30 vs daily_decr90



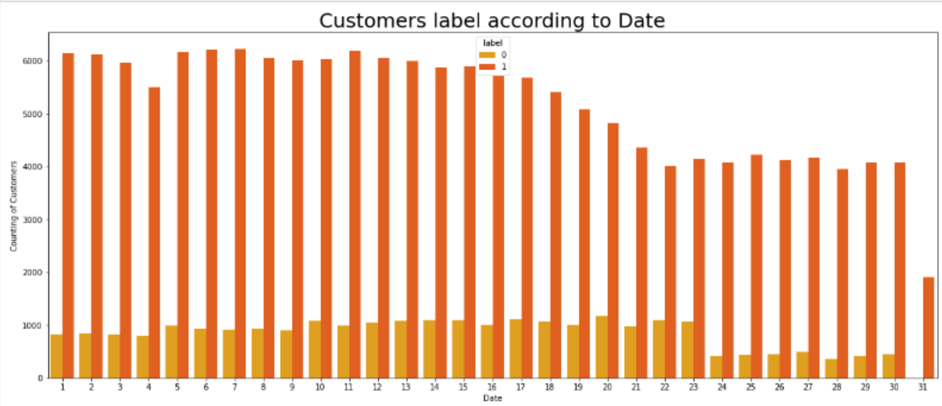
sumamnt_ma_rech30 vs daily_decr30



sumamnt_ma_rech30 vs daily_decr90



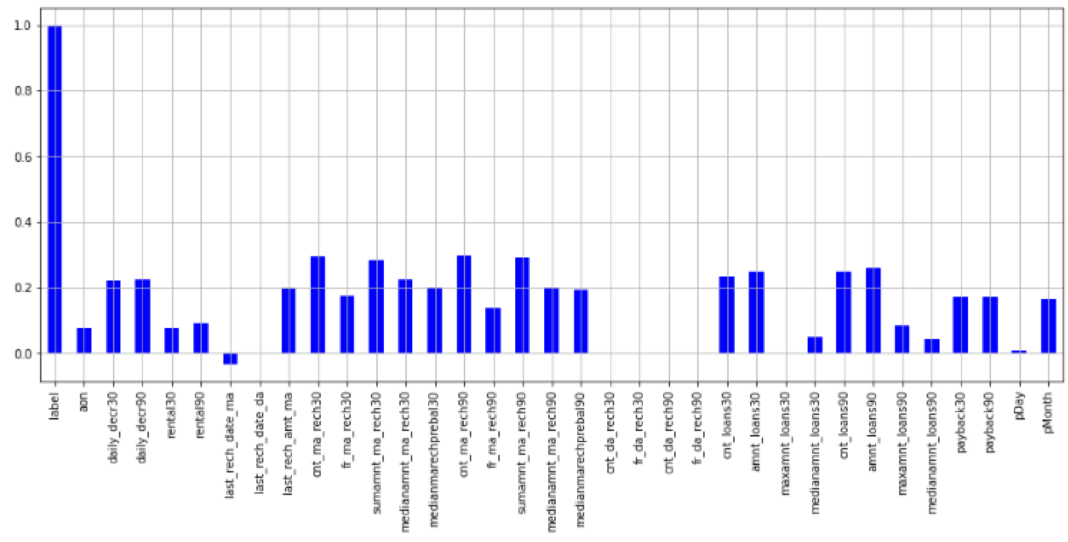

```
In [513]: #Customer Label according to Date
          plt.figure(figsize=(20,8))
          sns.countplot(x="pDay", hue='label', data=df, palette='autumn_r')
          plt.title("Customers label according to Date", fontsize=25)
          plt.xlabel('Date')
          plt.ylabel('Counting of Customers')
          plt.show()
```



Relation between label vs other feature columns:

```
#checking the correlation between attributes and target variable

df.corrwith(df['label']).plot(kind='bar',grid = True, figsize=(16,6), color='blue')
plt.show()
```



There is high correlation between label and other following columns (figure given above):

cnt_ma_rech30 & label

sumamnt_ma_rech30 vs & label

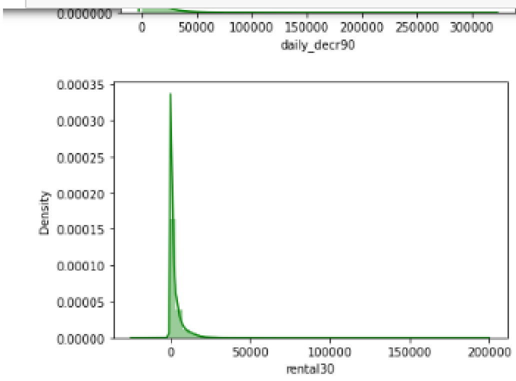
cnt_ma_rech90 & label

sumamnt_ma_rech90 vs & label

Checked the skewness in data with dist plot

```
#checking skewness

for col in df.describe().columns:
    sns.distplot(df[col],color='g')
    plt.show()
```



I found skewness in following columns:

'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', 'cnt_loans90','amnt_loans90', 'payback30','payback90']

I used df.skew() methos to double check the feature columns with outliers

I treated the columns with pecentile technique to remove outliers

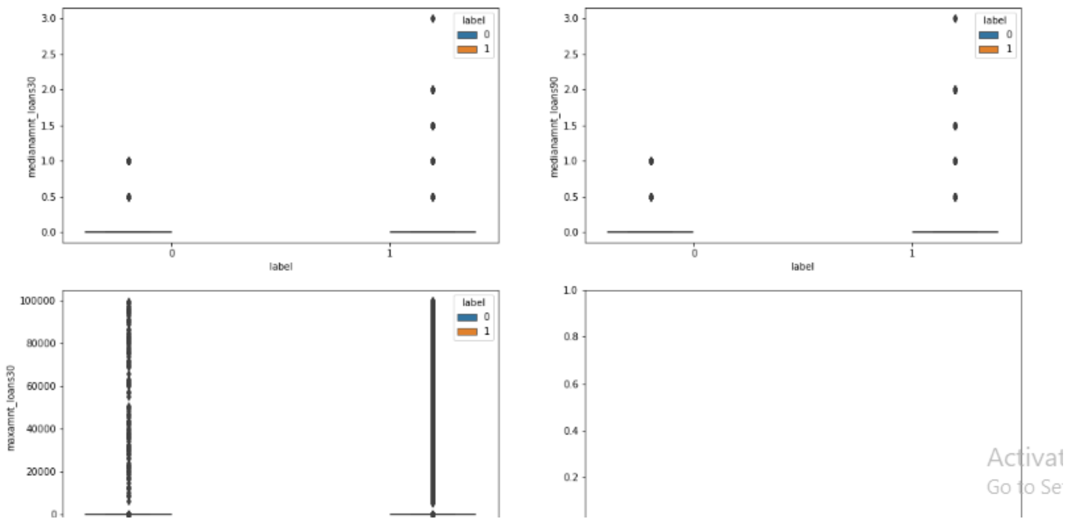
```
In [517]: columns = [ 'aon', 'daily_decr30', 'daily_decr90','rental30', 'rental90', 'last_rech_date_ma', 'last_rech_date_da','last_rech_amt
          for i in columns:
              iqr = df[i].quantile(0.75)-df[i].quantile(0.25)
              high = df[i].quantile(0.75)+(iqr*1.25)
              low  = df[i].quantile(0.25)-(iqr*1.25)
              df.loc[df[i]>high,i]=high
              df.loc[df[i]<low,i]=low
```

There were still outliers remaining in the features given below:

```
: #plotting outliers

  fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(nrows=2, ncols=2, figsize = (18, 10))
  sns.boxplot(ax=ax1, x = 'label', y = 'medianamnt_loans30', hue = 'label', data = df)
  sns.boxplot(ax=ax2, x = 'label', y = 'medianamnt_loans90', hue = 'label', data = df)
  sns.boxplot(ax=ax3, x = 'label', y = 'maxamnt_loans30', hue = 'label', data = df)

: <AxesSubplot:xlabel='label', ylabel='maxamnt_loans30'>
```



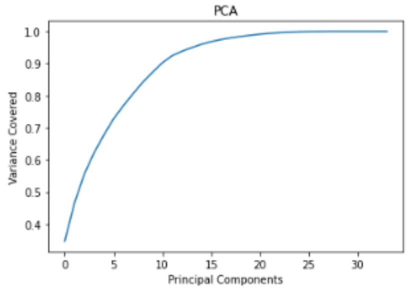
I later treated the data with zscore for remainng outliers, but there was too much loss of date, so I dropped the method

DATA TRANSFORMATION

I used the power transform method and standard scalar to reduce skewness in the data set

I used the PCA method for feature reduction and found only 25 features to be important

```python
#Let's plot Scree plot to check the best components
plt.figure()
plt.plot(np.cumsum(pca.explained_variance_ratio_))
plt.xlabel('Principal Components')
plt.ylabel("Variance Covered")
plt.title('PCA')
plt.show()
```

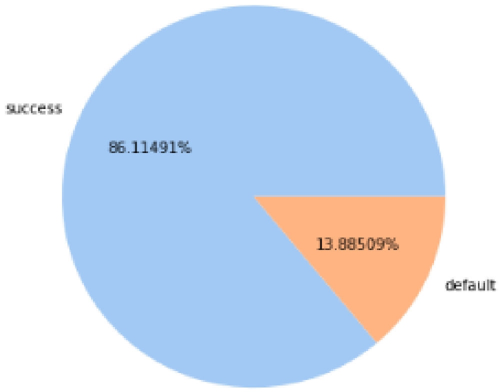



```python
#Around 25 Principal Components are able to explain > 98 % variance. Its safe to consider starting 25 PC's
pca=PCA(n_components=25)
new_pcomp=pca.fit_transform(X_scaler)
Princi_comp = pd.DataFrame(new_pcomp, columns=['PC1', 'PC2', 'PC3', 'PC4', 'PC5', 'PC6', 'PC7', 'PC8', 'PC9', 'PC10', 'PC11', 'PC
Princi_comp #PC 25 are the features
```

There is imbalance in data

```python
labels = ['success','default']
success = (df[df['label']==1].shape[0])/df.shape[0]
default = (df[df['label']==0].shape[0])/df.shape[0]
data= [success, default]
print("% of non defaulters anddefaulters: \n",data)
colors = sns.color_palette('pastel')[0:5]
plt.figure(figsize=(8,6), facecolor='white')
plt.pie(data, labels = labels, colors = colors, autopct='%2.5f%%')
plt.show()
```

```
% of non defaulters anddefaulters:
 [0.8611491438604404, 0.1388508561395596]
```



I used the SMOTE method to reduce the imbalance in feature data column

```
3]: from sklearn.model_selection import train_test_split
    from sklearn import metrics
    from imblearn.combine import SMOTETomek
    from imblearn.over_sampling import SMOTE
    ## RandomOverSampler to handle imbalanced data
    from collections import Counter
    from imblearn.over_sampling import RandomOverSampler
```

```
4]: smote=SMOTE()
    #fit predictor and target variable
    x_smote, y_smote=smote.fit_resample(Princi_comp,y)
```

```
5]: print('original datashape', Counter(y))
    print('original datashape', Counter(y_smote))

    original datashape Counter({1: 160383, 0: 25860})
    original datashape Counter({0: 160383, 1: 160383})
```

```
6]: print('original datashape', Counter(y))

    original datashape Counter({1: 160383, 0: 25860})
```

```
7]: x_smote.shape
7]: (320766, 25)
```

```
8]: y_smote.shape
8]: (320766,)
```

Hardware and Software Requirements and Tools Used:

SMOTE, LogisticRegression(), KNeighborsClassifier() , DecisionTreeClassifier() # Deciesion Tree, RandomForestClassifier() AdaBoostClassifier(), svm=SVC(), GradientBoostingClassifier() , XGBRegressor()

Model/s Development and Evaluation

- **Identification of possible problem-solving approaches (methods)**

  I have used logistic regression as an approach and worked with following algorithms:

  Logistic Regression, KNearest Neibour , Decision Tree, Random Forest, Adaboost Classifier, SVC() , Gradient Boosting Classifier , XGBRegressor(

- **Run and Evaluate selected models with key metrics**

I divided the data into train and split based on the smote scores

I trained the models and found their classification score

```
#trying with smote scores
x_smote, x_test, y_smote, y_test = train_test_split(Princi_comp,y, test_size=0.25, random_state=1)
```

```
lr.fit(x_smote, y_smote)
knn.fit(x_smote,y_smote)
dt.fit(x_smote,y_smote)
rf.fit(x_smote,y_smote)
adb.fit(x_smote,y_smote)
svm.fit(x_smote,y_smote)
gdboost.fit(x_smote,y_smote)
xgb.fit(x_smote,y_smote)
print("Model is trained")
```

```
Model is trained
```

```
print("Lr classification score",lr.score(x_smote, y_smote))
print("knn classification score",knn.score(x_smote, y_smote))
print("dt classification score",dt.score(x_smote, y_smote))
print("rf classification score",rf.score(x_smote, y_smote))
print("adb classification score",adb.score(x_smote, y_smote))
print("svm classification score",svm.score(x_smote, y_smote))
print("gdboost classification score",gdboost.score(x_smote, y_smote))
print("xgb classification score",xgb.score(x_smote, y_smote))
```

```
Lr classification score 1.0
knn classification score 0.999620566715826
dt classification score 1.0
rf classification score 1.0
adb classification score 1.0
svm classification score 1.0
gdboost classification score 1.0
xgb classification score 0.9994870160233361
```

I checked the ROC scores, they were all above 99%

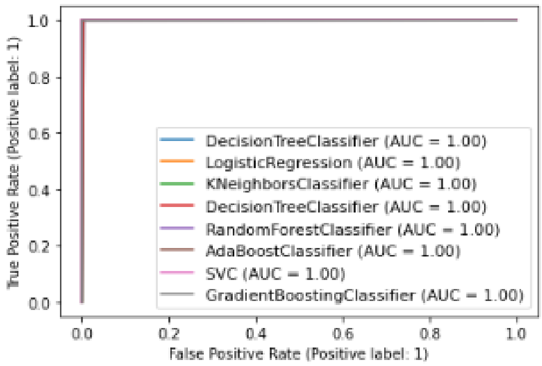
```
from sklearn.metrics import roc_curve, auc, roc_auc_score
from sklearn.metrics import plot_roc_curve
#importing the ric and auc from sklearn and predect the x_test and
#checking the roc_auc_score
print(roc_auc_score(y_test,lr.predict(x_test)))
print(roc_auc_score(y_test,knn.predict(x_test)))
print(roc_auc_score(y_test,dt.predict(x_test)))
print(roc_auc_score(y_test,rf.predict(x_test)))
print(roc_auc_score(y_test,adb.predict(x_test)))
print(roc_auc_score(y_test,svm.predict(x_test)))
print(roc_auc_score(y_test,gdboost.predict(x_test)))
```

```
1.0
0.9975765907332371
0.9977596911991641
0.9997666822211853
1.0
0.9998444548141234
0.9994959877853387
```

The results are determined by the ROC curve

```
#lets find roc curve to check best fittted model
disp = plot_roc_curve(dt,x_test,y_test)
plot_roc_curve(lr,x_test,y_test,ax=disp.ax_)  # here ax_ for axis with confustion matrics
plot_roc_curve(knn,x_test,y_test,ax=disp.ax_)
plot_roc_curve(dt,x_test,y_test,ax=disp.ax_)
plot_roc_curve(rf,x_test,y_test,ax=disp.ax_)
plot_roc_curve(adb,x_test,y_test,ax=disp.ax_)
plot_roc_curve(svm,x_test,y_test,ax=disp.ax_)
plot_roc_curve(gdboost,x_test,y_test,ax=disp.ax_)
plt.legend(prop = {'size':11}, loc ='lower right')
```

```
<matplotlib.legend.Legend at 0x2cc2b681100>
```



I checked the accuracy score for the following:

Gradient boosting- 99%

Logical regression - 1

KNN Neighbor-99%

Random Forest - 99%

Adaboost = 99%

SVM - 99%

```python
from sklearn.metrics import accuracy_score
print(accuracy_score(y_test,gdboost_yprad))
pd.crosstab(y_test,gdboost_yprad)
```

0.9998067051824489

col_0	0	1
label		
0	6423	6
1	3	40129

```python
#FOR LOGICTIC REGRESSION
from sklearn.metrics import accuracy_score
print(accuracy_score(y_test,lr_yprad))
pd.crosstab(y_test,lr_yprad)
```

1.0

col_0	0	1
label		
0	6429	0
1	0	40132

```python
#FOR KNN NEIGHBOR
from sklearn.metrics import accuracy_score
print(accuracy_score(y_test,knn_yprad))
pd.crosstab(y_test,knn_yprad)
```

0.9993127295375959

col_0	0	1
label		
0	6398	31
1	1	40131

```python
#FOR RANDOM FOREST
from sklearn.metrics import accuracy_score
print(accuracy_score(y_test,rf_yprad))
pd.crosstab(y_test,rf_yprad)
```

0.9999355683941497

col_0	0	1
label		
0	6426	3
1	0	40132

```
#FOR Adaboost FOREST
from sklearn.metrics import accuracy_score
print(accuracy_score(y_test,adb_yprad))
pd.crosstab(y_test,adb_yprad)
```

1.0

col_0	0	1
label		
0	6429	0
1	0	40132

```
#with SVM
from sklearn.metrics import accuracy_score
print(accuracy_score(y_test,svm_yprad))
pd.crosstab(y_test,svm_yprad)
```

0.9999570455960998

col_0	0	1
label		
0	6427	2
1	0	40132

•

Saving the model

```
Gradientboost=gdboost.fit(x_smote,y_smote)
```

```
import pickle as pkl
micro_finance_Model='micro_finance_Model.pickle'
pkl.dump(Gradientboost, open(micro_finance_Model,'wb'))
```

Interpretation of the Results

Conclusions:

- I have not used hyper tuning method as the classification and accuracy score is already high
- The data is imbalancedced with the target feature(87.5% for Non-defaulters and 12.5% for Defaulters).
- Few features had some unrealistic values such as 999860 days
- The imbalanced data was emoved with the SMOTE method
- With the PCA method, the top 25 columns were selected. The PCA curve was drawn to reduce the redundant columns
- The Distribution was not normal in the data set. The data was normalised with power transform and standard scalar
- There were also lots of outliers in the data. Percentile method was used to reduce outliers, however, inspire of that, there were outliers in 4 columns.These were reduced by standard scalar and power transform method

- Basically there are 2 type of observations made i.e, customer behavior for 30 days and 90 days, as well as two types of account held by customer main account and data account.
- 'Unnamed: 0' attribute has all unique values as same as index columns which has no importance for analysis.
- Approximately 90% of data in 'msisdn' has unique values, i.e, ID.
- 'payback30','payback90' has nearly 50% of the values having 0.
- More than 90% of 'last_rech_date_da', 'cnt_da_rech90 ','fr_da_rech90','medianamnt_loans30','medianamnt_loans90' has of values which is 0.
- 'pcircle' has only 1 unique value through out column and 'pdate' is a categorical column which can be dropped