FLIP ROBO

MICRO CREDIT PROJECT

Submitted by:

P. MANIVANNAN

INTRODUCTION

Problem Statement:

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

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

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

We are working with one such client that is in Telecom Industry. They are a fixed wireless telecommunications network provider. They have launched various products and have developed its business and organization based on the budget operator model, offering better products at Lower Prices to all value conscious customers through a strategy of disruptive innovation that focuses on the subscriber.

They understand the importance of communication and how it affects a person's life, thus, focusing on providing their services and products to low-income families and poor customers that can help them in the need of hour.

They are collaborating with an MFI to provide micro-credit on mobile balances to be paid back in 5 days. The Consumer is believed to be defaulter if he deviates from the path of paying back the loaned amount within the time duration of 5 days. For the loan amount of 5 (in Indonesian Rupiah), payback amount should be 6 (in Indonesian Rupiah), while, for the loan amount of 10 (in Indonesian Rupiah), the payback amount should be 12 (in Indonesian Rupiah).

The sample data is provided to us from our client database. It is hereby given to you for this exercise. In order to improve the selection of customers for the credit, the client wants some predictions that could help them in further investment and improvement in selection of customers.

• Conceptual Background of the Domain Problem

Understanding the importance of communication and how it affects a person’s life, thus, focusing on providing services and products to low-income families and poor customers that can help them in the need of hour.

• Review of Literature

The Research is done on Microfinance Institution and how each variable plays a role in predicting the customer habit to repay the loan with Data Science as tool the research is done. First Brief Exploratory Data Analysis is done which includes Skewness Removal, Outlier Treatment, Visualizing the Data to Understand Correlation between each feature. Then Data is Fed to Machine Learning Algorithms. after Evaluating and Testing Different Algorithms Best Algorithm is Selected. Based on Evaluation Metrics such as Accuracy Score, Cross Validation Score, ROC_AUC Curve etc.

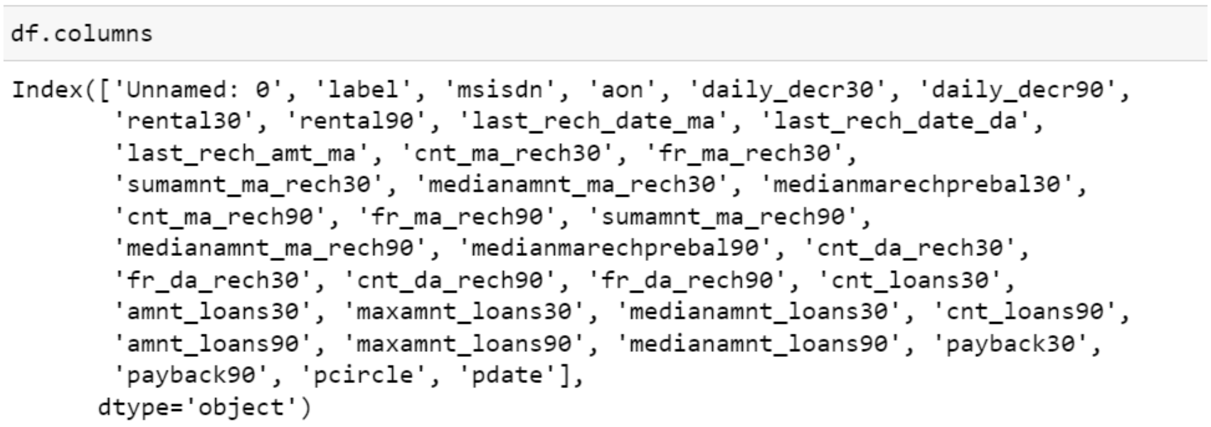
• Motivation for the Problem Undertaken

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

Analytical Problem Framing

- Data Sources and their formats

The data contains 209593 rows and 37 columns. The columns are



```
df.columns
```

```
Index(['Unnamed: 0', 'label', 'msisdn', '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',
       'payback90', 'pcircle', 'pdate'],
      dtype='object')
```

The data is in csv format.

- Data Pre-processing Done

1) We had categorical data which needs to be encoded. for this approach I used one hot encoder method as categorical data are not ordinal in nature.
2) Then Skewness Is checked for the data and Skewness is removed using power transform function.
3) Then outliers is checked using Z Score method. And removed.
4) The data are then standardized using Standard Scaler.
5) Used Principal Component Analysis to reduce the No of columns. To decide the No of columns I used pca.explained_variance_ratio_ method to find how much each feature (maximum variance) contribute to dataset.

Data Inputs- Logic- Output Relationships

Using Correlation function corr(). The correlation of features is analysed a nd how it affects our target variable. cnt_ma_rech30, cnt_ma_rech90, su mamnt_ma_rech90, sumamnt_ma_rech30, amnt_loans90, amnt_loans30 are positively correlated with people paying loan on time. fr_da_rech30, a on, medianmarechprebal30, fr_da_rech90 are negatively correlated with people paying loan on time

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

1) For handling the categorical one hot encoder method is used assuming the categorical data do not have ordinal relationship with each other.
2) Skewness is removed assuming the prediction is not affected by skewness removal.
3) Outliers are removed, assuming that those data are really outliers and do not contribute to dataset
4) Principal component analysis technique is used to reduce the column count. But the variance of features are retained by using pca.explained_variance_ratio_ method to find the variance exhibited by different features.

Hardware and Software Requirements and Tools Used

The Hardware requirements are 16 Gb RAM,500 GB SSD, at least 6 Cores processor

The software requirements are Windows, Anaconda Framework and libraries used are Pandas, NumPy, Sklearn, SciPy, Matplotlib, Seaborn

We use Pandas for reading the Dataset, Creating Dataframe, much more to handle data in dataset

Certain functions of NumPy are used like log , cbrt ,sqrt skewness transformation . absolute (abs) function used along with Zscore.

SKlearn Library is used for Machine Learning Algorithms like SVC, Random Forest, Linear Regression etc and Metrics for analysing them.

Scipy is scientific python library used for Zscore method etc.

Matplotlib and Seaborn are used for visualizations plotting graphs charts, etc

Finally joblib module is used for saving our model for future use.

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

  Since the Target Variable is classification type I should Use classification Approach to Solve this Problem and build a Classification Model

- Testing of Identified Approaches (Algorithms)
  1) Logistic Regression
  2) AdaBoostClassifier
  3) RandomForestClassifier
  4) KNeighborsClassifier
  5) SVC
  6) DecisionTreeClassifier

- ## Run and Evaluate selected models

  I have written a Custom Code Block for Running all Algorithms and Storing them in Data Frame and Ranking them based on their Accuracy Score. The best part is the code finds the best random state for each algorithm and prints them

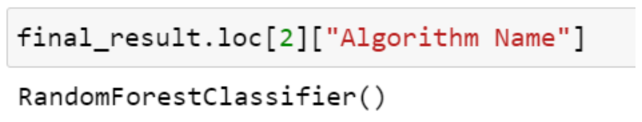
  Code:

```python
algo = [DecisionTreeClassifier(),KNeighborsClassifier(),RandomForestClassifier(),AdaBoostClassifier(),LogisticRegression()]
result = pd.DataFrame(columns=["Algorithm Name","Accuracy Score","Cross Validation Score"])

dtc=[]
knc=[]
rfc=[]
abc=[]
lr=[]
fl = [dtc,knc,rfc,abc,lr]

oo=0

for i in algo:
    rand=0
    acc=0

    for ii in range(2):
        train_x,test_x,train_y,test_y = train_test_split(x_final,y,test_size=.20,random_state=ii)
        cf = i
        cf.fit(train_x,train_y)
        pred = cf.predict(test_x)
        ac = accuracy_score(test_y,pred)
        if ac>acc:
            acc=ac
            rand=ii
    print(f' the best random state is {rand} for algorithm {i}')


    train_x,test_x,train_y,test_y = train_test_split(x_final,y,test_size=.20,random_state=rand)
    cf = i
    cf.fit(train_x,train_y)
    pred = cf.predict(test_x)
    ac = accuracy_score(test_y,pred)
    cv = cross_val_score(cf,x_final,y,cv=5).mean()
    fl[oo].insert(0,i)
    fl[oo].insert(1,ac)
    fl[oo].insert(2,cv)
    result.loc[oo] = fl[oo]
    oo +=1


final_result = result.sort_values(by=["Accuracy Score","Cross Validation Score"],ascending=False)
```

Output1:

	Algorithm Name	Accuracy Score	Cross Validation Score
2	(DecisionTreeClassifier(max_features='auto', r...	0.936179	0.937117
1	KNeighborsClassifier()	0.891801	0.891808
0	DecisionTreeClassifier()	0.878829	0.879705
3	(DecisionTreeClassifier(max_depth=1, random_st...	0.802140	0.800942
4	LogisticRegression()	0.777276	0.777445

Output2:

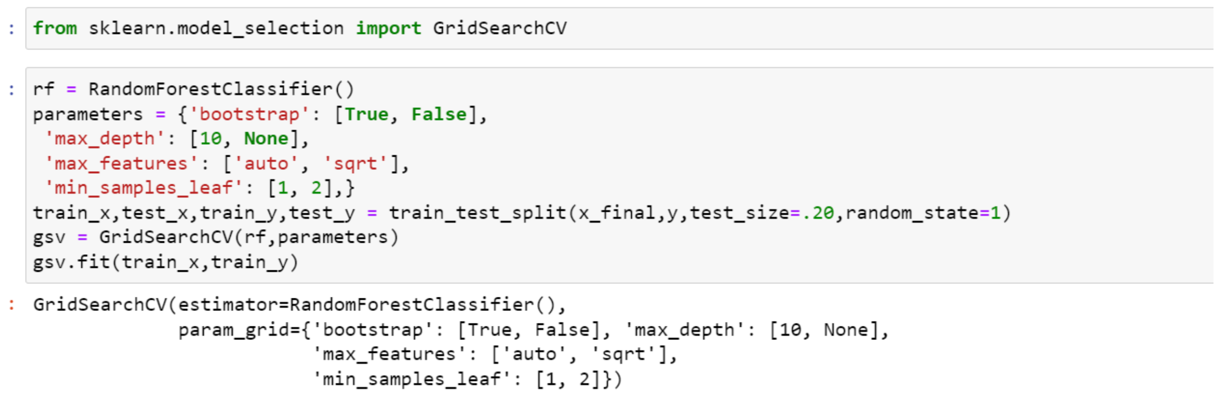


```
final_result.loc[2]["Algorithm Name"]

RandomForestClassifier()
```

Here you can see the Model Name, Accuracy Score, Cross Validation Score. Based on this best model is selected and further proceeded for hyperparameter tuning. Here Ridge Regression performed best compared to others.

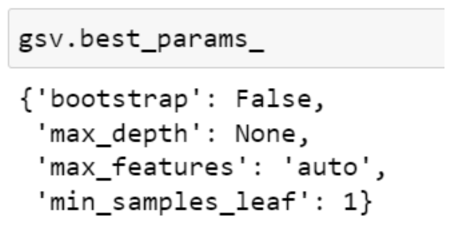
Hyperparameter Tuning

Code:



```
: from sklearn.model_selection import GridSearchCV

: rf = RandomForestClassifier()
  parameters = {'bootstrap': [True, False],
   'max_depth': [10, None],
   'max_features': ['auto', 'sqrt'],
   'min_samples_leaf': [1, 2],}
  train_x,test_x,train_y,test_y = train_test_split(x_final,y,test_size=.20,random_state=1)
  gsv = GridSearchCV(rf,parameters)
  gsv.fit(train_x,train_y)

: GridSearchCV(estimator=RandomForestClassifier(),
               param_grid={'bootstrap': [True, False], 'max_depth': [10, None],
                           'max_features': ['auto', 'sqrt'],
                           'min_samples_leaf': [1, 2]})
```

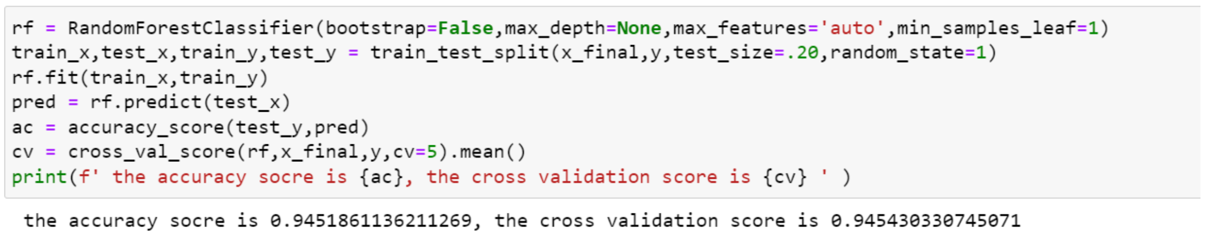
Output:



```
gsv.best_params_

{'bootstrap': False,
 'max_depth': None,
 'max_features': 'auto',
 'min_samples_leaf': 1}
```

The Best Parameters are listed now we need to run the model with the best parameters

Code:

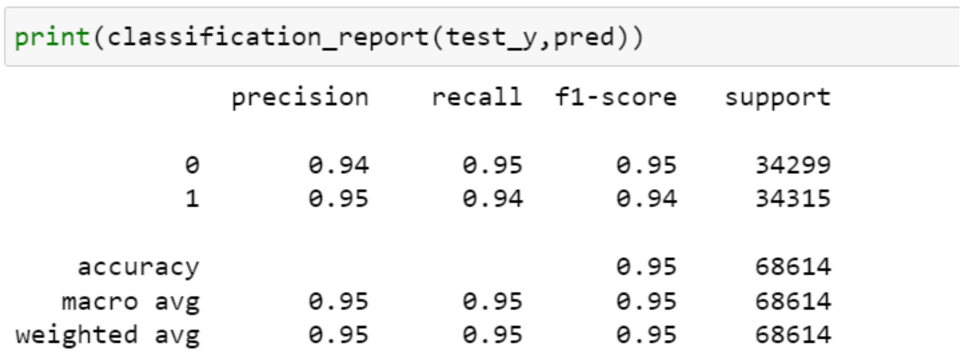


```
rf = RandomForestClassifier(bootstrap=False,max_depth=None,max_features='auto',min_samples_leaf=1)
train_x,test_x,train_y,test_y = train_test_split(x_final,y,test_size=.20,random_state=1)
rf.fit(train_x,train_y)
pred = rf.predict(test_x)
ac = accuracy_score(test_y,pred)
cv = cross_val_score(rf,x_final,y,cv=5).mean()
print(f' the accuracy socre is {ac}, the cross validation score is {cv} ' )
```

```
 the accuracy socre is 0.9451861136211269, the cross validation score is 0.945430330745071
```

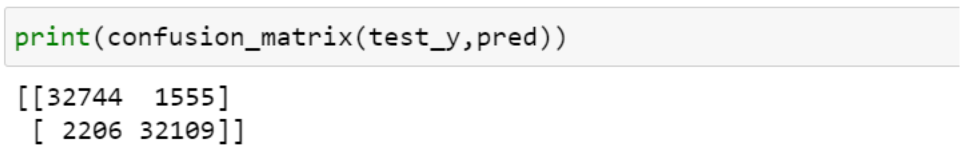
Output:

the accuracy score is 94% and cross validation score is 94%



```
print(classification_report(test_y,pred))
```

```
              precision    recall  f1-score   support

           0       0.94      0.95      0.95     34299
           1       0.95      0.94      0.94     34315

    accuracy                           0.95     68614
   macro avg       0.95      0.95      0.95     68614
weighted avg       0.95      0.95      0.95     68614
```



```
print(confusion_matrix(test_y,pred))
```

```
[[32744  1555]
 [ 2206 32109]]
```

The Best Training Score for Random Forest Regressor After Hyper parameter tuning for Training is 94% and Cross Validation Score is 94%

- Key Metrics for success in solving problem under consideration

Since it’s a Classification problem the key metrics used are Accuracy Score, Classification Method, Confusion Matrix, Cross Validation Report,ROC_AUC_Curve.

the accuracy score is 94% and cross validation score is 94%

```
print(classification_report(test_y,pred))
```

```
              precision    recall  f1-score   support

           0       0.94      0.95      0.95     34299
           1       0.95      0.94      0.94     34315

    accuracy                           0.95     68614
   macro avg       0.95      0.95      0.95     68614
weighted avg       0.95      0.95      0.95     68614
```

```
print(confusion_matrix(test_y,pred))
```

```
[[32744  1555]
 [ 2206 32109]]
```

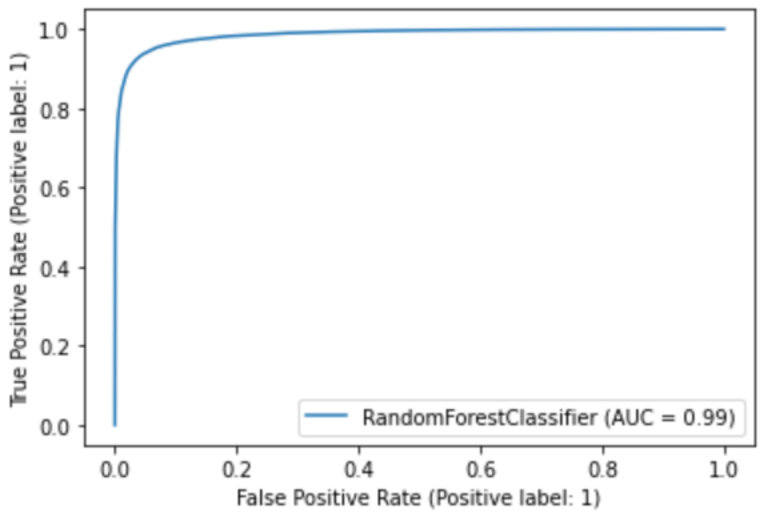
```
from sklearn.metrics import plot_roc_curve,roc_auc_score
```

```
print(f'The Roc Auc Score is {roc_auc_score(test_y,pred)}')
```

```
The Roc Auc Score is 0.9451883231009798
```

```
plot_roc_curve(rf,test_x,test_y)
```

```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x131a70a4df0>
```



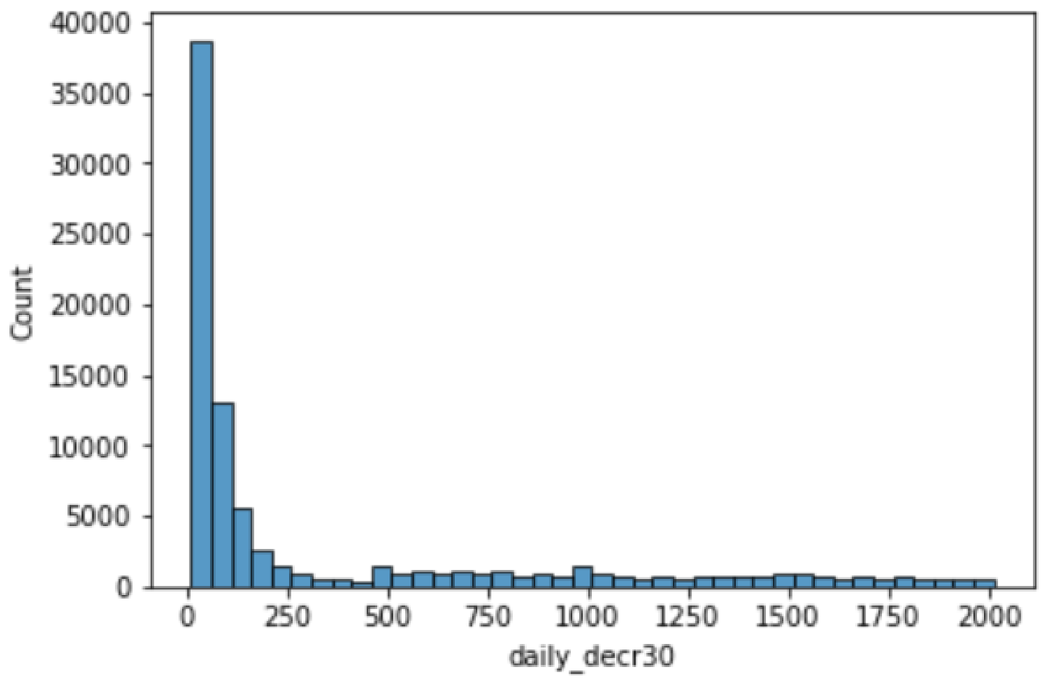
Visualizations

For visualizations the Matplotlib & Seaborn Library are used.

Code:

```
sns.histplot(df [ 'daily_decr30'],binwidth=50,binrange=(10,2000))
```

Output:



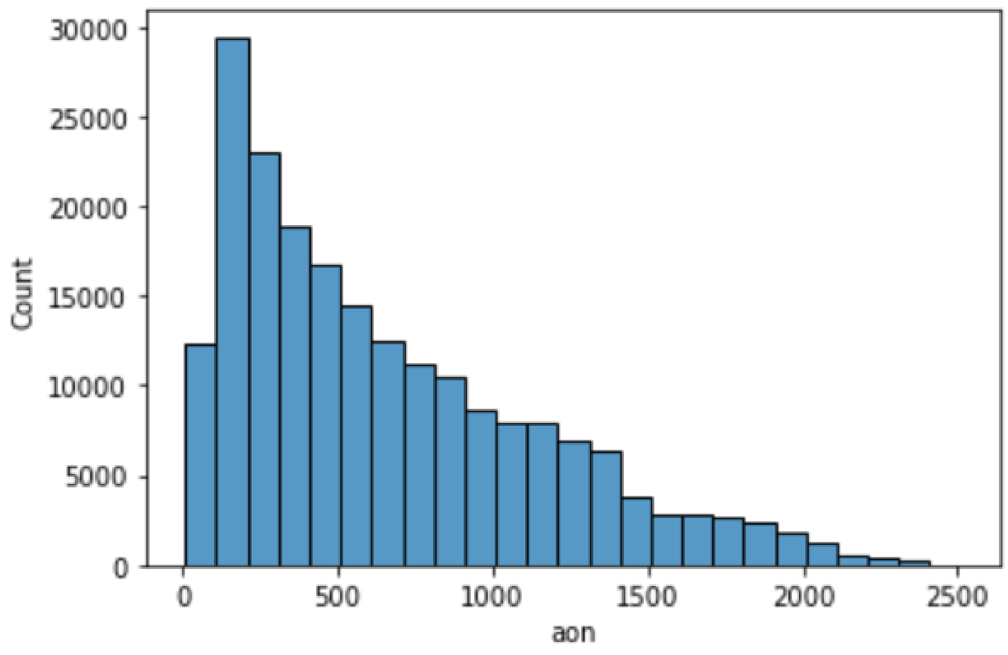
Observation:

Daily amount spent from main account, averaged over last 30 days are below 100 Indonesian rupiah on majority

Code:

```
sns.histplot(df ['aon'],binwidth=100,binrange=(10,2500))
```

Output:



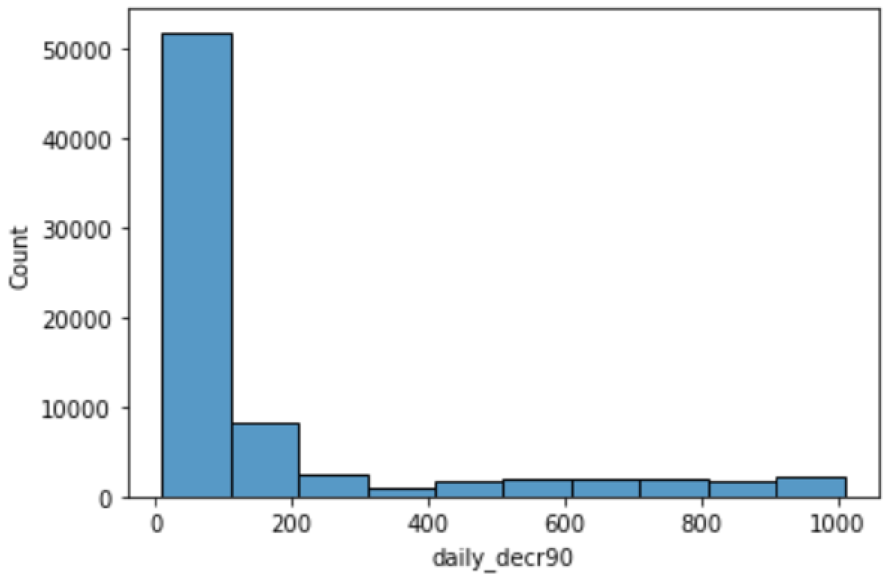
Observation:

age of cellular network below 500 days are higher in number

Code:

```
sns.histplot(df ['daily_decr90'],binwidth=100,binrange=(10,1000))
```

Output:



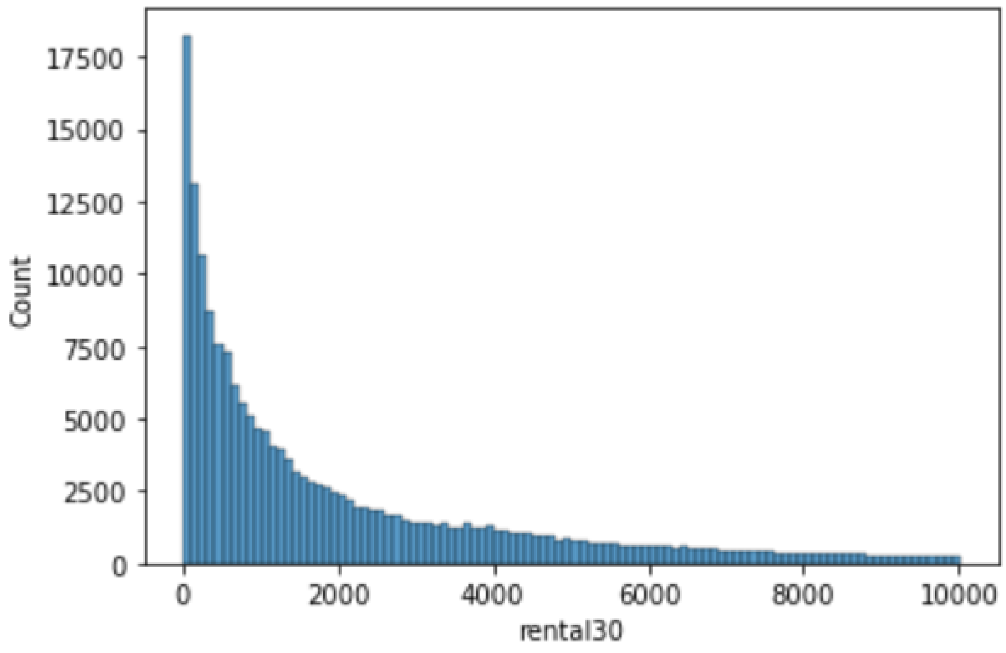
Observation:

Daily amount spent from main account, averaged over last 90 days (in Indonesian Rupiah) are below 200 on majority

Code:

```
sns.histplot(df ['rental30'],binwidth=100,binrange=(10,10000))
```

Output:



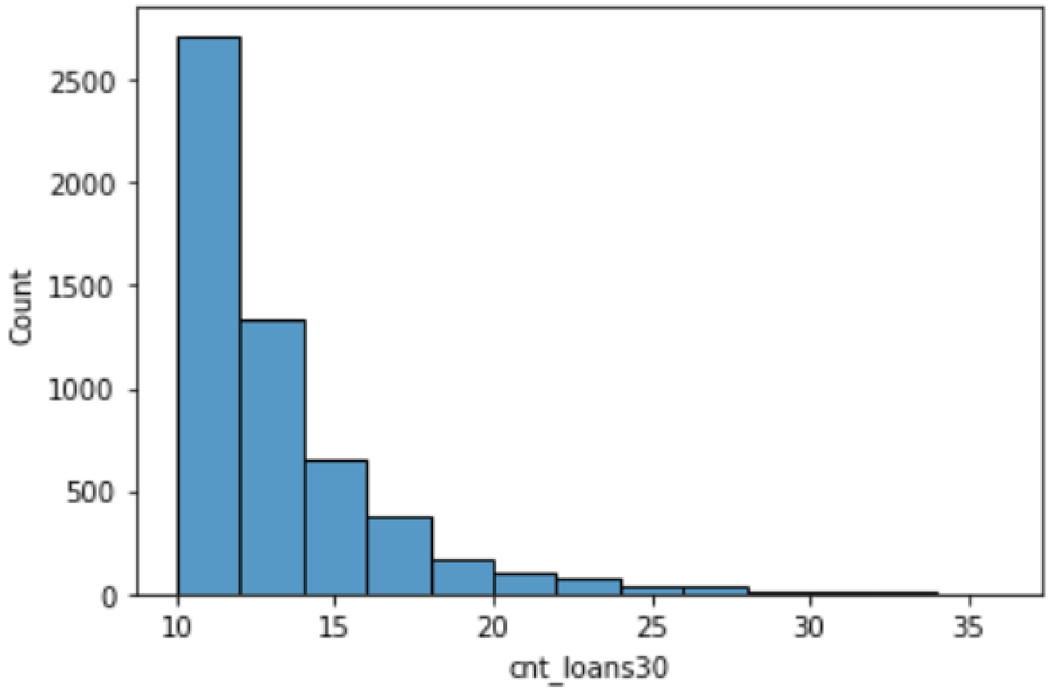
Observation:

Average main account balance over last 30 days are majory below 2000 indonesian rupiah

Code:

```
sns.histplot(df ['cnt_loans30'],binwidth=2,binrange=(10,35))
```

Output:



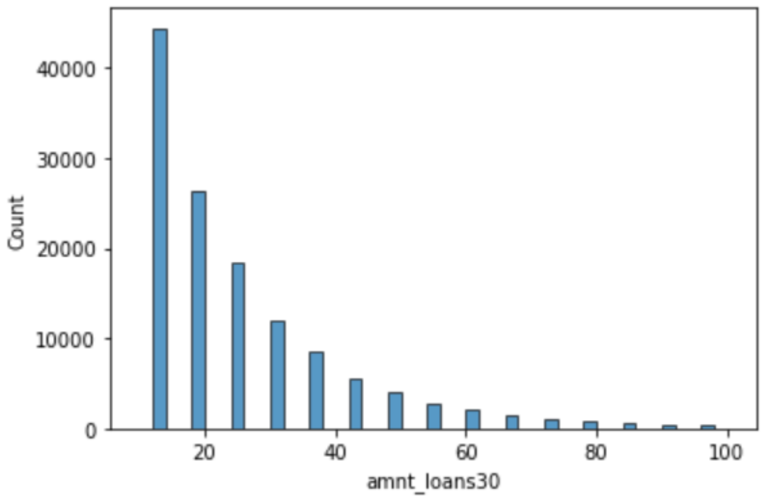
Observation:

Number of loans taken by user in last 30 days majority lies between 10 and 15

Code:

```
sns.histplot(df ['amnt_loans30'],binwidth=2,binrange=(10,100))
```

Output:

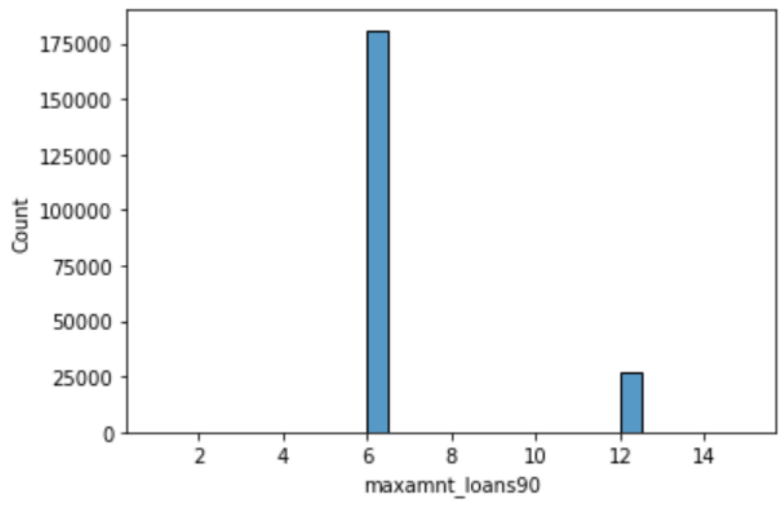


Observation:

Total amount of loans taken by user in last 30 days are majorly below 20 indonesian rupiah

```
sns.histplot(df ['maxamnt_loans90'],binwidth=0.5,binrange=(1,15))
```

```
<AxesSubplot:xlabel='maxamnt_loans90', ylabel='Count'>
```



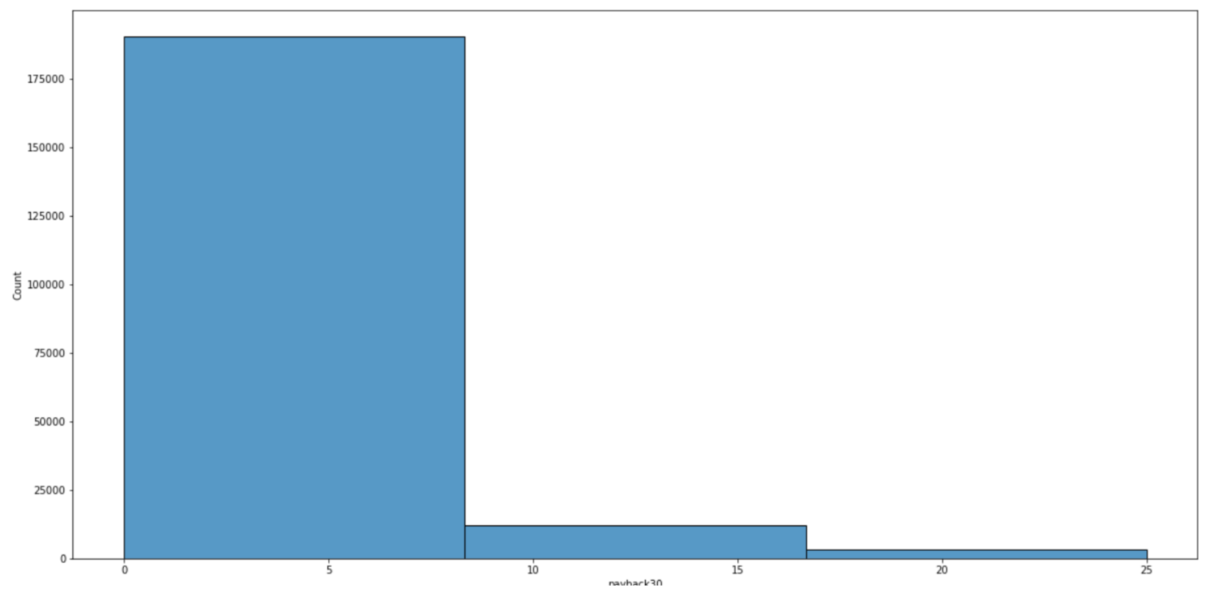
observation

maximum amount of loan taken by the user are majorly below 7 rupiah

Code:

```
plt.figure(figsize=(20,10))
sns.histplot(df['payback30'],binrange=(0,25),bins=3)
```

Output:



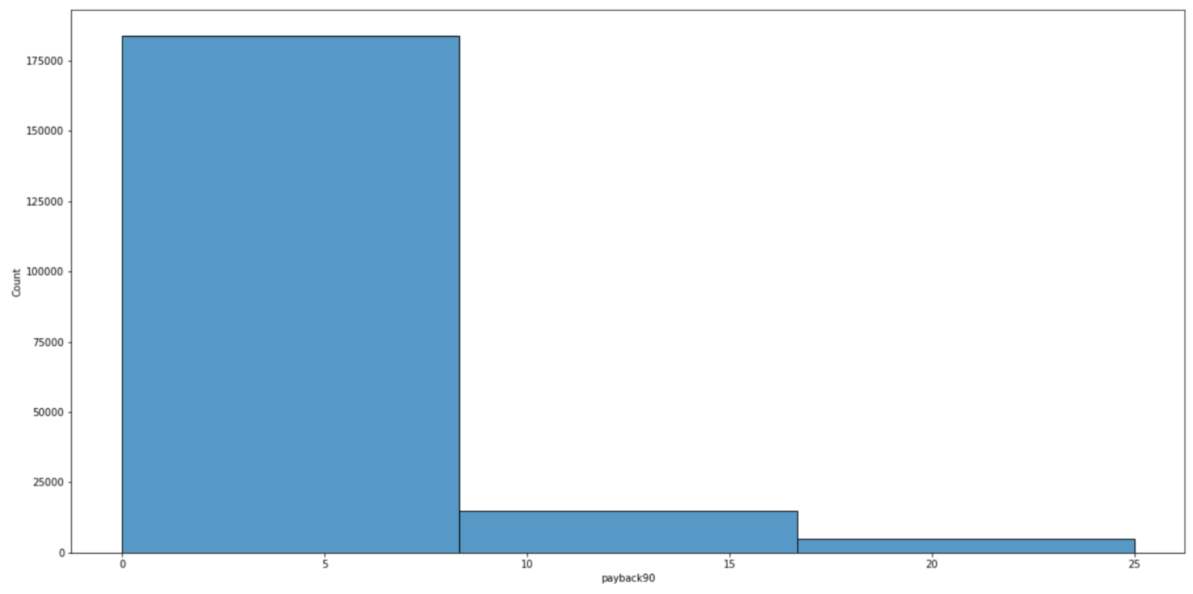
Observation:

Average payback time in days for past 30 days is mostly within 5 days

Code:

```
plt.figure(figsize=(20,10))
sns.histplot(df['payback90'],binrange=(0,25),bins=3)
```

Output:



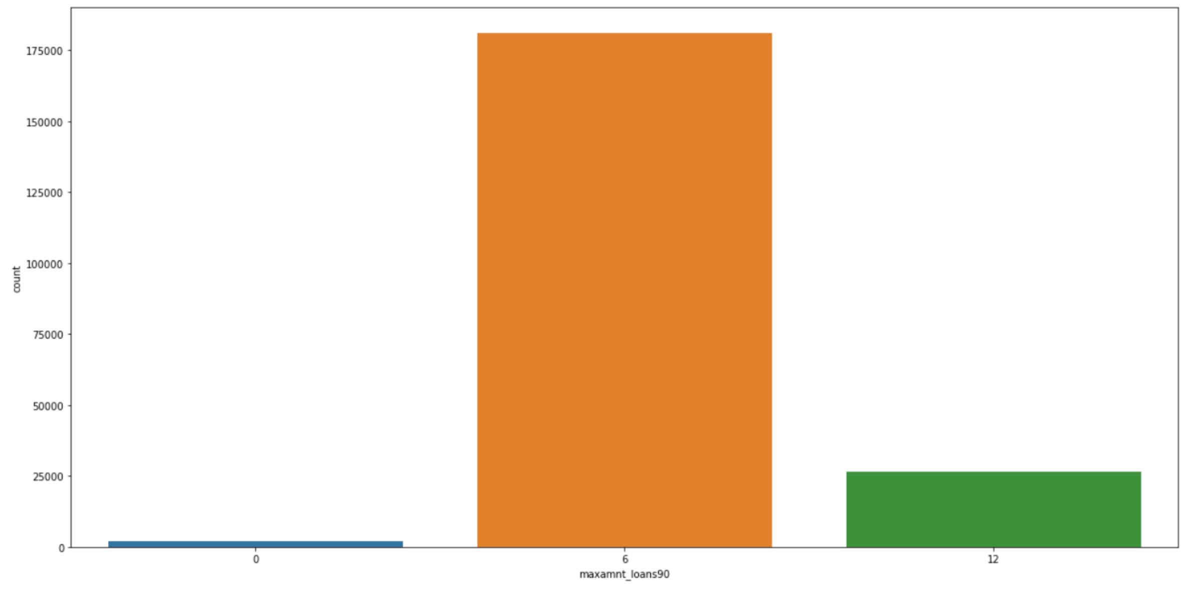
Observation:

Average payback time in days over last 90 day are mostly within 5 days

Code:

```
plt.figure(figsize=(20,10))
sns.countplot(df['maxamnt_loans90'])
```

Output:



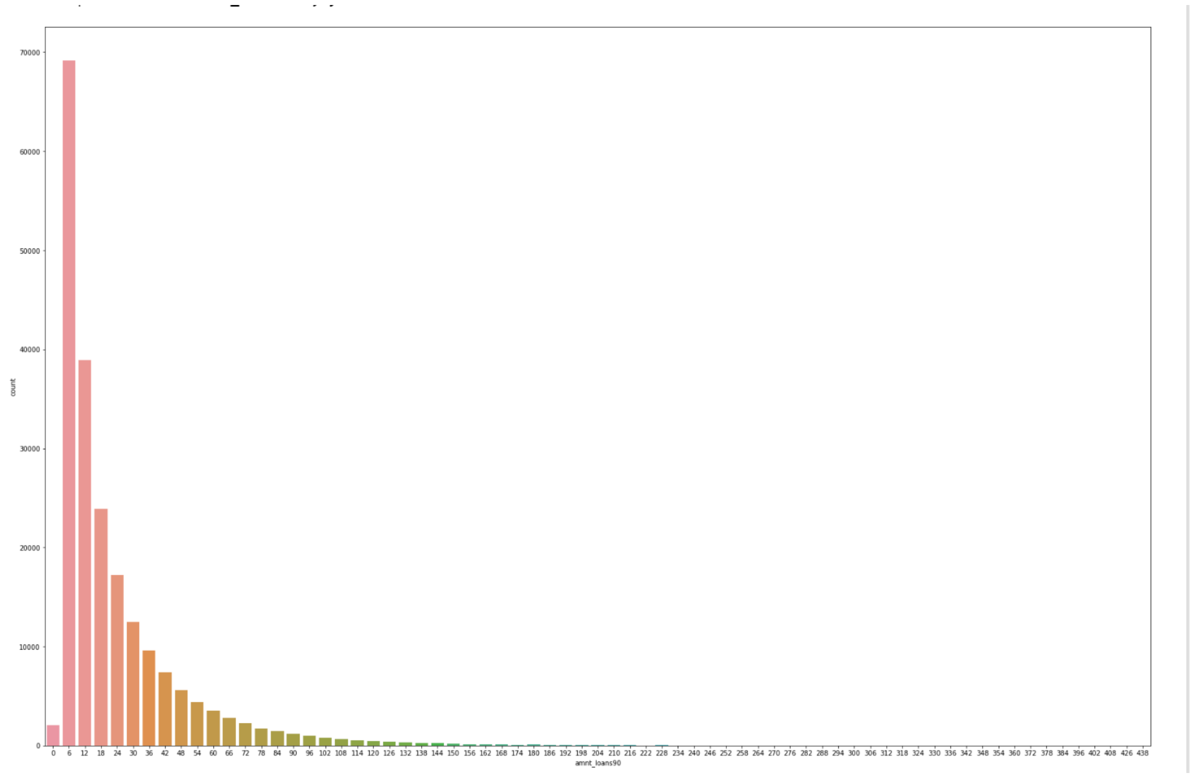
Observation:

maximum amount of loan taken by the user in last 90 days mostly are 6

Code:

```
plt.figure(figsize=(30,20))
sns.countplot(df['amnt_loans90'])
```

Output:



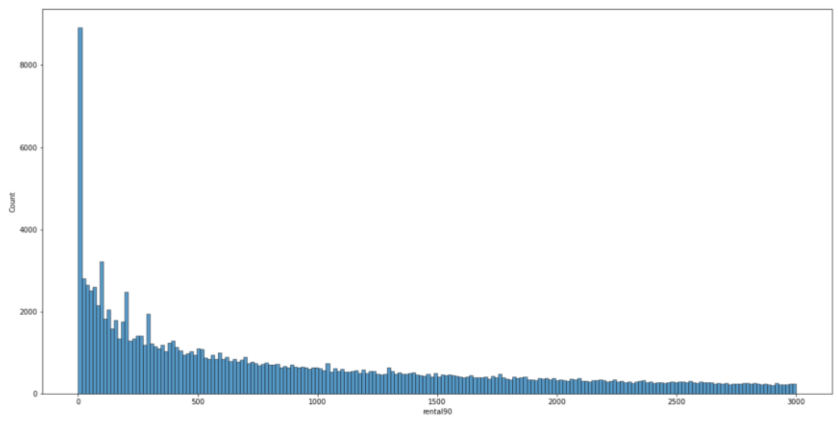
Observation:

Total amount of loans taken by user in last 90 days are mostly 6 and 12

Code:

```
plt.figure(figsize=(20,10))
sns.histplot(df['rental90'],bins=200,binrange=(0,3000))
```

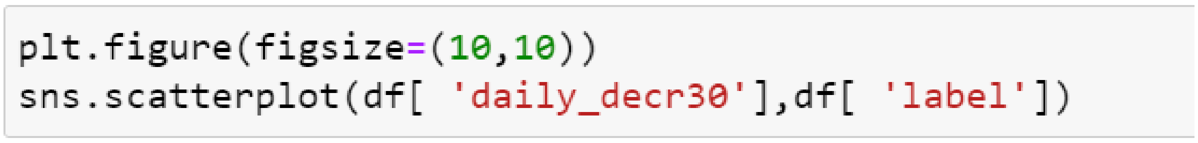
Output:



Observation:

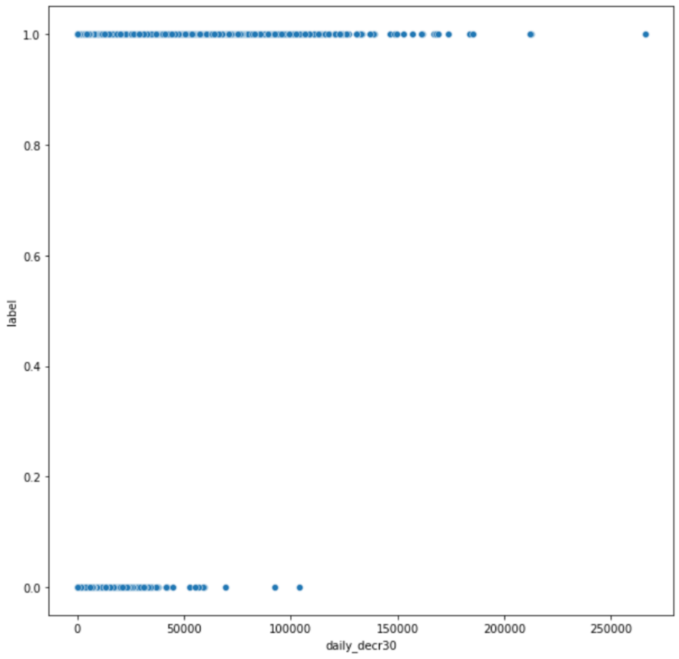
Average main account balance over last 90 days are majorly below 500

Code:



```
plt.figure(figsize=(10,10))
sns.scatterplot(df[ 'daily_decr30'],df[ 'label'])
```

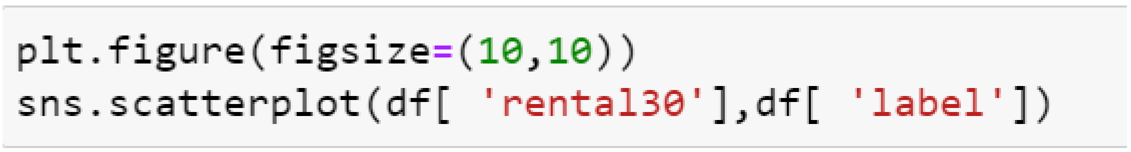
Output:



Observation:

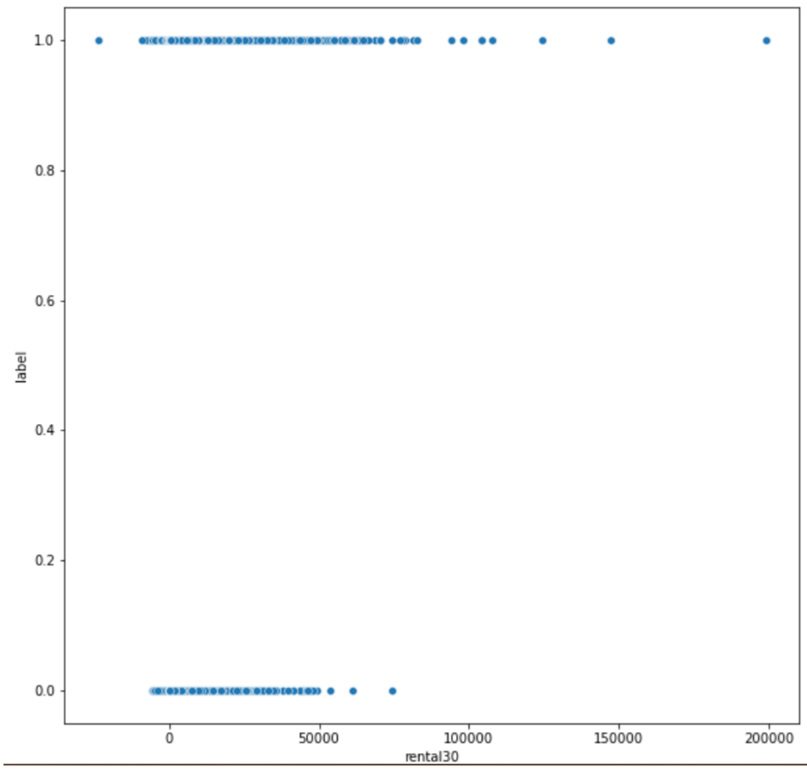
Daily amount spent from main account, averaged over last 30 days (in Indonesian Rupiah) increases for people who pay back loan in time

Code:



```
plt.figure(figsize=(10,10))
sns.scatterplot(df[ 'rental30'],df[ 'label'])
```

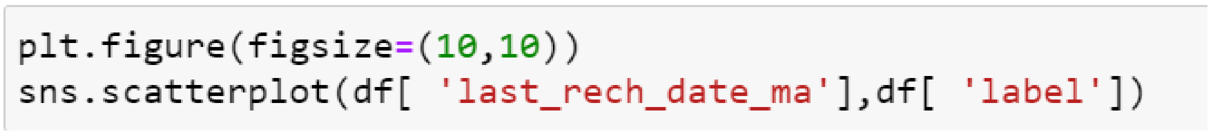
Output:



Observation:

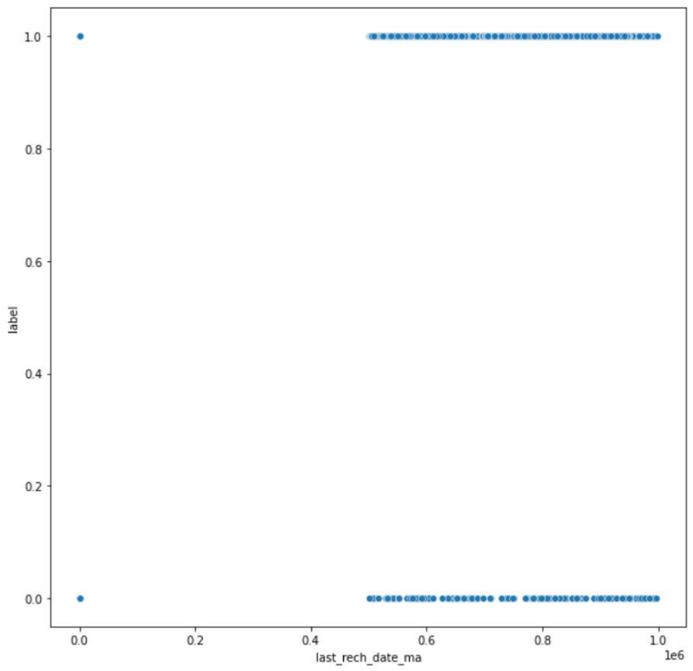
Average main account balance over last 30 days more than 50000 have moslty paid the loan on time

Code:



```
plt.figure(figsize=(10,10))
sns.scatterplot(df[ 'last_rech_date_ma'],df[ 'label'])
```

Output:



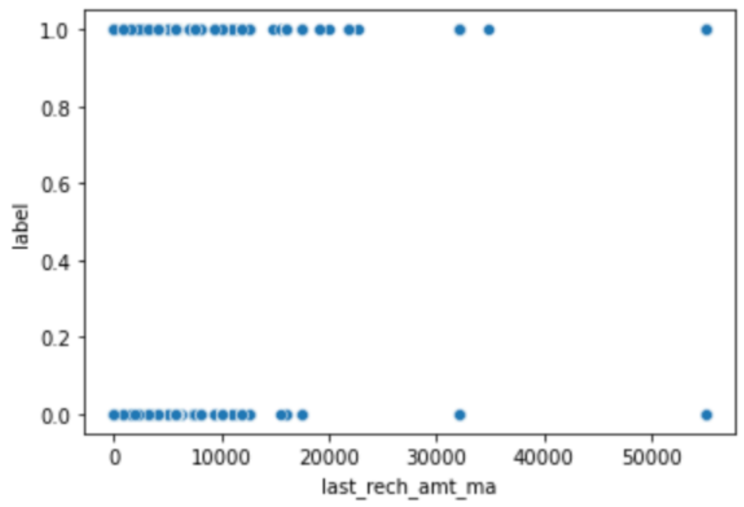
Observation:

Number of days till last recharge of main account are similar for both cases

Code:

```
sns.scatterplot(df[ 'last_rech_amt_ma'],df[ 'label'])
```

Output:



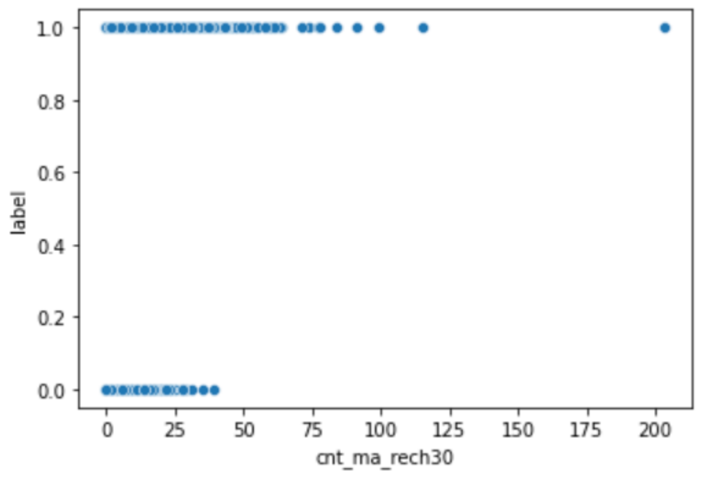
Observation:

for people paid loan on time have recharged higher amount on last recharge

Code:

```
sns.scatterplot(df[ 'cnt_ma_rech30'],df[ 'label'])
```

Output:



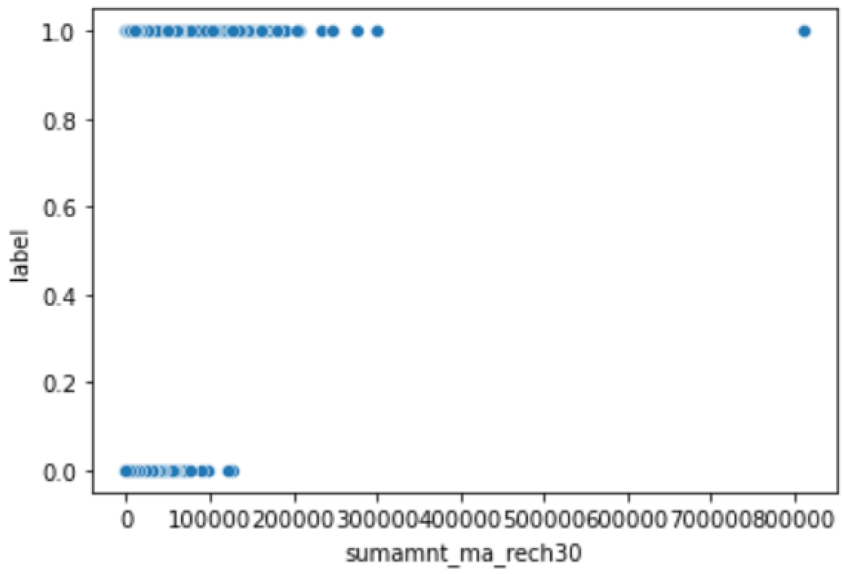
Observation:

People paid loan on time have recharged their account many number of times in last 30 days

Code:

```
sns.scatterplot(df[ 'sumamnt_ma_rech30'],df[ 'label'])
```

Output:



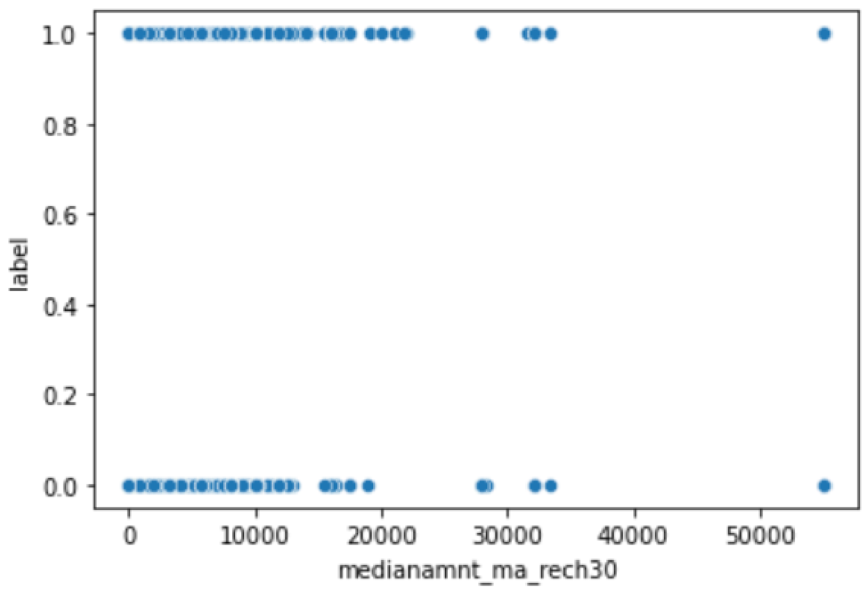
Observation:

People paid loan on time have higher Total amount of recharge in main account

Code:

```
sns.scatterplot(df[ 'medianamnt_ma_rech30'],df[ 'label'])
```

Output:



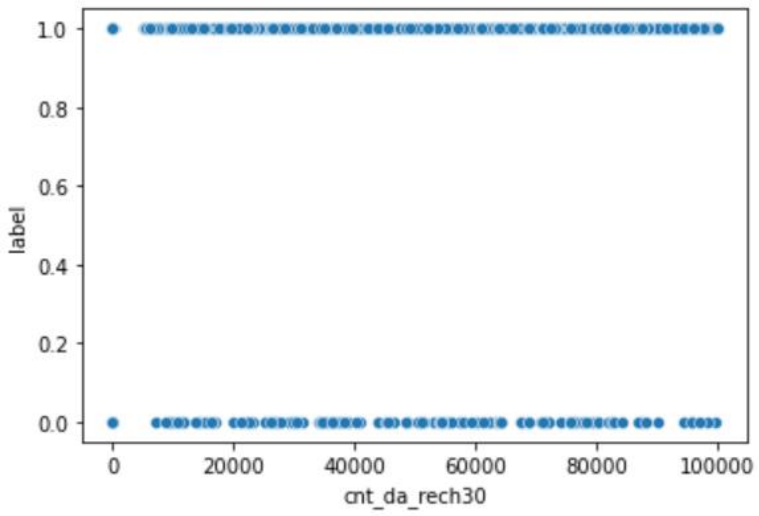
Observation:

People paid loan on time have higher Median amount of recharge done in last 30 days

Code:

```
sns.scatterplot(df[ 'cnt_da_rech30'],df[ 'label'])
```

Output:



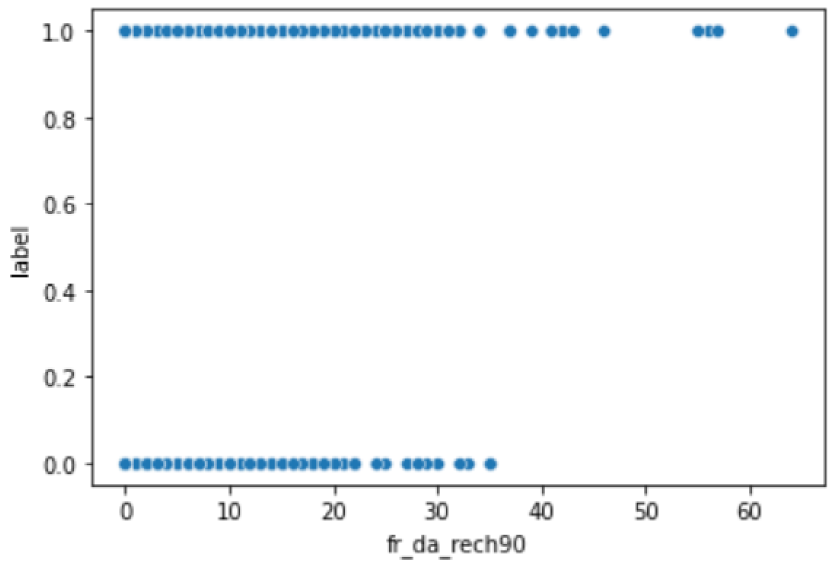
Observation:

People paid loan on time have higher number of data account recharge for past 30 days

Code:

```
sns.scatterplot(df[ 'fr_da_rech90'],df[ 'label'])
```

Output:



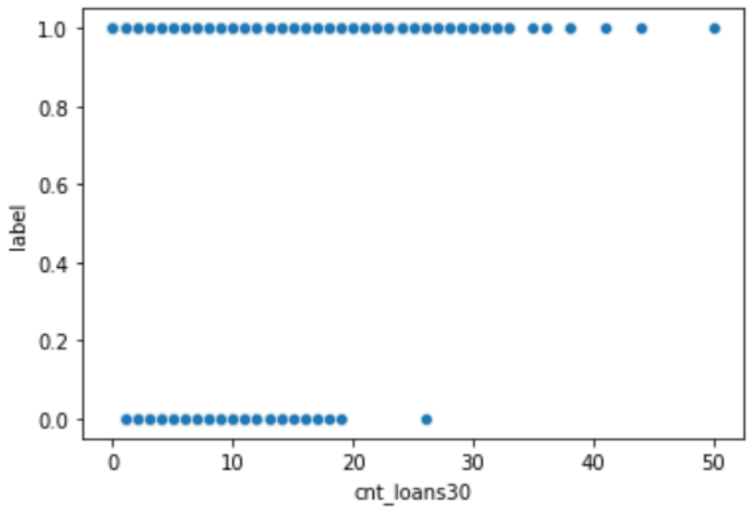
Observation:

People paid loan on time have higher Frequecy of data account got recharged in 90 days

Code:

```
sns.scatterplot(df[ 'cnt_loans30'],df[ 'label'])
```

Output:



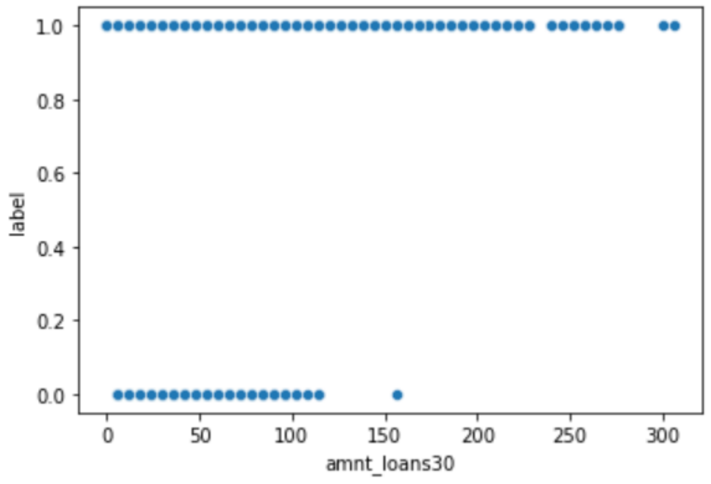
Observation:

People paid loan on time have higher Number of loans taken in last 30 days

Code:

```
sns.scatterplot(df[ 'amnt_loans30'],df[ 'label'])
```

Output:



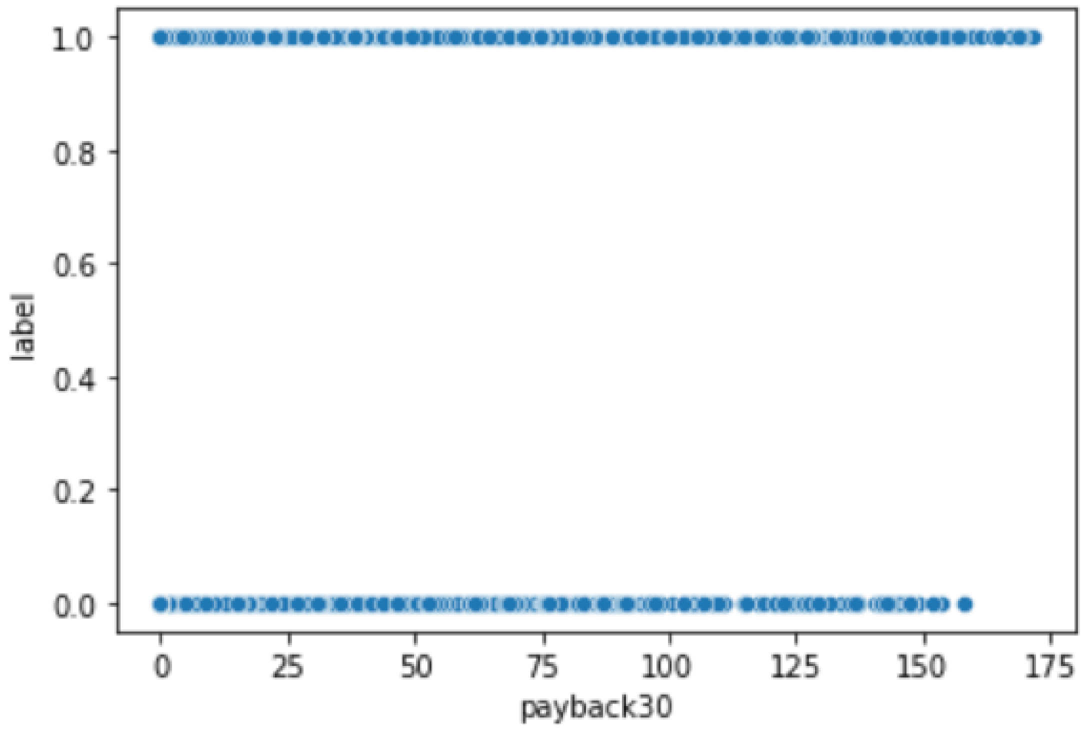
Observation:

People paid loan on time have higher Total amount of loans taken in last 30 days

Code:

```
sns.scatterplot(df[ 'payback30'],df[ 'label'])
```

Output:



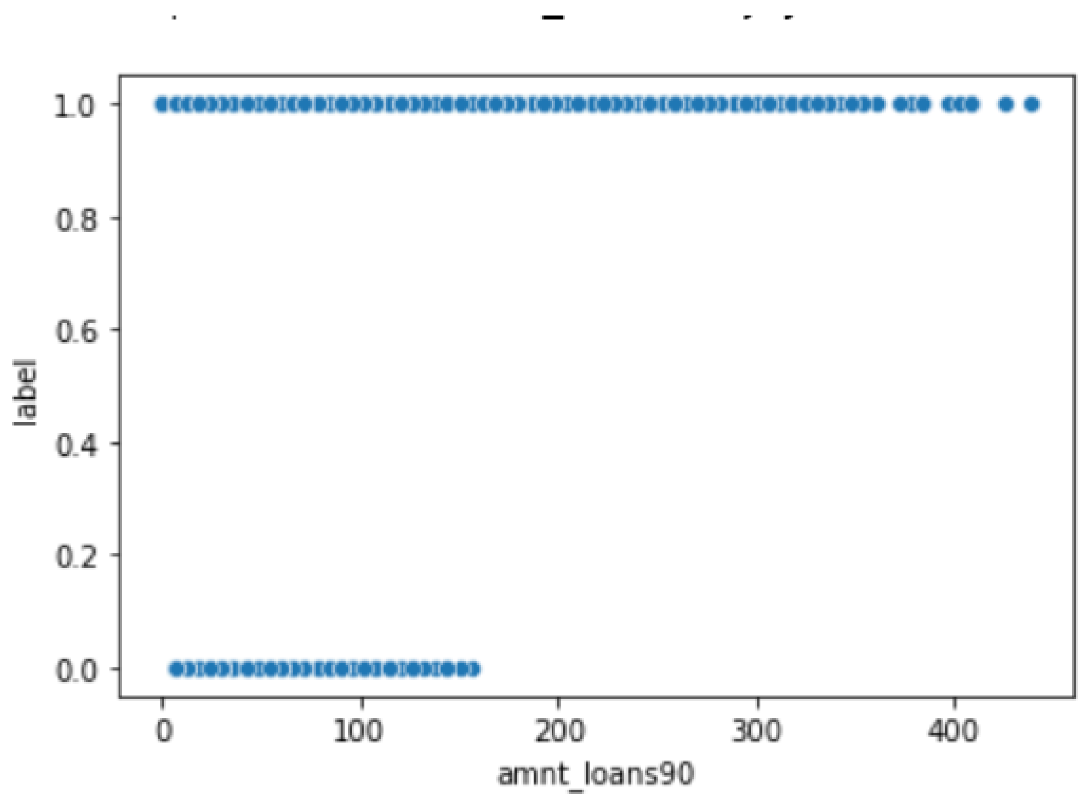
Observation:

Average payback time in days over last 30 days

Code:

```
sns.scatterplot(df[ 'amnt_loans90'],df[ 'label'])
```

Output:



Observation:

People Pay loan on time have higher Total amount of loans taken in last 90 days

- Interpretation of the Results

  Based on the visualizations I can see if a person's daily amount spends from main account, average main account balance, no of times account recharged, no of times data account recharged, no of times loan taken, total amount of loan increases he is more likely to pay loan on time

CONCLUSION

- Key Findings and Conclusions of the Study

  We can clearly see the person who most likely to pay loan on time have certain traits. If factors like daily amount spends from main account, average main account balance, no of times account recharged, no of times data account recharged, no of times loan taken, total amount of loan these increase he is more likely to pay loan on time

- Learning Outcomes of the Study in respect of Data Science

  Based on this study we can identify several Important factors which help us to determine a person quality to pay loan on time.