MICRO-CREDIT DEFAULTER

SUBMITTED BY:

SHUBHAM SHUKLA

ACKNOWLEDGMENT

I have taken references in this project for different materials in the below links:

1. https://stackoverflow.com/
2. https://www.geeksforgeeks.org/

INTRODUCTION

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.

Business Problem Framing

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 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).

Conceptual Background of the Domain Problem

The conceptual background of this domain problem that client wants to know whether their users are paying the loaned amount within the time frame or not, if not what are the criteria.
So, the sample data is provided to us from our client database. 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.
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.

Review of Literature

To predict in terms of a probability for each loan transaction, whether the customer will be paying back the loaned amount within 5 days of insurance of loan. In this case, Label ‘1’ indicates that the loan has been paid i.e., Non- defaulter, while, Label ‘0’ indicates that the loan has not been paid i.e., defaulter.
I have observed that there are no null values present in the dataset given by clients and also there are some customers who doesn’t have any loan history.
But the data set is quite imbalanced in terms of defaulter and non-defaulters, as the MFI to provide micro-credit on mobile balances to be paid back in 5 days. As per my research, I had seen that most of the users paid the amount within the time frame but if they missed to pay within the time frame of 5 days, they have paid almost like within 7 days, and for that, and I can observe that the client has charged rupiah 6 mostly to the customer.

Most number of loans are taken by Defaulter in the month of June 2016 and there are many valued customers who are good to our client and haven’t paid the amount within the deadline.

Data Sources and their formats

The data is been provided by one of our clients from telecom industry. They are a fixed wireless telecommunications network provider and 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.
The data is been given by Indonesian telecom company and they gave it to us in a CSV file, with data description file in excel format. They also had provided the problem statement by explaining what they need from us and also the required criteria to be satisfied.

Let's check the data now. Below I have attached the snapshot below to give an overview.

In [1]:
```
import pandas as pd
#Importing warning library to avoid any warnings
import warnings
warnings.filterwarnings('ignore')
df=pd.read_csv('C://Users/Admin/Downloads/Micro Credit Project/Data file.csv',parse_dates=['pdate'])
df.head()
```

Out[1]:

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

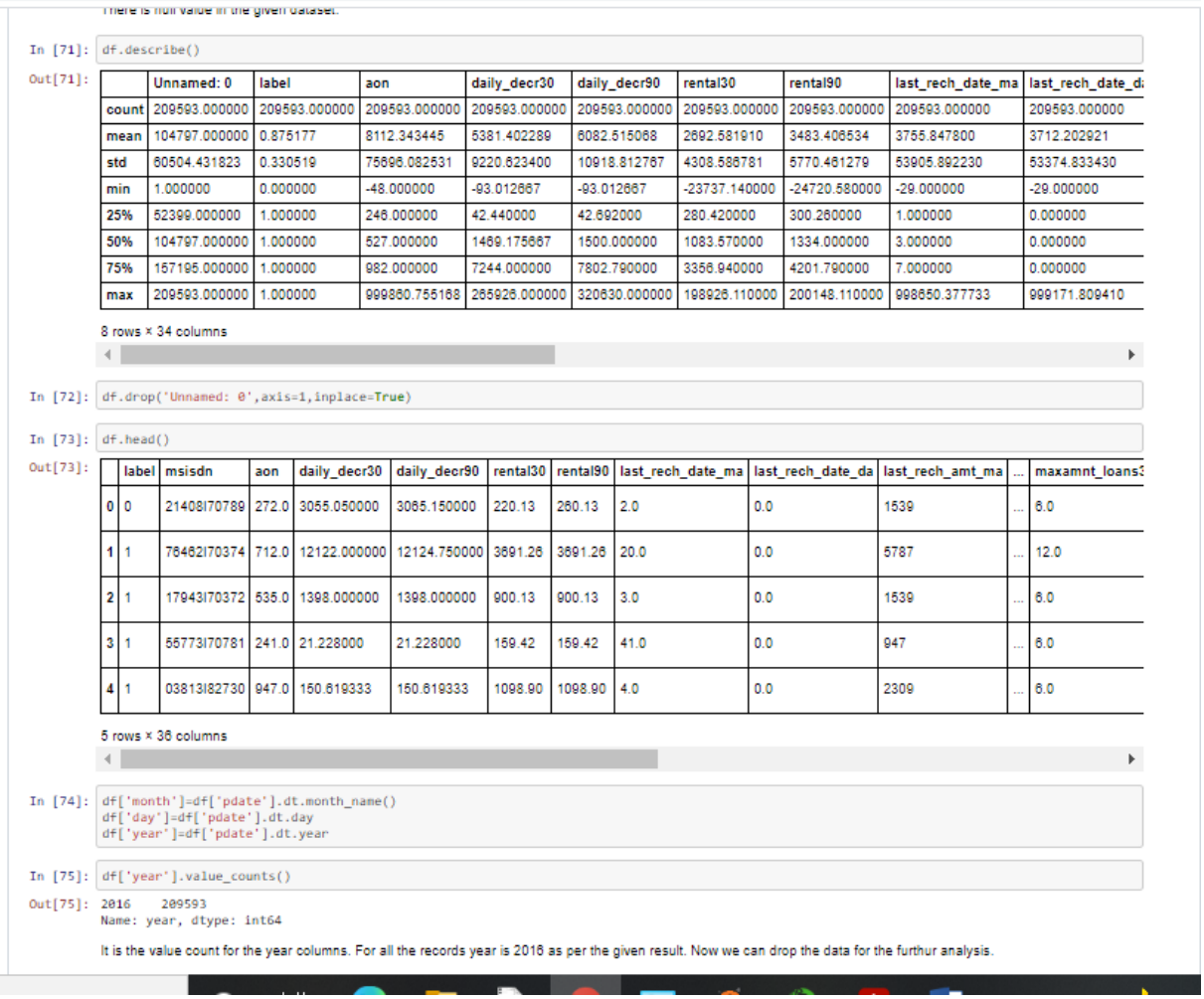
5 rows × 37 columns

In [2]:
```
#We can see that Unnamed:0 is just the index number.lets drop that column
df.drop('Unnamed: 0',axis=1,inplace=True)
```

In [3]:
```
df.shape
```

Data Pre-processing

Looking out at the summary of the dataset:



There is non value in the given dataset.

In [71]:
```
df.describe()
```

Out[71]:

	Unnamed: 0	label	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_date_da
count	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000
mean	104797.000000	0.875177	8112.343445	5381.402289	6082.515068	2692.581910	3483.406534	3755.847800	3712.202921
std	60504.431823	0.330519	75696.082531	9220.623400	10918.812767	4308.586781	5770.461279	53905.892230	53374.833430
min	1.000000	0.000000	-48.000000	-93.012667	-93.012667	-23737.140000	-24720.580000	-29.000000	-29.000000
25%	52399.000000	1.000000	246.000000	42.440000	42.692000	280.420000	300.260000	1.000000	0.000000
50%	104797.000000	1.000000	527.000000	1469.175667	1500.000000	1083.570000	1334.000000	3.000000	0.000000
75%	157195.000000	1.000000	982.000000	7244.000000	7802.790000	3356.940000	4201.790000	7.000000	0.000000
max	209593.000000	1.000000	999860.755168	265926.000000	320630.000000	198926.110000	200148.110000	998650.377733	999171.809410

8 rows × 34 columns

In [72]:
```
df.drop('Unnamed: 0',axis=1,inplace=True)
```

In [73]:
```
df.head()
```

Out[73]:

	label	msisdn	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_date_da	last_rech_amt_ma	...	maxamnt_loans3
0	0	21408I70789	272.0	3055.050000	3065.150000	220.13	260.13	2.0	0.0	1539	...	6.0
1	1	76462I70374	712.0	12122.000000	12124.750000	3691.26	3691.26	20.0	0.0	5787	...	12.0
2	1	17943I70372	535.0	1398.000000	1398.000000	900.13	900.13	3.0	0.0	1539	...	6.0
3	1	55773I70781	241.0	21.228000	21.228000	159.42	159.42	41.0	0.0	947	...	6.0
4	1	03813I82730	947.0	150.619333	150.619333	1098.90	1098.90	4.0	0.0	2309	...	6.0

5 rows × 36 columns

In [74]:
```
df['month']=df['pdate'].dt.month_name()
df['day']=df['pdate'].dt.day
df['year']=df['pdate'].dt.year
```

In [75]:
```
df['year'].value_counts()
```

Out[75]:
```
2016    209593
Name: year, dtype: int64
```

It is the value count for the year columns. For all the records year is 2016 as per the given result. Now we can drop the data for the furthur analysis.

On the off chance that we can see above, we will see that the information is very imbalanced. In certain segments the mean is more noteworthy than middle, with the goal that we can say the information which given by our customer is correct slanted from the beginning, and for certain sections the mean is lesser than the middle so I can say the information is left slanted. I likewise saw that there is an immense

contrast in 75% quartile and most extreme worth so from that I presume that the information contains gigantic anomalies, and furthermore in certain segments I have seen that there is no information till 75% quartile and at last the greatest information come in picture so we can simply drop that sections.

h_amt_ma	...	maxamnt_loans30	medianamnt_loans30	cnt_loans90	amnt_loans90	maxamnt_loans90	medianamnt_loans90	payback30	payback90	month	day
1539	...	6.0	0.0	2.0	12	6	0.0	29.000000	29.000000	July	20
5787	...	12.0	0.0	1.0	12	12	0.0	0.000000	0.000000	August	10
1539	...	6.0	0.0	1.0	6	6	0.0	0.000000	0.000000	August	19
947	...	6.0	0.0	2.0	12	6	0.0	0.000000	0.000000	June	6
2309	...	6.0	0.0	7.0	42	6	0.0	2.333333	2.333333	June	22

Checked the maximum amount of loan taken by the user in last 30 days and found that the data have so much of outliers as per the description given by the client that the loan amount can be paid by the customer is either rupiah 6 or 12 so that I have dropped all the loan amount that shows the loan is taken more than 12.

Checked for number of defaulters in dataset.

Checking the data of defaulters alone

```
defaulters_data=df.loc[df['label'] == 0 ]
defaulters_data
```

	label	msisdn	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_date_da	last_rech_amt_ma	...	maxamnt_loans3
0	0	21408I70789	272.0	3055.050000	3065.150000	220.13	260.13	2.0	0.0	1539	...	6.
11	0	82417I90848	82.0	65.166667	65.166667	326.20	326.20	17.0	0.0	7526	...	6.
15	0	24075I89239	1037.0	12.000000	12.000000	1216.80	1216.80	0.0	0.0	0	...	6.
16	0	82053I85350	1583.0	1000.000000	1000.000000	1000.80	1087.88	0.0	0.0	0	...	6.
21	0	75522I70784	378.0	514.693333	515.200000	56.26	58.20	2.0	0.0	773	...	6.
...	...	...	...	...	...	...	...	...	...	...	...	
209547	0	32172I88688	153.0	5670.733333	5672.200000	1817.08	2764.88	0.0	0.0	0	...	6.
209549	0	59552I90843	843.0	729.235000	758.470000	7470.90	9537.90	1.0	0.0	770	...	6.
209554	0	49076I89233	744.0	1454.491667	1461.750000	559.73	655.28	31.0	0.0	2309	...	6.
209571	0	59768I84453	827.0	1867.668667	1881.180000	1875.72	2312.65	14.0	0.0	1924	...	6.
209584	0	70387I89237	945.0	0.000000	0.000000	78.30	78.30	0.0	0.0	0	...	6.

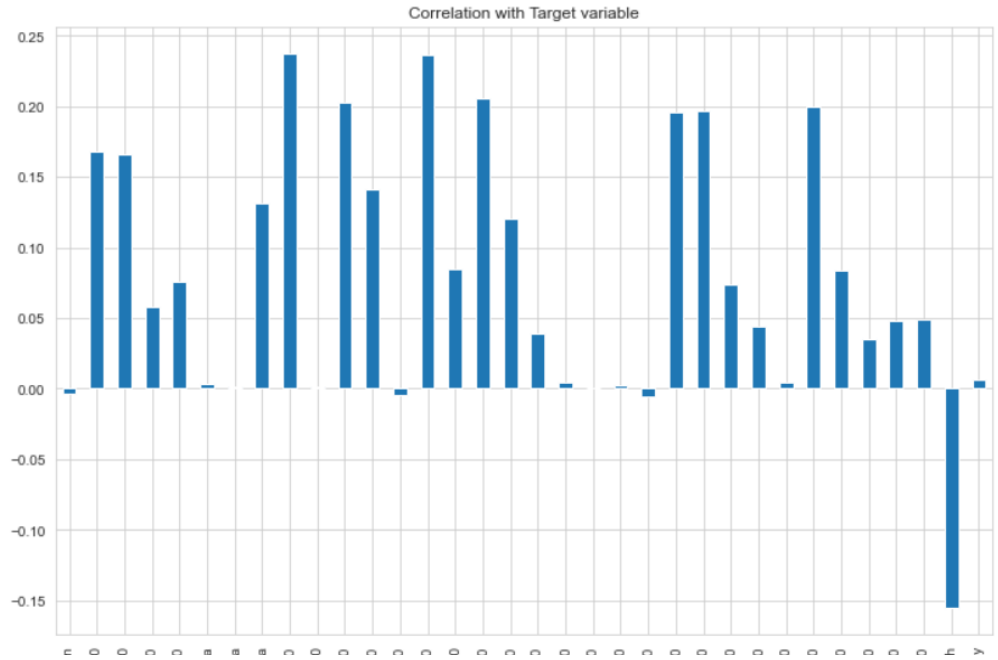
26033 rows × 36 columns

Data Inputs- Logic- Output Relationships

As the information given to us is in csv configuration and customer has portrayed it well that we need to foresee whether the defaulters are paying the sum inside 5 days or not, it's exceptionally obvious to us that we need to anticipate the double worth which is Name '1' demonstrates that the advance has been paid i.e., Non-defaulter, while, Mark '0' shows that the advance has not been paid i.e., defaulter. Based on correlational information with my yield acquired, we had plotted the figure and it is given beneath:

```
plt.figure(figsize=(12,8))
df.drop('label',axis=1).corrwith(df['label']).plot(kind='bar',grid=True)
plt.title('Correlation with Target variable')
```

```
Text(0.5, 1.0, 'Correlation with Target variable')
```



By observing the relation of input output I found out that all the data is positively correlated apart from few columns data. Foe eg: Month column data was negatively correlated.

Hardware and Software Requirements and Tools Used

In Jupyter notebook, I had used lots of python libraries to carry out this project and I have mentioned

1.Pandas- a library which is used to read the data, visualisation and analysis of data.

2. NumPy- used for working with array and various mathematical techniques. 3.Seaborn- visualization tool for plotting different types of plot.

4. Matplotlib- It provides an object-oriented API for embedding plots into applications.

5. Jupyter Notebook

Testing of Identified Approaches

I used Logistic Regression, GaussianNB, DecisionTreeClassifier and KNeighborsClassifier algorithms for finding out the best model among those.

Also, I used Ensemble Techniques like Random Forest, Adaboost and Gradient Boosting Classifier algorithms to find the best performing model.

```
models= []
models.append(('Logistic Regression',LR))
models.append(('GaussianNB',gnb))
models.append(('DecisionTreeClassifier',dtc))
models.append(('KNeighborsClassifier',knc))
models.append(('RandomForestClassifier',rfc))
models.append(('AdaBoostClassifier',abc))
models.append(('GradientBoostingClassifier',gbc))
```

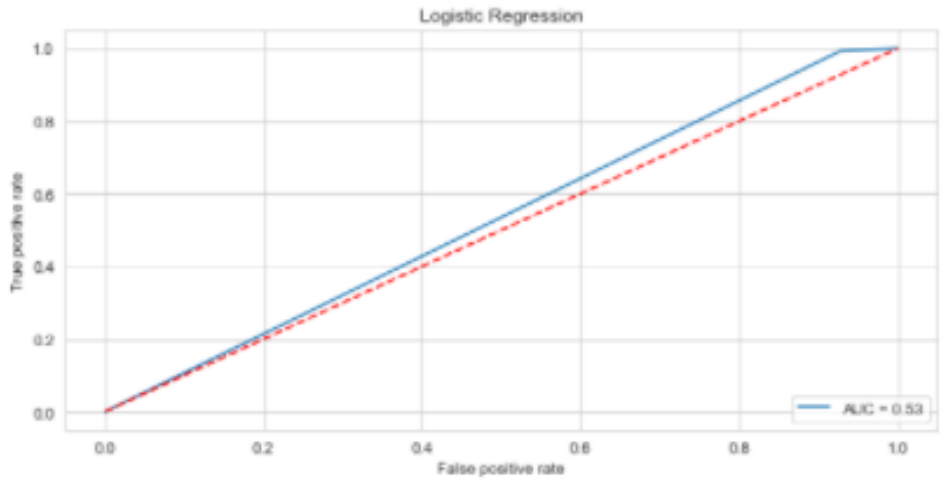
The algorithms I have used to study this research are listed above in image.

Looking out at the code performed for models:

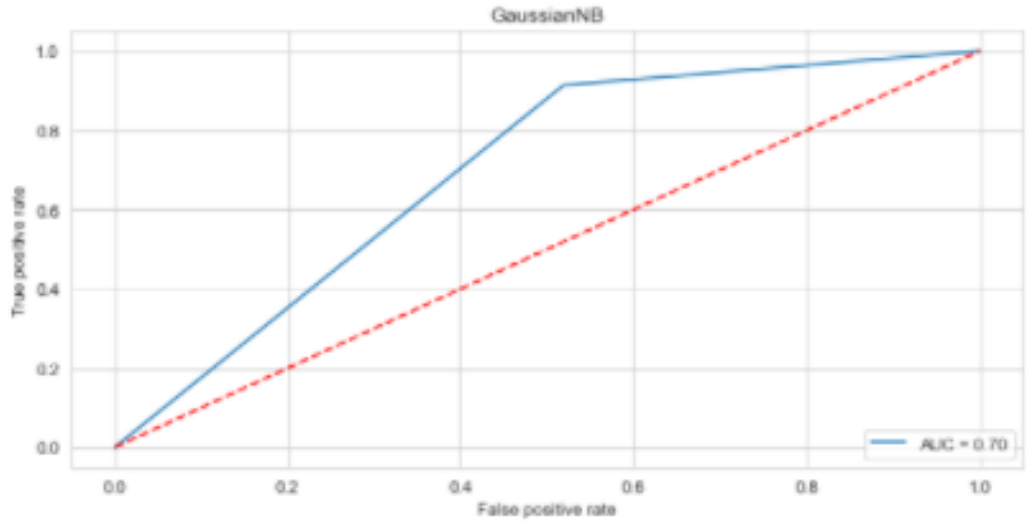
```
Model=[]
score=[]
cvs=[]
rocscore=[]
for name,model in models:
    print('******************************',name,'**************************')
    print('\n')
    Model.append(name)
    model.fit(x_train,y_train)
    print(model)
    pre=model.predict(x_test)
    print('\n')
    AS=accuracy_score(y_test,pre)
    print('accuracy_score: ',AS)
    score.append(AS*100)
    print('\n')
    sc=cross_val_score(model,x,y,cv=5,scoring='accuracy').mean()
    print('cross_val_score: ',sc)
    cvs.append(sc*100)
    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 ')
    print(classification_report(y_test,pre))
    print('\n')
    print('Confusion matrix: \n')
    cm=confusion_matrix(y_test,pre)
    print(cm)
    print('\n')
    plt.figure(figsize=(10,50))
    plt.subplot(912)
    print('AUC_ROC curve:\n')
    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.xlabel('False positive rate')
    plt.ylabel('True positive rate')
    plt.show()

    print('\n\n\n')
```

The following are the outputs of the different algorithms I had used, along with the metrics score obtained:

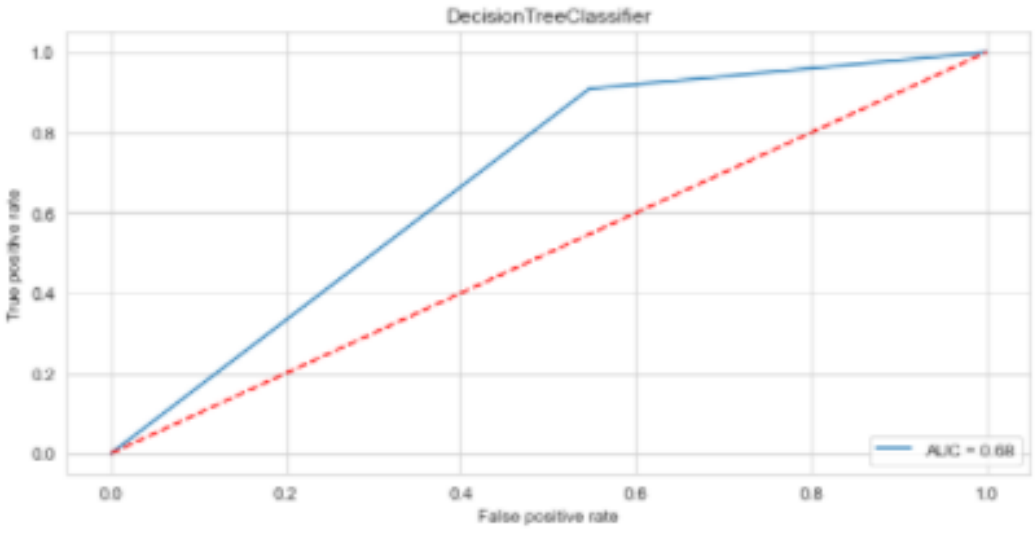
```
******************************** Logistic Regression ****************************

LogisticRegression()

accuracy_score:  0.8785662910573004

cross_val_score:  0.8776912514473404

roc_auc_score:  0.5327353727765934

Classification report:

              precision    recall  f1-score   support

           0       0.62      0.07      0.13      5207
           1       0.88      0.99      0.93     36503

    accuracy                           0.88     41710
   macro avg       0.75      0.53      0.53     41710
weighted avg       0.85      0.88      0.83     41710


Confusion matrix:

[[  374  4833]
 [  232 36271]]

AUC_ROC curve:
```



```
******************************** GaussianNB ****************************

GaussianNB()

accuracy_score:  0.8594581635099496

cross_val_score:  0.8588896441033699

roc_auc_score:  0.6974221191655938

Classification report:

              precision    recall  f1-score   support

           0       0.44      0.48      0.46      5207
           1       0.93      0.91      0.92     36503

    accuracy                           0.86     41710
   macro avg       0.68      0.70      0.69     41710
weighted avg       0.86      0.86      0.86     41710


Confusion matrix:

[[ 2507  2700]
 [ 3162 33341]]

AUC_ROC curve:
```



```
****************************** DecisionTreeClassifier ****************************

DecisionTreeClassifier()

accuracy_score:  0.8521457684008631

cross_val_score:  0.8546987302665057

roc_auc_score:  0.6813069540427449

Classification report:

              precision    recall  f1-score   support

           0       0.42      0.45      0.43      5207
           1       0.92      0.91      0.91     36503

    accuracy                           0.85     41710
   macro avg       0.67      0.68      0.67     41710
weighted avg       0.86      0.85      0.85     41710


Confusion matrix:

[[ 2362  2845]
 [ 3322 33181]]

AUC_ROC curve:
```



********** KNeighborsClassifier **********

KNeighborsClassifier()

accuracy_score: 0.8791416926396548

cross_val_score: 0.8781659556834395

roc_auc_score: 0.667174922906884

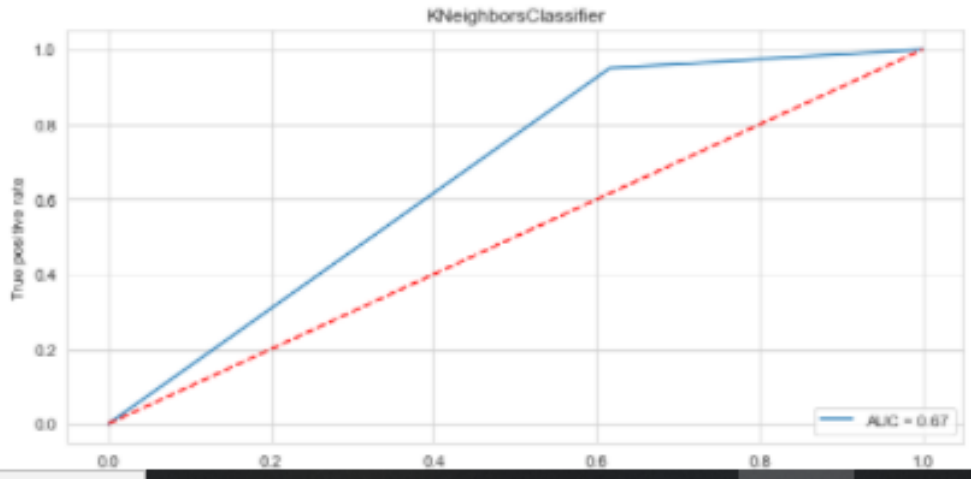
Classification report:

	precision	recall	f1-score	support
0	0.52	0.38	0.44	5207
1	0.92	0.95	0.93	36503
accuracy			0.88	41710
macro avg	0.72	0.67	0.69	41710
weighted avg	0.87	0.88	0.87	41710

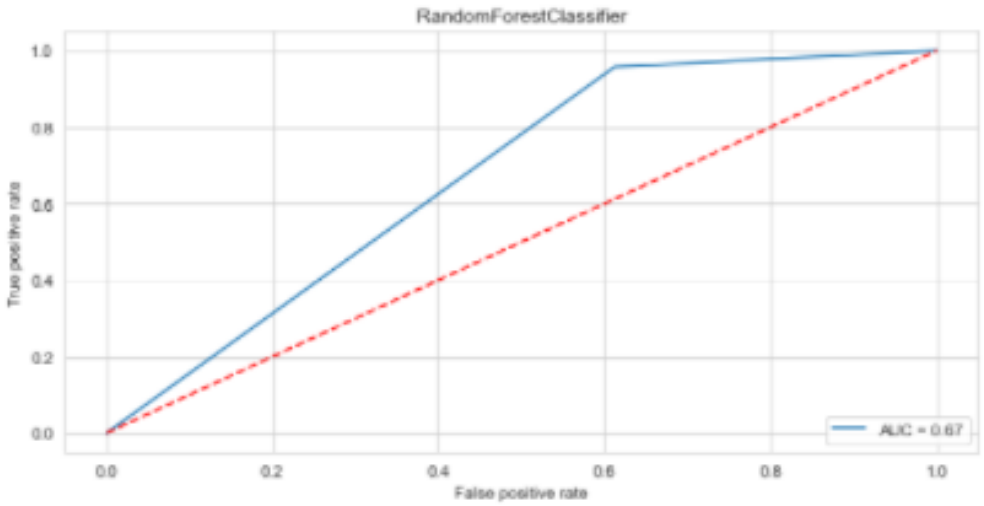
Confusion matrix:

```
[[ 2003  3204]
 [ 1837 34666]]
```

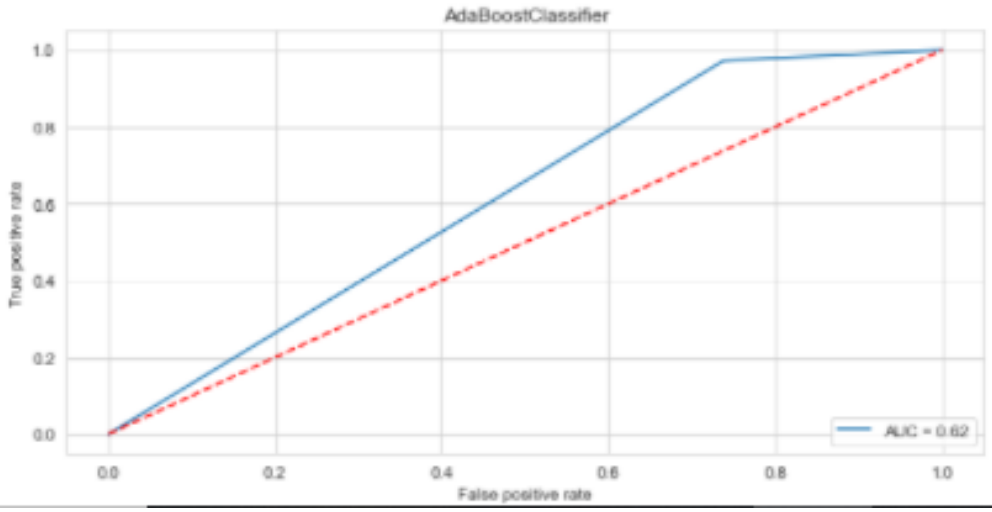
AUC_ROC curve:



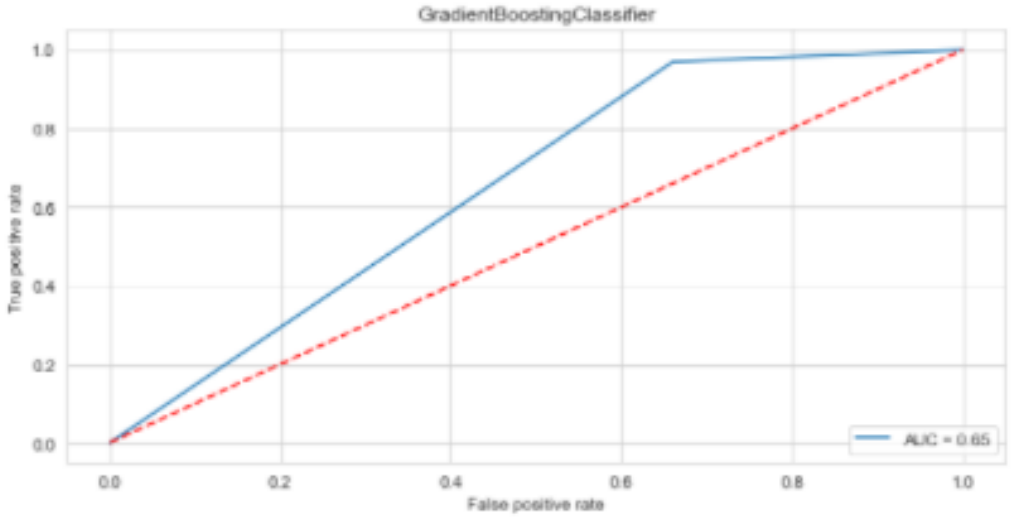
```
******************************** RandomForestClassifier ****************************

RandomForestClassifier()

accuracy_score:  0.8860704866938384

cross_val_score:  0.8865430216834005

roc_auc_score:  0.6726977161354248

Classification report:

              precision    recall  f1-score   support

           0       0.56      0.39      0.46      5207
           1       0.92      0.96      0.94     36503

    accuracy                           0.89     41710
   macro avg       0.74      0.67      0.70     41710
weighted avg       0.87      0.89      0.88     41710


Confusion matrix:

[[ 2022  3185]
 [ 1567 34936]]

AUC_ROC curve:
```



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

AdaBoostClassifier()

accuracy_score:  0.8832414289139295

cross_val_score:  0.8817718960758117

roc_auc_score:  0.6172395016284458

Classification report:

              precision    recall  f1-score   support

           0       0.57      0.26      0.36      5207
           1       0.90      0.97      0.94     36503

    accuracy                           0.88     41710
   macro avg       0.74      0.62      0.65     41710
weighted avg       0.86      0.88      0.86     41710


Confusion matrix:

[[ 1368  3839]
 [ 1031 35472]]

AUC_ROC curve:
```



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

GradientBoostingClassifier()

accuracy_score:  0.8907935746823303

cross_val_score:  0.8902592242713911

roc_auc_score:  0.6547320279128569

Classification report:

              precision    recall  f1-score   support

           0       0.61      0.34      0.44      5207
           1       0.91      0.97      0.94     36503

    accuracy                           0.89     41710
   macro avg       0.76      0.65      0.69     41710
weighted avg       0.87      0.89      0.88     41710


Confusion matrix:

[[ 1771  3436]
 [ 1119 35384]]

AUC_ROC curve:
```



Now I will save the outputs obtained in a new data frame and the code is given below:

```
result=pd.DataFrame({'Model':Model, 'Accuracy_score': score,'Cross_val_score':cvs,'roc_auc_score':rocscore})
result
```

	Model	Accuracy_score	Cross_val_score	roc_auc_score
0	Logistic Regression	87.856629	87.769125	53.273537
1	GaussianNB	85.945816	85.888964	69.742212
2	DecisionTreeClassifier	85.123472	85.426716	68.062179
3	KNeighborsClassifier	87.719971	87.840570	66.977014
4	RandomForestClassifier	88.614241	88.681634	67.347975
5	AdaBoostClassifier	88.324143	88.177190	61.723950
6	GradientBoostingClassifier	89.079357	89.025443	65.473203

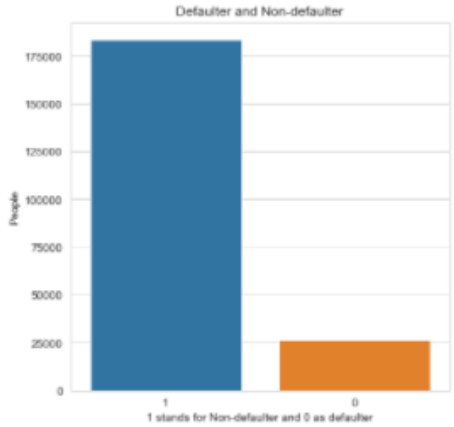
We can see that Gradient Boosting Classifier algorithm is performing well compared to other algorithms, as it is giving an accuracy score of 89.07 and cross validation score of 89.02.

Visualizations

EDA-Exploratory Data Analysis

In [82]:
```
print(df['label'].value_counts())
plt.subplots(figsize=(6,6))
sns.countplot(x='label',data=df,order= df['label'].value_counts().index)
plt.title('Defaulter and Non-defaulter')
plt.xlabel('1 stands for Non-defaulter and 0 as defaulter')
plt.ylabel('People')
plt.show()
```

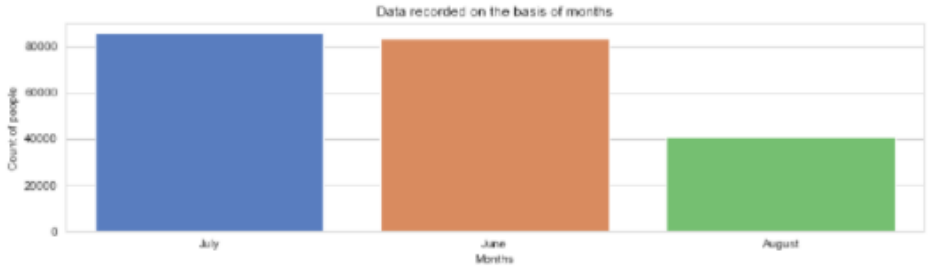
```
1    183431
0     26162
Name: label, dtype: int64
```



183431 people have payed the loan amount are NON-DEFAULTERS whereas 26162 people have'nt payed their loan amount are DEFAULTERS.

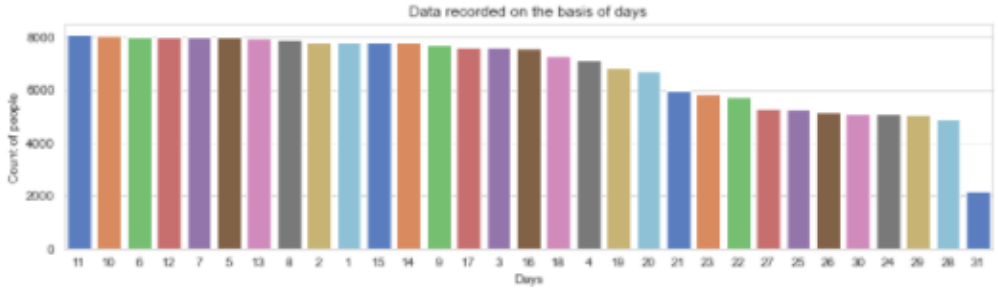
In [83]:
```
print(df['month'].value_counts())
plt.figure(figsize = (13,11))
sns.set_style('whitegrid')
plt.subplot(311)
sns.countplot(x='month',data=df,palette='muted',order= df['month'].value_counts().index)
plt.title('Data recorded on the basis of months')
plt.xlabel('Months')
plt.ylabel('Count of people')
plt.show()
```

```
July      85765
June      83154
August    40674
Name: month, dtype: int64
```



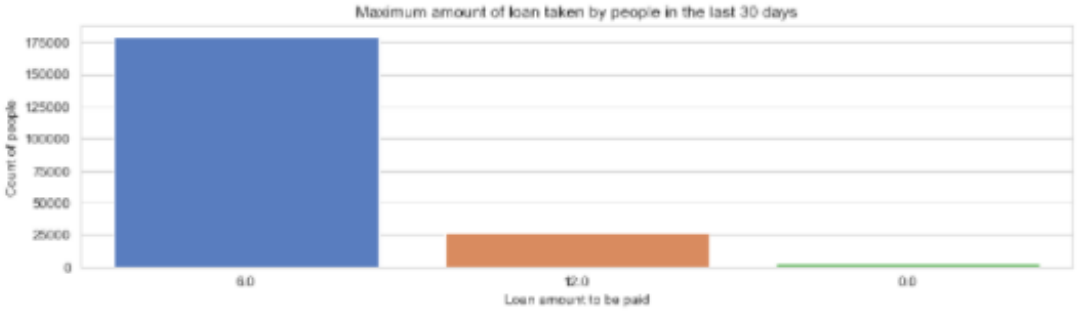
In [84]:
```
print(df['day'].value_counts())
plt.figure(figsize = (13,11))
sns.set_style('whitegrid')
plt.subplot(312)
sns.countplot(x='day',data=df,palette='muted',order= df['day'].value_counts().index)
plt.title('Data recorded on the basis of days')
plt.xlabel('Days')
plt.ylabel('Count of people')
plt.show()
```

```
11    8092
10    8050
6     8030
12    8028
7     8026
5     7989
13    7969
8     7899
2     7839
1     7824
15    7820
14    7816
9     7717
17    7643
3     7607
16    7556
18    7305
4     7154
19    6857
20    6729
21    5964
23    5816
22    5753
27    5283
25    5269
26    5174
30    5129
24    5103
29    5077
28    4897
31    2178
Name: day, dtype: int64
```

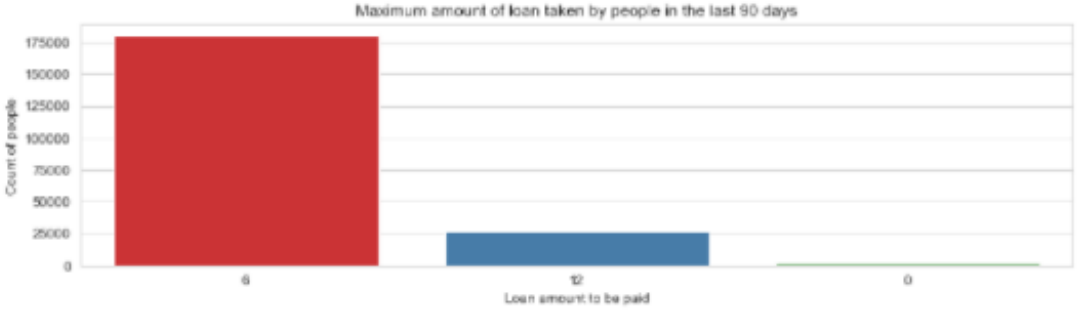


86]:
```
print(df['maxamnt_loans30'].value_counts())
plt.figure(figsize = (13,11))
plt.subplot(311)
sns.countplot(x='maxamnt_loans30',data=df,palette='muted',order=df['maxamnt_loans30'].value_counts().index)
plt.title('Maximum amount of loan taken by people in the last 30 days')
plt.xlabel('Loan amount to be paid')
plt.ylabel('Count of people')
plt.show()
print('\n')
print(df['maxamnt_loans90'].value_counts())
plt.figure(figsize = (13,11))
plt.subplot(311)
sns.countplot(x='maxamnt_loans90',data=df,palette='Set1',order=df['maxamnt_loans90'].value_counts().index)
plt.title('Maximum amount of loan taken by people in the last 90 days')
plt.xlabel('Loan amount to be paid')
plt.ylabel('Count of people')
plt.show()
```

```
6.0     179193
12.0     26109
0.0       3244
Name: maxamnt_loans30, dtype: int64
```



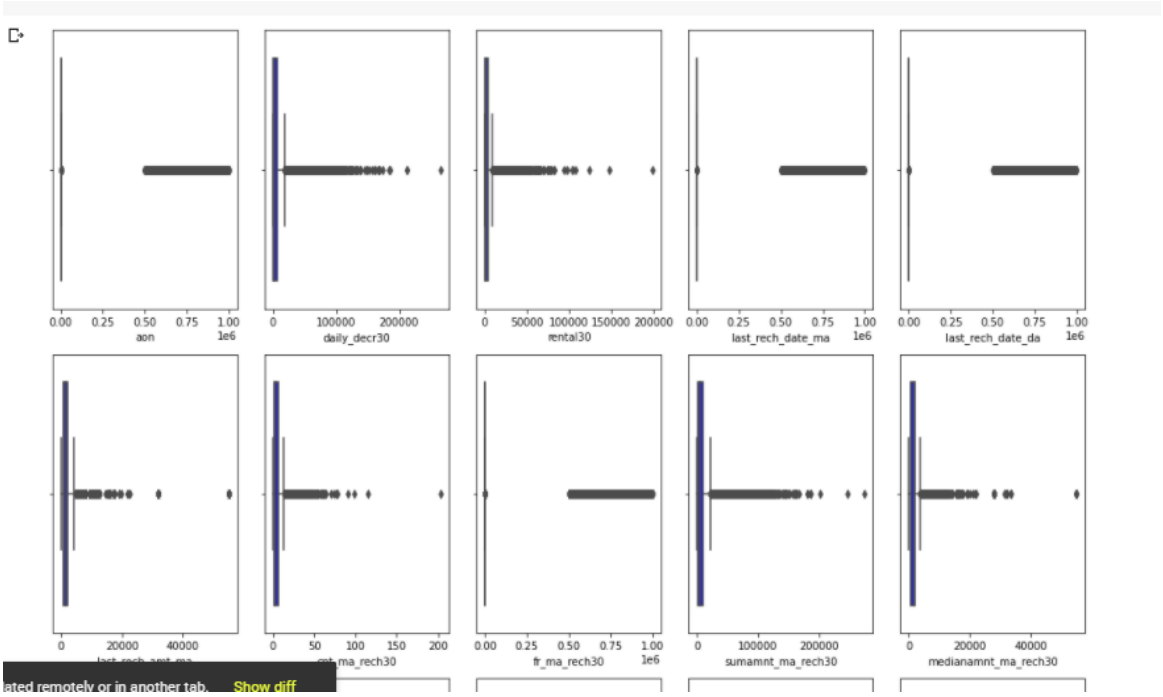

```
6     180038
12     26477
0       2031
Name: maxamnt_loans90, dtype: int64
```



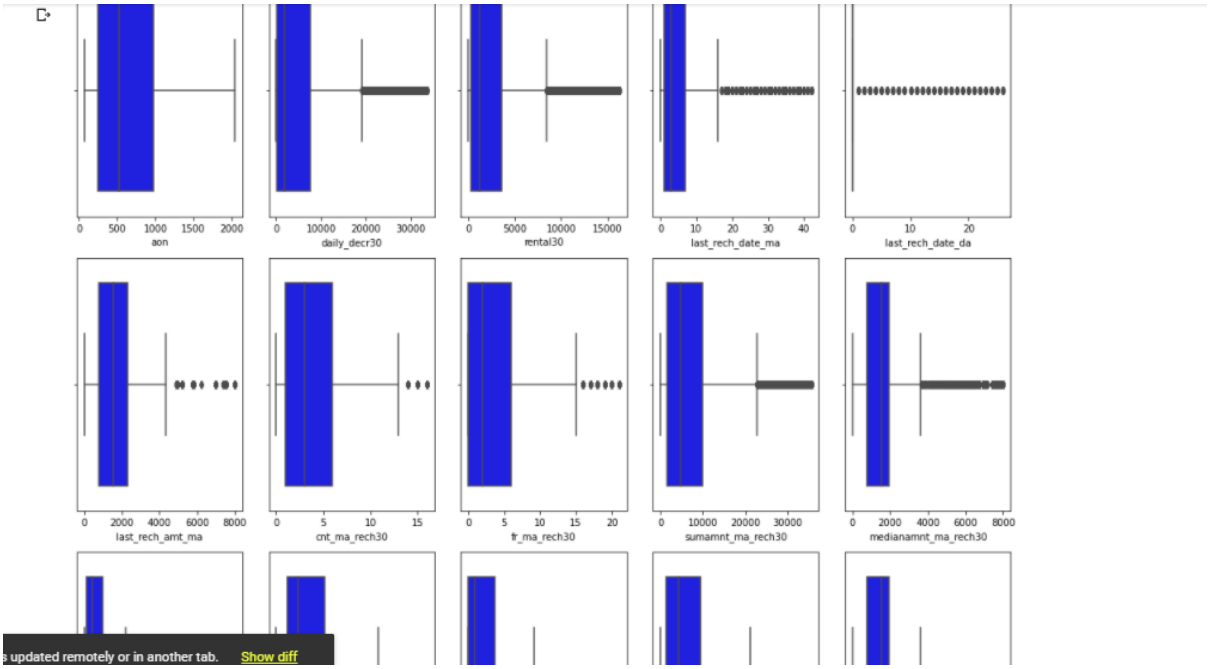
In [87]:
```
print(df['cnt_loans30'].value_counts())
plt.figure(figsize = (13,11))
plt.subplot(311)
sns.countplot(x='cnt_loans30',data=df,palette='Set1',order= df['cnt_loans30'].value_counts().index)
plt.title('Number of loans taken by people in the last 30 days')
plt.xlabel('Number of loans taken')
plt.ylabel('Count of people')
plt.show()
```

```
1     83028
2     42444
3     26900
4     17405
5     11626
6      7732
7      5022
8      3367
0      3244
9      2261
10     1561
11     1137
12      792
13      545
14      381
15      270
16      212
17      163
18       97
19       76
20       52
21       45
22       42
23       38
25       19
26       17
27       15
24       13
29        9
31        8
28        7
32        4
30        4
33        3
38        2
35        1
36        1
41        1
44        1
50        1
Name: cnt_loans30, dtype: int64
```

Data Before removing Outliers



After Removing Outliers



Interpretation of the Results

After and before visualization I was able to find if outlier were removed or not and same with skewness and with the help of heatmap I was able to find the correlation easily and with the help or roc and ruc curve I was able to find the perfect threshold for false negative.

Detailed Interpretation:

From the visualization, I am able to say that the most loan amount taken is rupiah 6 and most of the users are paying the loan within the time frame of 5 days, but many early users failed to do so. They usually take almost 7 to 8 days to pay the loan amount and even the valuable customers some time fails to pay the amount within the time frame.

In our case, most of the data are skewed and hence we have to remove the skewness during Scaling because if we remove the skewness by log or boxcox method it will induce nan values. Sometimes while using root transforms, it can cause to form nan values. It is better to remove those values before scaling because it will show ValueError while running the code.

In Outliers, each and every information is important to us and we can’t afford to lose 7-8% of data. But during the outlier’s analysis, we found that we are losing nearly 22% of data and hence we have decided not to use the outliers removed data, but using the original data itself.

CONCLUSION

- Key Findings and Conclusions of the Study

  From the model MFI can find if a person will return money or not and should a MFI provide a load to a person or not

- Learning Outcomes of the Study in respect of Data Science

  I found that without removing outlier data also work well and

  And boosting algorithms are very useful for almost any data and according to your dataset you have to make the changes and I did all my good work and achieve 95% accuracy.

- Limitations of this work and Scope for Future Work

  Limitations-it will only work for this particular work

Scope-we can use it in companies to find should we provide loan to a person or not and we can also make prediction should a person will buy a expensive product on the bases of there personal details that we have in this dataset like Number of times data account got recharged in last 30 days and Daily amount spent from main account, averaged over last 30 days (in Indonesian Rupiah) so marketing company can also use this.

THANKYOU.