



Micro-Credit Defaulter Model

Submitted by
Saurabh Upadhyay

Table of Contents

Acknowledgment.....	2
Introduction.....	3
Business Problem Framing	4
Conceptual Background of the Domain Problem	4
Review of Literature	4
Motivation for the Problem Undertaken.....	4
Analytical Problem Framing	4
Mathematical/ Analytical Modelling of the Problem	5
Data Sources and their formats.....	5
Data Pre-processing Done	6
Hardware and Software Requirements and Tools Used	6
Model/s Development and Evaluation	8
Identification of possible problem-solving approaches (methods).....	8
Testing of Identified Approaches (Algorithms)	8
Run and Evaluate selected models	8
Key Metrics for success in solving problem under consideration	19
Visualizations	19
Interpretation of the Results	25
Conclusion	26
Key Findings and Conclusions of the Study.....	26
Learning Outcomes of the Study in respect of Data Science	26

Acknowledgment

Following are the external references which I used:

www.w3school.com

www.stackoverflow.com

www.google.com

www.geeksforgeeks.org

Business Problem Framing

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

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

Basically here a one telecom industry provide the 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

Since we know that telecom sector is very much competitive so this data is very helpful in understanding the problem for the lower class people specially by providing them the facility of network and the credit amount provided by the help of MFI and MFS. From this data we get to know that what the criteria to become defaulters and successor are. And the useful information from the data to know how much amount people spend on data recharge or on the main balance recharge.

Review of Literature

From the dataset I get to know that it is a classification problem and there are two categories which are successor and the defaulters. And there are so many features which help to find it.

Motivation for the Problem Undertaken

From this project I get to know of different kind of information every recharge done by the user on which kind of recharge user is using mostly and the data service or the main balance the frequency of recharge in 30 day or 90 days. It is really quite interesting to know that each column contributed to make you close to know more about the data and in prediction you can do in many ways

Analytical Problem Framing

Mathematical/ Analytical Modelling of the Problem

The statistical figure I get to know by the data.describe() so many information the min max standard deviation the 25 percentile the 50th percentile the 75 percentile .Then by the help of correlation function I get to know the correlation of each columns with each other. From the heatmap I can visualized to see them clearly that they are positive correlated or the negative correlated the dark side is show the negative correlation among each other the lighter side represent the positive correlation among the each other. **The z-score** function computes the relative **Z-score** of the input data, relative to the sample mean and standard deviation.

Data Sources and their formats

Data I get form the Flip Robo the format was in CSV (Comma Separated Values).The number of columns and row are 209593 and columns are 36.

The data descriptions are as follow:-

Label	Flag indicating whether the user paid back the credit amount within 5 days of issuing the loan{ 1:success, 0:failure}
Msisdn	mobile number of user
Aon	age on cellular network in days
daily_decr30	Daily amount spent from main account, averaged over last 30 days (in Indonesian Rupiah)
daily_decr90	Daily amount spent from main account, averaged over last 90 days (in Indonesian Rupiah)
rental30	Average main account balance over last 30 days
rental90	Average main account balance over last 90 days
last_rech_date_ma	Number of days till last recharge of main account
last_rech_date_da	Number of days till last recharge of data account
last_rech_amt_ma	Amount of last recharge of main account (in Indonesian Rupiah)
cnt_ma_rech30	Number of times main account got recharged in last 30 days
fr_ma_rech30	Frequency of main account recharged in last 30 days
sumamnt_ma_rech30	Total amount of recharge in main account over last 30 days (in Indonesian Rupiah)
medianamnt_ma_rech30	Median of amount of recharges done in main account over last 30 days at user level (in Indonesian Rupiah)
medianmarechprebal30	Median of main account balance just before recharge in last 30 days at user level (in Indonesian Rupiah)
cnt_ma_rech90	Number of times main account got recharged in last 90 days
fr_ma_rech90	Frequency of main account recharged in last 90 days
sumamnt_ma_rech90	Total amount of recharge in main account over last 90 days (in Indonesian Rupee)
medianamnt_ma_rech90	Median of amount of recharges done in main account over last 90 days at user level (in Indonesian Rupee)
medianmarechprebal90	Median of main account balance just before recharge in last 90 days at user level (in Indonesian Rupee)

cnt_da_rech30	Number of times data account got recharged in last 30 days
fr_da_rech30	Frequency of data account recharged in last 30 days
cnt_da_rech90	Number of times data account got recharged in last 90 days
fr_da_rech90	Frequency of data account recharged in last 90 days
cnt_loans30	Number of loans taken by user in last 30 days
amnt_loans30	Total amount of loans taken by user in last 30 days
maxamnt_loans30	maximum amount of loan taken by the user in last 30 days
medianamnt_loans30	Median of amounts of loan taken by the user in last 30 days
cnt_loans90	Number of loans taken by user in last 90 days
amnt_loans90	Total amount of loans taken by user in last 90 days
maxamnt_loans90	maximum amount of loan taken by the user in last 90 days
medianamnt_loans90	Median of amounts of loan taken by the user in last 90 days
payback30	Average payback time in days over last 30 days
payback90	Average payback time in days over last 90 days
Pcircle	telecom circle
Pdate	Date

Data Pre-processing Done

There were no null value was present in the dataset but there are some outliers which also get too removed, approximately 48128 outliers get removed from the data. After that categorical are change to integer or float with the help of **LabelEncoder**. Then I used updated data for the correlation for splitting it into x and y with the help of standard scalar it will transform the data in such way that its distribution will have a mean value 0 and standard deviation of 1. In case of multivariate data, this is done feature-wise (in other words independently for each column of the data).

Hardware and Software Requirements and Tools Used

Hardware – Laptop

Software - anaconda jupyter notebook

Libraries- numpy, pandas, seaborn, matplotlib.pyplot, warning

From sklearn.preprocessing import StandardScaler

As these columns are different in **scale**, they are **standardized** to have common **scale** while building machine learning model. This is useful when you want to compare data that correspond to different units.

from sklearn.preprocessing import Label Encoder

Label Encoder and One Hot Encoder. These two encoders are parts of the SciKit Learn library in Python, and they are used to convert categorical data, or text data, into numbers, which our predictive models can better understand.

from sklearn.model_selection import train_test_split, cross_val_score

Train_test_split is a function in Sklearn model selection for splitting data arrays into two subsets: for training data and for testing data. With this function, you don't need to divide the dataset manually. By default, Sklearn train_test_split will make random partitions for the two subsets.

The algorithm is trained and tested K times, each time a new set is used as testing set while remaining sets are used for training. Finally, the result of the K-Fold Cross-Validation is the average of the results obtained on each set.

from sklearn.neighbors import KNeighborsClassifier

K Nearest Neighbor(KNN) is a very simple, easy to understand, versatile and one of the topmost machine learning algorithms. KNN used in the variety of applications such as finance, healthcare, political science, handwriting detection, image recognition and video recognition

from sklearn.linear_model import LogisticRegression

The library sklearn can be used to perform logistic regression in a few lines as shown using the LogisticRegression class. It also supports multiple features. It requires the input values to be in a specific format hence they have been reshaped before training using the fit method.

from sklearn.tree import DecisionTreeClassifier

Decision Tree is a white box type of ML algorithm. It shares internal decision-making logic, which is not available in the black box type of algorithms such as Neural Network. Its training time is faster compared to the neural network algorithm. The time complexity of decision trees is a function of the number of records and number of attributes in the given data. The decision tree is a distribution-free or non-parametric method, which does not depend upon probability distribution assumptions. Decision trees can handle high dimensional data with good accuracy

from sklearn.naive_bayes import GaussianNB

Naive Bayes are a group of supervised machine learning classification algorithms based on the Bayes theorem. It is a simple classification technique, but has high functionality.

Model/s Development and Evaluation

Identification of possible problem-solving approaches (methods)

Descriptive statistics are used to describe the basic features of the data in a study which are mean count max standard deviations 25% , 75% , 50 % it all help me to understand the data in terms of statistically for the problem solving

Testing of Identified Approaches (Algorithms)

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

- KNN=KNeighborsClassifier(n_neighbors=6)
- LR=LogisticRegression()
- DT=DecisionTreeClassifier(random_state=6)
- GNB=GaussianNB()

I applied all these algorithms in the dataset.

Run and Evaluate selected models

```
jupyter PROJECT - I FLIP ROBO Last Checkpoint: an hour ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help
(113025,) (48440,)

In [64]: 1 from sklearn.neighbors import KNeighborsClassifier
2 from sklearn.linear_model import LogisticRegression
3 from sklearn.tree import DecisionTreeClassifier
4 from sklearn.naive_bayes import GaussianNB

In [65]: 1 KNN=KNeighborsClassifier(n_neighbors=6)
2 LR=LogisticRegression()
3 DT=DecisionTreeClassifier(random_state=6)
4 GNB=GaussianNB()

In [66]: 1 models = []
2 models.append(('KNeighborsClassifier',KNN))
3 models.append(('LogisticRegression',LR))
4 models.append(('DecisionTreeClassifier',DT))
5 models.append(('GaussianNB',GNB))

In [67]: 1 from sklearn.metrics import classification_report,confusion_matrix,accuracy_score,roc_curve,auc

In [68]: 1 Model = []
2 score = []
3 cvs=[]
4 rocscore=[]
5 for name,model in models:
6     print('*****',name,'*****')
7     print('\n')
8     Model.append(name)
9     model.fit(x_train,y_train)
10    print(model)
11    pre=model.predict(x_test)
12    print('\n')
13    AS=accuracy_score(y_test,pre)
14    print('Accuracy_Score = ',AS)
15    score.append(AS*100)
16    print('\n')
17    sc = cross_val_score(model, x, y, cv=10, scoring='accuracy').mean()
18    print('Cross_Val Score = ',sc)
```



```
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3
Code
14 print('Accuracy_Score = ',AS)
15 score.append(AS*100)
16 print('\n')
17 sc = cross_val_score(model, X, y, cv=10, scoring='accuracy').mean()
18 print('Cross_Val_Score = ',sc)
19 cvs.append(sc*100)
20 print('\n')
21 false_positive_rate, true_positive_rate, thresholds = roc_curve(y_test,pre)
22 roc_auc = auc(false_positive_rate, true_positive_rate)
23 print('roc_auc_score = ',roc_auc)
24 rocscore.append(roc_auc*100)
25 print('\n')
26 print('Classification_report\n',classification_report(y_test,pre))
27 print('\n')
28 cm=confusion_matrix(y_test,pre)
29 print(cm)
30 print('\n')
31 plt.figure(figsize=(10,40))
32 plt.subplot(911)
33 plt.title(name)
34 print(sns.heatmap(cm,annot=True))
35 plt.subplot(912)
36 plt.title(name)
37 plt.plot(false_positive_rate, true_positive_rate, label='AUC = %0.2f'% roc_auc)
38 plt.legend(loc='lower right')
39 plt.ylabel('True Positive Rate')
40 plt.xlabel('False Positive Rate')
41 print('\n\n')
42

***** KNeighborsClassifier *****

KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                    metric_params=None, n_jobs=None, n_neighbors=6, p=2,
                    weights='uniform')

Accuracy_Score = 0.9672997522708505
```

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

KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',

metric_params=None, n_jobs=None, n_neighbors=6, p=2,

weights='uniform')

Accuracy_Score = 0.9672997522708505

Cross_Val_Score = 0.969002580351303

roc_auc_score = 0.8960001198465963

Classification_report

	<i>precision</i>	<i>recall</i>	<i>f1-score</i>	<i>support</i>
<i>0</i>	<i>0.96</i>	<i>0.80</i>	<i>0.87</i>	<i>6720</i>
<i>1</i>	<i>0.97</i>	<i>0.99</i>	<i>0.98</i>	<i>41720</i>
<i>accuracy</i>		<i>0.97</i>	<i>48440</i>	
<i>macro avg</i>	<i>0.96</i>	<i>0.90</i>	<i>0.93</i>	<i>48440</i>
<i>weighted avg</i>	<i>0.97</i>	<i>0.97</i>	<i>0.97</i>	<i>48440</i>

[[5358 1362]

[222 41498]]

AxesSubplot(0.125,0.808774;0.62x0.0712264)

****** LogisticRegression ******

LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
intercept_scaling=1, l1_ratio=None, max_iter=100,
multi_class='auto', n_jobs=None, penalty='l2',

```
random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
warm_start=False)
```

Accuracy_Score = 0.94023534269199

Cross_Val_Score = 0.9400489319409511

roc_auc_score = 0.8276680848513902

Classification_report

	precision	recall	f1-score	support
0	0.87	0.67	0.76	6720
1	0.95	0.98	0.97	41720
accuracy		0.94		48440
macro avg	0.91	0.83	0.86	48440
weighted avg	0.94	0.94	0.94	48440

[[4515 2205]

[690 41030]]

AxesSubplot(0.125,0.808774;0.62x0.0712264)

****** DecisionTreeClassifier ******

*DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
max_depth=None, max_features=None, max_leaf_nodes=None,
min_impurity_decrease=0.0, min_impurity_split=None,
min_samples_leaf=1, min_samples_split=2,
min_weight_fraction_leaf=0.0, presort='deprecated',
random_state=6, splitter='best')*

Accuracy_Score = 0.9559661436829067

Cross_Val_Score = 0.958009483368885

roc_auc_score = 0.9066484899328859

Classification_report

precision recall f1-score support

0 0.84 0.84 0.84 6720

1 0.97 0.97 0.97 41720

accuracy 0.96 48440

macro avg 0.91 0.91 0.91 48440

weighted avg 0.96 0.96 0.96 48440

[[5634 1086]

[1047 40673]]

AxesSubplot(0.125,0.808774;0.62x0.0712264)

***** GaussianNB *****

GaussianNB(priors=None, var_smoothing=1e-09)

Accuracy_Score = 0.8293146160198184

Cross_Val_Score = 0.8319636247034256

roc_auc_score = 0.8022870725471396

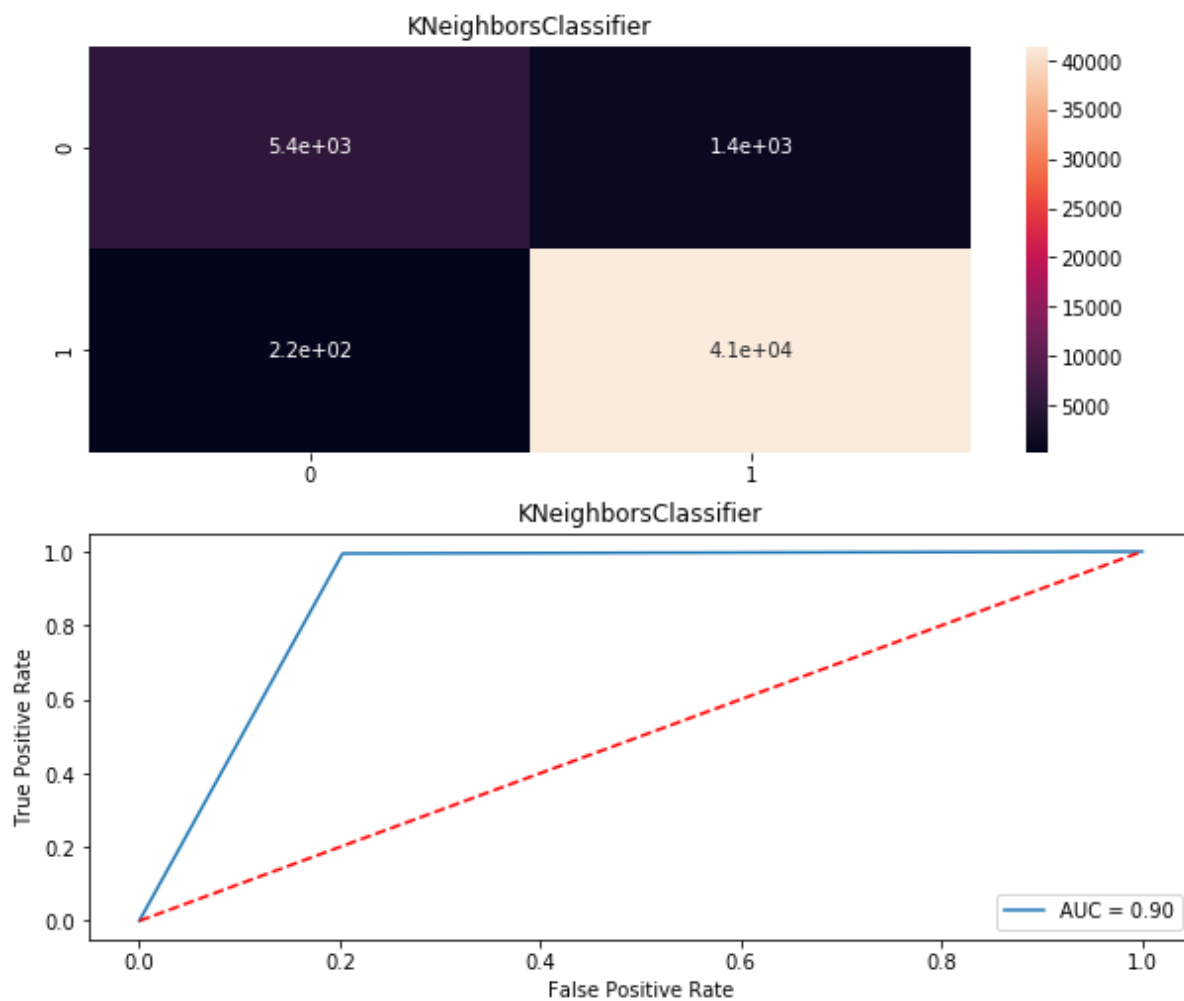
Classification_report

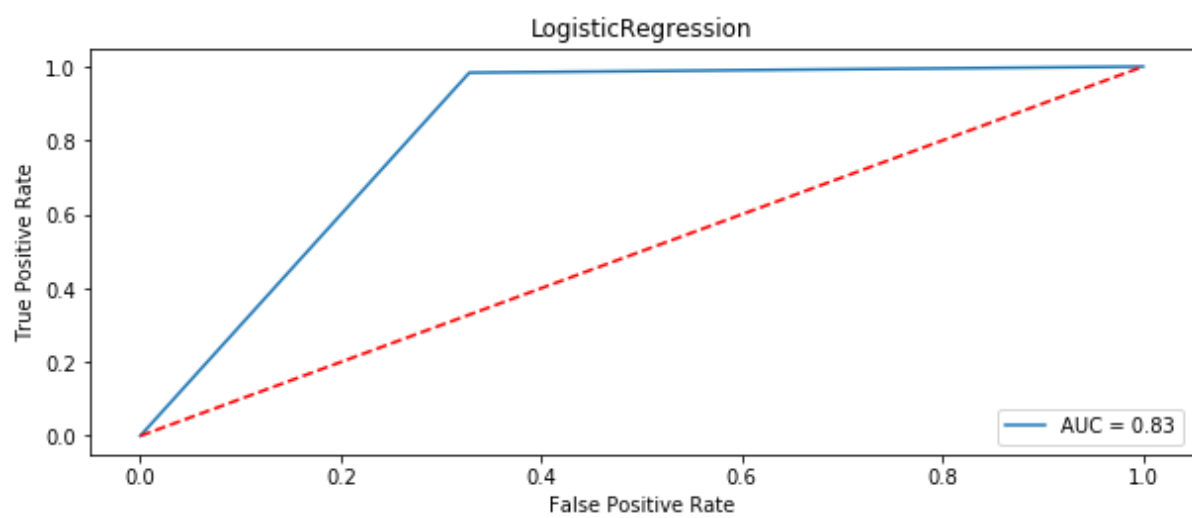
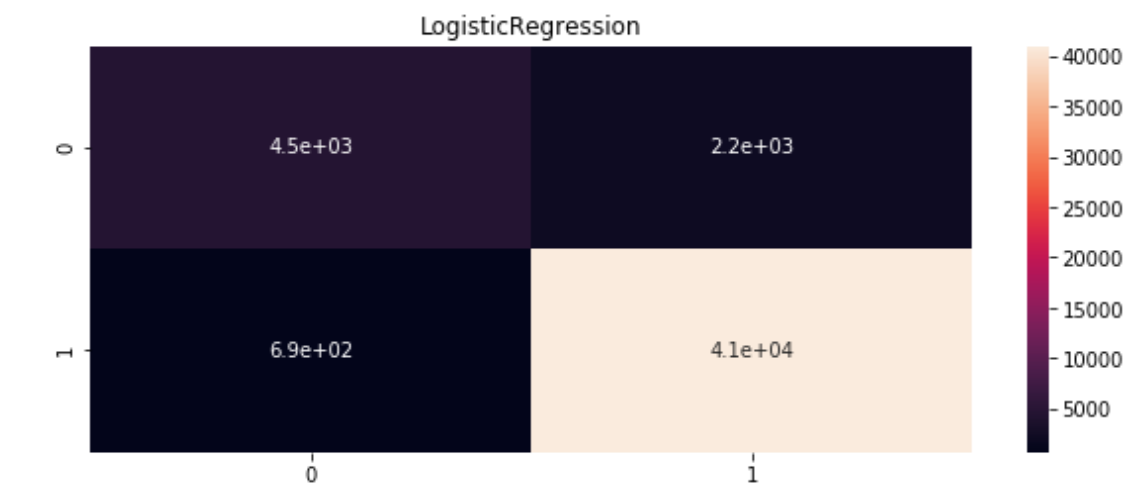
	<i>precision</i>	<i>recall</i>	<i>f1-score</i>	<i>support</i>
<i>0</i>	<i>0.43</i>	<i>0.76</i>	<i>0.55</i>	<i>6720</i>
<i>1</i>	<i>0.96</i>	<i>0.84</i>	<i>0.89</i>	<i>41720</i>
<i>accuracy</i>		<i>0.83</i>	<i>48440</i>	
<i>macro avg</i>	<i>0.70</i>	<i>0.80</i>	<i>0.72</i>	<i>48440</i>
<i>weighted avg</i>	<i>0.88</i>	<i>0.83</i>	<i>0.85</i>	<i>48440</i>

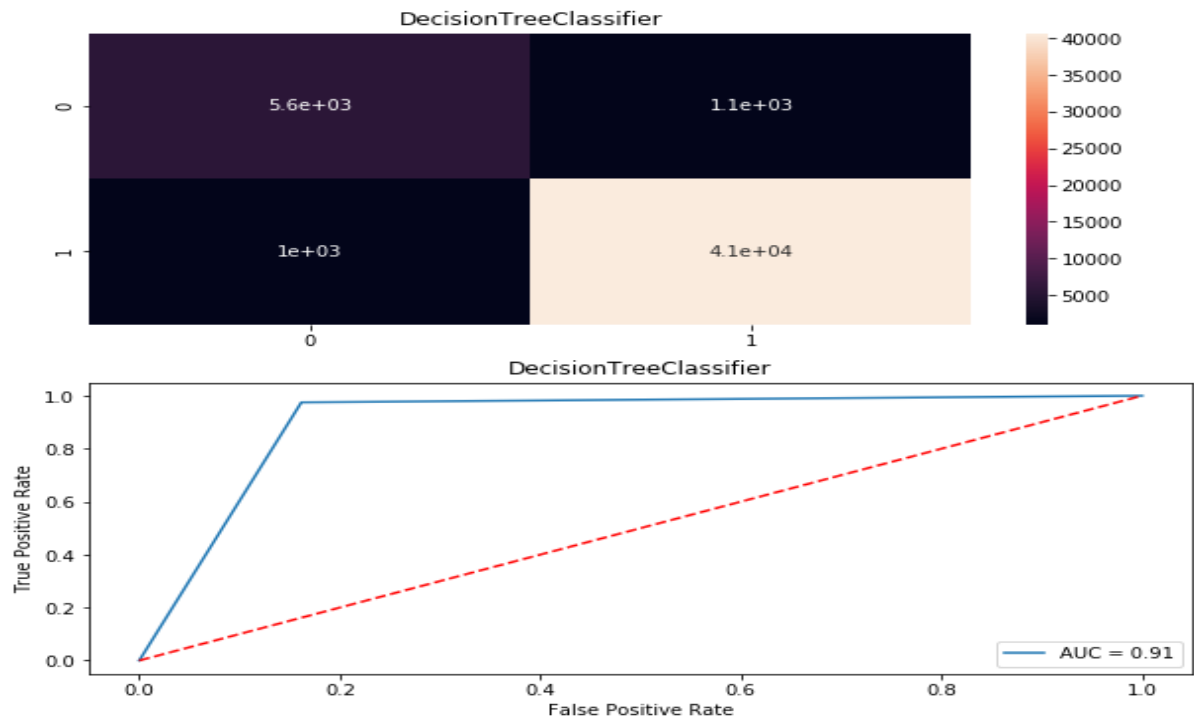
[[5140 1580]

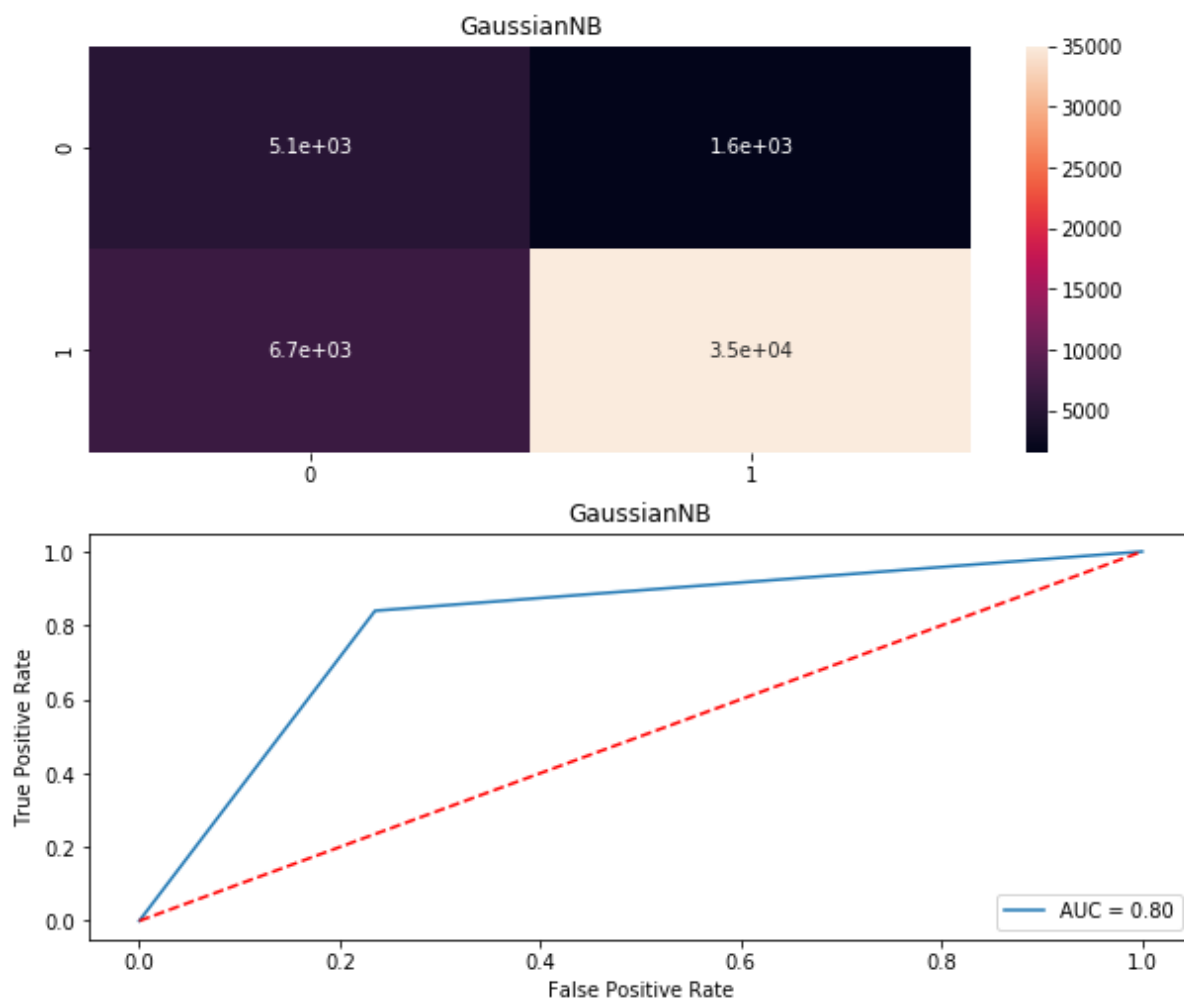
[6688 35032]]

AxesSubplot(0.125,0.808774;0.62x0.0712264)









```

+ 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656 657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696 697 698 699 700 701 702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720 721 722 723 724 725 726 727 728 729 730 731 732 733 734 735 736 737 738 739 740 741 742 743 744 745 746 747 748 749 750 751 752 753 754 755 756 757 758 759 760 761 762 763 764 765 766 767 768 769 770 771 772 773 774 775 776 777 778 779 780 781 782 783 784 785 786 787 788 789 790 791 792 793 794 795 796 797 798 799 800 801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817 818 819 820 821 822 823 824 825 826 827 828 829 830 831 832 833 834 835 836 837 838 839 840 841 842 843 844 845 846 847 848 849 850 851 852 853 854 855 856 857 858 859 860 861 862 863 864 865 866 867 868 869 870 871 872 873 874 875 876 877 878 879 880 881 882 883 884 885 886 887 888 889 890 891 892 893 894 895 896 897 898 899 900 901 902 903 904 905 906 907 908 909 910 911 912 913 914 915 916 917 918 919 920 921 922 923 924 925 926 927 928 929 930 931 932 933 934 935 936 937 938 939 940 941 942 943 944 945 946 947 948 949 950 951 952 953 954 955 956 957 958 959 960 961 962 963 964 965 966 967 968 969 970 971 972 973 974 975 976 977 978 979 980 981 982 983 984 985 986 987 988 989 990 991 992 993 994 995 996 997 998 999 1000

```

Out[69]:

	Model	Accuracy_score	Cross_val_score	Roc_auc_curve
0	KNeighborsClassifier	96.729975	96.900258	89.600012
1	LogisticRegression	94.023534	94.004893	82.766808
2	DecisionTreeClassifier	95.596614	95.800948	90.664849
3	GaussianNB	82.931462	83.196362	80.228707

```

1 Since from the above table I see that KNeighborsClassifier,LogisticRegression,DecisionTreeClassifier and GaussianNB all
2 are performing very well.
3 I choose KNeighborsClassifier as my final model because it perform well on the dataset
4 Accuracy_score = 96.75
5 Cross_val_score = 96.90
6 Roc_auc_curve = 89.63

In [70]: 1 from sklearn.externals import joblib
2         #save the model as a pickel in a file
3

In [71]: 1 joblib.dump(KNN,'Micro-Credit Defaulter Model.pkl')
2

Out[71]: ['Micro-Credit Defaulter Model.pkl']

```

Key Metrics for success in solving problem under consideration

Precision: can be seen as a measure of quality, **higher precision** means that an algorithm returns more relevant results than irrelevant ones

Recall is used as a measure of quantity and high recall means that an algorithm returns most of the relevant results.

Accuracy score is used when the True Positives and True negatives are more important. **Accuracy** can be used when the class distribution is similar

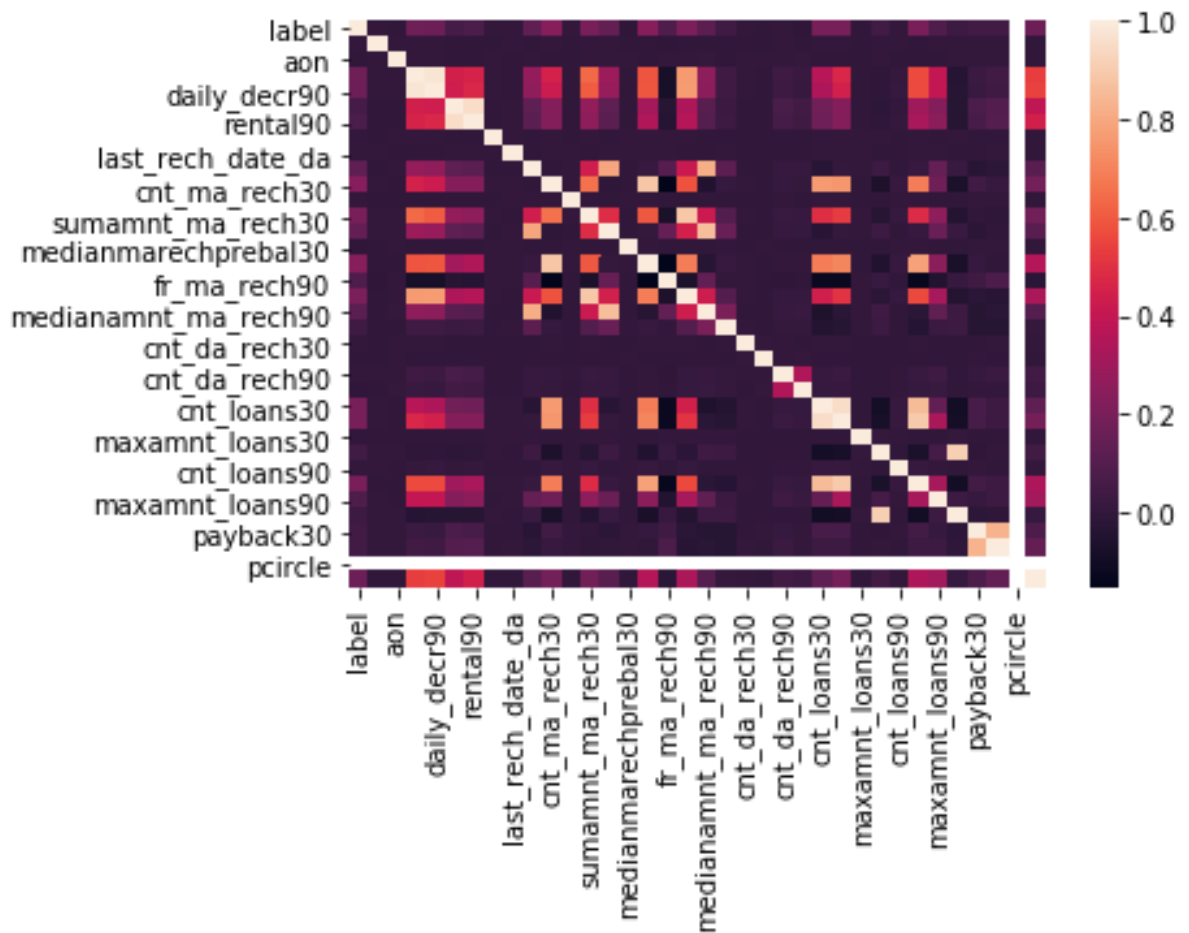
F1-score is used when the False Negatives and False Positives are crucial. While F1-score is a better metric when there are imbalanced classes.

Cross_val_score :- To run **cross-validation** on multiple metrics and also to return train **scores**, fit times and **score** times. Get predictions from each split of **cross-validation** for diagnostic purposes. Make a scorer from a performance metric or loss function.

roc_auc_score :- ROC curve. It is a plot of the false positive rate (x-axis) versus the true positive rate (y-axis) for a number of different candidate threshold values between 0.0 and 1.0

Visualizations

sns.heatmap(dfcor) From this code I get the below picture which represent the correlation among different columns since darker side represents the negative correlation and the higher side represent the positive correlation.



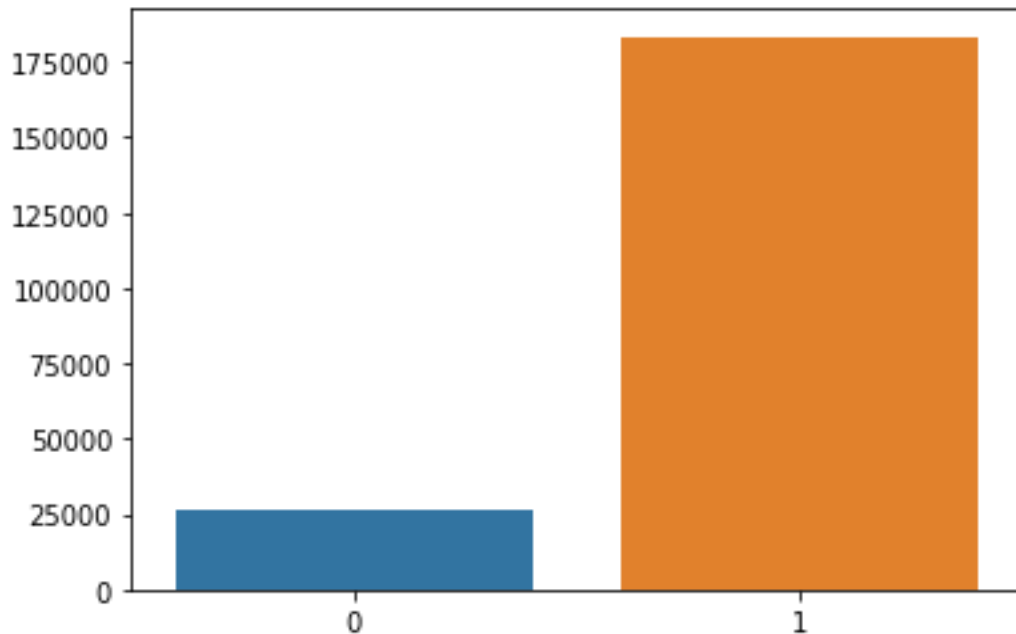
Code:-

```
y=df['label'].value_counts()
```

```
sns.barplot(y.index, y.values)
```

through the above code I get the graphical representation from it

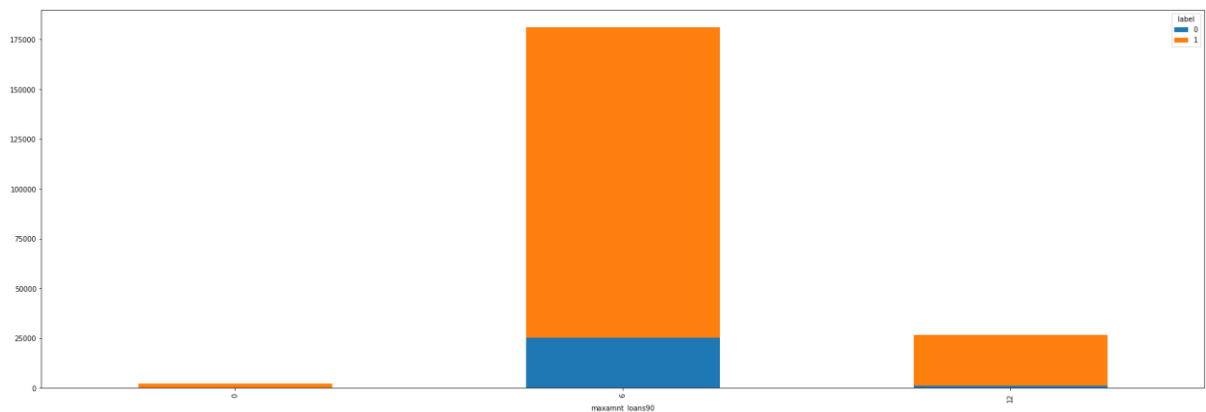
pay back credit amount of successor are 175000 and failure to payback credit amount are 250000



Code-

```
df.groupby(['maxamnt_loans90','label']).size().unstack().plot(kind='bar',stacked=True,figsize=(30,10))
```

The maximum amount of w=loan was payed by the successors



```
pd.crosstab(df.label,df.maxamnt_loans90).plot(kind='bar',figsize=(15,6),color=['#1CA53B','#AA1111'])
```

```
plt.title('Frequency of label who take maximum amount of loan')
```

```
plt.xlabel('label(0=defaulter, 1=successor)')
```

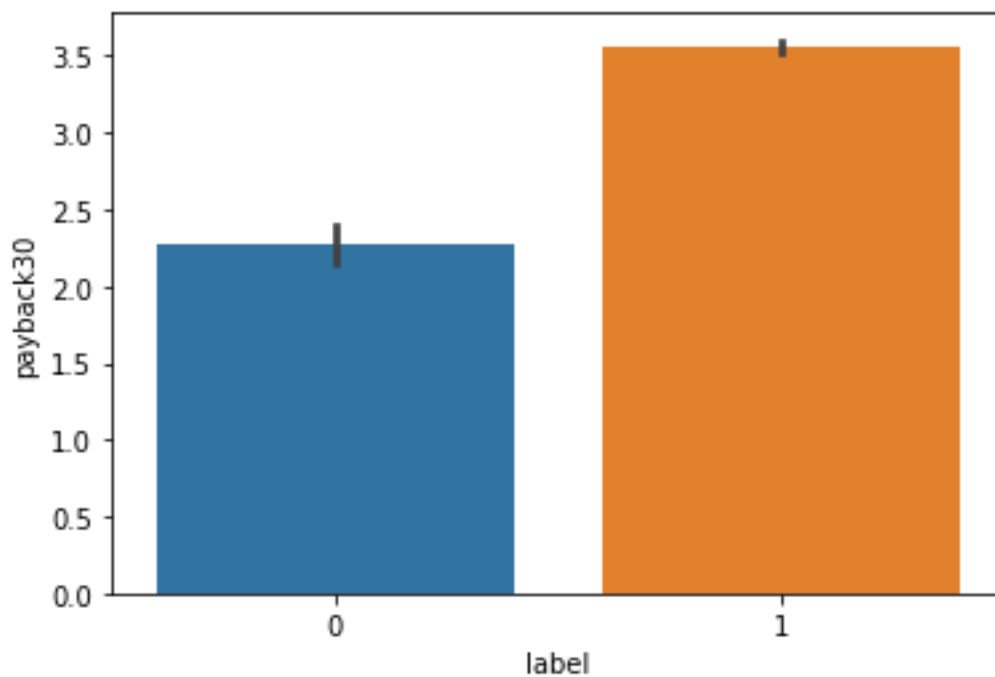
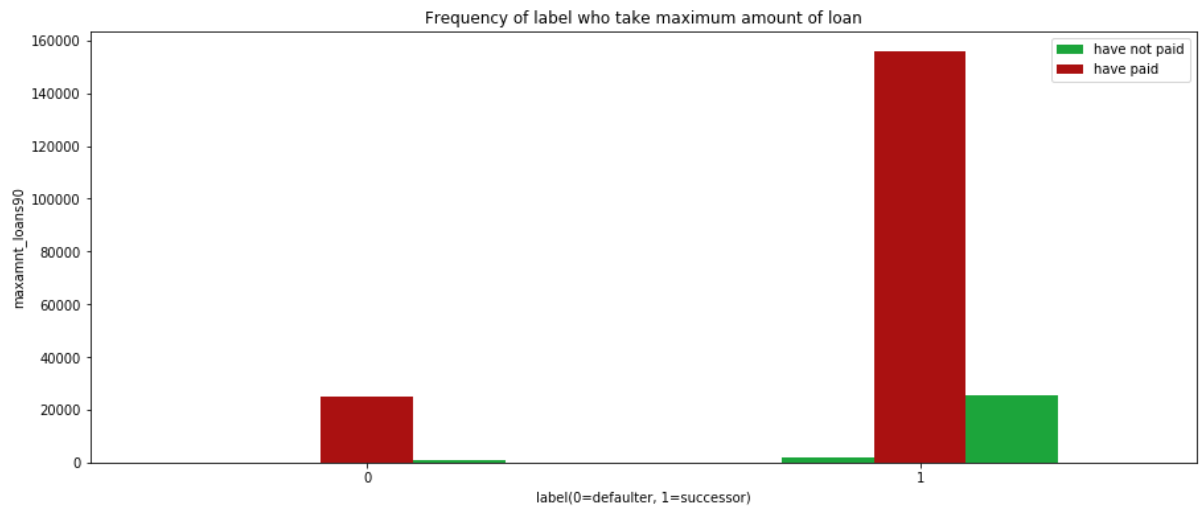
```
plt.xticks(rotation=0)
```

```
plt.legend(['have not paid', 'have paid'])
```

```
plt.ylabel('maxamnt_loans90')
```

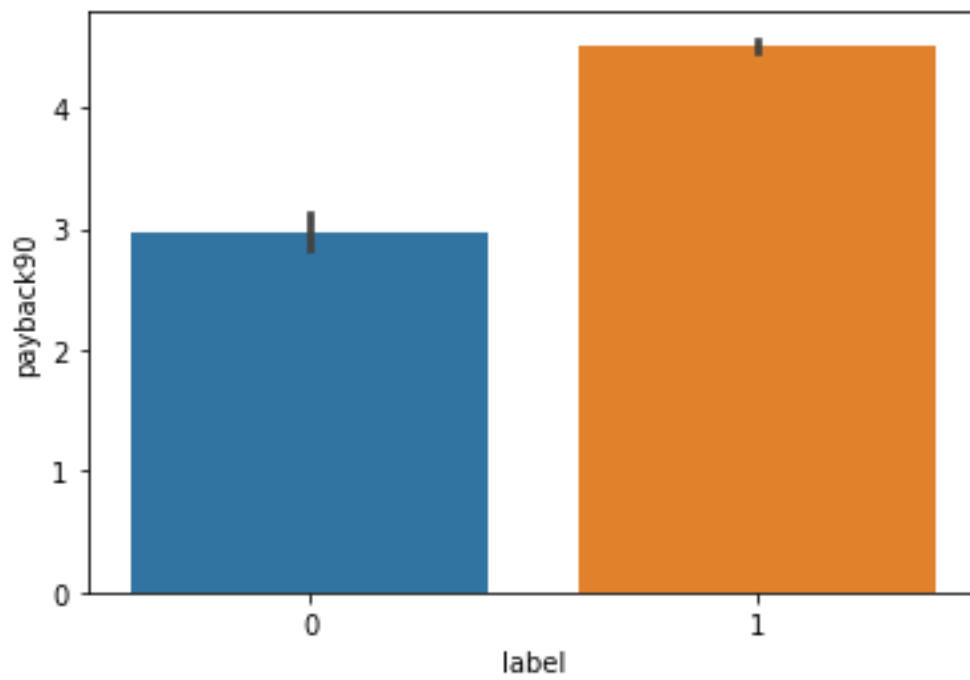
```
plt.show()
```

Maximum amount of loan taken by the user in last 90 days and who have paid is the successor which range is high as compare to the person who have not paid called as defaulter.



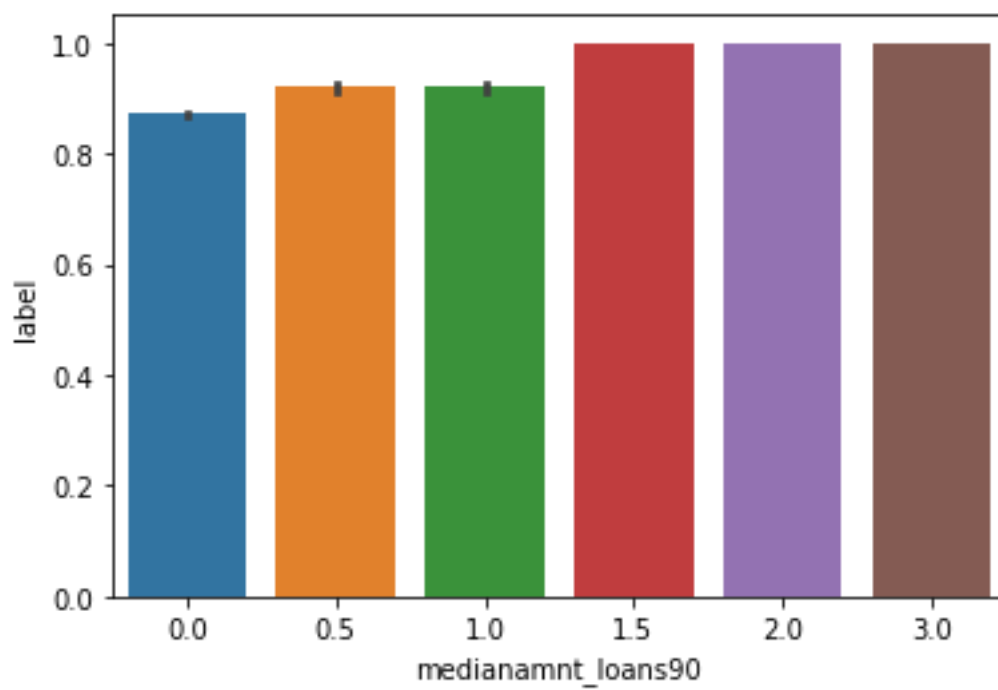
```
sns.barplot(x=df['label'],y=df['payback30'],data=df)
```

```
plt.show()
```

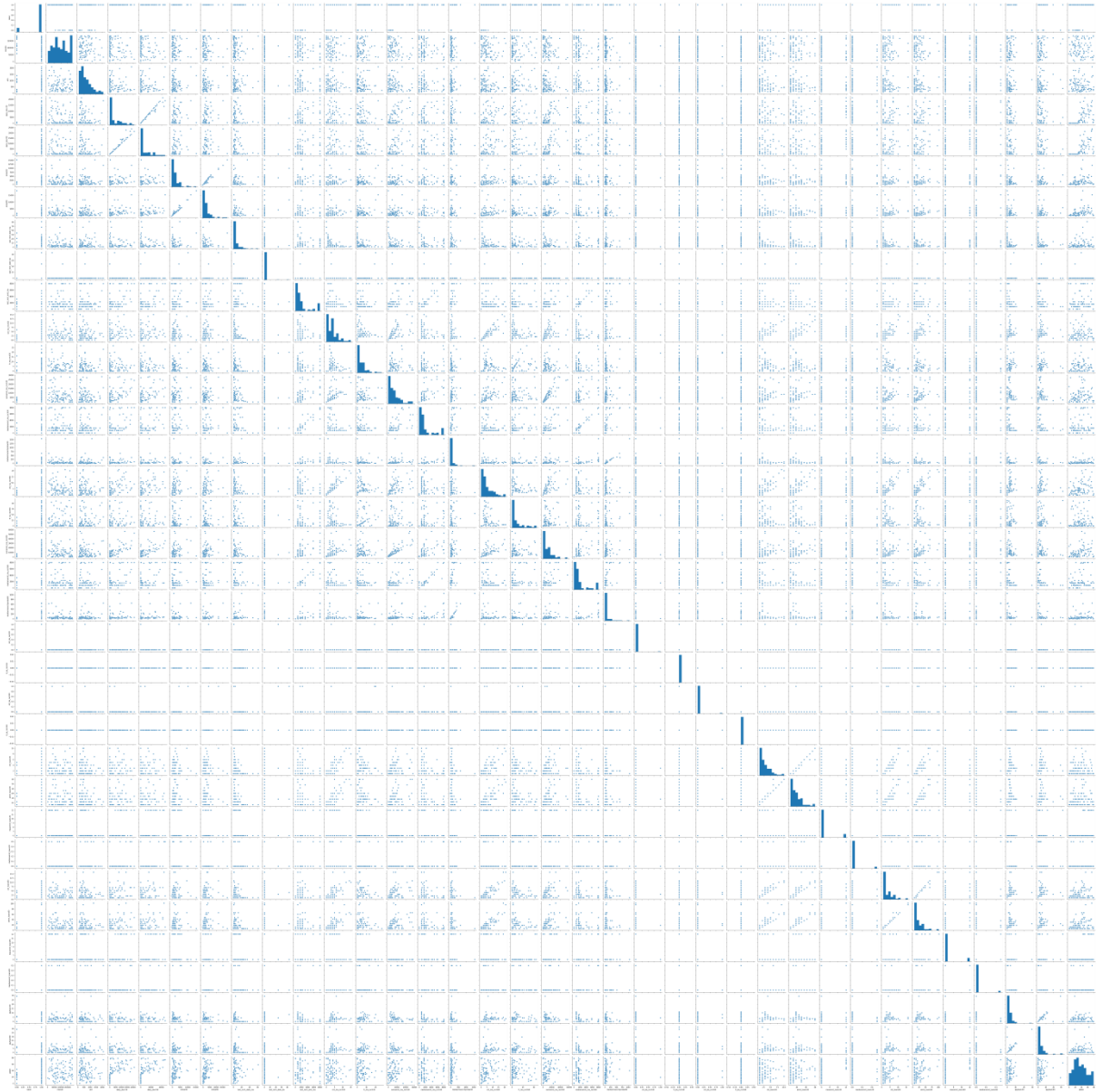


```
sns.barplot(x=df['label'],y=df['payback90'],data=df)
```

```
plt.show()
```



```
sns.barplot(x='medianamnt_loans90',y='label',data=df)
plt.show()
```



```
sns.pairplot(df_new.sample(100))
```

With the help of above code I will get the above graphical representation.

Interpretation of the Results

.

In the Pre-processing it is imported by the Label Encoder the library is “**from sklearn.preprocessing import Label Encoder**”.

Label Encoder can be used in the following:-

- Normalize labels.
- It transform data to encode the target values i.e. y target and not the input x.

Following are the syntax of Label Encoder which we use in the data set of label:

```
from sklearn.preprocessing import LabelEncoder  
le=LabelEncoder()  
y=le.fit_transform(y)  
y
```

StandardScaler - The idea behind the StandardScaler method is that it will transform our data in such a way that its distribution will have a mean value of 0 and the standard deviation of 1.

The library which is used by for StandardScaler is following –

From sklearn.preprocessing import StandardScaler

The syntax which I used in the data is following –

```
from sklearn.preprocessing import StandardScaler  
sc=StandardScaler()  
x=sc.fit_transform(df_new)  
x=pd.DataFrame(x,columns=df_new.columns)
```

Conclusion

Key Findings and Conclusions of the Study

The key findings:

From this dataset I get to know that each feature play a very import role to understand the data. Data format plays a very important role in the visualization and Appling the models and algorithms.

Learning Outcomes of the Study in respect of Data Science

My learnings :- the power of visualization is helpful for the understanding of data into the graphical representation its help me to understand that what data is trying to say, Data cleaning is one of the most important step to remove missing value or null value fill it by mean median or by mode or by 0.

Various algorithms I used in this dataset and to get out best result and save that model.The best algorithm is KNeighboursClassifier.

The challenges I faced while working on this project basically I was trying to face issue in running the SVC algorithm and during the pair plot also because to huge rows and columns I face the issue to run it since it take more than hour to run I overcome by taking the help of Google I am able to run the sample 100 from the huge data.