|  |  |
| --- | --- |
| **NAME** | Vishnu Chopra |
| **REGISTRATION NUMBER** | 11805967 |
| **SECTION** | KM046, group 1 |
| **ROLL NUMBER** | A26 |
| **COURSE CODE** | INT-247 |
| **SUBMITTED TO:** | Sanjay Kumar Singh |



GITHUB REPOSITORY- <https://github.com/vishnu190/PYTHON-PROJECT-CA3>

REPORT

**DATASET DEFAULT CREDIT CARD PAYMENT**

**AIM:** The task is to see which variables are the strongest predictors of default, and to make predictions on which customers are likely to default.

**DATA SET LOOKS LIKE THIS:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | **LIMIT\_BAL** | **SEX** | **EDUCATION** | **MARRIAGE** | **AGE** | **PAY\_0** | **PAY\_2** | **PAY\_3** | **PAY\_4** |  |  |  | BILL\_AMT2 | BILL\_AMT3 | BILL\_AMT4 | BILL\_AMT5 | BILL\_AMT6 | PAY\_AMT1 | PAY\_AMT2 | PAY\_AMT3 | PAY\_AMT4 | PAY\_AMT5 | PAY\_AMT6 | default payment next month | | |
| 1 | 20000 | 2 | 2 | 1 | 24 | 2 | 2 | -1 | -1 |  |  |  | 3102 | 689 | 0 | 0 | 0 | 0 | 689 | 0 | 0 | 0 | 0 | 1 |  |  |
| 2 | 120000 | 2 | 2 | 2 | 26 | -1 | 2 | 0 | 0 |  |  |  | 1725 | 2682 | 3272 | 3455 | 3261 | 0 | 1000 | 1000 | 1000 | 0 | 2000 | 1 |  |  |
| 3 | 90000 | 2 | 2 | 2 | 34 | 0 | 0 | 0 | 0 |  |  |  | 14027 | 13559 | 14331 | 14948 | 15549 | 1518 | 1500 | 1000 | 1000 | 1000 | 5000 | 0 |  |  |
| 4 | 50000 | 2 | 2 | 1 | 37 | 0 | 0 | 0 | 0 |  |  |  | 48233 | 49291 | 28314 | 28959 | 29547 | 2000 | 2019 | 1200 | 1100 | 1069 | 1000 | 0 |  |  |
| 5 | 50000 | 1 | 2 | 1 | 57 | -1 | 0 | -1 | 0 |  |  |  | 5670 | 35835 | 20940 | 19146 | 19131 | 2000 | 36681 | 10000 | 9000 | 689 | 679 | 0 |  |  |
| 6 | 50000 | 1 | 1 | 2 | 37 | 0 | 0 | 0 | 0 |  |  |  | 57069 | 57608 | 19394 | 19619 | 20024 | 2500 | 1815 | 657 | 1000 | 1000 | 800 | 0 |  |  |
| 7 | 500000 | 1 | 1 | 2 | 29 | 0 | 0 | 0 | 0 |  |  |  | 412023 | 445007 | 542653 | 483003 | 473944 | 55000 | 40000 | 38000 | 20239 | 13750 | 13770 | 0 |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **PAY\_5** | **PAY\_6** | **BILL\_AMT1** | **BILL\_AMT2** | **BILL\_AMT3** | **BILL\_AMT4** | **BILL\_AMT5** | **BILL\_AMT6** | **PAY\_AMT1** |  |  | **PAY\_AMT4** | **PAY\_AMT5** | **PAY\_AMT6** |
| -2 | -2 | 3913 | 3102 | 689 | 0 | 0 | 0 | 0 |  |  | 0 | 0 | 0 |
| 0 | 2 | 2682 | 1725 | 2682 | 3272 | 3455 | 3261 | 0 |  |  | 1000 | 0 | 2000 |
| 0 | 0 | 29239 | 14027 | 13559 | 14331 | 14948 | 15549 | 1518 |  |  | 1000 | 1000 | 5000 |
| 0 | 0 | 46990 | 48233 | 49291 | 28314 | 28959 | 29547 | 2000 |  |  | 1100 | 1069 | 1000 |
| 0 | 0 | 8617 | 5670 | 35835 | 20940 | 19146 | 19131 | 2000 |  |  | 9000 | 689 | 679 |
| 0 | 0 | 64400 | 57069 | 57608 | 19394 | 19619 | 20024 | 2500 |  |  | 1000 | 1000 | 800 |
| 0 | 0 | 367965 | 412023 | 445007 | 542653 | 483003 | 473944 | 55000 |  |  | 20239 | 13750 | 13770 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **PAY\_AMT2** | **PAY\_AMT3** | **PAY\_AMT4** | **PAY\_AMT5** | **PAY\_AMT6** | **default payment next month** | | |
| 689 | 0 | 0 | 0 | 0 | 1 |  |  |
| 1000 | 1000 | 1000 | 0 | 2000 | 1 |  |  |
| 1500 | 1000 | 1000 | 1000 | 5000 | 0 |  |  |
| 2019 | 1200 | 1100 | 1069 | 1000 | 0 |  |  |
| 36681 | 10000 | 9000 | 689 | 679 | 0 |  |  |
| 1815 | 657 | 1000 | 1000 | 800 | 0 |  |  |
| 40000 | 38000 | 20239 | 13750 | 13770 | 0 |  |  |

**NUMBER OF INSTANCES = 25**

**TARGET CLASS = “default payment next month”**

**UNIQUE VALUES OF TARGET CLASS= [1, 0]**

**\***clearly no ONE-HOT encoding is required

CODE written on Google Colaboratory:

How to import dataset on colaboratory?

Step1) from google.colab import drive

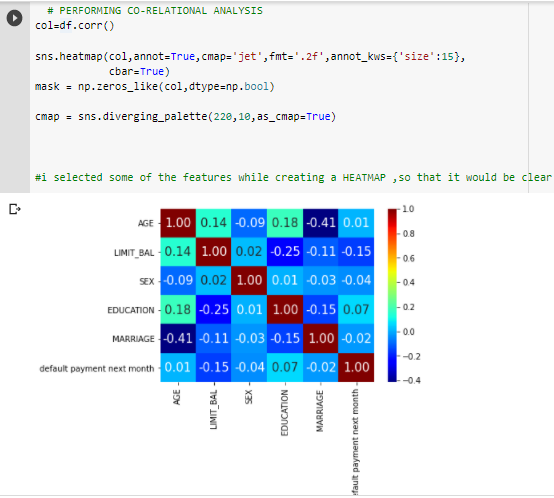
Step2) Enter the generated code on Colab and press ENTER

Step3) Now the file is imported to the Colab

Step4) Write this code to load dataset:-

drive.mount('/content/drive')

**HEATMAP FOR COORELATIONAL ANALYSIS**:



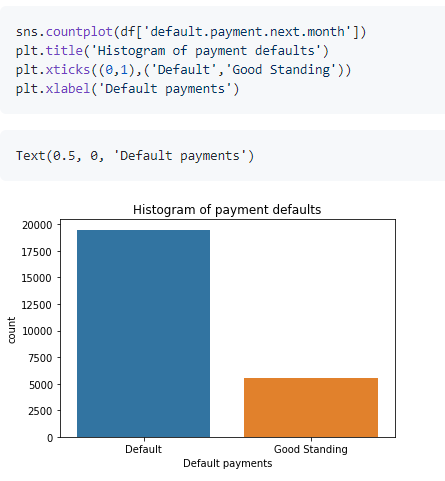
It is commonly applied for describing density or intensity of variables; visualize patterns, variance, and even anomalies.

Heatmaps will have a higher impact as they are not the conventional way of displaying this sort of data.

They’ll lose some accuracy, especially in this case

But overall, they would still be able to display patterns and summarize the periods in our data.

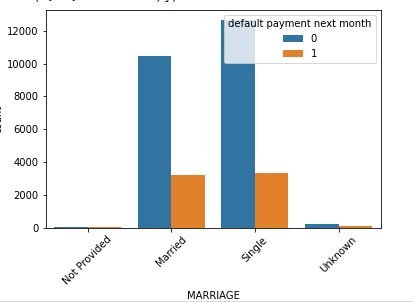
CHECKING THE DEFAULTERS



|  |
| --- |
| Total defaults=1875  Good standing=51oo  RATIO=1875/5100  = 3.54  # About one in every 3.5-4 people will end up defaulting. |

|  |
| --- |
| YOUNG POPULATION DENSITY of age 28,30,32,34,36 and 38 |

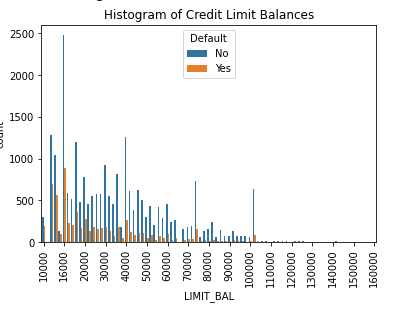
CHECKING MARITAL STATUS WITH DEFAULTERS



|  |
| --- |
| MARITAL STATUS ANALYSIS=======  in case of MARRIED less defaults  in case of SINGLE less defaults  0 =NO 1= YES |

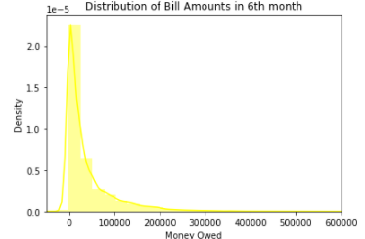
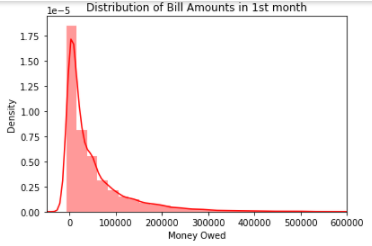
CHECKING RELATION BETWEEN LIMIT BALANCE AND POPULATION DENSITY

CHECKING:

****

Kind of a hard figure to see but shows the reationship between a persons balance limit and thier defualt. Generally a higher percentage of people with lower limit balances will end up defaulting. If you're a 'high risk' applicant the bank usually will only approve you for a smaller line of credit. This graph might help show why that is.

CHECKING BILL AMOUNT OF 1st AND 6th MONTH:

****

|  |
| --- |
| sns.distplot(data['BILL\_AMT1'],color='red')  sns.distplot(data['BILL\_AMT6'],color='yellow') |

|  |
| --- |
| Almost exactly the same. This seems to make sense. Most people would not want have too much debt. The vast majority of people have debt below about -50,000Rs |

**PERFORMING CLASSIFICATION ON DATASET**

WHAT IS CLASSIFICATION?

In simple terms classification is the process of predicting the class of given data points of a given Dataset.

Classes are sometimes called as Instances or Target or Labels or Categories.

Example > Email spam detection in email service providers can be identified as a classification problem. This is a binary classification since there are only 2 classes as spam and not spam.

COMPARISON AMONG CLASSIFIERS WITH GIVEN DATASET:

Code:

from sklearn.linear\_model import Perceptron

from sklearn.linear\_model import LogisticRegression

from sklearn.svm import SVC

from sklearn.neighbors import KNeighborsClassifier

from sklearn.naive\_bayes import GaussianNB

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

clf1=Perceptron(eta0=1)

clf2=LogisticRegression(penalty='l2',C=10)

clf3=SVC(C=100,kernel='rbf') # kernel='linear'

clf4=KNeighborsClassifier(n\_neighbors=3) # 3-NN

clf5=GaussianNB()

clf6=DecisionTreeClassifier(max\_depth=5)

clf7=RandomForestClassifier(max\_depth=5)

clf=[clf1,clf2,clf3,clf4,clf5,clf6,clf7]

clf\_names=['prec','Logistic Regression','SVM','K Nearest Neighbour','Gaussian NB','Decision Tree','Random Forest']

test={}

T={}

import time

for model,name in zip(clf,clf\_names):

  st=time.time()

  model.fit(x\_train\_std,y\_train)

  y\_pred=model.predict(x\_test\_std)

  et=time.time()

  acc=accuracy\_score(y\_test,y\_pred)

  test[name]=np.round(acc\*100,decimals=1)

  T[name]=np.round((et-st)\*1000,decimals=1) # ms

print(test)

print(T)

Output:

{'**prec**': 70.6, **'Logistic Regression**': 81.7, '**SVM**': 80.3, **'K Nearest Neighbour**': 77.4, **'Gaussian NB**': 58.9, **'Decision Tree**': 82.4, '**Random Forest'**: 81.8}

\*clearly Decision tree Classifier throws the highest accuracy (of 82.4) among others classifiers.