**Project No. 1: House Loan Data Analysis**

Step1: Import the required libraries import numpy as np

import pandas as pd

import pandas as pd

import sklearn

import numpy as np

import matplotlib.pyplot as plt

import os

import warnings

import seaborn as sns

Step2: Import scikit-learn from sklearn.preprocessing import OneHotEncoder

from sklearn.datasets import make\_blobs

from sklearn.impute import SimpleImputer

from sklearn.pipeline import Pipeline

from sklearn.compose import ColumnTransformer

from sklearn.preprocessing import StandardScaler

from sklearn.svm import LinearSVC

from sklearn.metrics import roc\_auc\_score

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import roc\_auc\_score

from sklearn.calibration import CalibratedClassifierCV

from sklearn.metrics import confusion\_matrix

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score

from sklearn.linear\_model import SGDClassifier

import plotly.offline as py

import plotly.graph\_objs as go

from plotly.offline import init\_notebook\_mode, iplot

from sklearn.model\_selection import train\_test\_split

init\_notebook\_mode(connected=True)

import cufflinks as cf

cf.go\_offline()

import pickle

import gc

import lightgbm as lgb

warnings.filterwarnings('ignore')

%matplotlib inline

Step3: Load the dataset house\_loan=pd.read\_csv('/content/loan\_data (1).csv')

house\_loan.describe()

house\_loan.columns

house\_loan.info()

Step4: Check for null values in the dataset

house\_loan.isnull().sum()

house\_loan.head()

Step5: Print percentage of default to payer of the dataset for the TARGET column

defaulters=(house\_loan.TARGET==1).sum()

payers=(house\_loan.TARGET==0).sum()

print((defaulters/payers)\*100)

without\_id=[column for column in house\_loan.columns if column!='SK\_ID\_CURR']

Step6: check for duplicate values

na=house\_loan[house\_loan.duplicated(subset=without\_id,keep=False)]

print("Duplicates are: ",na.shape[0])

house\_loan.TARGET.value\_counts().plot(kind='pie',autopct='%1.1f%%')

Step7: Balance the dataset if the data is imbalanced

import matplotlib as plt

shuffled\_data=house\_loan.sample(frac=1,random\_state=3)

unpaid\_home\_loan=shuffled\_data.loc[shuffled\_data['TARGET']==1]

paid\_home\_loan=shuffled\_data.loc[shuffled\_data['TARGET']==0].sample(n=24825,random\_state=69)

normalised\_home\_loan=pd.concat([unpaid\_home\_loan,paid\_home\_loan])

normalised\_home\_loan.TARGET.value\_counts().plot(kind='pie',autopct="%1.1f%%")

Step8: importing tensorflow for model tracking

import tensorflow as tf

normalised\_home\_loan.info()

normalised\_home\_loan.head

normalised\_home\_loan.dropna(axis=0)

normalised\_home\_loan.info()

normalised\_home\_loan.isnull().sum()

print(pd.unique(normalised\_home\_loan.AMT\_REQ\_CREDIT\_BUREAU\_DAY))

print(pd.unique(normalised\_home\_loan.AMT\_REQ\_CREDIT\_BUREAU\_WEEK))

print(pd.unique(normalised\_home\_loan.AMT\_REQ\_CREDIT\_BUREAU\_MON))

print(pd.unique(normalised\_home\_loan.AMT\_REQ\_CREDIT\_BUREAU\_QRT))

print(pd.unique(normalised\_home\_loan.AMT\_REQ\_CREDIT\_BUREAU\_YEAR))

normalised\_home\_loan.dropna(axis=0)

print(normalised\_home\_loan.info())

print(normalised\_home\_loan.isnull().sum())

Step9: Plot the balanced data or imbalanced data

normalised\_home\_loan.TARGET.value\_counts().plot(kind='pie',autopct="%1.1f%%")

normalised\_home\_loan.NAME\_CONTRACT\_TYPE.value\_counts().plot(kind='pie',autopct="%1.1f%%")

normalised\_home\_loan.CODE\_GENDER.value\_counts().plot(kind='pie',autopct="%1.1f%%")

normalised\_home\_loan.FLAG\_OWN\_CAR.value\_counts().plot(kind='pie',autopct="%1.1f%%")

normalised\_home\_loan.CNT\_CHILDREN.value\_counts().plot(kind='pie',autopct="%1.1f%%")

!pip install chart\_studio

cf.set\_config\_file(theme='polar')

normalised\_home\_loan[normalised\_home\_loan['AMT\_INCOME\_TOTAL'] < 2000000]['AMT\_INCOME\_TOTAL'].iplot(kind='histogram', bins=100,

   xTitle = 'Total Income', yTitle ='Count of applicants',

             title='Distribution of AMT\_INCOME\_TOTAL')

(normalised\_home\_loan[normalised\_home\_loan['AMT\_INCOME\_TOTAL']>1000000]['TARGET'].value\_counts())/len(normalised\_home\_loan[normalised\_home\_loan['AMT\_INCOME\_TOTAL'] > 1000000])\*100

print((normalised\_home\_loan[normalised\_home\_loan['CNT\_CHILDREN']>2]['TARGET'].value\_counts())/len(normalised\_home\_loan[normalised\_home\_loan['CNT\_CHILDREN'] > 2])\*100)

print((normalised\_home\_loan[normalised\_home\_loan['CNT\_CHILDREN']>5]['TARGET'].value\_counts())/len(normalised\_home\_loan[normalised\_home\_loan['CNT\_CHILDREN'] > 5])\*100)

print((normalised\_home\_loan[normalised\_home\_loan['FLAG\_OWN\_CAR']=='N']['TARGET'].value\_counts())/len(normalised\_home\_loan[normalised\_home\_loan['FLAG\_OWN\_CAR'] =='N'])\*100)

print((normalised\_home\_loan[normalised\_home\_loan['FLAG\_OWN\_CAR']=='Y']['TARGET'].value\_counts())/len(normalised\_home\_loan[normalised\_home\_loan['FLAG\_OWN\_CAR'] =='Y'])\*100)

print((normalised\_home\_loan[normalised\_home\_loan['CODE\_GENDER']=='M']['TARGET'].value\_counts())/len(normalised\_home\_loan[normalised\_home\_loan['CODE\_GENDER'] =='M'])\*100)

print((normalised\_home\_loan[normalised\_home\_loan['CODE\_GENDER']=='F']['TARGET'].value\_counts())/len(normalised\_home\_loan[normalised\_home\_loan['CODE\_GENDER'] =='F'])\*100)

print((normalised\_home\_loan[normalised\_home\_loan['NAME\_CONTRACT\_TYPE']=='Cash loans']['TARGET'].value\_counts())/len(normalised\_home\_loan[normalised\_home\_loan['NAME\_CONTRACT\_TYPE']=='Cash loans'])\*100)

print((normalised\_home\_loan[normalised\_home\_loan['NAME\_CONTRACT\_TYPE']=='Revolving loans']['TARGET'].value\_counts())/len(normalised\_home\_loan[normalised\_home\_loan['NAME\_CONTRACT\_TYPE']=='Revolving loans'])\*100)

normalised\_home\_loan=normalised\_home\_loan.sample(frac=1,random\_state=5)

Step10: Encode the columns that is required for the model

from sklearn.preprocessing import OrdinalEncoder

ordenc=OrdinalEncoder()

normalised\_home\_loan['NAME\_CONTRACT\_TYPE\_CODE']=ordenc.fit\_transform(normalised\_home\_loan[['NAME\_CONTRACT\_TYPE']])

print(normalised\_home\_loan[['NAME\_CONTRACT\_TYPE','NAME\_CONTRACT\_TYPE\_CODE']].head(20))

print(normalised\_home\_loan['NAME\_CONTRACT\_TYPE\_CODE'].value\_counts())

normalised\_home\_loan['CODE\_GENDER\_CODE']=ordenc.fit\_transform(normalised\_home\_loan[['CODE\_GENDER']])

print(normalised\_home\_loan[['CODE\_GENDER','CODE\_GENDER\_CODE']].head(20))

print(normalised\_home\_loan['CODE\_GENDER\_CODE'].value\_counts())

normalised\_home\_loan.loc[normalised\_home\_loan['CODE\_GENDER\_CODE']==2]

normalised\_home\_loan['FLAG\_OWN\_CAR\_CODE']=ordenc.fit\_transform(normalised\_home\_loan[['FLAG\_OWN\_CAR']])

print(normalised\_home\_loan[['FLAG\_OWN\_CAR','FLAG\_OWN\_CAR\_CODE']].head(20))

print(normalised\_home\_loan['FLAG\_OWN\_CAR\_CODE'].value\_counts())

normalised\_home\_loan['CNT\_CHILDREN\_CODE']=ordenc.fit\_transform(normalised\_home\_loan[['CNT\_CHILDREN']])

print(normalised\_home\_loan[['CNT\_CHILDREN\_CODE','CNT\_CHILDREN']].head(20))

print(normalised\_home\_loan['CNT\_CHILDREN\_CODE'].value\_counts())

normalised\_home\_loan=normalised\_home\_loan.sample(frac=1,random\_state=45)

normalised\_home\_loan['TARGET'].value\_counts()

y=normalised\_home\_loan.TARGET

normalised\_home\_loan\_features=['SK\_ID\_CURR','NAME\_CONTRACT\_TYPE\_CODE','CNT\_CHILDREN\_CODE','FLAG\_OWN\_CAR\_CODE','CODE\_GENDER\_CODE']

Step11: Train the model

from sklearn.model\_selection import train\_test\_split

X=normalised\_home\_loan[normalised\_home\_loan\_features]

blobs\_random\_seed = 42

centers = [(0,0), (5,5)]

cluster\_std = 1

frac\_test\_split = 0.33

num\_features\_for\_samples = 2

num\_samples\_total = 49650

Step12: Generate data

inputs, targets = make\_blobs(n\_samples = num\_samples\_total, centers = centers, n\_features = num\_features\_for\_samples, cluster\_std = cluster\_std)

X\_train,X\_test,y\_train,y\_test=train\_test\_split(inputs,targets,test\_size=0.33,random\_state=45)

print(X\_train.shape, X\_test.shape, y\_train.shape, y\_test.shape)

plt.pyplot.scatter(X\_train[:,0], X\_train[:,1])

plt.pyplot.title('Linearly separable data')

plt.pyplot.xlabel('X1')

plt.pyplot.ylabel('X2')

plt.pyplot.show()

Step13: Calculate Sensitivity as a metrice

from sklearn import svm

from sklearn.metrics import plot\_confusion\_matrix

clf=svm.SVC(kernel='linear')

clf=clf.fit(X\_train,y\_train)

predictions = clf.predict(X\_test)

Step14: Calculate area under receiver operating characteristics curve

matrix = plot\_confusion\_matrix(clf, X\_test, y\_test, cmap=plt.cm.Blues, normalize='true')

plt.pyplot.title('Confusion matrix for our classifier')

plt.pyplot.show(matrix)

plt.pyplot.show()

from sklearn.metrics import precision\_score, recall\_score,f1\_score

print(precision\_score(y\_test, predictions))

print(recall\_score(y\_test, predictions))

print(f1\_score(y\_test,predictions,average=None))

support\_vectors = clf.support\_vectors\_

plt.pyplot.scatter(X\_train[:,0], X\_train[:,1])

plt.pyplot.scatter(support\_vectors[:,0], support\_vectors[:,1], color='red')

plt.pyplot.title('Linearly separable data with support vectors')

plt.pyplot.xlabel('X1')

plt.pyplot.ylabel('X2')

plt.pyplot.show()

from mlxtend.plotting import plot\_decision\_regions

plot\_decision\_regions(X\_test, y\_test, clf=clf, legend=2)

plt.pyplot.show()

# Screenshot of the Output

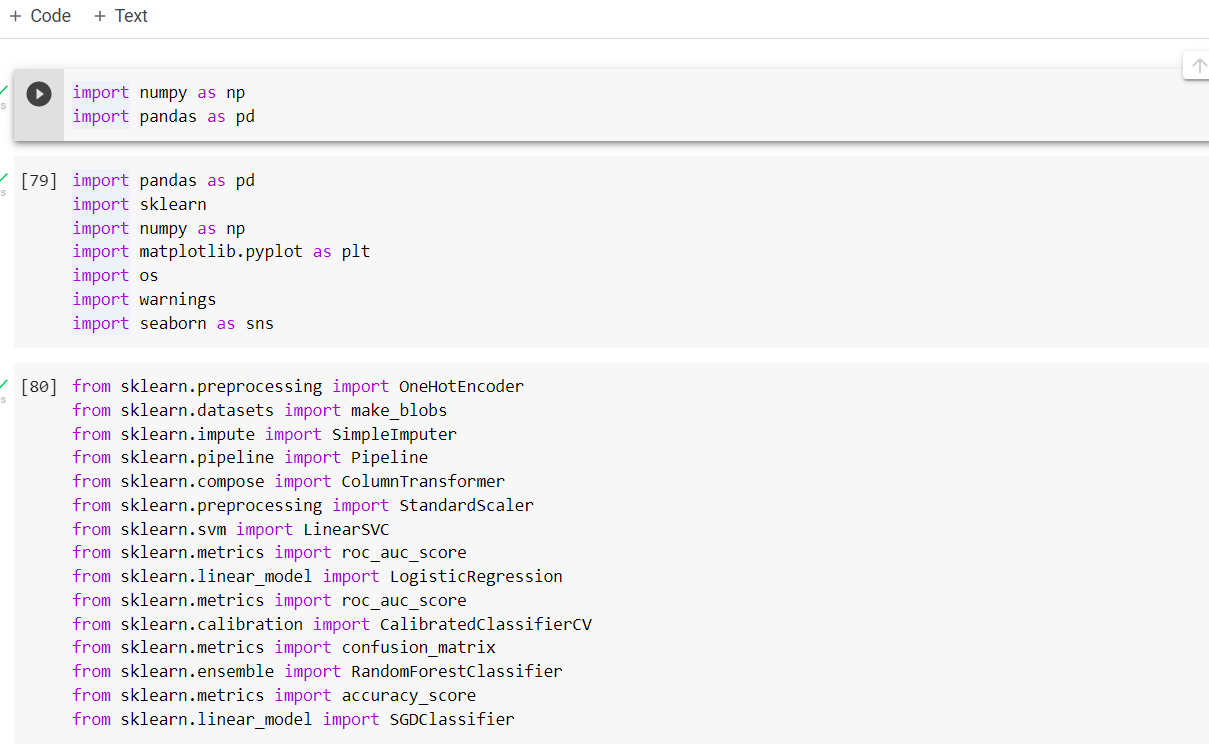
## Q1. Read the data from the Given Dataset

Step3: Load the dataset house\_loan=pd.read\_csv('/content/loan\_data (1).csv')

house\_loan.describe()

house\_loan.columns

house\_loan.info()

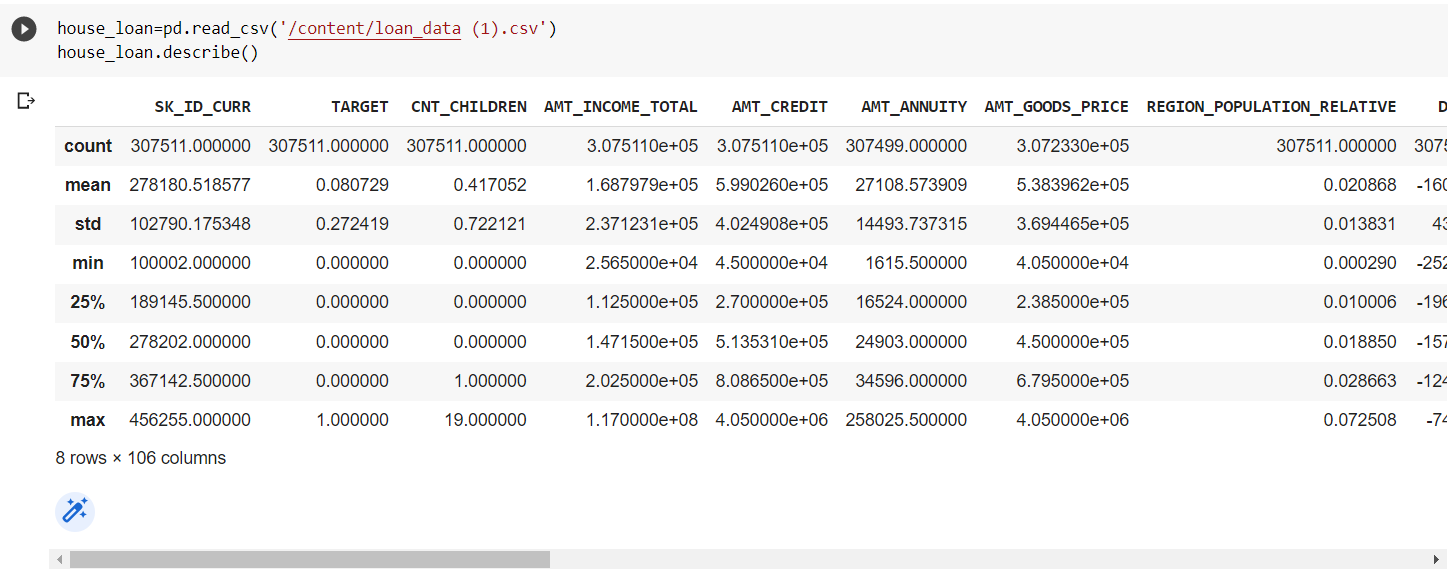


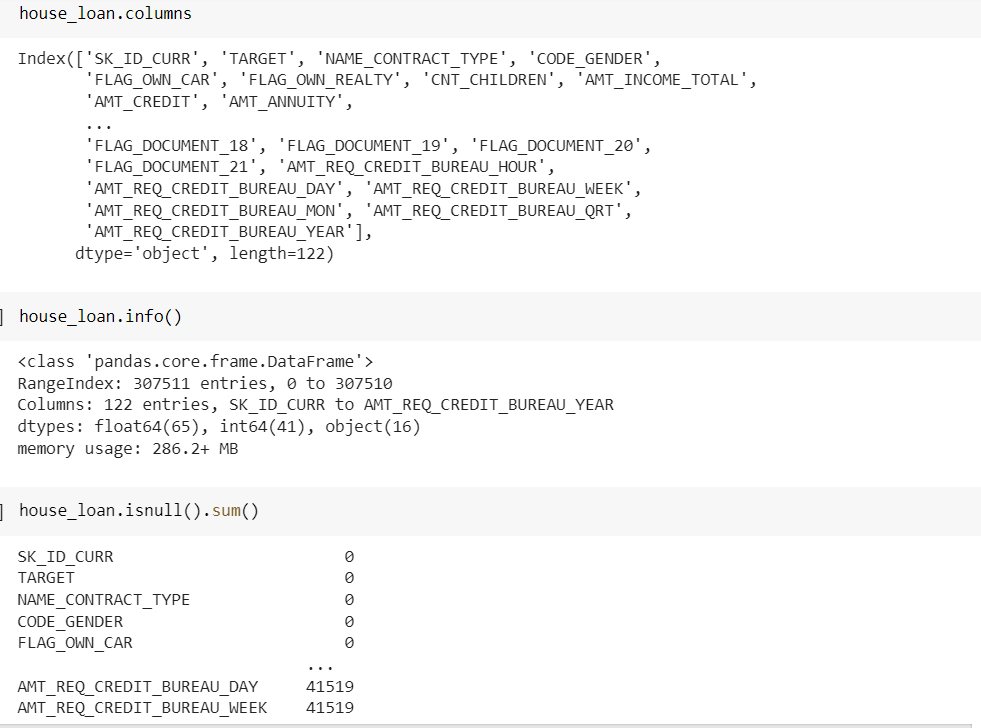
## Q2. Check for null values in the dataset

Step4: Check for null values in the dataset

house\_loan.isnull().sum()

house\_loan.head()





## Q3. Print percentage of default to payer of the dataset for the TARGET column

Step5: Print percentage of default to payer of the dataset for the TARGET column

defaulters=(house\_loan.TARGET==1).sum()

payers=(house\_loan.TARGET==0).sum()

print((defaulters/payers)\*100)

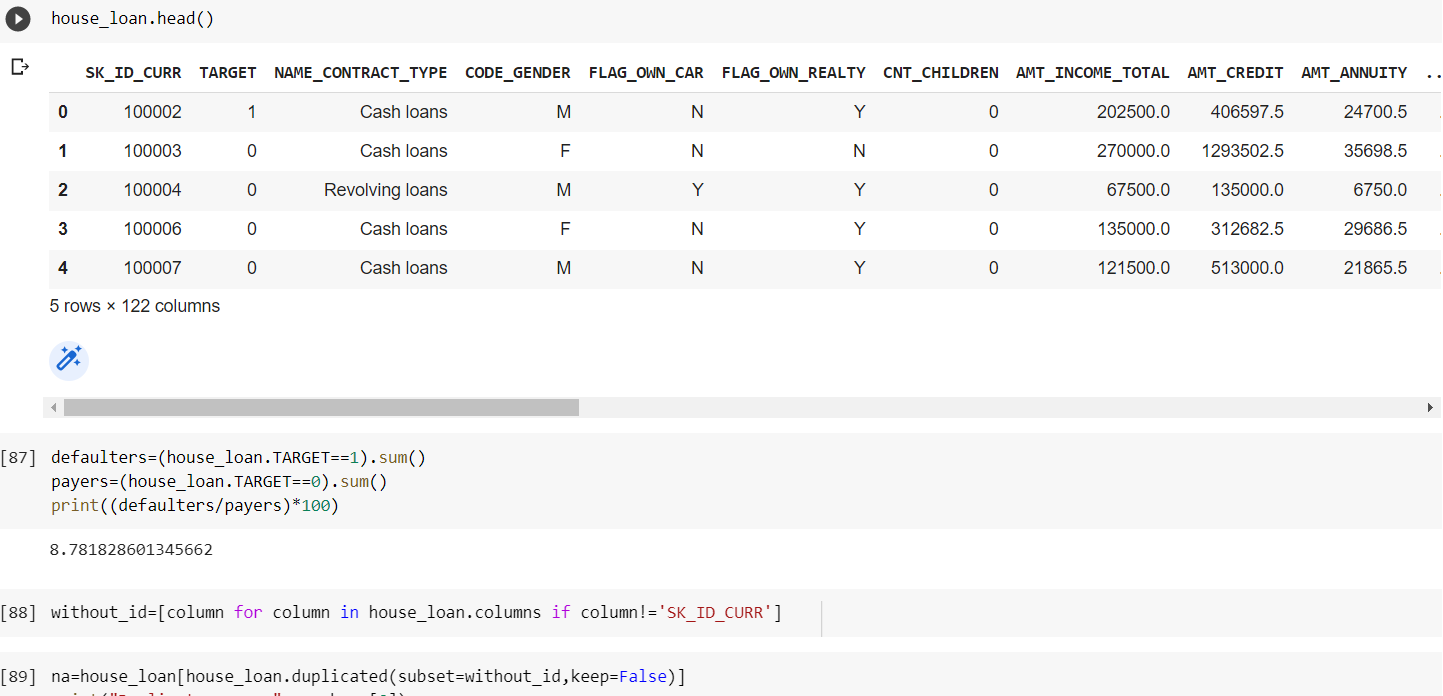
without\_id=[column for column in house\_loan.columns if column!='SK\_ID\_CURR']

Step6: check for duplicate values

na=house\_loan[house\_loan.duplicated(subset=without\_id,keep=False)]

print("Duplicates are: ",na.shape[0])

house\_loan.TARGET.value\_counts().plot(kind='pie',autopct='%1.1f%%')



## Q4. Balance the dataset if the data is imbalanced

Step7: Balance the dataset if the data is imbalanced

import matplotlib as plt

shuffled\_data=house\_loan.sample(frac=1,random\_state=3)

unpaid\_home\_loan=shuffled\_data.loc[shuffled\_data['TARGET']==1]

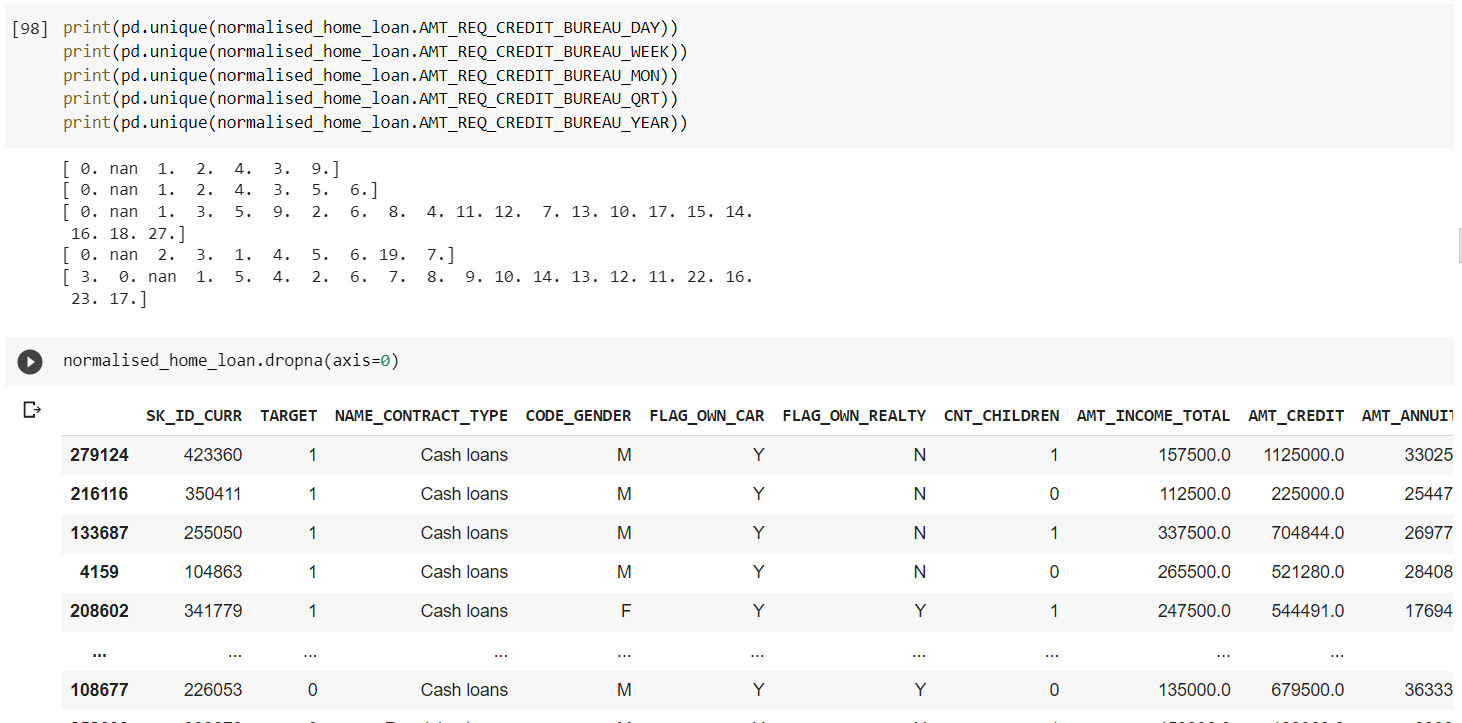
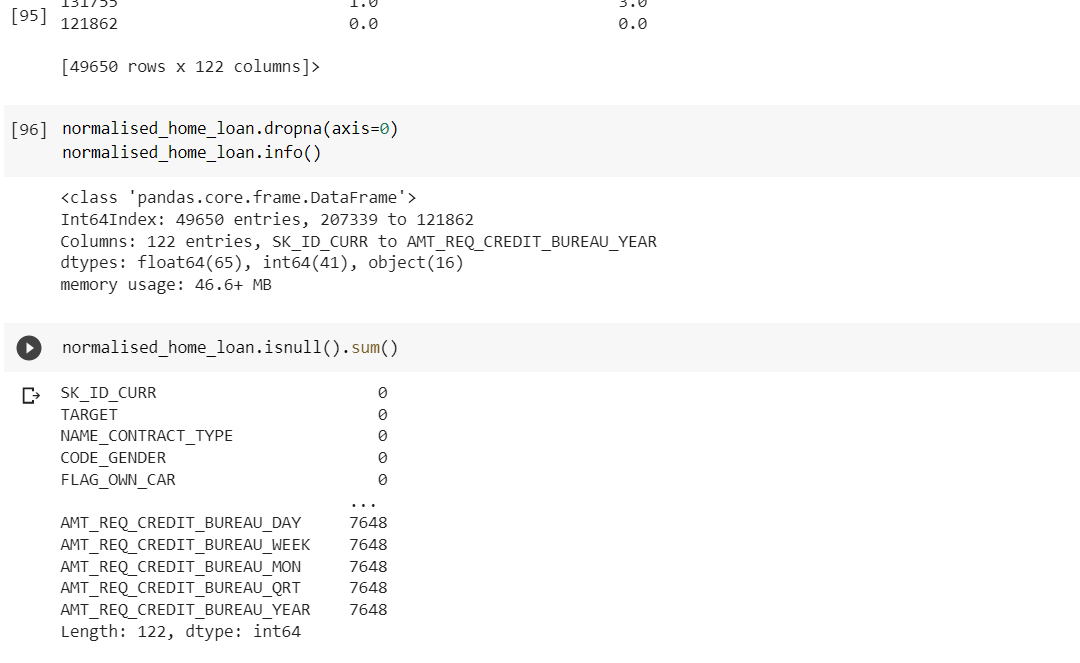
paid\_home\_loan=shuffled\_data.loc[shuffled\_data['TARGET']==0].sample(n=24825,random\_state=69)

normalised\_home\_loan=pd.concat([unpaid\_home\_loan,paid\_home\_loan])

normalised\_home\_loan.TARGET.value\_counts().plot(kind='pie',autopct="%1.1f%%")







## Q5. Plot the balanced data or imbalanced data

Step9: Plot the balanced data or imbalanced data

normalised\_home\_loan.TARGET.value\_counts().plot(kind='pie',autopct="%1.1f%%")

normalised\_home\_loan.NAME\_CONTRACT\_TYPE.value\_counts().plot(kind='pie',autopct="%1.1f%%")

normalised\_home\_loan.CODE\_GENDER.value\_counts().plot(kind='pie',autopct="%1.1f%%")

normalised\_home\_loan.FLAG\_OWN\_CAR.value\_counts().plot(kind='pie',autopct="%1.1f%%")

normalised\_home\_loan.CNT\_CHILDREN.value\_counts().plot(kind='pie',autopct="%1.1f%%")

!pip install chart\_studio

cf.set\_config\_file(theme='polar')

normalised\_home\_loan[normalised\_home\_loan['AMT\_INCOME\_TOTAL'] < 2000000]['AMT\_INCOME\_TOTAL'].iplot(kind='histogram', bins=100,

   xTitle = 'Total Income', yTitle ='Count of applicants',

             title='Distribution of AMT\_INCOME\_TOTAL')

(normalised\_home\_loan[normalised\_home\_loan['AMT\_INCOME\_TOTAL']>1000000]['TARGET'].value\_counts())/len(normalised\_home\_loan[normalised\_home\_loan['AMT\_INCOME\_TOTAL'] > 1000000])\*100

print((normalised\_home\_loan[normalised\_home\_loan['CNT\_CHILDREN']>2]['TARGET'].value\_counts())/len(normalised\_home\_loan[normalised\_home\_loan['CNT\_CHILDREN'] > 2])\*100)

print((normalised\_home\_loan[normalised\_home\_loan['CNT\_CHILDREN']>5]['TARGET'].value\_counts())/len(normalised\_home\_loan[normalised\_home\_loan['CNT\_CHILDREN'] > 5])\*100)

print((normalised\_home\_loan[normalised\_home\_loan['FLAG\_OWN\_CAR']=='N']['TARGET'].value\_counts())/len(normalised\_home\_loan[normalised\_home\_loan['FLAG\_OWN\_CAR'] =='N'])\*100)

print((normalised\_home\_loan[normalised\_home\_loan['FLAG\_OWN\_CAR']=='Y']['TARGET'].value\_counts())/len(normalised\_home\_loan[normalised\_home\_loan['FLAG\_OWN\_CAR'] =='Y'])\*100)

print((normalised\_home\_loan[normalised\_home\_loan['CODE\_GENDER']=='M']['TARGET'].value\_counts())/len(normalised\_home\_loan[normalised\_home\_loan['CODE\_GENDER'] =='M'])\*100)

print((normalised\_home\_loan[normalised\_home\_loan['CODE\_GENDER']=='F']['TARGET'].value\_counts())/len(normalised\_home\_loan[normalised\_home\_loan['CODE\_GENDER'] =='F'])\*100)

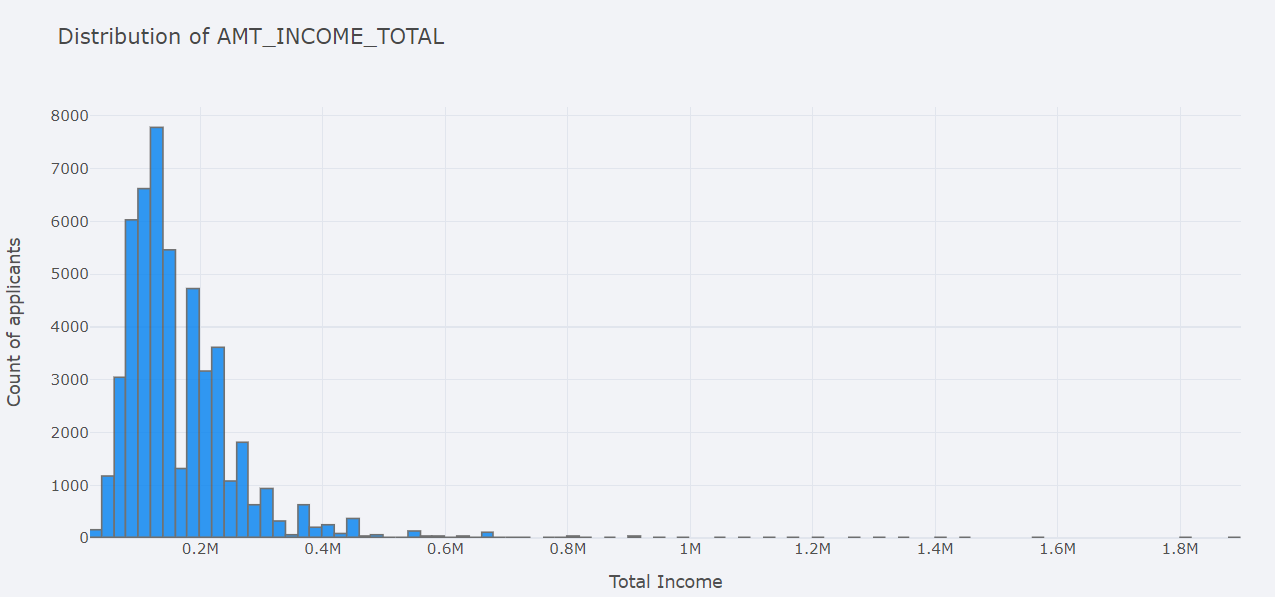
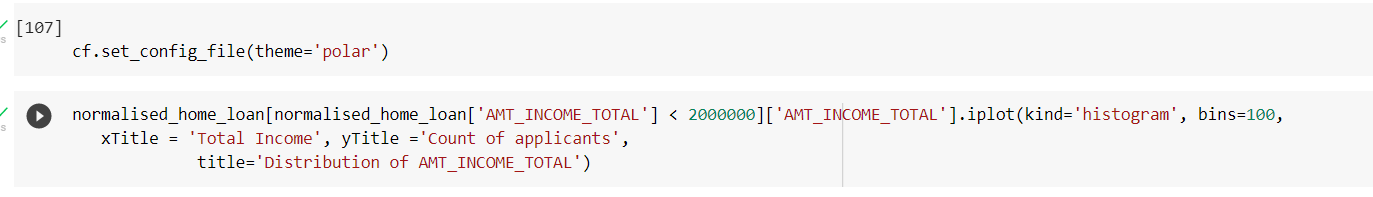
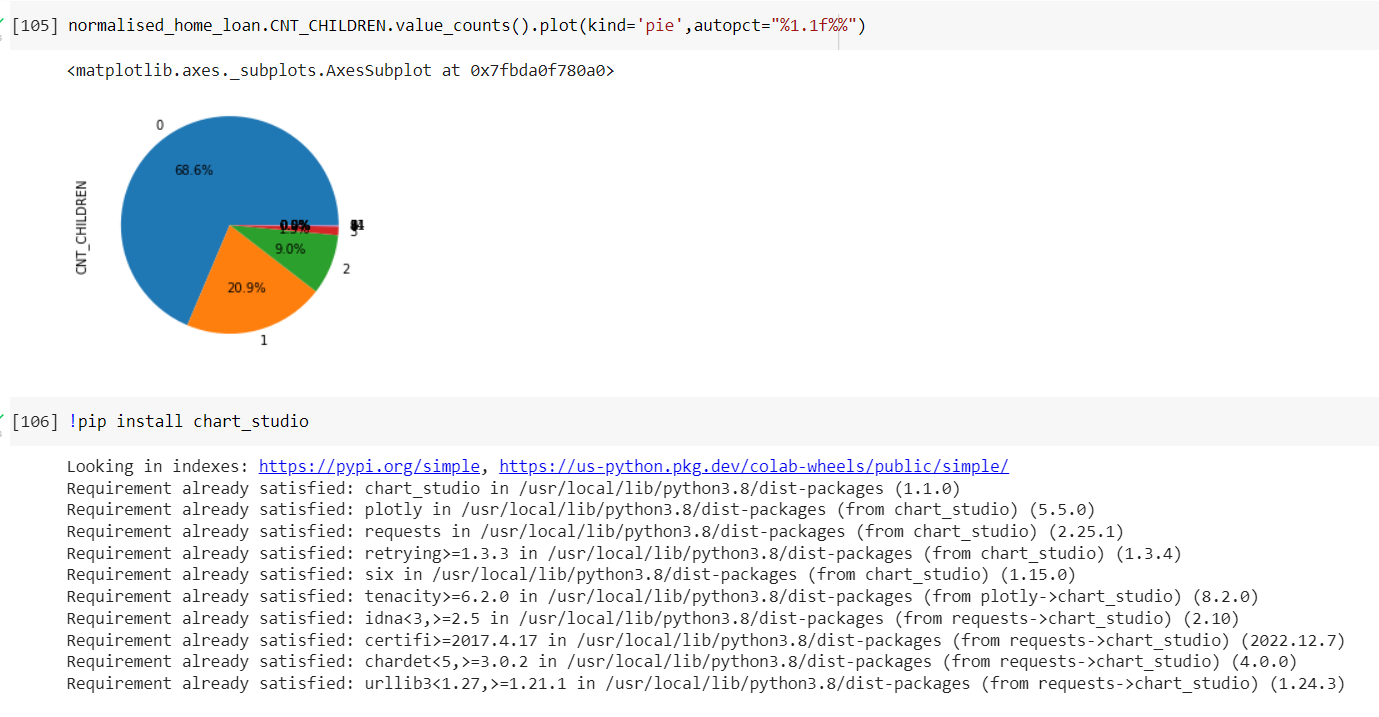
print((normalised\_home\_loan[normalised\_home\_loan['NAME\_CONTRACT\_TYPE']=='Cash loans']['TARGET'].value\_counts())/len(normalised\_home\_loan[normalised\_home\_loan['NAME\_CONTRACT\_TYPE']=='Cash loans'])\*100)

print((normalised\_home\_loan[normalised\_home\_loan['NAME\_CONTRACT\_TYPE']=='Revolving loans']['TARGET'].value\_counts())/len(normalised\_home\_loan[normalised\_home\_loan['NAME\_CONTRACT\_TYPE']=='Revolving loans'])\*100)

normalised\_home\_loan=normalised\_home\_loan.sample(frac=1,random\_state=5)









## Q6. Encode the columns that is required for the model

Step10: Encode the columns that is required for the model

from sklearn.preprocessing import OrdinalEncoder

ordenc=OrdinalEncoder()

normalised\_home\_loan['NAME\_CONTRACT\_TYPE\_CODE']=ordenc.fit\_transform(normalised\_home\_loan[['NAME\_CONTRACT\_TYPE']])

print(normalised\_home\_loan[['NAME\_CONTRACT\_TYPE','NAME\_CONTRACT\_TYPE\_CODE']].head(20))

print(normalised\_home\_loan['NAME\_CONTRACT\_TYPE\_CODE'].value\_counts())

normalised\_home\_loan['CODE\_GENDER\_CODE']=ordenc.fit\_transform(normalised\_home\_loan[['CODE\_GENDER']])

print(normalised\_home\_loan[['CODE\_GENDER','CODE\_GENDER\_CODE']].head(20))

print(normalised\_home\_loan['CODE\_GENDER\_CODE'].value\_counts())

normalised\_home\_loan.loc[normalised\_home\_loan['CODE\_GENDER\_CODE']==2]

normalised\_home\_loan['FLAG\_OWN\_CAR\_CODE']=ordenc.fit\_transform(normalised\_home\_loan[['FLAG\_OWN\_CAR']])

print(normalised\_home\_loan[['FLAG\_OWN\_CAR','FLAG\_OWN\_CAR\_CODE']].head(20))

print(normalised\_home\_loan['FLAG\_OWN\_CAR\_CODE'].value\_counts())

normalised\_home\_loan['CNT\_CHILDREN\_CODE']=ordenc.fit\_transform(normalised\_home\_loan[['CNT\_CHILDREN']])

print(normalised\_home\_loan[['CNT\_CHILDREN\_CODE','CNT\_CHILDREN']].head(20))

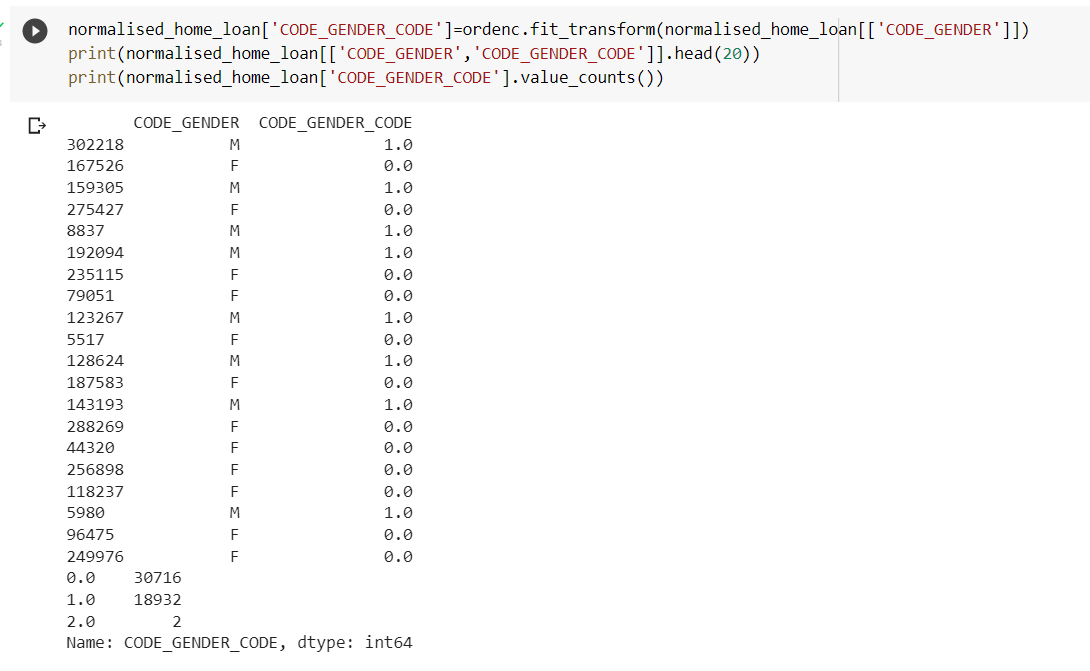
print(normalised\_home\_loan['CNT\_CHILDREN\_CODE'].value\_counts())

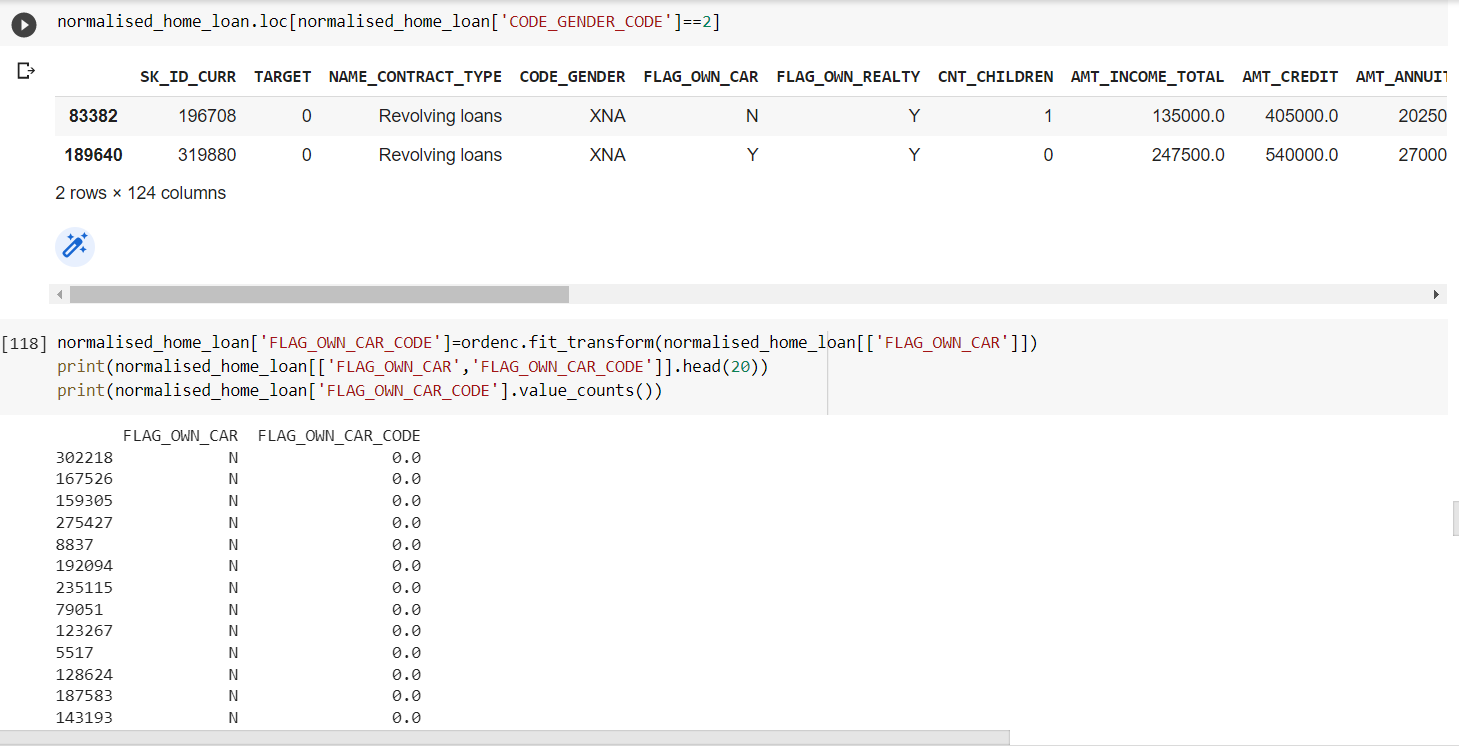
normalised\_home\_loan=normalised\_home\_loan.sample(frac=1,random\_state=45)

normalised\_home\_loan['TARGET'].value\_counts()

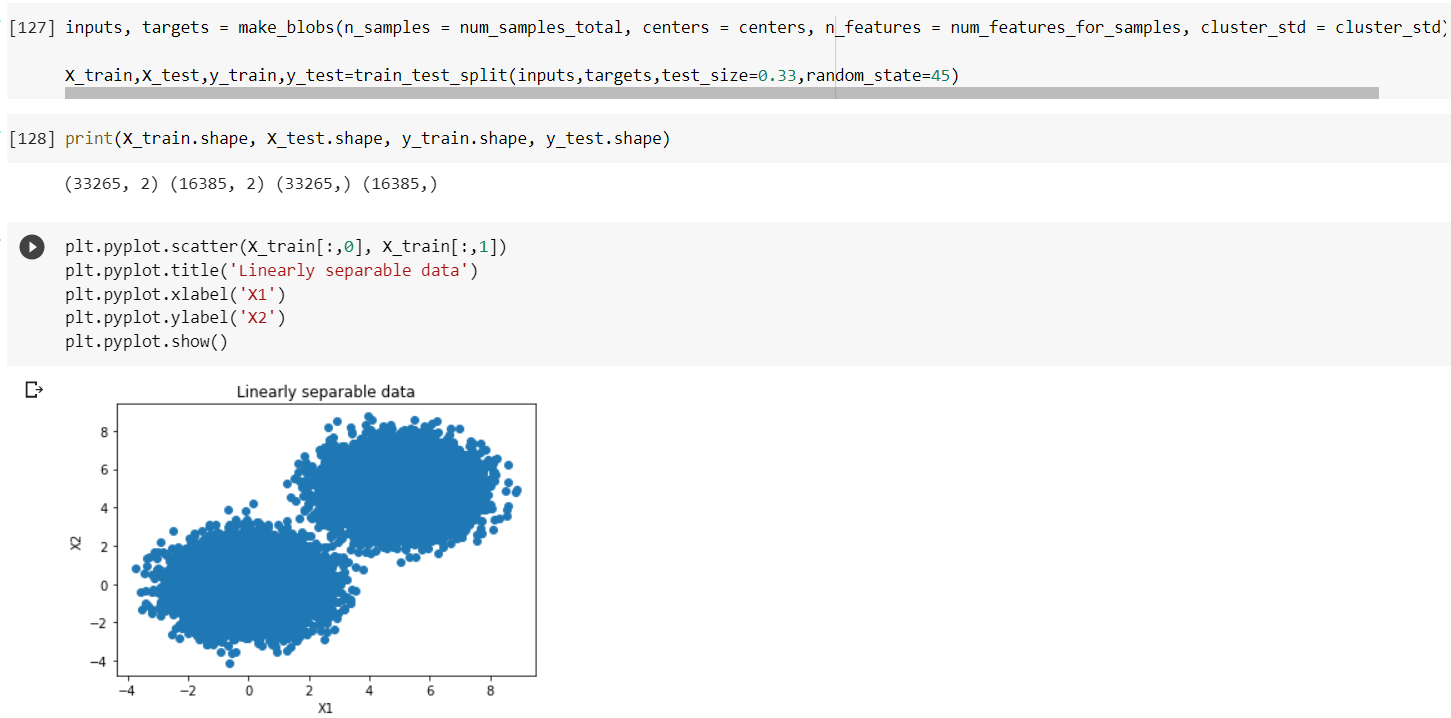
y=normalised\_home\_loan.TARGET

normalised\_home\_loan\_features=['SK\_ID\_CURR','NAME\_CONTRACT\_TYPE\_CODE','CNT\_CHILDREN\_CODE','FLAG\_OWN\_CAR\_CODE','CODE\_GENDER\_CODE']









## Q7. Calculate Sensitivity as a metrice and Calculate area under receiver operating characteristics curve

Step13: Calculate Sensitivity as a metrice

from sklearn import svm

from sklearn.metrics import plot\_confusion\_matrix

clf=svm.SVC(kernel='linear')

clf=clf.fit(X\_train,y\_train)

predictions = clf.predict(X\_test)

Step14: Calculate area under receiver operating characteristics curve

matrix = plot\_confusion\_matrix(clf, X\_test, y\_test,cmap=plt.cm.Blues, normalize='true')

plt.pyplot.title('Confusion matrix for our classifier')

plt.pyplot.show(matrix)

plt.pyplot.show()

from sklearn.metrics import precision\_score, recall\_score,f1\_score

print(precision\_score(y\_test, predictions))

print(recall\_score(y\_test, predictions))

print(f1\_score(y\_test,predictions,average=None))

support\_vectors = clf.support\_vectors\_

plt.pyplot.scatter(X\_train[:,0], X\_train[:,1])

plt.pyplot.scatter(support\_vectors[:,0], support\_vectors[:,1], color='red')

plt.pyplot.title('Linearly separable data with support vectors')

plt.pyplot.xlabel('X1')

plt.pyplot.ylabel('X2')

plt.pyplot.show()

from mlxtend.plotting import plot\_decision\_regions

plot\_decision\_regions(X\_test, y\_test, clf=clf, legend=2)

plt.pyplot.show()

