LOAN PREDICTION USING MACHINE LEARING ALGORITHMS (BATCH-9)

# Group Members:

## Nagineni Sai Lasya -19BCE7493

## CHADALAWADA V V KASYAP-20BCE7457

## HARSHAVARDINI K - 20BCE7439

## CHITTURI SRIJA CHOWDARY-20BCE7583

# DATA VISUALIZATION:

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

load\_dataset = pd.read\_csv("Loan.csv")

load\_dataset.columns

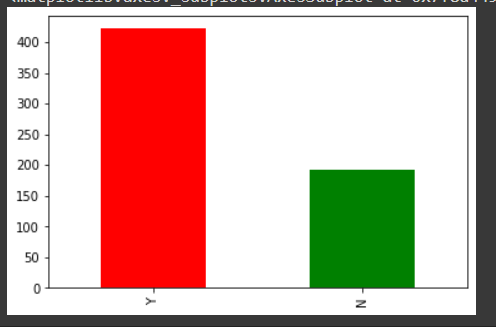
Index(['Loan\_ID', 'Gender', 'Married', 'Dependents', 'Education',

'Self\_Employed', 'ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',

'Loan\_Amount\_Term', 'Credit\_History', 'Property\_Area', 'Loan\_Status'],

dtype='object')

load\_dataset["Loan\_Status"].value\_counts().plot.bar(color=["red","green"])



plt.figure(1)

plt.subplot(221)

load\_dataset["Gender"].value\_counts(normalize='True').plot.bar(figsize=(8,8),title="Gender",color=["orange","green"])

plt.subplot(222)

load\_dataset["Married"].value\_counts(normalize='True').plot.bar(figsize=(8,8),title="Married",color=["orange","green"])

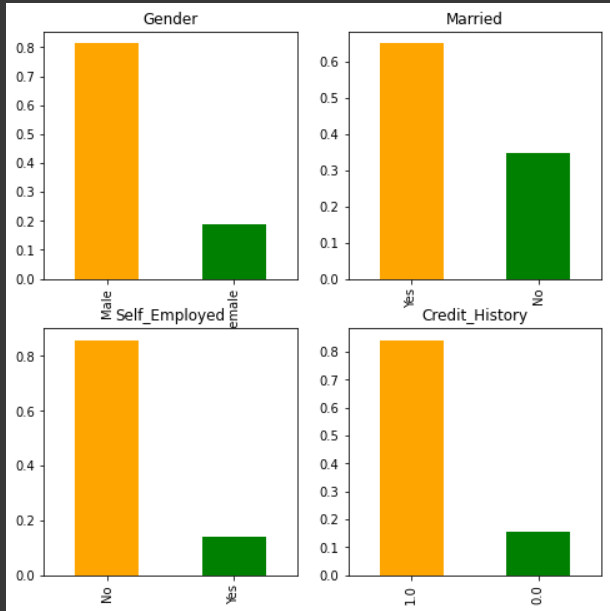
plt.subplot(223)

load\_dataset["Self\_Employed"].value\_counts(normalize='True').plot.bar(figsize=(8,8),title="Self\_Employed",color=["orange","green"])

plt.subplot(224)

load\_dataset["Credit\_History"].value\_counts(normalize='True').plot.bar(figsize=(8,8),title="Credit\_History",color=["orange","green"])

plt.show()



plt.figure(1)

plt.subplot(131)

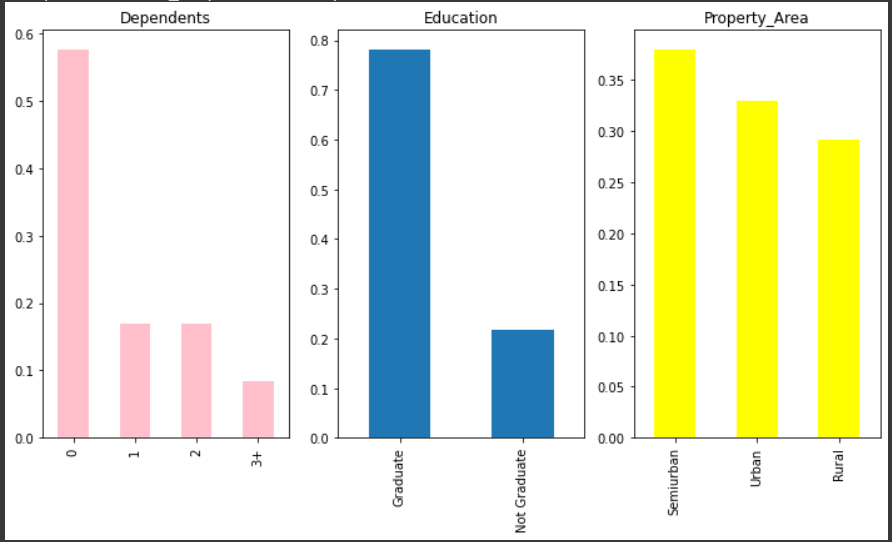
load\_dataset["Dependents"].value\_counts(normalize=True).plot.bar(figsize=(12,6),color="pink",title="Dependents")

plt.subplot(132)

load\_dataset["Education"].value\_counts(normalize=True).plot.bar(figsize=(12,6),title="Education")

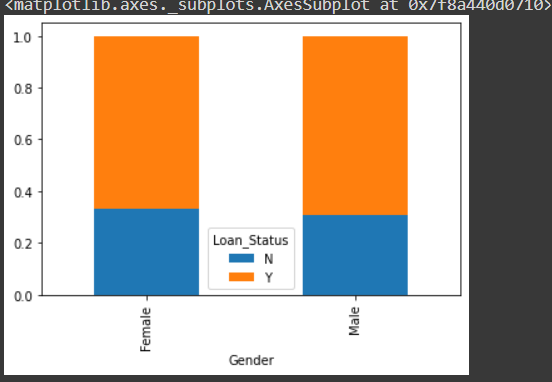
plt.subplot(133)

load\_dataset["Property\_Area"].value\_counts(normalize=True).plot.bar(figsize=(12,6),color="yellow",title="Property\_Area")



Gender=pd.crosstab(load\_dataset['Gender'],train['Loan\_Status'])

Gender.div(Gender.sum(1).astype(float), axis=0).plot(kind="bar", stacked=True)



Married=pd.crosstab(load\_dataset['Married'],train['Loan\_Status'])

Married.div(Married.sum(1).astype(float), axis=0).plot(kind="bar", stacked=True, figsize=(4,4))



Employed=pd.crosstab(load\_dataset['Self\_Employed'],load\_dataset['Loan\_Status'])

Employed.div(Employed.sum(1).astype(float), axis=0).plot(kind="bar", stacked=True, figsize=(4,4))

Dependents=pd.crosstab(load\_dataset['Dependents'],load\_dataset['Loan\_Status'])

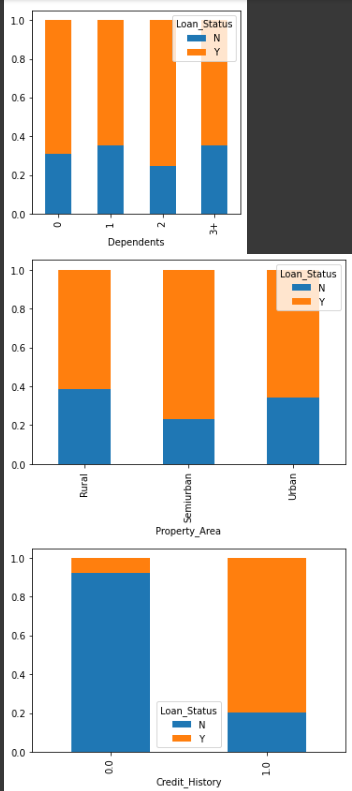
Dependents.div(Dependents.sum(1).astype(float), axis=0).plot(kind="bar", stacked=True, figsize=(4,4))

property\_a = pd.crosstab(load\_dataset["Property\_Area"],load\_dataset["Loan\_Status"])

property\_a.div(property\_a.sum(1).astype(float),axis=0).plot(kind="bar",stacked=True)

credit\_hist = pd.crosstab(load\_dataset["Credit\_History"],load\_dataset["Loan\_Status"])

credit\_hist.div(credit\_hist.sum(1).astype(float),axis=0).plot(kind="bar",stacked="True")



# CODE FOR DATASET LOADING AND REMOVING NULL’S:

import numpy as np

import pandas as pd

import seaborn as sns

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

from sklearn import svm

from sklearn.linear\_model import LogisticRegression

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import accuracy\_score

loan\_dataset = pd.read\_csv('Loan.csv')

loan\_dataset.head

<bound method NDFrame.head of Loan\_ID Gender Married Dependents Education Self\_Employed \

0 LP001002 Male No 0 Graduate No

1 LP001003 Male Yes 1 Graduate No

2 LP001005 Male Yes 0 Graduate Yes

3 LP001006 Male Yes 0 Not Graduate No

4 LP001008 Male No 0 Graduate No

.. ... ... ... ... ... ...

609 LP002978 Female No 0 Graduate No

610 LP002979 Male Yes 3+ Graduate No

611 LP002983 Male Yes 1 Graduate No

612 LP002984 Male Yes 2 Graduate No

613 LP002990 Female No 0 Graduate Yes

ApplicantIncome CoapplicantIncome LoanAmount Loan\_Amount\_Term \

0 5849 0.0 NaN 360.0

1 4583 1508.0 128.0 360.0

2 3000 0.0 66.0 360.0

3 2583 2358.0 120.0 360.0

4 6000 0.0 141.0 360.0

.. ... ... ... ...

609 2900 0.0 71.0 360.0

610 4106 0.0 40.0 180.0

611 8072 240.0 253.0 360.0

612 7583 0.0 187.0 360.0

613 4583 0.0 133.0 360.0

Credit\_History Property\_Area Loan\_Status

0 1.0 Urban Y

1 1.0 Rural N

2 1.0 Urban Y

3 1.0 Urban Y

4 1.0 Urban Y

.. ... ... ...

609 1.0 Rural Y

610 1.0 Rural Y

611 1.0 Urban Y

612 1.0 Urban Y

613 0.0 Semiurban N

[614 rows x 13 columns]>

loan\_dataset.isnull().sum()

Loan\_ID 0

Gender 13

Married 3

Dependents 15

Education 0

Self\_Employed 32

ApplicantIncome 0

CoapplicantIncome 0

LoanAmount 22

Loan\_Amount\_Term 14

Credit\_History 50

Property\_Area 0

Loan\_Status 0

dtype: int64

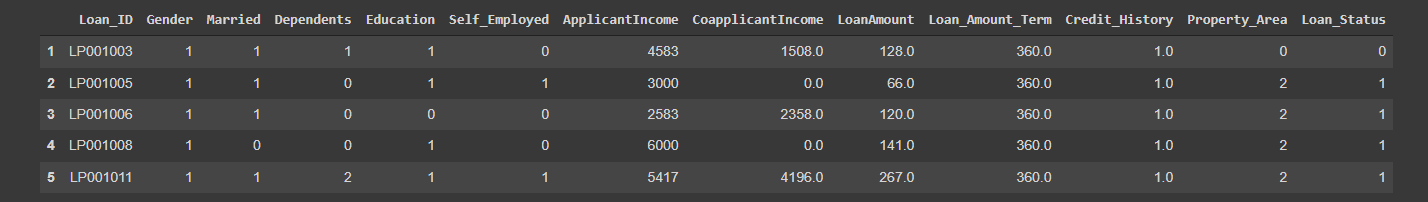
loan\_dataset.replace({"Loan\_Status":{'N':0,'Y':1}},inplace=True)

loan\_dataset = loan\_dataset.replace(to\_replace='3+', value=4)

loan\_dataset.replace({'Married':{'No':0,'Yes':1},'Gender':{'Male':1,'Female':0},'Self\_Employed':{'No':0,'Yes':1},

                      'Property\_Area':{'Rural':0,'Semiurban':1,'Urban':2},'Education':{'Graduate':1,'Not Graduate':0}},inplace=True)

loan\_dataset.head()

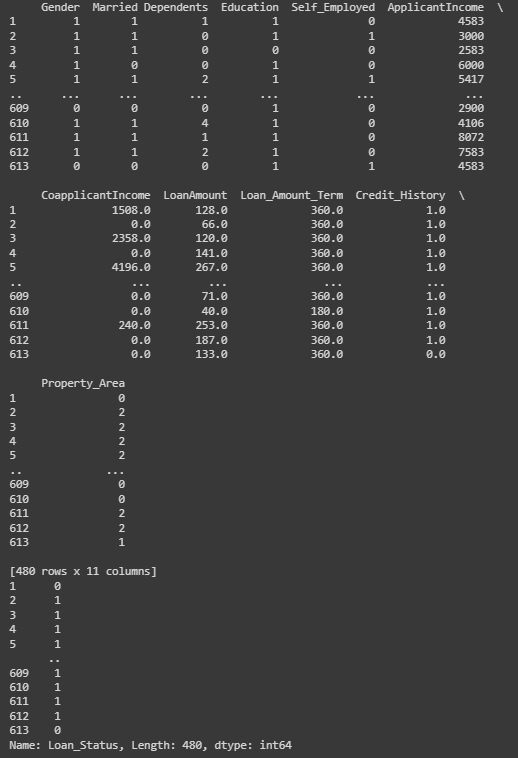


X = loan\_dataset.drop(columns=['Loan\_ID','Loan\_Status'],axis=1)

Y = loan\_dataset['Loan\_Status']

print(X)

print(Y)



scaler = StandardScaler()

scaler.fit(X)

standardized\_data = scaler.transform(X)

X = standardized\_data

X\_train, X\_test,Y\_train,Y\_test = train\_test\_split(X,Y,test\_size=0.1,stratify=Y,random\_state=2)

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

(480, 11) (432, 11) (48, 11)

# SVM:

classifier = svm.SVC(kernel='linear')

classifier.fit(X\_train,Y\_train)

X\_train\_prediction = classifier.predict(X\_train)

training\_data\_accuray = accuracy\_score(X\_train\_prediction,Y\_train)

print('Accuracy on training data : ', training\_data\_accuray)

Accuracy on training data : 0.8055555555555556

X\_test\_prediction = classifier.predict(X\_test)

test\_data\_accuray = accuracy\_score(X\_test\_prediction,Y\_test)

print('Accuracy on test data : ', test\_data\_accuray)

Accuracy on test data : 0.8333333333333334

new\_id=(1,1,4,1,0,3036,2504,2000,360,0,1)

new\_id1=np.asarray(new\_id)

new\_id\_reshape=new\_id1.reshape(1,-1)

st\_data=scaler.transform(new\_id\_reshape)

prediction=classifier.predict(new\_id\_reshape)

if (prediction[0] == 0):

  print('The loan is rejectd')

else:

  print('The loan is approved')

The loan is approved

/usr/local/lib/python3.7/dist-packages/sklearn/base.py:451: UserWarning: X does not have valid feature names, but StandardScaler was fitted with feature names

"X does not have valid feature names, but"

# LOGISTIC REGRESSION:

model = LogisticRegression()

model.fit(X\_train, Y\_train)

LogisticRegression()

pred\_test = model.predict(X\_test)

accuracy\_score(Y\_test,pred\_test)

0.8333333333333334

# RANDOM FOREST:

from sklearn.ensemble import RandomForestClassifier

random = RandomForestClassifier()

random.fit(X\_train,Y\_train)



RF = random.score(X\_test, Y\_test)

print("Random Forest: {}".format(RF))

Random Forest: 0.8333333333333334

# DECISION TREE:

from sklearn import tree

model = tree.DecisionTreeClassifier()

model.fit(X\_train,Y\_train)



model.score(X\_test,Y\_test)

0.7708333333333334

# NAIVE BAYES:

from sklearn.naive\_bayes import GaussianNB

classifier = GaussianNB()

classifier.fit(X\_train,Y\_train)



RF = classifier.score(X\_test, Y\_test)

print("Naive Bayes: {}".format(RF))

Naive Bayes: 0.8125

# XGBOOST:

from numpy import loadtxt

from xgboost import XGBClassifier

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

from sklearn.metrics import accuracy\_score

model = XGBClassifier()

model.fit(X\_train, Y\_train)



y\_pred = model.predict(X\_test)

predictions = [round(value) for value in y\_pred]

accuracy = accuracy\_score(Y\_test, predictions)

print("Accuracy: %.2f%%" % (accuracy \* 100.0))

Accuracy: 79.17%