LOAN PREDICTION USING MACHINE LEARING ALGORITHMS (BATCH-9)

Group Members:

Nagineni Sai Lasya -19BCE7493

CHADALAWADA V V KASYAP-20BCE7457

HARSHAVARDINI K - 20BCE7439

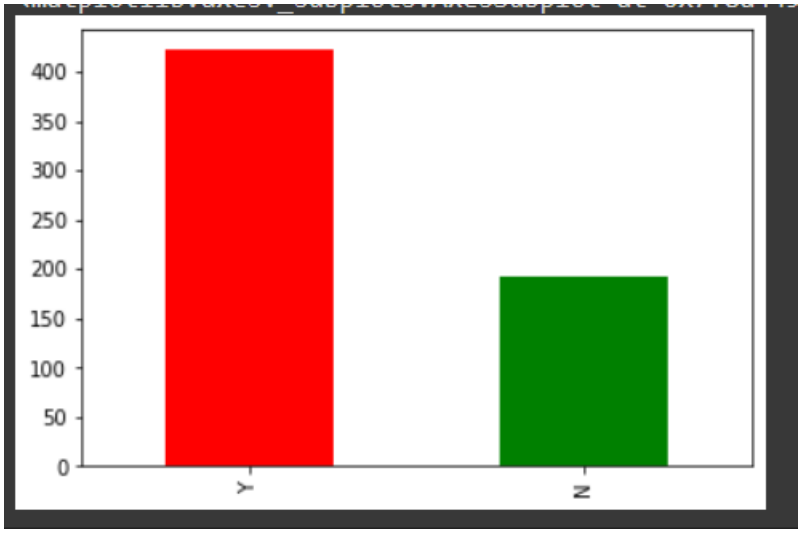
CHITTURI SRIJA CHOWDARY-20BCE7583

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
```

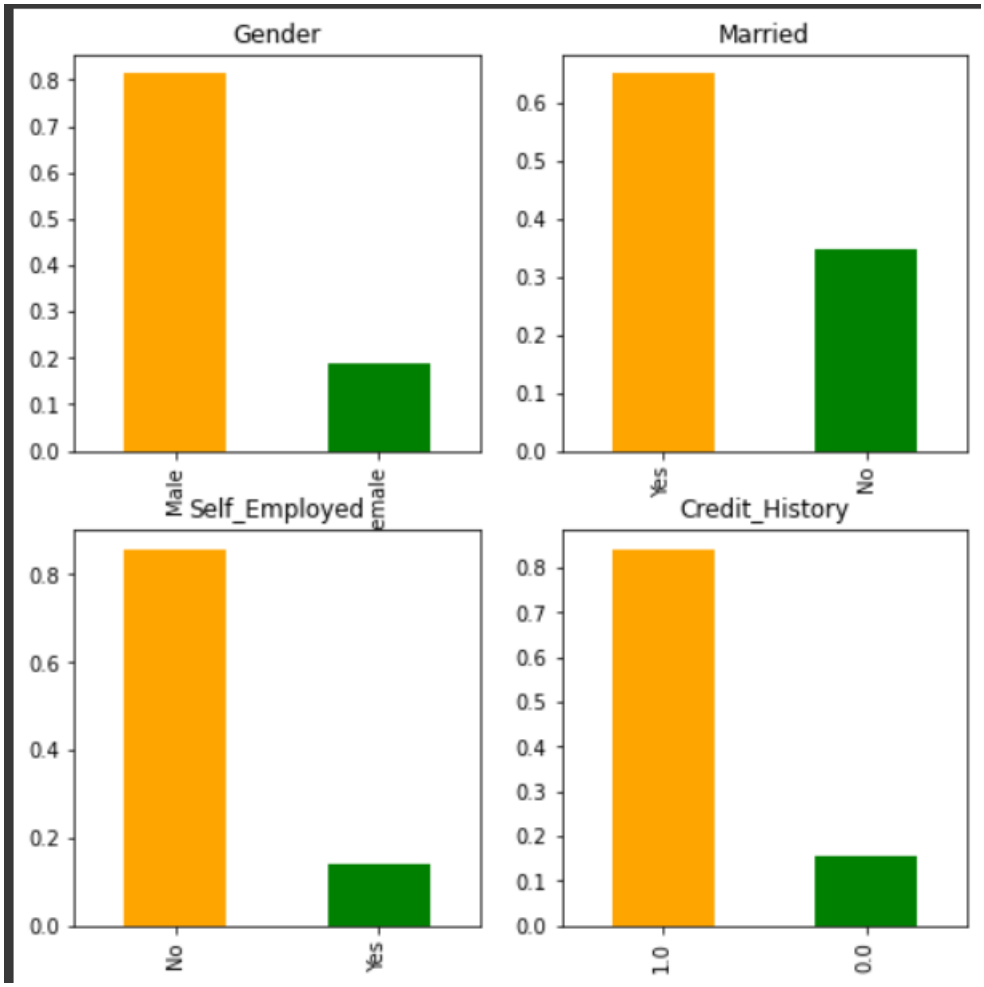
```
Index(['Loan_ID', 'Gender', 'Married', 'Dependents', 'Education',
       'Self_Employed', 'ApplicantIncome', 'CoapplicantIncome',
'LoanAmount',
       'Loan_Amount_Term', 'Credit_History', 'Property_Area',
'Loan_Status'],
      dtype='object')
```

```
load_dataset["Loan_Status"].value_counts().plot.bar(color=["red","green
"])
```

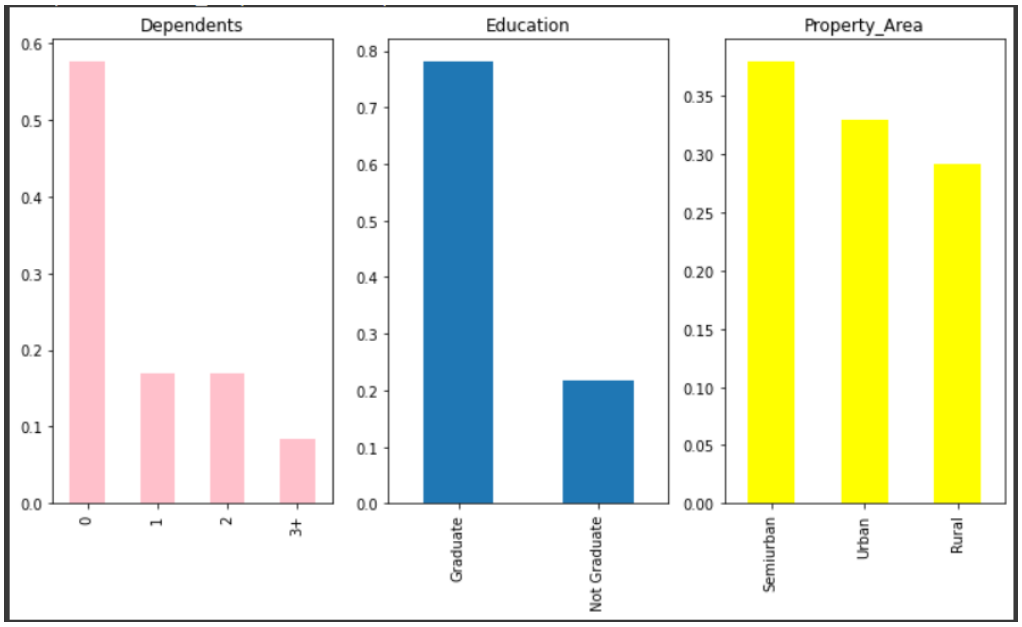



```
plt.figure(1)
plt.subplot(221)
load_dataset["Gender"].value_counts(normalize='True').plot.bar(figsize=
(8,8),title="Gender",color=["orange","green"])
plt.subplot(222)
```

```
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(f
igsize=(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()
```

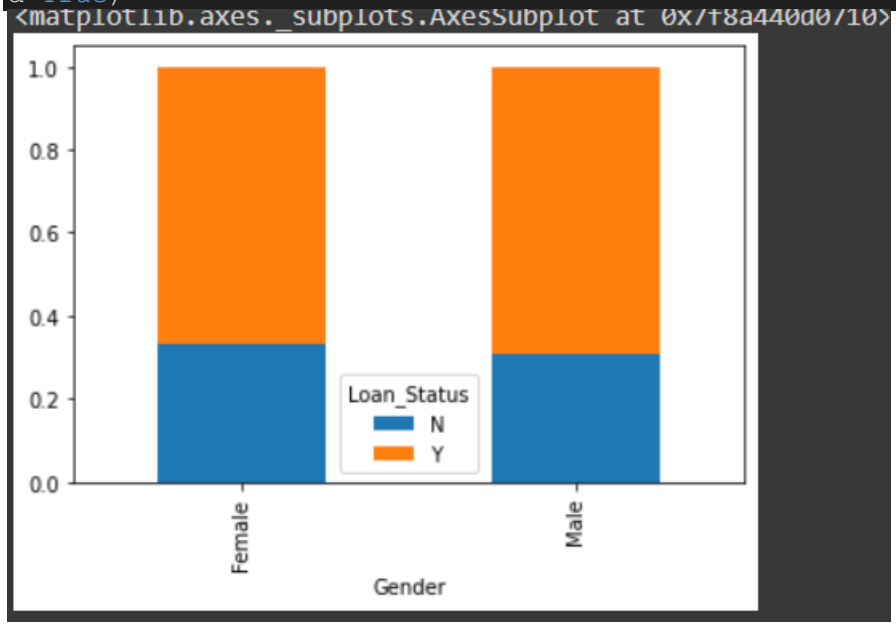



```
plt.figure(1)
plt.subplot(131)
load_dataset["Dependents"].value_counts(normalize=True).plot.bar(figsiz
e=(12,6),color="pink",title="Dependents")
plt.subplot(132)
load_dataset["Education"].value_counts(normalize=True).plot.bar(figsize
=(12,6),title="Education")
plt.subplot(133)
load_dataset["Property_Area"].value_counts(normalize=True).plot.bar(fig
size=(12,6),color="yellow",title="Property_Area")
```



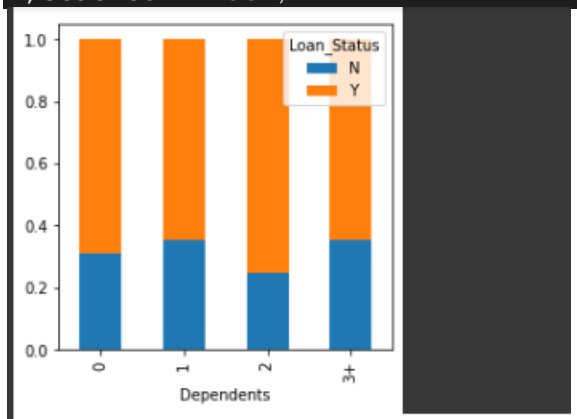

```
Gender=pd.crosstab(load_dataset['Gender'],train['Loan_Status'])
Gender.div(Gender.sum(1).astype(float), axis=0).plot(kind="bar", stacke
d=True)
```

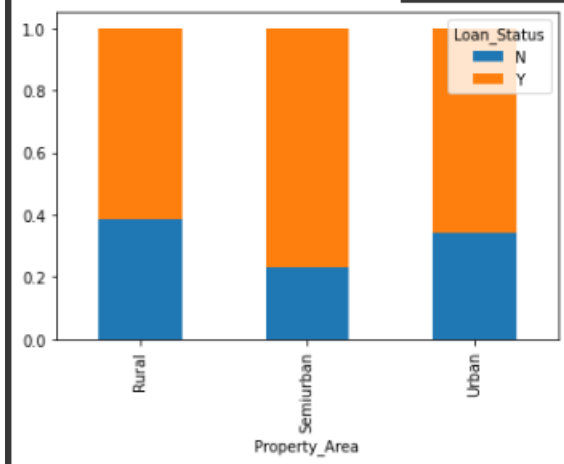
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8a440d0710>
```

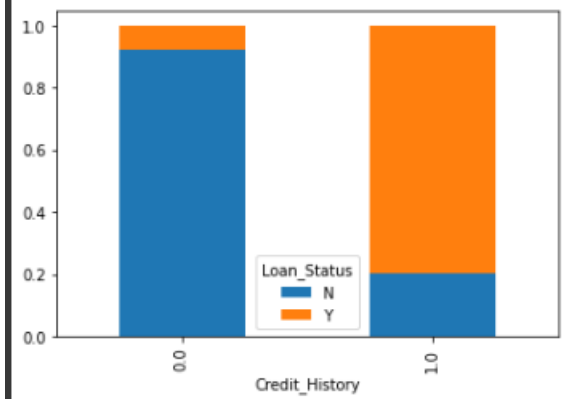



```
Married=pd.crosstab(load_dataset['Married'],train['Loan_Status'])
Married.div(Married.sum(1).astype(float), axis=0).plot(kind="bar", stac
ked=True, figsize=(4,4))
```

```
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,'Fe
male':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()
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
1	LP001003	1	1	1	1	0	4583	1508.0	128.0	360.0	1.0	0	0
2	LP001005	1	1	0	1	1	3000	0.0	66.0	360.0	1.0	2	1
3	LP001006	1	1	0	0	0	2583	2358.0	120.0	360.0	1.0	2	1
4	LP001008	1	0	0	1	0	6000	0.0	141.0	360.0	1.0	2	1
5	LP001011	1	1	2	1	1	5417	4196.0	267.0	360.0	1.0	2	1

```
X = loan_dataset.drop(columns=['Loan_ID','Loan_Status'],axis=1)
Y = loan_dataset['Loan_Status']
print(X)
print(Y)
```

```
     Gender  Married Dependents  Education  Self_Employed  ApplicantIncome  \
1         1        1          1          1              0             4583
2         1        1          0          1              1             3000
3         1        1          0          0              0             2583
4         1        0          0          1              0             6000
5         1        1          2          1              1             5417
..      ...      ...        ...        ...            ...              ...
609       0        0          0          1              0             2900
610       1        1          4          1              0             4106
611       1        1          1          1              0             8072
612       1        1          2          1              0             7583
613       0        0          0          1              1             4583

     CoapplicantIncome  LoanAmount  Loan_Amount_Term  Credit_History  \
1               1508.0       128.0             360.0             1.0
2                  0.0        66.0             360.0             1.0
3               2358.0       120.0             360.0             1.0
4                  0.0       141.0             360.0             1.0
5               4196.0       267.0             360.0             1.0
..                 ...         ...               ...             ...
609                0.0        71.0             360.0             1.0
610                0.0        40.0             180.0             1.0
611              240.0       253.0             360.0             1.0
612                0.0       187.0             360.0             1.0
613                0.0       133.0             360.0             0.0

     Property_Area
1                0
2                2
3                2
4                2
5                2
..             ...
609              0
610              0
611              2
612              2
613              1

[480 rows x 11 columns]
1      0
2      1
3      1
4      1
5      1
      ..
609    1
610    1
611    1
612    1
613    0
Name: Loan_Status, Length: 480, dtype: int64
```

```
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,str
atify=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)
```

```
RandomForestClassifier()
```

```
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)
```

```
DecisionTreeClassifier()
```

```
model.score(X_test,Y_test)
```

```
0.7708333333333334
```

NAIVE BAYES:

```
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train,Y_train)
```

```
GaussianNB()
```

```
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)
```

```
XGBClassifier()
```

```
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%
```