Start coding or generate with AI.
#Importing Neccessaryy library and module
#Importing Classifier from sklearn.ensemble import RandomForestClassifier
#Importing numpy for multidimensional array import numpy as np
#importing matplotlib for Plotting Graphimport matplotlib.pyplot as plt
#Loadig model selection for splitting the dataset from sklearn.model_selection import train_test_split
#Importing models for classification efficiency from sklearn.metrics import accuracy_score, classification_report
#======Importing library for dealing with DataFrame===== import pandas as pd
#Loading the dataset loan = pd.read_csv('/content/sample_data/loan.csv')
#Visualising the Dataset

loan.head(5)												
₹		Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_Hist
	0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	
	1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	
	2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	
	3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	
	4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	

len(loan)

⋺ 614

111111

We have to check if the dataset contain null values so as we can replace them with mean values to improve our classification

and

we have to Encode string with Numerical values such as Gender and Education and Loan status $\,$

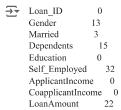
 $\overline{\Rightarrow}$

"\nWe have to check if the dataset contain null values so as we \ncan replace them with mean values to improve our classification \n\nand \n\nwe have to Encode string with Numerical values such as Gender and Education\nand Loan status\n'

#-----Checking the presence of Null Values-----

#==== We use built in Function isnull() and sum()

loan.isnull().sum()





```
Loan Amount Term
      Credit History
                      50
      Property_Area
                       0
      Loan Status
                       0
      dtype: int64
import pandas as pd
from sklearn.impute import SimpleImputer
# Assuming 'loan' is your DataFrame
# Replace '3+' with 3 in the 'Dependents' column
loan['Dependents'] = loan['Dependents'].replace('3+', 3)
# Selecting the columns to be imputed
loan_n = loan.iloc[:, 8:11]
# Create a SimpleImputer instance with strategy 'mean'
imputer = SimpleImputer(missing_values=pd.NA, strategy='mean')
# Fit the imputer on loan_n and transform the data
loan_n = imputer.fit_transform(loan_n)
# Converting the numpy array back to a DataFrame
loan\_d = pd.DataFrame(loan\_n, columns=loan.columns[8:11])
# Assign the imputed values back to the original DataFrame
loan['LoanAmount'] = loan\_d['LoanAmount']
loan['Loan_Amount_Term'] = loan_d['Loan_Amount_Term']
loan['Credit_History'] = loan_d['Credit_History']
# Display the resulting DataFrame
print(loan.head())
\overline{\mathcal{Z}}
        0 LP001002 Male
                          No
                                   0 Graduate
                                                      No
      1 LP001003 Male
                          Yes
                                   1
                                       Graduate
                                                      No
      2 LP001005 Male
                          Yes
                                       Graduate
                                                      Yes
      3 LP001006 Male
                          Yes
                                   0 Not Graduate
                                                       No
      4 LP001008 Male
                                   0 Graduate
                          No
                                                      No
       ApplicantIncome CoapplicantIncome LoanAmount Loan_Amount_Term \
      0
              5849
                           0.0 146.412162
                                                 360.0
              4583
                          1508.0 128.000000
                                                  360.0
      2
              3000
                           0.0 66.000000
                                                 360.0
                          2358.0 120.000000
                                                  360.0
      3
              2583
                           0.0 141.000000
                                                 360.0
      4
              6000
        Credit_History Property_Area Loan_Status
      0
              1.0
                      Urban
      1
              1.0
                      Rural
                                 Ν
      2
                      Urban
                                 Y
      3
              1.0
                      Urban
                                 Y
      4
              1.0
                      Urban
loan.columns
     Index(['Loan_ID', 'Gender', 'Married', 'Dependents', 'Education',
          'Self_Employed', 'ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
          'Loan\_Amount\_Term', 'Credit\_History', 'Property\_Area', 'Loan\_Status'],
         dtype='object')
Start coding or generate with AI.
loan.isnull().sum()
    Loan_ID
                      0
      Gender
                     13
      Married
                     3
      Dependents
                      15
      Education
                      0
      Self Employed
      ApplicantIncome
      CoapplicantIncome
                        0
      LoanAmount
      Loan_Amount_Term
```

4

Credit_History

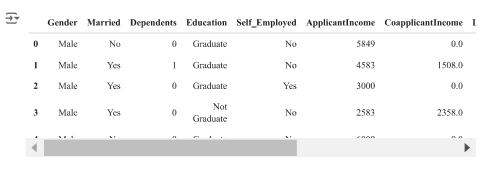
```
0
      Property Area
                         0
      Loan Status
      dtype: int64
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.impute import SimpleImputer
# Assuming 'loan' is your DataFrame
# Replace '3+' with 3 in the 'Dependents' column
loan['Dependents'] = loan['Dependents'].replace('3+', 3)
dependat = loan['Dependents'] # Define 'dependat'
# Selecting the columns to be imputed
loan_n = loan.iloc[:, 8:11]
# Create a SimpleImputer instance with strategy 'mean'
imputer = SimpleImputer (missing\_values = pd.NA, strategy = 'mean')
# Fit the imputer on loan n and transform the data
loan\_n = imputer.fit\_transform(loan\_n)
# Converting the numpy array back to a DataFrame
loan_d = pd.DataFrame(loan_n, columns=loan.columns[8:11])
# Assign the imputed values back to the original DataFrame
loan['LoanAmount'] = loan_d['LoanAmount']
loan['Loan\_Amount\_Term'] = loan\_d['Loan\_Amount\_Term']
loan['Credit\_History'] = loan\_d['Credit\_History']
# Fill missing values for categorical columns
loan['Self Employed'] = loan['Self Employed'].fillna("Yes")
loan['Property\_Area'] = loan['Property\_Area'].fillna("Rural")
loan['Gender'] = loan["Gender"].fillna("Male")
loan['Married'] = loan["Married"].fillna("Yes")
loan['Education'] = loan["Education"].fillna("Graduate")
loan['Dependents'] = loan['Dependents'].fillna(1)
# Initialize LabelEncoder
label_encoder = LabelEncoder()
# Encode categorical columns
loan['Gender'] = label_encoder.fit_transform(loan['Gender'])
loan['Married'] = label encoder.fit transform(loan['Married'])
loan['Self Employed'] = label encoder.fit transform(loan['Self Employed'])
loan['Property\_Area'] = label\_encoder.fit\_transform(loan['Property\_Area'])
loan['Education'] = label encoder.fit transform(loan['Education'])
loan['Loan\_Status'] = label\_encoder.fit\_transform(loan['Loan\_Status'])
# Display the resulting DataFrame
print(loan.head())
         Loan_ID Gender Married Dependents Education Self_Employed \
      0 LP001002
                             0
                                    0
                                           0
                                                    0
      1 LP001003
                                    1
                                           0
                                                    0
      2 LP001005
                             1
                                    0
                                           0
                                                    1
      3 LP001006
                                    0
                                                    0
                                           1
      4 LP001008
                                           0
                                                    0
                                    0
        ApplicantIncome CoapplicantIncome LoanAmount Loan Amount Term \
      0
               5849
                             0.0 146.412162
                                                     360.0
                4583
                                                      360.0
                            1508.0 128.000000
                3000
                             0.0 66.000000
                                                    360.0
      3
               2583
                            2358.0 120.000000
                                                      360.0
      4
                             0.0 141.000000
                                                     360.0
               6000
        Credit_History Property_Area Loan_Status
      0
                1.0
                                   1
      1
                1.0
                           0
                                   0
      2
                1.0
                           2
                                   1
                           2
      3
               1.0
                                   1
      4
               1.0
```

 $loan['Education'], loan['Self_Employed'], loan['Property_Area'], loan['Married''], loan['Loan_Status'], loan['Dependents'] = education, employ, P_area, gender, status, loan_status, loan_Status'], loan['Dependents'] = education, employ, P_area, gender, status, loan_status, loa$

loan.head() loan.to_csv("Loan Processed data.csv")

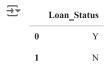
data.head(5)

data = loan.iloc[:, 1:-1] target = loan.iloc[:, -1:]



7

target.head(2)



data.to_csv("Data.csv")
target.to_csv("Target.csv")

 $x_train, x_test, y_train, y_test = train_test_split(data, target, test_size = 1/5, random_state = 10)$

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy score
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive bayes import MultinomialNB, GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
# Assuming 'loan' is your DataFrame
# Replace '3+' with 3 in the 'Dependents' column
loan['Dependents'] = loan['Dependents'].replace('3+', 3)
# Fill missing values for categorical columns
loan['Self_Employed'] = loan['Self_Employed'].fillna("Yes")
loan['Property_Area'] = loan['Property_Area'].fillna("Rural")
loan['Gender'] = loan["Gender"].fillna("Male")
loan['Married'] = loan["Married"].fillna("Yes")
loan['Education'] = loan["Education"].fillna("Graduate")
loan['Dependents'] = loan['Dependents'].fillna(1)
# Initialize LabelEncoder
label encoder = LabelEncoder()
# Encode categorical columns
loan['Gender'] = label_encoder.fit_transform(loan['Gender'])
loan['Married'] = label encoder.fit transform(loan['Married'])
loan['Self\_Employed'] = label\_encoder.fit\_transform(loan['Self\_Employed'])
loan['Property\_Area'] = label\_encoder.fit\_transform(loan['Property\_Area'])
loan['Education'] = label encoder.fit transform(loan['Education'])
loan['Loan_Status'] = label_encoder.fit_transform(loan['Loan_Status'])
# Define features (X) and target (y)
X = loan.drop(columns=['Loan_Status', 'Loan_ID']) # Exclude 'Loan_ID' as it is not a feature
y = loan['Loan_Status']
# Split the data into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Train and evaluate different models
results = \{\}
# Random Forest Classifier
model a = RandomForestClassifier()
model a.fit(x train, y train)
pred = model_a.predict(x_test)
results['Random Forest Classifier'] = accuracy_score(y_test, pred)
# K Nearest Neighbors
model_knn = KNeighborsClassifier()
model_knn.fit(x_train, y_train)
pred = model knn.predict(x test)
results['K Nearest Neighbors'] = accuracy_score(y_test, pred)
# Multinomial Naive Bayes
model_mnb = MultinomialNB()
model_mnb.fit(x_train, y_train)
pred = model\_mnb.predict(x\_test)
results['Multinomial NB'] = accuracy_score(y_test, pred)
# Gaussian Naive Bayes
model gnb = GaussianNB()
model_gnb.fit(x_train, y_train)
pred = model \ gnb.predict(x \ test)
results['Gaussian NB'] = accuracy_score(y_test, pred)
# Logistic Regression
model_lr = LogisticRegression(max_iter=200)
model_lr.fit(x_train, y_train)
pred = model lr.predict(x test)
results['Logistic Regression'] = accuracy_score(y_test, pred)
# Decision Tree Classifier
model dtc = DecisionTreeClassifier()
model_dtc.fit(x_train, y_train)
```



predictions = model.predict(m)

print(predictions)

```
nred = model_dtc nredict(x_test)
!pip install joblib
import joblib
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (1.4.2)
  print(1 \text{inoder_name}, \text{accuracy..+1} )
joblib.dump(model_a, "Forest.pkl")
→ ['Forest.pkl']
       Muitinomiai NB: U.48/8
n = data.iloc[:,:].values \\
      Decision Tree Classifier: 0.7236
m = n[0]
m = np.array([m])
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import load_iris
# Load sample data (replace this with your actual data)
iris = load\_iris()
X, y = iris.data, iris.target
# Define and train the model
model = RandomForestClassifier()
model.fit(X, y)
# Now you can use the model to make predictions
m = \hbox{\tt [[5.1, 3.5, 1.4, 0.2]] \# Example input data}
```

