PROJECT : Loan Status Prediction

Import required libraries

In [ ]:
```
import numpy as np
import pandas as pd
df=pd.read_csv('/content/LoanApprovalPrediction.csv')
df
```

Out[ ]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	1.0	Urban	Y
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y
...	...	...	...	...	...	...	...	...	...	...	...	...	...
609	LP002978	Female	No	0	Graduate	No	2900	0.0	71.0	360.0	1.0	Rural	Y
610	LP002979	Male	Yes	3+	Graduate	No	4106	0.0	40.0	180.0	1.0	Rural	Y
611	LP002983	Male	Yes	1	Graduate	No	8072	240.0	253.0	360.0	1.0	Urban	Y
612	LP002984	Male	Yes	2	Graduate	No	7583	0.0	187.0	360.0	1.0	Urban	Y
613	LP002990	Female	No	0	Graduate	Yes	4583	0.0	133.0	360.0	0.0	Semiurban	N

614 rows × 13 columns

In [ ]:
```
df.head()
```

Out[ ]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	1.0	Urban	Y
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y

In [ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Loan_ID            614 non-null    object
 1   Gender             601 non-null    object
 2   Married            611 non-null    object
 3   Dependents         599 non-null    object
 4   Education          614 non-null    object
 5   Self_Employed      582 non-null    object
 6   ApplicantIncome    614 non-null    int64
 7   CoapplicantIncome  614 non-null    float64
 8   LoanAmount         592 non-null    float64
 9   Loan_Amount_Term   600 non-null    float64
 10  Credit_History     564 non-null    float64
 11  Property_Area      614 non-null    object
 12  Loan_Status        614 non-null    object
dtypes: float64(4), int64(1), object(8)
memory usage: 62.5+ KB
```

Checking missing values

In [ ]:
```
df.isna().sum()
```

Out[ ]:
```
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
```

In [ ]:
```
df.dtypes
```

Out[ ]:
```
Loan_ID              object
Gender               object
Married              object
Dependents           object
Education            object
Self_Employed        object
ApplicantIncome       int64
CoapplicantIncome   float64
LoanAmount          float64
Loan_Amount_Term    float64
Credit_History      float64
Property_Area        object
Loan_Status          object
dtype: object
```

Fill the missing value

In [ ]:
```
col=['Gender','Married','Dependents','Self_Employed']
for i in col:
  df[i]=df[i].fillna(df[i].mode()[0])
```

In [ ]:
```
col1=['LoanAmount','Loan_Amount_Term','Credit_History']
for i in col1:
  df[i]=df[i].fillna(df[i].mean())
```

In [ ]:
```
df.isna().sum()
```

Out[ ]:
```
Loan_ID              0
Gender               0
Married              0
Dependents           0
Education            0
Self_Employed        0
ApplicantIncome      0
CoapplicantIncome    0
LoanAmount           0
Loan_Amount_Term     0
Credit_History       0
Property_Area        0
Loan_Status          0
dtype: int64
```

Encoding using LabelEncoder

In [ ]:
```
col3=['Loan_ID','Gender','Married','Dependents','Education','Self_Employed','Property_Area']
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
for i in col3:
  df[i]=le.fit_transform(df[i])
df
```

Out[ ]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	0	1	0	0	0	0	5849	0.0	146.412162	360.0	1.0	2	Y
1	1	1	1	1	0	0	4583	1508.0	128.000000	360.0	1.0	0	N
2	2	1	1	0	0	1	3000	0.0	66.000000	360.0	1.0	2	Y
3	3	1	1	0	1	0	2583	2358.0	120.000000	360.0	1.0	2	Y
4	4	1	0	0	0	0	6000	0.0	141.000000	360.0	1.0	2	Y
...	...	...	...	...	...	...	...	...	...	...	...	...	...
609	609	0	0	0	0	0	2900	0.0	71.000000	360.0	1.0	0	Y
610	610	1	1	3	0	0	4106	0.0	40.000000	180.0	1.0	0	Y
611	611	1	1	1	0	0	8072	240.0	253.000000	360.0	1.0	2	Y
612	612	1	1	2	0	0	7583	0.0	187.000000	360.0	1.0	2	Y
613	613	0	0	0	0	1	4583	0.0	133.000000	360.0	0.0	1	N

614 rows × 13 columns

Separate input features and target

In [ ]:
```
x=df.iloc[:,:-1]
x
```

Out[ ]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area
0	0	1	0	0	0	0	5849	0.0	146.412162	360.0	1.0	2
1	1	1	1	1	0	0	4583	1508.0	128.000000	360.0	1.0	0
2	2	1	1	0	0	1	3000	0.0	66.000000	360.0	1.0	2
3	3	1	1	0	1	0	2583	2358.0	120.000000	360.0	1.0	2
4	4	1	0	0	0	0	6000	0.0	141.000000	360.0	1.0	2
...	...	...	...	...	...	...	...	...	...	...	...	...
609	609	0	0	0	0	0	2900	0.0	71.000000	360.0	1.0	0
610	610	1	1	3	0	0	4106	0.0	40.000000	180.0	1.0	0
611	611	1	1	1	0	0	8072	240.0	253.000000	360.0	1.0	2
612	612	1	1	2	0	0	7583	0.0	187.000000	360.0	1.0	2
613	613	0	0	0	0	1	4583	0.0	133.000000	360.0	0.0	1

614 rows × 12 columns

In [ ]:
```
y=df.iloc[:,-1]
y
```

Out[ ]:
```
0      Y
1      N
2      Y
3      Y
4      Y
      ..
609    Y
610    Y
611    Y
612    Y
613    N
Name: Loan_Status, Length: 614, dtype: object
```

Feature selection using chi-squared

In [ ]:
```
from sklearn.feature_selection import chi2
score=chi2(x,y)
score
```

Out[ ]:
```
(array([8.72362729e+00, 3.62343084e-02, 1.78242499e+00, 8.59527587e-02,
        3.54050246e+00, 7.28480330e-03, 9.39049635e+01, 1.13420416e+04,
        3.92115449e+01, 3.26731521e+00, 2.59988672e+01, 3.77837464e-01]),
 array([3.14112478e-03, 8.49032435e-01, 1.81851834e-01, 7.69386856e-01,
        5.98873168e-02, 9.31982300e-01, 3.31042936e-22, 0.00000000e+00,
        3.80284952e-10, 7.06731966e-02, 3.41617751e-07, 5.38762867e-01]))
```

In [ ]:
```
f_value=pd.Series(score[0],index=x.columns)
f_value.sort_values(ascending=False)
```

Out[ ]:
```
CoapplicantIncome    11342.041603
ApplicantIncome         93.904964
LoanAmount              39.211545
Credit_History          25.998867
Loan_ID                  8.723627
Education                3.540502
Loan_Amount_Term         3.267315
Married                  1.782425
Property_Area            0.377837
Dependents               0.085953
Gender                   0.036234
Self_Employed            0.007285
dtype: float64
```

In [ ]:
```
s_value=pd.Series(score[1],index=x.columns)
s_value.sort_values(ascending=False)
```

Out[ ]:
```
Self_Employed        9.319823e-01
Gender               8.490324e-01
Dependents           7.693869e-01
Property_Area        5.387629e-01
Married              1.818518e-01
Loan_Amount_Term     7.067320e-02
Education            5.988732e-02
Loan_ID              3.141125e-03
Credit_History       3.416178e-07
LoanAmount           3.802850e-10
ApplicantIncome      3.310429e-22
CoapplicantIncome    0.000000e+00
dtype: float64
```

Remove unwanted columns

In [ ]:
```
df.drop(['Loan_ID','Married','Self_Employed','Gender','Property_Area','Dependents','Education'],axis=1,inplace=True)
df.head()
```

Out[ ]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Loan_Status
0	5849	0.0	146.412162	360.0	1.0	Y
1	4583	1508.0	128.000000	360.0	1.0	N
2	3000	0.0	66.000000	360.0	1.0	Y
3	2583	2358.0	120.000000	360.0	1.0	Y
4	6000	0.0	141.000000	360.0	1.0	Y

Separate input features and target After Feature selection

In [ ]:
```
x=df.iloc[:,:-1].values
y=df.iloc[:,-1].values
```

Training & Testing data

In [ ]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30,random_state=42)
x_train
```

Out[ ]:
```
array([[2.45400000e+03, 2.33300000e+03, 1.81000000e+02, 3.60000000e+02,
        0.00000000e+00],
       [2.89400000e+03, 2.79200000e+03, 1.55000000e+02, 3.60000000e+02,
        1.00000000e+00],
       [2.06000000e+03, 2.20900000e+03, 1.34000000e+02, 3.60000000e+02,
        1.00000000e+00],
       ...,
       [3.23700000e+03, 0.00000000e+00, 3.00000000e+01, 3.60000000e+02,
        1.00000000e+00],
       [1.00470000e+04, 0.00000000e+00, 1.46412162e+02, 2.40000000e+02,
        1.00000000e+00],
       [1.36500000e+04, 0.00000000e+00, 1.46412162e+02, 3.60000000e+02,
        1.00000000e+00]])
```

In [ ]:
```
x_test
y_train
y_test
```

Out[ ]:
```
array(['Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'N', 'N', 'Y', 'Y',
       'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y',
       'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'N', 'N', 'Y', 'Y',
       'N', 'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'N', 'N', 'N', 'N', 'N', 'N',
       'N', 'N', 'Y', 'N', 'N', 'Y', 'N', 'Y', 'Y', 'N', 'N', 'Y', 'Y',
       'Y', 'Y', 'Y', 'N', 'N', 'N', 'Y', 'N', 'N', 'N', 'Y', 'Y', 'N',
       'N', 'Y', 'N', 'Y', 'N', 'N', 'Y', 'Y', 'Y', 'N', 'Y', 'N', 'N',
       'N', 'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'Y',
       'Y', 'N', 'Y', 'N', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'Y',
       'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'N', 'N', 'Y',
       'N', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'N',
       'N', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'N', 'N', 'Y', 'Y',
       'Y', 'Y', 'N', 'N', 'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'N',
       'Y', 'N', 'N', 'N', 'N', 'Y', 'Y', 'Y', 'N', 'Y', 'N', 'Y', 'Y',
       'Y', 'Y', 'N'], dtype=object)
```

Data Normalization

In [ ]:
```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
scaler.fit(x_train)
x_train=scaler.transform(x_train)
x_test=scaler.transform(x_test)
```

Model Creation using

- KNN
- Naivebayes
- SVM

In [ ]:
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import BernoulliNB
from sklearn.svm import SVC
knn=KNeighborsClassifier(n_neighbors=7)
model=BernoulliNB()
model3=SVC()
lst=[knn,model,model3]
```

Performance Evaluation

In [ ]:
```
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report
for i in lst:
  i.fit(x_train,y_train)
  y_pred=i.predict(x_test)
  print(i)
  print(confusion_matrix(y_test,y_pred))
  print("accuracy_score:",accuracy_score(y_test,y_pred))
  print("classification_report:",classification_report(y_test,y_pred))
```

```
KNeighborsClassifier(n_neighbors=7)
[[ 27  38]
 [  2 118]]
accuracy_score: 0.7837837837837838
classification_report:               precision    recall  f1-score   support

           N       0.93      0.42      0.57        65
           Y       0.76      0.98      0.86       120

    accuracy                           0.78       185
   macro avg       0.84      0.70      0.71       185
weighted avg       0.82      0.78      0.76       185

BernoulliNB()
[[ 35  30]
 [ 12 108]]
accuracy_score: 0.772972972972973
classification_report:               precision    recall  f1-score   support

           N       0.74      0.54      0.62        65
           Y       0.78      0.90      0.84       120

    accuracy                           0.77       185
   macro avg       0.76      0.72      0.73       185
weighted avg       0.77      0.77      0.76       185

SVC()
[[ 27  38]
 [  2 118]]
accuracy_score: 0.7837837837837838
classification_report:               precision    recall  f1-score   support

           N       0.93      0.42      0.57        65
           Y       0.76      0.98      0.86       120

    accuracy                           0.78       185
   macro avg       0.84      0.70      0.71       185
weighted avg       0.82      0.78      0.76       185
```