Loan Prediction

About Company

Dream Housing Finance company deals in all home loans. They have presence across all urban, semi urban and rural areas. Customer first apply for home loan after that company validates the customer eligibility for loan.

Problem

Company wants to automate the loan eligibility process (real time) based on customer detail provided while filling online application form. These details are Gender, Marital Status, Education, Number of Dependents, Income, Loan Amount, Credit History and others. To automate this process, they have given a problem to identify the customers segments, those are eligible for loan amount so that they can specifically target these customers. Here they have provided a partial data set.

In [1]:
```
import numpy as np
import pandas as pd
```

Reading the file

In [2]:
```
# Importing traing data
train_data=pd.read_csv(r'F:\Prthon Programming\Loan Prediction\train.csv',inde
x_col=0,
                       header=0)


# Importing traing data
test_data=pd.read_csv(r'F:\Prthon Programming\Loan Prediction\test.csv',
                       index_col=0,header=0)
```

In [3]:
```
train_data.head()
```

Out[3]:

Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coapplica
LP001002	Male	No	0	Graduate	No	5849	
LP001003	Male	Yes	1	Graduate	No	4583	
LP001005	Male	Yes	0	Graduate	Yes	3000	
LP001006	Male	Yes	0	Not Graduate	No	2583	
LP001008	Male	No	0	Graduate	No	6000	

In [4]:
```
test_data.head()
```

Out[4]:

Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coapplica
LP001015	Male	Yes	0	Graduate	No	5720	
LP001022	Male	Yes	1	Graduate	No	3076	
LP001031	Male	Yes	2	Graduate	No	5000	
LP001035	Male	Yes	2	Graduate	No	2340	
LP001051	Male	No	0	Not Graduate	No	3276	

In [5]:
```
train_data.shape
```

Out[5]: (614, 12)

In [6]:
```
test_data.shape
```

Out[6]: (367, 11)

Finding the missing value

In [7]:
```
train_data.isnull().sum()
```

Out[7]:
```
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 [8]:
```
test_data.isnull().sum()
```

Out[8]:
```
Gender               11
Married               0
Dependents           10
Education             0
Self_Employed        23
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount            5
Loan_Amount_Term      6
Credit_History       29
Property_Area         0
dtype: int64
```

In [9]:
```
for value in ['Gender','Married','Dependents','Self_Employed','Loan_Amount_Ter
m']:
    train_data[value].fillna(train_data[value].mode()[0],inplace=True)

train_data['LoanAmount'].fillna(round(train_data['LoanAmount'].mean(),0),inpla
ce=True)

train_data['Credit_History'].fillna(value=0,inplace=True)
```

In [10]:
```
train_data.isnull().sum()
```

Out[10]:
```
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
```

In [11]:
```
for value in ['Gender','Dependents','Self_Employed','Loan_Amount_Term']:
    test_data[value].fillna(test_data[value].mode()[0],inplace=True)

test_data['LoanAmount'].fillna(round(test_data['LoanAmount'].mean(),0),inplace
=True)

test_data['Credit_History'].fillna(value=0,inplace=True)
```

In [12]:
```
test_data.isnull().sum()
```

Out[12]:
```
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
dtype: int64
```

Convert Data from Factors to Numerical

In [13]:
```
colname=[]
for x in train_data.columns[:]:
    if train_data[x].dtype=='object':
        colname.append(x)
colname
```

Out[13]:
```
['Gender',
 'Married',
 'Dependents',
 'Education',
 'Self_Employed',
 'Property_Area',
 'Loan_Status']
```

In [14]:
```
from sklearn import preprocessing

le = preprocessing.LabelEncoder()

for x in colname:
    train_data[x]=le.fit_transform(train_data[x])
```

In [15]:
```
train_data.head()
```

Out[15]:

Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coapplica
LP001002	1	0	0	0	0	5849	
LP001003	1	1	1	0	0	4583	
LP001005	1	1	0	0	1	3000	
LP001006	1	1	0	1	0	2583	
LP001008	1	0	0	0	0	6000	

In [16]:
```
colname=[]
for x in test_data.columns[:]:
    if test_data[x].dtype=='object':
        colname.append(x)
colname
```

Out[16]:
```
['Gender',
 'Married',
 'Dependents',
 'Education',
 'Self_Employed',
 'Property_Area']
```

In [17]:
```
from sklearn import preprocessing

le = preprocessing.LabelEncoder()

for x in colname:
    test_data[x]=le.fit_transform(test_data[x])
```

In [18]:
```
test_data.head()
```

Out[18]:

Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coapplica
LP001015	1	1	0	0	0	5720	
LP001022	1	1	1	0	0	3076	
LP001031	1	1	2	0	0	5000	
LP001035	1	1	2	0	0	2340	
LP001051	1	0	0	1	0	3276	

Creating (X) And (Y)

In [19]:
```
X_train = train_data.values[:,0:-1]
Y_train = train_data.values[:,-1]
Y_train = Y_train.astype(int)
Y_train.shape
```

Out[19]: (614,)

In [20]:
```
X_test = test_data.values[:,:]
```

Scaler

In [21]:
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

scaler.fit(X_train)

X_train = scaler.transform(X_train)

X_test = scaler.transform(X_test)

print(X_train)
```

```
[[ 0.47234264 -1.37208932 -0.73780632 ...  0.2732313   0.54095432
   1.22329839]
 [ 0.47234264  0.72881553  0.25346957 ...  0.2732313   0.54095432
  -1.31851281]
 [ 0.47234264  0.72881553 -0.73780632 ...  0.2732313   0.54095432
   1.22329839]
 ...
 [ 0.47234264  0.72881553  0.25346957 ...  0.2732313   0.54095432
   1.22329839]
 [ 0.47234264  0.72881553  1.24474546 ...  0.2732313   0.54095432
   1.22329839]
 [-2.11710719 -1.37208932 -0.73780632 ...  0.2732313  -1.84858491
  -0.04760721]]
```

In [22]:
```
from sklearn import svm

svc_model = svm.SVC(kernel='rbf',C=1.0,gamma=0.1)

svc_model.fit(X_train,Y_train)

Y_pred = svc_model.predict(X_test)

print(list(Y_pred))
```

```
[1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1,
1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1,
1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1,
1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1,
1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1,
1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1,
1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1,
1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1,
1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0,
1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1,
1, 0, 1, 1]
```

Adding column in test file Loan_status

In [23]:
```
new_test_data=pd.read_csv(r'F:\Prthon Programming\Loan Prediction\test.csv',in
dex_col=0,
                          header=0)

new_test_data["Loan_Status"]=Y_pred

new_test_data.head()
```

Out[23]:

Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coapplica
LP001015	Male	Yes	0	Graduate	No	5720	
LP001022	Male	Yes	1	Graduate	No	3076	
LP001031	Male	Yes	2	Graduate	No	5000	
LP001035	Male	Yes	2	Graduate	No	2340	
LP001051	Male	No	0	Not Graduate	No	3276	

In [24]:
```
classifier=svm.SVC(kernel='rbf',C=1.0,gamma=0.1)

#performing kfold_cross_validation
from sklearn.model_selection import KFold
kfold_cv=KFold(n_splits=10)
print(kfold_cv)

from sklearn.model_selection import cross_val_score
#running the model using scoring metric as accuracy
kfold_cv_result=cross_val_score(estimator=classifier,X=X_train,
y=Y_train, cv=kfold_cv)
print(kfold_cv_result)
#finding the mean
print(kfold_cv_result.mean())
```

```
KFold(n_splits=10, random_state=None, shuffle=False)
[0.74193548 0.82258065 0.75806452 0.72580645 0.7704918  0.67213115
 0.75409836 0.75409836 0.78688525 0.80327869]
0.7589370703331569
```

In [25]:
```
classifier=svm.SVC(kernel='rbf',C=10.0,gamma=0.001)

#performing kfold_cross_validation
from sklearn.model_selection import KFold
kfold_cv=KFold(n_splits=10)
print(kfold_cv)

from sklearn.model_selection import cross_val_score
#running the model using scoring metric as accuracy
kfold_cv_result=cross_val_score(estimator=classifier,X=X_train,
y=Y_train, cv=kfold_cv)
print(kfold_cv_result)
#finding the mean
print(kfold_cv_result.mean())
```

```
KFold(n_splits=10, random_state=None, shuffle=False)
[0.77419355 0.82258065 0.74193548 0.72580645 0.7704918  0.68852459
 0.80327869 0.7704918  0.7704918  0.83606557]
0.770386039132734
```

In [26]:
```
from sklearn.linear_model import LogisticRegression

import warnings
warnings.filterwarnings("ignore")

classifier=(LogisticRegression())

from sklearn.model_selection import KFold
kfold_cv=KFold(n_splits=10)
print(kfold_cv)

from sklearn.model_selection import cross_val_score
#running the model using scoring metric as accuracy
kfold_cv_result=cross_val_score(estimator=classifier,X=X_train,
y=Y_train, cv=kfold_cv)
print(kfold_cv_result)
#finding the mean
print(kfold_cv_result.mean())
```

```
KFold(n_splits=10, random_state=None, shuffle=False)
[0.77419355 0.82258065 0.74193548 0.72580645 0.7704918  0.68852459
 0.80327869 0.7704918  0.78688525 0.83606557]
0.772025383395029
```

In [27]:
```
new_test_data=new_test_data.drop(['Gender','Married','Dependents','Education',
'Self_Employed','ApplicantIncome',
                                  'CoapplicantIncome','LoanAmount','Loan_Amoun
t_Term','Credit_History','Property_Area'],axis=1)
print(new_test_data)
```

Loan_ID	Loan_Status
LP001015	1
LP001022	1
LP001031	1
LP001035	0
LP001051	1
LP001054	1
LP001055	1
LP001056	0
LP001059	1
LP001067	1
LP001078	1
LP001082	1
LP001083	0
LP001094	0
LP001096	1
LP001099	1
LP001105	1
LP001107	1
LP001108	1
LP001115	1
LP001121	1
LP001124	1
LP001128	1
LP001135	1
LP001149	1
LP001153	1
LP001163	0
LP001169	1
LP001174	0
LP001176	1
...	...
LP002856	1
LP002857	1
LP002858	0
LP002860	1
LP002867	1
LP002869	1
LP002870	1
LP002876	1
LP002878	1
LP002879	0
LP002885	1
LP002890	1
LP002891	1
LP002899	1
LP002901	1
LP002907	1
LP002920	1
LP002921	0
LP002932	1
LP002935	1
LP002952	1
LP002954	0
LP002962	1
LP002965	0

```
LP002969            1
LP002971            1
LP002975            1
LP002980            0
LP002986            1
LP002989            1

[367 rows x 1 columns]
```

In [28]:
```
new_test_data.to_csv(r'F:\Prthon Programming\Loan Prediction\Pred_test.csv')
```