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
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.metrics import accuracy_score
```

In [2]:
```
#loading the dataset
loan_dataset = pd.read_csv("Loan Dataset.csv")
```

In [3]:
```
type(loan_dataset)
```

Out[3]: pandas.core.frame.DataFrame

In [4]:
```
#Visualize the data
loan_dataset.head()
```

Out[4]:

	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 [5]:
```
#Dependence mean(0) -they are not dependent
#(1) - they are dependent
#Loan amount are in $
#credit_history -(0) they have low credit score
#(1) - means they have good credit score
```

In [7]:
```
#Checking total no of rows and columns
loan_dataset.shape
```

Out[7]: (614, 13)

In [8]:
```
#get some stats measures
loan_dataset.describe()
```

Out[8]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
count	614.000000	614.000000	592.000000	600.00000	564.000000
mean	5403.459283	1621.245798	146.412162	342.00000	0.842199
std	6109.041673	2926.248369	85.587325	65.12041	0.364878
min	150.000000	0.000000	9.000000	12.00000	0.000000
25%	2877.500000	0.000000	100.000000	360.00000	1.000000
50%	3812.500000	1188.500000	128.000000	360.00000	1.000000
75%	5795.000000	2297.250000	168.000000	360.00000	1.000000
max	81000.000000	41667.000000	700.000000	480.00000	1.000000

In [9]:
```
#25% people having income less than 2877$
#50% people having income less then 3812
```

In [11]:
```
#checking the missing values
loan_dataset.isnull().sum()
```

Out[11]:
```
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 [12]:
```
#Here we are not taking the mean value because the values are in the form of categorical(yes,no)
#So we are droping the missing values
```

In [13]:
```
loan_dataset=loan_dataset.dropna()
```

In [14]:
```
#Now again checking the missing values
loan_dataset.isnull().sum()
```

Out[14]:
```
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
```

In [15]:
```
#Visualization of data(label encoding)
#now we are doing "Z" for yes and "N" for no, changing the categorical values to numerical values
```

In [16]:
```
loan_dataset.replace({"Loan_Status":{'N':0,'Y':1}},inplace=True)
```

In [17]:
```
loan_dataset.head()
```

Out[17]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	0
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	1
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	1
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	1
5	LP001011	Male	Yes	2	Graduate	Yes	5417	4196.0	267.0	360.0	1.0	Urban	1

In [18]:
```
#Dependence column values
loan_dataset['Dependents'].value_counts()
```

Out[18]:
```
0     274
2      85
1      80
3+     41
Name: Dependents, dtype: int64
```

In [19]:
```
#274 people do not have any dependency
#85 people have 2 dependency
#80 people have 1 dependency
#and 41 people have more than 3 dependency
```

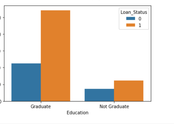
In [20]:
```
#Replacing the value of 3+ to 4
loan_dataset = loan_dataset.replace(to_replace='3+', value=4)
```

In [21]:
```
#Dependent values
loan_dataset['Dependents'].value_counts()
```

Out[21]:
```
0    274
2     85
1     80
4     41
Name: Dependents, dtype: int64
```

In [22]:
```
#Data visualization
#Education and loan status
sns.countplot(x="Education", hue="Loan_Status",data=loan_dataset)
```

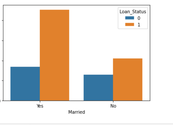
Out[22]:
```
<AxesSubplot:xlabel='Education', ylabel='count'>
```



In [23]:
```
#0-Loan is Rejected
#1-Loan is approved
```

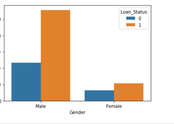
In [24]:
```
# # marital status and loan status
sns.countplot(x="Married", hue="Loan_Status",data=loan_dataset)
```

Out[24]:
```
<AxesSubplot:xlabel='Married', ylabel='count'>
```



In [25]:
```
#Gender and loan status
sns.countplot(x="Gender", hue="Loan_Status", data=loan_dataset)
```

Out[25]:
```
<AxesSubplot:xlabel='Gender', ylabel='count'>
```



In [26]:
```
#Convert text values to the numbers(Like graduate or not, yes or No)
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)
```

In [27]:
```
loan_dataset.head()
```

Out[27]:

	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

In [28]:
```
#So all the values are in numerical
```

In [30]:
```
##Seperating the data and label
X = loan_dataset.drop(columns=["Loan_ID","Loan_Status"],axis=1)
Y = loan_dataset['Loan_Status']
```

In [31]:
```
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
```

In [32]:
```
#Splitting the data into train and test data
X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size=0.2,stratify=Y,random_state=2)
```

In [33]:
```
print(X.shape, X_train.shape, X_test.shape)
```

```
(480, 11) (384, 11) (96, 11)
```

In [35]:
```
#Training the Model using SVM

classifier = svm.SVC(kernel="linear")

#Training the svm
classifier.fit(X_train,Y_train)
```

Out[36]: SVC(kernel='linear')

In [38]:
```
#Evaluating the accuracy of the model
X_train_prediction = classifier.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction, Y_train)
```

In [39]:
```
print("Accuracy on training data : ", training_data_accuracy)
```

```
Accuracy on training data :  0.7786458333333334
```

In [40]:
```
#We can get high accuracy if we have more data
```

In [42]:
```
#Evaluating the test data
X_test_prediction = classifier.predict(X_test)
test_data_accuracy = accuracy_score(X_test_prediction, Y_test)
```

In [43]:
```
print("Accuracy on test data : ", test_data_accuracy)
```

```
Accuracy on test data :  0.8125
```

In [ ]: