LOAN STATUS PREDICTION

Importing required libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Importing data into notebook

In [2]:
```
df = pd.read_csv("train_u6lujuX_CVtuZ9i (1).csv")
df.head()
```

Out[2]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAı
0	LP001002	Male	No	0	Graduate	No	5849	0.0	
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	
4	LP001008	Male	No	0	Graduate	No	6000	0.0	

Evaluating the Dataset

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

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

In [4]:
```
df.shape
```

Out[4]:
```
(614, 13)
```

In [5]:
```
df.describe()
```

Out[5]:

	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 [6]:
```
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
```

EXPLORATORY DATA ANALYSIS

Finding Missing Values

In [7]:
```
df.isna().sum()
```

Out[7]:
```
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
```

Dropping the missing Values

In [8]:
```
df = df.dropna()
```

In [9]:
```
df.isna().sum()
```

Out[9]:
```
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
```

Converting Target Feature to numeric in nature

In [10]:
```
#The Loan Status is attributed into Categorical column, so we need to convert for further
df.replace({"Loan_Status":{"N":0, "Y":1}}, inplace = True)
```

In [11]:
```
df.head()
```

Out[11]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAı
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	
4	LP001008	Male	No	0	Graduate	No	6000	0.0	
5	LP001011	Male	Yes	2	Graduate	Yes	5417	4196.0	

In [12]:
```
df['Dependents'].value_counts()
```

Out[12]:
```
0     274
2      85
1      80
3+     41
Name: Dependents, dtype: int64
```

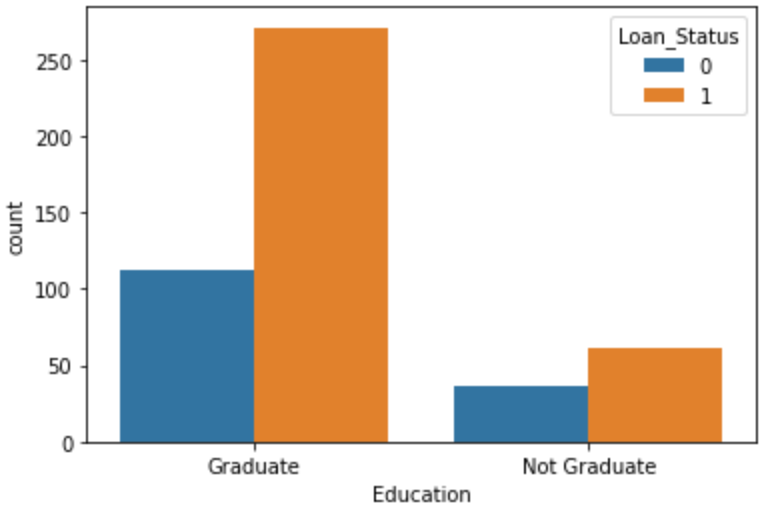
In [13]:
```
#The 3+ number will present errors while handling into model, so conversion is required
df = df.replace(to_replace = '3+', value = 4)
```

In [14]:
```
df['Dependents'].value_counts()
```

Out[14]:
```
0    274
2     85
1     80
4     41
Name: Dependents, dtype: int64
```

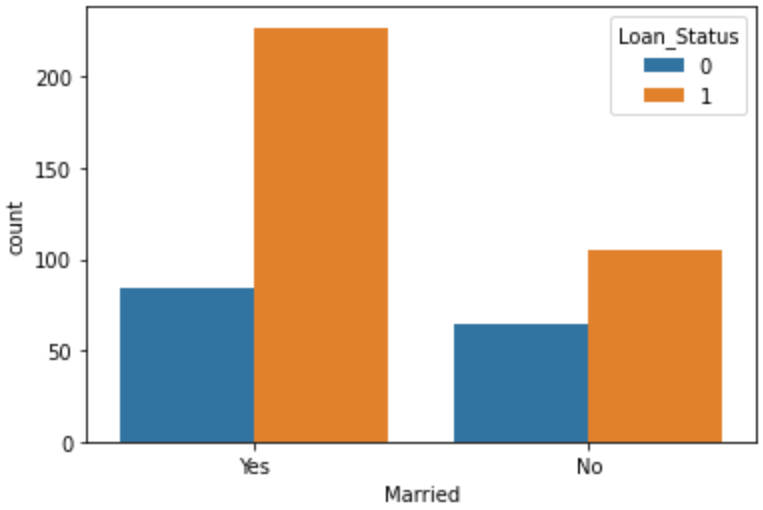
Visualuizing the Segments for Education

In [15]:
```
sns.countplot(x = "Education", hue = "Loan_Status", data = df)
plt.show()
```



Visualuizing the Segments for Marriage Segment

In [16]:
```
sns.countplot(x = "Married", hue = "Loan_Status", data = df)
plt.show()
```



Removing non required Feature

In [17]:
```
df.drop("Loan_ID", axis = 1, inplace = True)
```

In [18]:
```
df
```

Out[18]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount
1	Male	Yes	1	Graduate	No	4583	1508.0	128.0
2	Male	Yes	0	Graduate	Yes	3000	0.0	66.0
3	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0
4	Male	No	0	Graduate	No	6000	0.0	141.0
5	Male	Yes	2	Graduate	Yes	5417	4196.0	267.0
...	...	...	...	...	...	...	...	...
609	Female	No	0	Graduate	No	2900	0.0	71.0

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	
610	Male	Yes	4	Graduate	No	4106	0.0	40.0	
611	Male	Yes	1	Graduate	No	8072	240.0	253.0	
612	Male	Yes	2	Graduate	No	7583	0.0	187.0	
613	Female	No	0	Graduate	Yes	4583	0.0	133.0	

480 rows × 12 columns

The dataset requires to be submerged into numerical Values

In [19]:
```
df = pd.get_dummies(df, drop_first = True)
```

In [20]:
```
df
```

Out[20]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Loan_Status	Gender_Ma
1	4583	1508.0	128.0	360.0	1.0	0	
2	3000	0.0	66.0	360.0	1.0	1	
3	2583	2358.0	120.0	360.0	1.0	1	
4	6000	0.0	141.0	360.0	1.0	1	
5	5417	4196.0	267.0	360.0	1.0	1	
...	...	...	...	...	...	...	
609	2900	0.0	71.0	360.0	1.0	1	
610	4106	0.0	40.0	180.0	1.0	1	
611	8072	240.0	253.0	360.0	1.0	1	
612	7583	0.0	187.0	360.0	1.0	1	
613	4583	0.0	133.0	360.0	0.0	0	

480 rows × 15 columns

Splitting data into X And Y features

In [21]:
```
X = df.drop("Loan_Status", axis = 1)
y =df["Loan_Status"]
```

In [22]:
```
X
```

Out[22]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Gender_Male	Married_Y
1	4583	1508.0	128.0	360.0	1.0	1	
2	3000	0.0	66.0	360.0	1.0	1	
3	2583	2358.0	120.0	360.0	1.0	1	
4	6000	0.0	141.0	360.0	1.0	1	

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Gender_Male	Married_Y
5	5417	4196.0	267.0	360.0	1.0	1	
...	...	...	...	...	...	...	
609	2900	0.0	71.0	360.0	1.0	0	
610	4106	0.0	40.0	180.0	1.0	1	
611	8072	240.0	253.0	360.0	1.0	1	
612	7583	0.0	187.0	360.0	1.0	1	
613	4583	0.0	133.0	360.0	0.0	0	

480 rows × 14 columns

In [23]:
```
y
```

Out[23]:
```
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 [24]:
```
#Importing train_test_split to the dataset for implementing into the model
from sklearn.model_selection import train_test_split
```

In [25]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.1, random_state =
```

In [26]:
```
X_train.head()
```

Out[26]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Gender_Male	Married_Y
54	11500	0.0	286.0	360.0	0.0	0	
78	3167	4000.0	180.0	300.0	0.0	1	
325	8666	4983.0	376.0	360.0	0.0	1	
352	2666	2083.0	95.0	360.0	1.0	1	
359	5167	3167.0	200.0	360.0	1.0	1	

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

Out[27]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Gender_Male	Married_Y
552	3333	3250.0	158.0	360.0	1.0	1	

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Gender_Male	Married_Y
389	5488	0.0	125.0	360.0	1.0	1	
546	3358	0.0	80.0	36.0	1.0	1	
234	3155	1779.0	140.0	360.0	1.0	1	
40	3600	0.0	80.0	360.0	1.0	1	

In [28]:
```
y_train.head()
```

Out[28]:
```
54     0
78     0
325    0
352    1
359    1
Name: Loan_Status, dtype: int64
```

In [29]:
```
y_test.head()
```

Out[29]:
```
552    1
389    1
546    0
234    1
40     0
Name: Loan_Status, dtype: int64
```

In [30]:
```
X.shape
```

Out[30]:
```
(480, 14)
```

In [31]:
```
X_train.shape
```

Out[31]:
```
(432, 14)
```

In [32]:
```
X_test.shape
```

Out[32]:
```
(48, 14)
```

Importing SVM Library

In [33]:
```
from sklearn import svm
```

In [34]:
```
#assigning the SVM Classifier to a variable
classifier = svm.SVC(kernel='linear')
```

In [35]:
```
classifier
```

Out[35]:
```
SVC(kernel='linear')
```

In [36]:
```
#Fitting the data to the classifier model
classifier.fit(X_train, y_train)
```

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

In [37]:
```
#For Checking how accuarate the model is, we need to import accuracy score
from sklearn.metrics import accuracy_score
```

In [38]:
```
#predicting the value for the required Target Variable
X_train_prediction = classifier.predict(X_train)
```

In [39]:
```
#Calculating the Accuracy for the model
training_data_accuracy  =  accuracy_score(X_train_prediction, y_train)
```

In [40]:
```
print (" the Accuaracy for the Training Dataset is ",round(training_data_accuracy*100,2),'
```

```
 the Accuaracy for the Training Dataset is  78.47 %
```

In [41]:
```
#Similarly applying the testing dataset to the model
X_test_prediction = classifier.predict(X_test)
test_data_accuracy  =  accuracy_score(X_test_prediction, y_test)
```

In [42]:
```
print (" the Accuaracy for the Test Dataset is ",round(test_data_accuracy*100,2),"%")
```

```
 the Accuaracy for the Test Dataset is  81.25 %
```

Thus the Accuracy for the model performed here is predicting 81.25% Accurately.

In [ ]:
```

```