Importing the Dependencies

In [43]:
```python
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
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
```

Data Collection and Processing

In [2]:
```python
# Loading the dataset to pandas DataFrame
loan_dataset = pd.read_csv(r'C:\Users\student\Downloads\dataset\loan-status
loan_dataset
```

Out[2]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coa
0	LP001002	Male	No	0	Graduate	No	5849	
1	LP001003	Male	Yes	1	Graduate	No	4583	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	
3	LP001006	Male	Yes	0	Not Graduate	No	2583	
4	LP001008	Male	No	0	Graduate	No	6000	
5	LP001011	Male	Yes	2	Graduate	Yes	5417	
6	LP001013	Male	Yes	0	Not Graduate	No	2333	
7	LP001014	Male	Yes	3+	Graduate	No	3036	
8	LP001018	Male	Yes	2	Graduate	No	4006	
9	LP001020	Male	Yes	1	Graduate	No	12841	

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

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

In [4]:
```
# printing the first 5 rows of the dataframe
loan_dataset.head()
```

Out[4]:

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

In [5]:
```
# printing the last 5 rows of the dataframe
loan_dataset.tail()
```

Out[5]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coapp
609	LP002978	Female	No	0	Graduate	No	2900	
610	LP002979	Male	Yes	3+	Graduate	No	4106	
611	LP002983	Male	Yes	1	Graduate	No	8072	
612	LP002984	Male	Yes	2	Graduate	No	7583	
613	LP002990	Female	No	0	Graduate	Yes	4583	

In [6]:
```
# number of rows and columns
loan_dataset.shape
```

Out[6]:

(614, 13)

In [7]:
```
# statistical measures
loan_dataset.describe()
```

Out[7]:

	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 [8]:
```
# number of missing values in each column
loan_dataset.isnull().sum()
```

Out[8]:
```
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 [9]:
```
# dropping the missing values
loan_dataset = loan_dataset.dropna()
```

In [10]:
```
# number of missing values in each column
loan_dataset.isnull().sum()
```

Out[10]:
```
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 [ ]:

```
# label encoding
loan_dataset.replace({"Loan_Status":{'N':0,'Y':1}},inplace=True)
```

In [12]:

```
# Dependent column values
loan_dataset['Dependents'].value_counts()
```

Out[12]:

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

[13]:

```
# replacing the value of 3+ to 4 loan_dataset =
loan_dataset.replace(to_replace='3+', value=4)
```

In [14]:

```
# dependent values
loan_dataset['Dependents'].value_counts()
```

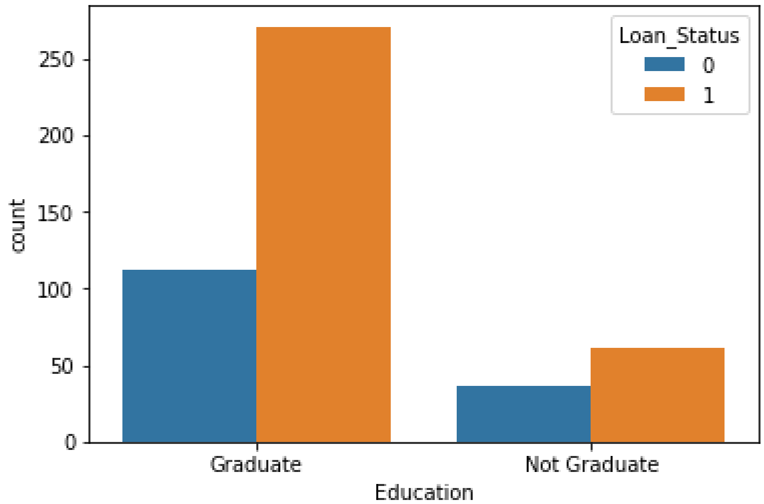
Out[14]:

```
0     274
2      85
1      80
4      41
Name: Dependents, dtype: int64
```

Data Visualization

In [15]:
```
# education & Loan Status
sns.countplot(x='Education',hue='Loan_Status',data=loan_dataset)
```

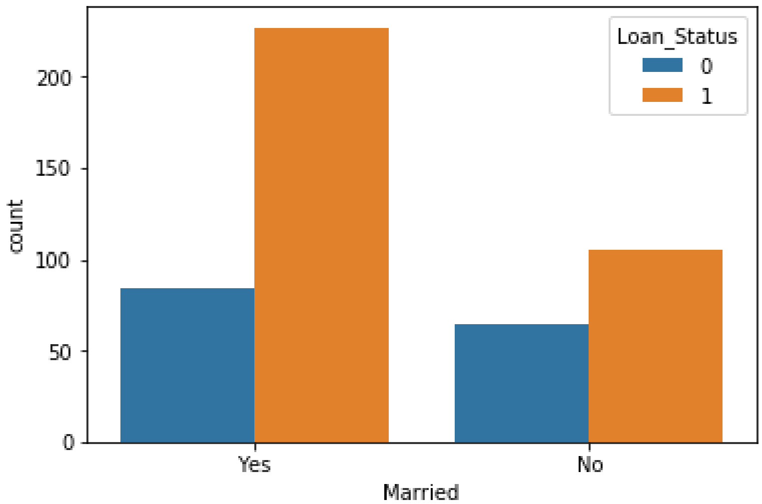
Out[15]: <matplotlib.axes._subplots.AxesSubplot at 0xa1b8370>



[16]:
```
# marital status & Loan Status
sns.countplot(x='Married',hue='Loan_Status',data=loan_dataset)
```

Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0xa49fdf0>

In [17]:



In [18]:

```
# convert categorical columns to numerical values
loan_dataset.replace({'Married':{'No':0,'Yes':1},'Gender':{'Male':1,'Female
                      'Property_Area':{'Rural':0,'Semiurban':1,'Urban':2},'
```

```
loan_dataset.head()
```

Out[18]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coapplic
1	LP001003	1	1	1	1	0	4583	
2	LP001005	1	1	0	1	1	3000	
3	LP001006	1	1	0	0	0	2583	
4	LP001008	1	0	0	1	0	6000	
5	LP001011	1	1	2	1	1	5417	

In [19]:

```
# separating the data and label
X = loan_dataset.drop(columns=['Loan_ID','Loan_Status'],axis=1)
Y = loan_dataset['Loan_Status']
```

[20]:

```
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
6        1        1          0          0              0             2333
7        1        1          4          1              0             3036
8        1        1          2          1              0             4006
9        1        1          1          1              0            12841
10       1        1          2          1              0             3200
12       1        1          2          1              0             3073
13       1        0          0          1              0             1853
14       1        1          2          1              0             1299
15       1        0          0          1              0             4950
17       0        0          0          1              0             3510
18       1        1          0          0              0             4887
20       1        1          0          0              0             7660
21       1     1      1      1      0      5955
```

Train Test Split

In [21]:

```
X_train, X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.1,stratif
```

In [22]:

```
print(X.shape, X_train.shape, X_test.shape)
```

```
(480, 11) (432, 11) (48, 11)
```

Training the model:

Support Vector Machine Model

In [33]:

```
clf = DecisionTreeClassifier()
clf
```

Out[33]:

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
  max_features=None, max_leaf_nodes=None,
  min_impurity_decrease=0.0, min_impurity_split=None,
  min_samples_leaf=1, min_samples_split=2,
```

[34]:

```
clf.fit(X_train, Y_train)
```

Out[34]:
```
DecisionTreeClassifier(class_weight=None, criterion='gini',
    max_depth=None,            max_features=None, max_leaf_nodes=None,
    min_impurity_decrease=0.0, min_impurity_split=None,
    min_samples_leaf=1, min_samples_split=2,
                min_weight_fraction_leaf=0.0, presort=False,
    random_state=None,            splitter='best')
```

Model Evaluation

In [36]:

```
y_pred = clf.predict(X_test)
y_pred
```

Out[36]:
```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1,
 0, 1, 1,       0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1,
                                                          1, 0, 1,
           1, 1, 1, 1], dtype=int64)
```

In [37]:

```
accuracy = accuracy_score(Y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.7291666666666666
```

[38]:
```
1 result_loan_dataset = pd.DataFrame({'Actual': Y_test, 'Predicted': y_pred}) 2 result_loan_dataset
```

Out[38]:

	Actual	Predicted
368	1	1
74	1	1
135	0	1
53	0	1
96	1	1
388	1	1
345	1	1

8	1	1
549	1	1
99	1	0
49	1	1
513	0	1
43	1	1
92	1	1
555	1	1
609	1	1
221	1	1
454	1	0
607	1	1
179	0	0
277	1	1
488	1	1
150	0	0
585	0	1
168	0	0
267	1	0
543	1	1
520	1	1
22	0	0
69	0	0
91	1	0
250	0	1
416	0	1
154	1	1
415	1	1
291	0	0
253	1	0
97	1	1
393	1	1
399	0	1
537	1	1

15	1	1
469	0	0
371	1	1
300	0	1
189	1	1
224	1	1
201	1	1

In [40]:

```
cm = confusion_matrix(Y_test, y_pred)
cm
```

Out[40]:

```
array([[ 7,  8],
       [ 5, 28]], dtype=int64)
```

In [44]:

```
# Create a heatmap for the confusion matrix
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Y','N'], y
plt.xlabel('Predicted label')
plt.ylabel('True label')
plt.title('Confusion Matrix')
plt.show()
```