There is a finance company that gives loan for people.So before approving the loan this company analyzes various credential of the person.There are several aspects like whether the person is graduated or not/Whether the perosn is eduacated or not/Whether he is married or not or he is a single perosn.So there are several parameters that finance company looks.So this company wants to automate this Loan approval process.SO the user or person who wants the loan will fill an online application form and based on the information given by the user so we need to develop a machine learning system that can tell the company that this perosn is eligible for a loan or this person is not eligible for a loan,this is the problem statement.

Work Flow
- First we need the data.
- Data Preprocessing(Make the data suitable for ML Model)

We can not give raw data to the Machine Learning Model

- Train Test Split
- Once we split the data into train - test we will feed this training data to ML Model.
- We use Support Vector Machine which is a supervised learning model

In this case there are two labels,the one is the loan will be approved and the other is the loan will be rejected.

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
```

Data Collection and Preprocessing

In [2]:
```
# Loading the dataset
loan_dataset = pd.read_csv(r'C:\Users\Arvind Kumar Yadav\Downloads\train_loan.csv')
```

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

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

In [4]:
```
loan_dataset.head()
# 1 represents very good Credit history
# 0 represents not good Credit histrory
```

Out[4]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	

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

Out[5]: (614, 13)

In [6]:
```
loan_dataset.describe()
```

Out[6]:

	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 [7]:
```
# Missing values
loan_dataset.isnull().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
```

In [8]:
```
# Dropping Missing values
loan_dataset = loan_dataset.dropna()
```

In [9]:
```
loan_dataset.isnull().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
```

In [10]:
```
# label encoding
loan_dataset.replace({'Loan_Status':{'N':0,'Y':1}},inplace=True)
```

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

Out[11]:

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

In [12]:
```
# Dependents column value
loan_dataset['Dependents'].value_counts()
```

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

In [13]:
```
# Replacing the value of 3+ to 4
loan_dataset = loan_dataset.replace(to_replace='3+',value=4)
```

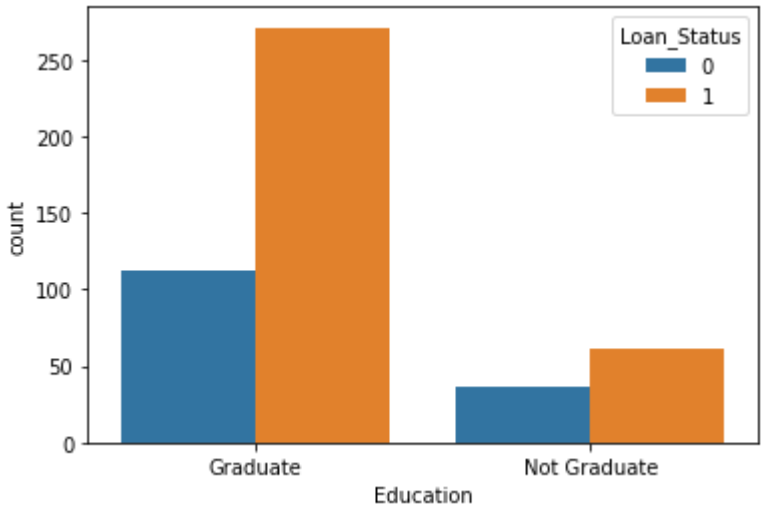
In [14]:
```
loan_dataset['Dependents'].value_counts()
```

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

Data Visualization

In [16]:
```
# Education and loan_status
sns.countplot(x='Education',hue='Loan_Status',data=loan_dataset)
```

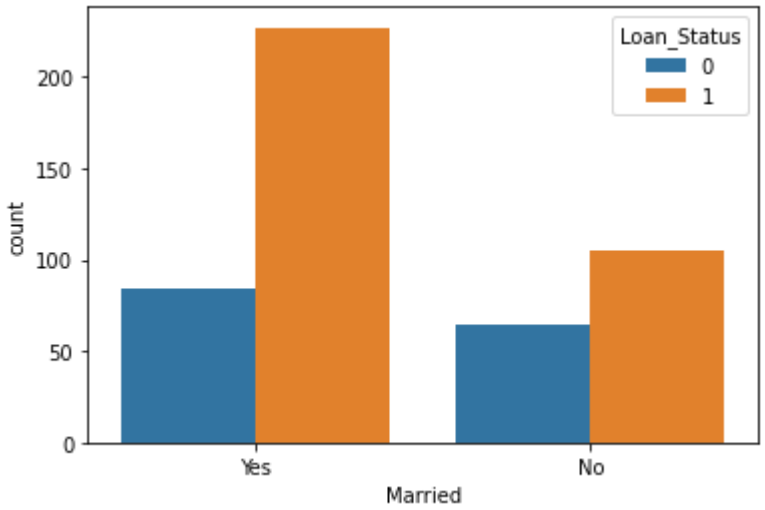
Out[16]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x274abf3a490>
```




```
Loan is approved if the person is Graduated in most cases.The no.of loan approved for the Graduated
people is more compare to non-graduated.
```

In [17]:
```
# Marital status and loan_status
sns.countplot(x='Married',hue='Loan_Status',data=loan_dataset)
```

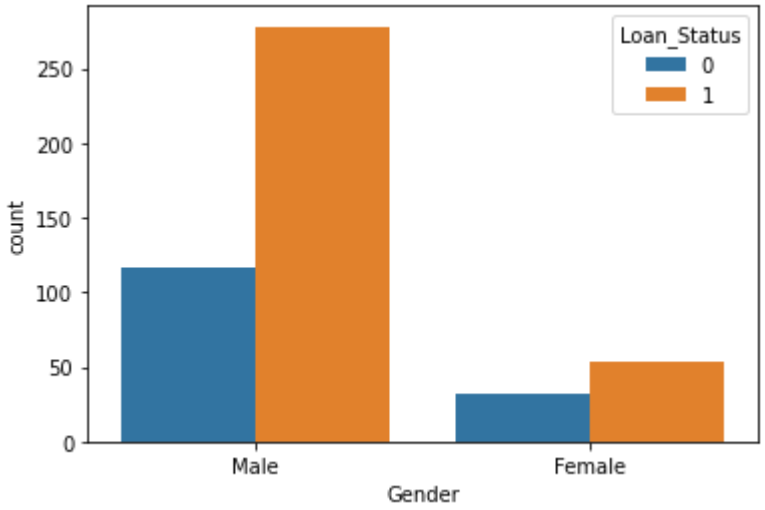
Out[17]: <matplotlib.axes._subplots.AxesSubplot at 0x274abb50520>



If a person is married there is a high chance their loan will be approved. If a person is not married,the no.of loan approval is less. The reason is that if a person is married then both husband and wife can contribute to setteling the loan.

In [19]:
```
1 sns.countplot(x='Gender',hue='Loan_Status',data=loan_dataset)
```

Out[19]: <matplotlib.axes._subplots.AxesSubplot at 0x274ad5b8790>



In [20]:
```
1 # convert categorical columns to numerical values
2 loan_dataset.replace({'Married':{'No':0,'Yes':1},'Gender':{'Male':1,'Female':0},'Self_Employed':{'No':0,'Ye
3                       'Property_Area':{'Rural':0,'Semiurban':1,'Urban':2},'Education':{'Graduate':1,'Not Gr
```

In [21]:
```
1 loan_dataset.head()
```

Out[21]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_
1	LP001003	1	1	1	1	0	4583	1508.0	128.0	
2	LP001005	1	1	0	1	1	3000	0.0	66.0	
3	LP001006	1	1	0	0	0	2583	2358.0	120.0	
4	LP001008	1	0	0	1	0	6000	0.0	141.0	
5	LP001011	1	1	2	1	1	5417	4196.0	267.0	

In [22]:
```
1 # separate the data and label
2 X = loan_dataset.drop(columns=['Loan_ID','Loan_Status'],axis=1)
3 Y = loan_dataset['Loan_Status']
```

In [25]:
```
1 X.head()
```

Out[25]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Cred
1	1	1	1	1	0	4583	1508.0	128.0	360.0	
2	1	1	0	1	1	3000	0.0	66.0	360.0	
3	1	1	0	0	0	2583	2358.0	120.0	360.0	
4	1	0	0	1	0	6000	0.0	141.0	360.0	
5	1	1	2	1	1	5417	4196.0	267.0	360.0	

In [26]:
```
1 Y.head()
```

Out[26]:
```
1    0
2    1
3    1
4    1
5    1
Name: Loan_Status, dtype: int64
```

Train - Test Split

In [35]:
```
1 X_train,X_test,y_train,y_test = train_test_split(X,Y,test_size=0.2,stratify=Y,random_state=1)
```

In [29]:
```
1 X.shape, X_train.shape, X_test.shape
```

Out[29]:
```
((480, 11), (384, 11), (96, 11))
```

Training the model: Support Vector Machine Model

In [36]:
```
1 classifier = svm.SVC(kernel='linear')
```

In [37]:
```
1 # Training the support vector Machine Model
2 classifier.fit(X_train,y_train)
```

Out[37]:
```
SVC(kernel='linear')
```

Model Evaluation

In [38]:
```
1 # Accuracy score on training data
2 X_train_pred = classifier.predict(X_train)
3 training_data_accuracy = accuracy_score(X_train_pred,y_train)
```

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

```
Accuracy on training data:  0.7994791666666666
```

In [40]:
```
# Accuracy score on test data
X_test_pred = classifier.predict(X_test)
test_data_accuracy = accuracy_score(X_test_pred,y_test)
```

In [41]:
```
print('Accuracy on test data: ',test_data_accuracy)
```

```
Accuracy on test data:  0.8229166666666666
```