Double-click (or enter) to edit

First we have to read .CSV file. We will do this with the help of pandas library.

```
import pandas as pd
data1=pd.read_csv('dataset.csv')
data1
```

	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
...	...	...	...	...	...	...	...	...	...	...	...	...	...
609	LP002978	Female	No	0	Graduate	No	2900	0.0	71.0	360.0	1.0	Rural	Y
610	LP002979	Male	Yes	3+	Graduate	No	4106	0.0	40.0	180.0	1.0	Rural	Y
611	LP002983	Male	Yes	1	Graduate	No	8072	240.0	253.0	360.0	1.0	Urban	Y
612	LP002984	Male	Yes	2	Graduate	No	7583	0.0	187.0	360.0	1.0	Urban	Y
613	LP002990	Female	No	0	Graduate	Yes	4583	0.0	133.0	360.0	0.0	Semiurban	N

614 rows × 13 columns

Lets see the first five rows of our data.

```
data1.head()
```

	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

Lets see the last five rows of our data.

```
data1.tail()
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
609	LP002978	Female	No	0	Graduate	No	2900	0.0	71.0	360.0	1.0	Rural	Y
610	LP002979	Male	Yes	3+	Graduate	No	4106	0.0	40.0	180.0	1.0	Rural	Y
611	LP002983	Male	Yes	1	Graduate	No	8072	240.0	253.0	360.0	1.0	Urban	Y
612	LP002984	Male	Yes	2	Graduate	No	7583	0.0	187.0	360.0	1.0	Urban	Y
613	LP002990	Female	No	0	Graduate	Yes	4583	0.0	133.0	360.0	0.0	Semiurban	N

Let us see the all columns present in our dataset in one place.

```
data1.columns
```

```
Index(['Loan_ID', 'Gender', 'Married', 'Dependents', 'Education',
       'Self_Employed', 'ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
       'Loan_Amount_Term', 'Credit_History', 'Property_Area', 'Loan_Status'],
      dtype='object')
```

Similarly we can know the shape of our dataset.

```
data1.shape
```

```
(614, 13)
```

If we want to view row from 2 to 6 ,then it can be done as:

```
data1[5:9]
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
5	LP001011	Male	Yes	2	Graduate	Yes	5417	4196.0	267.0	360.0	1.0	Urban	Y
6	LP001013	Male	Yes	0	Not Graduate	No	2333	1516.0	95.0	360.0	1.0	Urban	Y
7	LP001014	Male	Yes	3+	Graduate	No	3036	2504.0	158.0	360.0	0.0	Semiurban	N
8	LP001018	Male	Yes	2	Graduate	No	4006	1526.0	168.0	360.0	1.0	Urban	Y

To get the maximum and minimum applicant income we can follow the following steps:

```
data1['ApplicantIncome'].max()
```

```
81000
```

```
data1['ApplicantIncome'].min()
```

```
150
```

To know which LoanID is not graduate, we can do the following steps:

```
data1['Loan_ID'][data1['Education'] == 'Not Graduate']
```

```
3      LP001006
6      LP001013
16     LP001034
18     LP001038
20     LP001043
         ...
595    LP002940
596    LP002941
601    LP002950
605    LP002960
607    LP002964
Name: Loan_ID, Length: 134, dtype: object
```

```
data1.describe() #We can analysis our complete dataset from this process
```

	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

```
data1.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
```

If we want to see where is nmissing value in our data we can see that by following steps

```
data1.isna() #We will see true where data is missing
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	False	False	False	False	False	False	False	False	True	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...
609	False	False	False	False	False	False	False	False	False	False	False	False	False
610	False	False	False	False	False	False	False	False	False	False	False	False	False
611	False	False	False	False	False	False	False	False	False	False	False	False	False
612	False	False	False	False	False	False	False	False	False	False	False	False	False
613	False	False	False	False	False	False	False	False	False	False	False	False	False

614 rows × 13 columns

Now to get the more detail of missing value.

```
data1.isna().sum()
```

```
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
```

Now we first drop the missing value then we will check again if there is any missing value in our dataset

```
data1=data1.dropna()
data1.isna().sum()
```

```
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
```

Since our loan status is in yes and no,we will change it into 0 and 1 for our analysis

```
data1.replace({'Loan_Status':{'N':0,'Y':1}},inplace=True)
```

```
data1.head()
```

	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

To know how many different values are in dependents columns

```
data1['Dependents'].value_counts()
```

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

Now we will change the value of 3+ to 4

```
data1=data1.replace(to_replace="3+",value=4)
```

```
data1['Dependents'].value_counts()
```

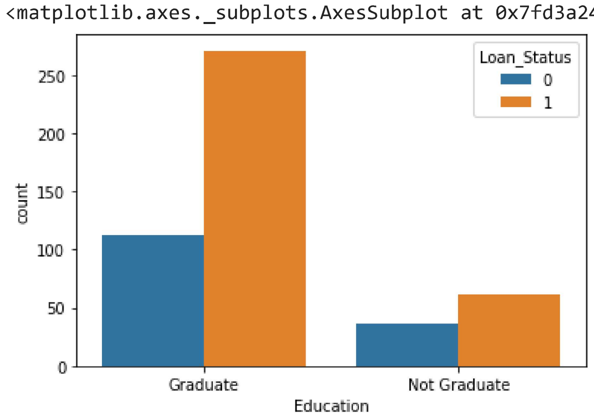
```
0    274
2     85
1     80
4     41
Name: Dependents, dtype: int64
```

Now we will visualise our data and analysis our data . In that ways we will able to analysis the factors in which loan status depends. we will also import some libraries for this

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

```
sns.countplot(x='Education', hue='Loan_Status',data=data1)
```

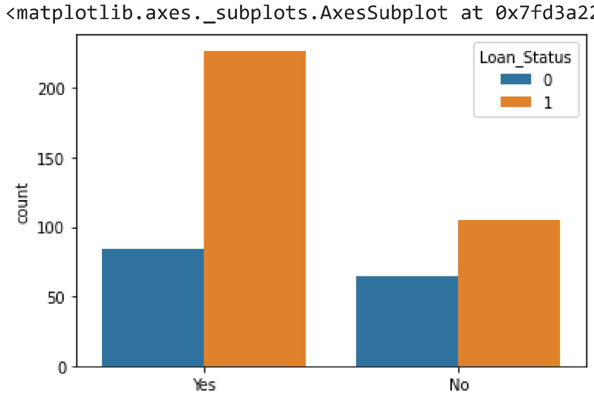
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd3a241aa10>
```



We can see here person with graduation have high chances of getting home loans. Now we can check the same for married status

```
sns.countplot(x='Married',hue='Loan_Status',data=data1)
```

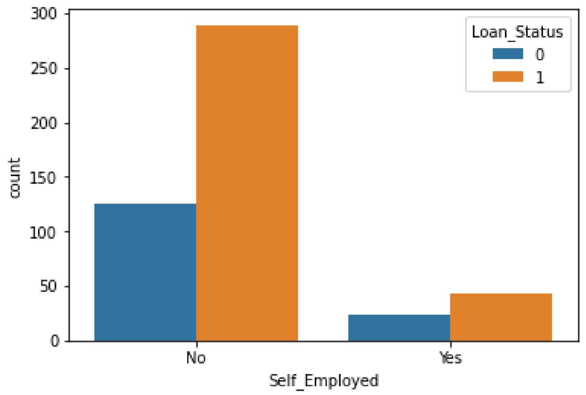
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd3a226fd50>
```



We can clearly see the the person with "married" status have got more home loan approvred . Now we will see it for Self Employment

```
sns.countplot(x='Self_Employed',hue='Loan_Status',data=data1)
```

<matplotlib.axes._subplots.AxesSubplot at 0x7fd3a2258bd0>



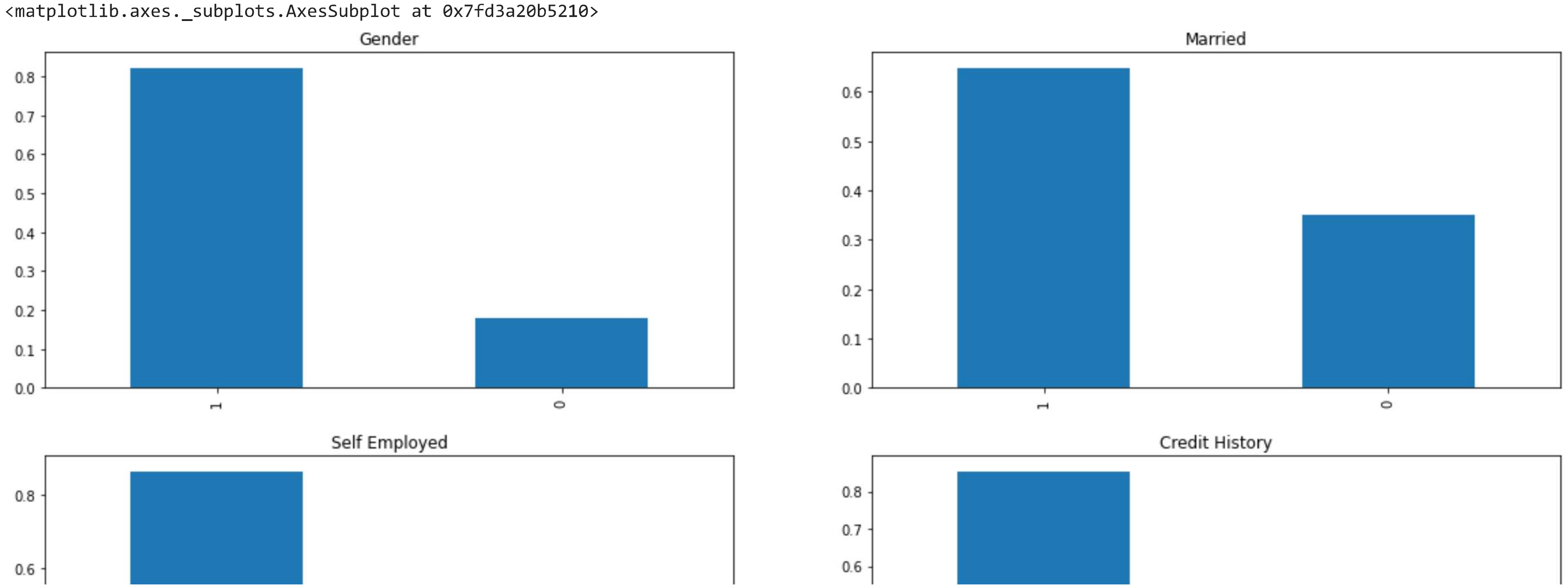

```
#Now we will convert all categorical columns to numerical values
data1.replace({'Married':{'No':0,'Yes':1},'Gender':{'Male':1,'Female':0},'Self_Employed':{'No':0,'Yes':1},'Property_Area':{'Rural':0,'Semiurban':1,'Urban':2},'Education':{'Graduate':1,'Not Graduate':0}},inplace=True)
```

```
import matplotlib.pyplot as plt
import matplotlib.axes
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")
```

```
#Study of categorical features like Gender, Married, Self_Employed and Credit_History
```

```
plt.figure(figsize=(20,10))
plt.subplot(2,2,1)
data1['Gender'].value_counts(normalize=True).plot.bar(title='Gender')
plt.subplot(2,2,2)
data1['Married'].value_counts(normalize=True).plot.bar(title='Married')
plt.subplot(2,2,3)
data1['Self_Employed'].value_counts(normalize=True).plot.bar(title='Self Employed')
plt.subplot(2,2,4)
data1['Credit_History'].value_counts(normalize=True).plot.bar(title='Credit History')
```

<matplotlib.axes._subplots.AxesSubplot at 0x7fd3a20b5210>



we can see that all value are in 0 and 1

```
data1.head()
```

	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

```
#seprating tha data and label
X=data1.drop(columns=['Loan_ID','Loan_Status'],axis=1)
Y=data1['Loan_Status']
```

```
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
```

Train Test Split

```
X_train, X_test, Y_train,Y_test = train_test_split(X,Y,test_size=0.1,stratify=Y,random_state=2)
```

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

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

Training the model:

Support Vector Machine Model:

```
classifier=svm.SVC(kernel='linear')
```

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

```
SVC(kernel='linear')
```

```
#accuracy score on training data
X_train_prediction=classifier.predict(X_train)
training_data_accuracy=accuracy_score(X_train_prediction,Y_train)
```

```
print('Accuracy on training data:', training_data_accuracy)
```

```
Accuracy on training data: 0.7986111111111112
```

```
#Accuracy score on testing data
X_test_prediction=classifier.predict(X_test)
test_data_accuracy=accuracy_score(X_test_prediction,Y_test)
```

```
print('Accuracy on test data:',test_data_accuracy)
```

```
Accuracy on test data: 0.8333333333333334
```

So in this way we can visualise and analysis our data, also there is only little difference between testing and training data. so this model works for us.