```
import  pandas as pd
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

loan_dataset =
pd.read_csv(r'F:/ML_Projects/Loan_Prediction/train_u6.csv')
loan_dataset.head()
```

```
    Loan_ID Gender Married Dependents     Education Self_Employed  \
0  LP001002   Male      No          0      Graduate            No
1  LP001003   Male     Yes          1      Graduate            No
2  LP001005   Male     Yes          0      Graduate           Yes
3  LP001006   Male     Yes          0  Not Graduate            No
4  LP001008   Male      No          0      Graduate            No

   ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
0             5849                0.0         NaN             360.0
1             4583             1508.0       128.0             360.0
2             3000                0.0        66.0             360.0
3             2583             2358.0       120.0             360.0
4             6000                0.0       141.0             360.0

   Credit_History Property_Area Loan_Status
0             1.0         Urban           Y
1             1.0         Rural           N
2             1.0         Urban           Y
3             1.0         Urban           Y
4             1.0         Urban           Y
```

```
# Printing the last 5rows of the DataFrame;
loan_dataset.tail()
```

```
      Loan_ID  Gender Married Dependents Education Self_Employed  \
609  LP002978  Female      No          0  Graduate            No
610  LP002979    Male     Yes         3+  Graduate            No
611  LP002983    Male     Yes          1  Graduate            No
612  LP002984    Male     Yes          2  Graduate            No
613  LP002990  Female      No          0  Graduate           Yes

     ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term
\
609             2900                0.0        71.0             360.0

610             4106                0.0        40.0             180.0

611             8072              240.0       253.0             360.0
```

```
612             7583                0.0       187.0             360.0

613             4583                0.0       133.0             360.0


     Credit_History Property_Area Loan_Status
609             1.0         Rural           Y
610             1.0         Rural           Y
611             1.0         Urban           Y
612             1.0         Urban           Y
613             0.0     Semiurban           N
```

```
# number of rows and columns
loan_dataset.shape
```

```
(614, 13)
```

```
# statistical measures
loan_dataset.describe()
```

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

       Credit_History
count      564.000000
mean         0.842199
std          0.364878
min          0.000000
25%          1.000000
50%          1.000000
75%          1.000000
max          1.000000
```

```
# number of missing values in each column
loan_dataset.isnull().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
```

```
# dropping the missing values
loan_dataset = loan_dataset.dropna()

# number of missing values in each column
loan_dataset.isnull().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
```

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

loan_dataset.head()
```

```
    Loan_ID Gender Married Dependents     Education Self_Employed  \
1  LP001003   Male     Yes          1      Graduate            No
2  LP001005   Male     Yes          0      Graduate           Yes
3  LP001006   Male     Yes          0  Not Graduate            No
4  LP001008   Male      No          0      Graduate            No
5  LP001011   Male     Yes          2      Graduate           Yes

   ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
1             4583             1508.0       128.0             360.0
2             3000                0.0        66.0             360.0
```

```
3              2583             2358.0       120.0             360.0
4              6000                0.0       141.0             360.0
5              5417             4196.0       267.0             360.0

   Credit_History Property_Area  Loan_Status
1             1.0         Rural            0
2             1.0         Urban            1
3             1.0         Urban            1
4             1.0         Urban            1
5             1.0         Urban            1
```

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

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

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

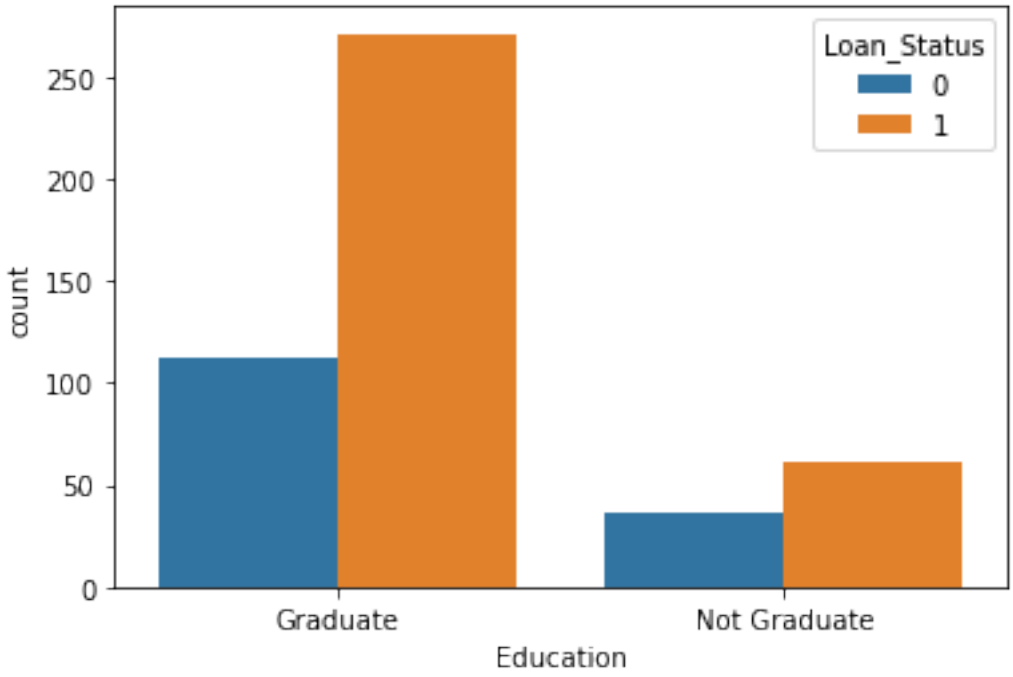
```
# Dependent Values
loan_dataset['Dependents'].value_counts()
```

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

Data Visualization

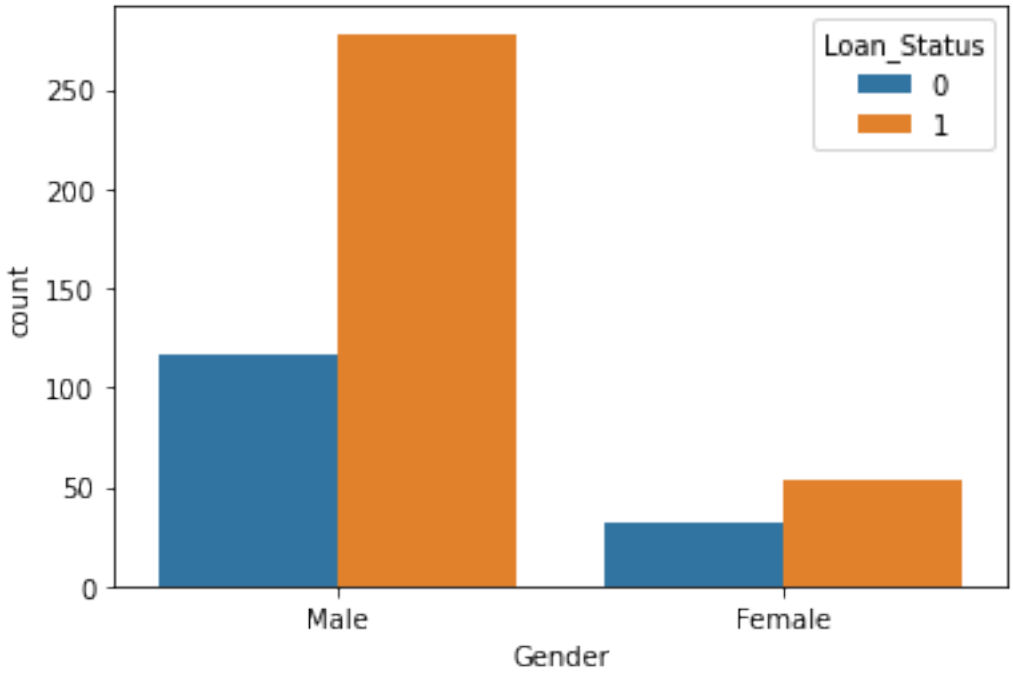
```
# Education & Loan Status:
sns.countplot(x = 'Education', hue = 'Loan_Status', data =
loan_dataset)
```

```
<AxesSubplot:xlabel='Education', ylabel='count'>
```



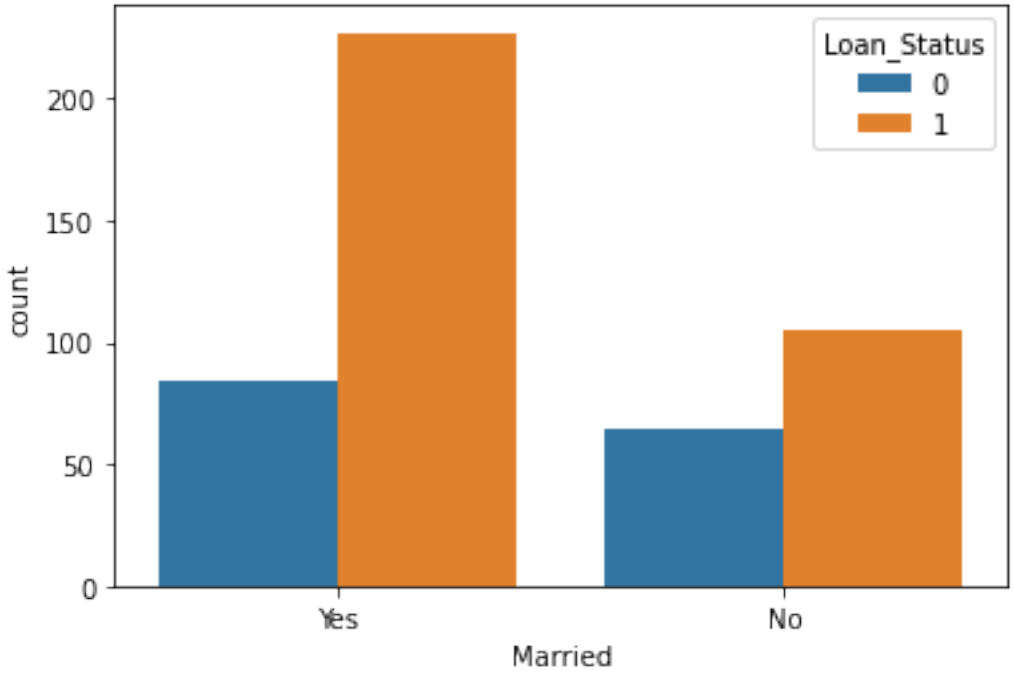

```
# Gender & Loan_ Status:
sns.countplot(x = 'Gender', hue = 'Loan_Status', data = loan_dataset)
```

```
<AxesSubplot:xlabel='Gender', ylabel='count'>
```



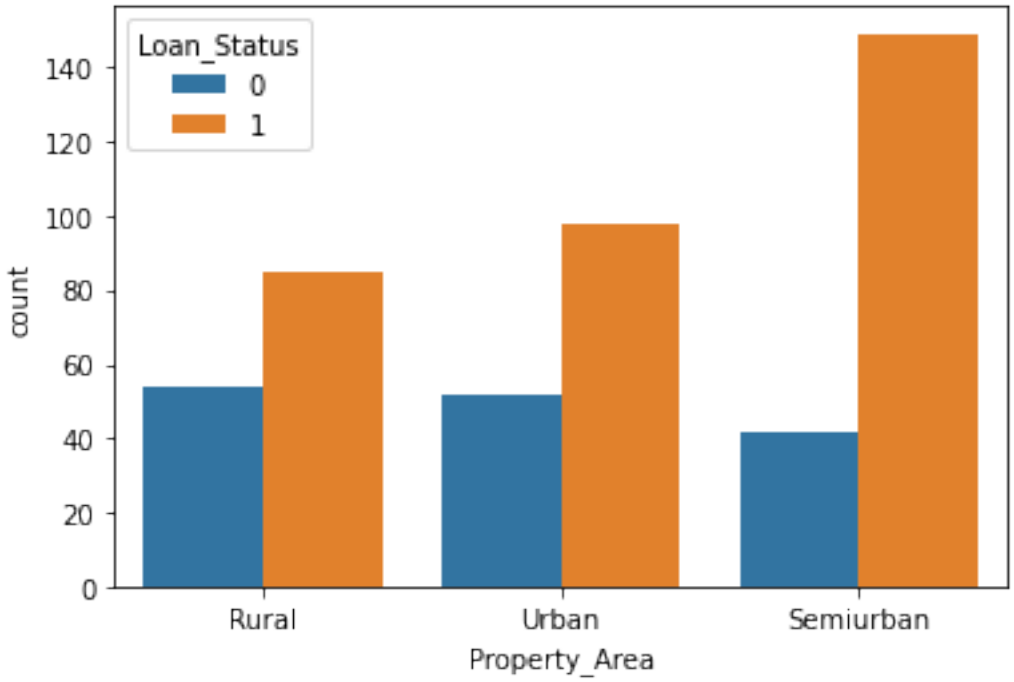

```
# Maritial Staus (column = Married) & Loan_ Status:
sns.countplot(x = 'Married', hue = 'Loan_Status', data = loan_dataset)
```

```
<AxesSubplot:xlabel='Married', ylabel='count'>
```




```
# Property Area & Loan_ Status:
sns.countplot(x = 'Property_Area', hue = 'Loan_Status', data =
loan_dataset)
```

```
<AxesSubplot:xlabel='Property_Area', ylabel='count'>
```



```
#  convert categorical columns to numerical values:
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)
```

```
loan_dataset.head()
```

```
    Loan_ID  Gender  Married Dependents  Education  Self_Employed  \
1  LP001003       1        1           1          1              0
2  LP001005       1        1           0          1              1
3  LP001006       1        1           0          0              0
4  LP001008       1        0           0          1              0
5  LP001011       1        1           2          1              1

   ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
1             4583             1508.0       128.0             360.0
2             3000                0.0        66.0             360.0
3             2583             2358.0       120.0             360.0
4             6000                0.0       141.0             360.0
5             5417             4196.0       267.0             360.0

   Credit_History  Property_Area  Loan_Status
1             1.0              0            0
2             1.0              2            1
3             1.0              2            1
4             1.0              2            1
5             1.0              2            1
```

```
# Separating the data & label:
X = loan_dataset.drop(['Loan_ID','Loan_Status'], axis = 1)
Y = loan_dataset['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
```

```
print(loan_dataset.std())
```

```
Gender                  0.383892
Married                 0.478118
Education               0.401973
Self_Employed           0.344734
ApplicantIncome      5668.251251
CoapplicantIncome    2617.692267
LoanAmount             80.508164
Loan_Amount_Term       65.212401
Credit_History          0.353307
Property_Area           0.776411
Loan_Status             0.462287
dtype: float64

C:\Users\DELL\AppData\Local\Temp/ipykernel_7412/829098530.py:1:
FutureWarning: Dropping of nuisance columns in DataFrame reductions
(with 'numeric_only=None') is deprecated; in a future version this
will raise TypeError.  Select only valid columns before calling the
reduction.
  print(loan_dataset.std())
```

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')

# Training the Support Vector Machine Model:
classifier.fit(X_train, Y_train)
```

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

Model Evaluation

```
# Accuracy Score On training Data:
X_train_prediction = classifier.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction, Y_train)

print('Accuracy Score on Training Data :',training_data_accuracy)
```

```
Accuracy Score on Training Data : 0.7986111111111112
```

```
# Accuracy Score On testing Data:
X_test_prediction = classifier.predict(X_test)
testing_data_accuracy = accuracy_score(X_test_prediction, Y_test)

print('Accuracy Score on Testing Data :',testing_data_accuracy)
```

```
Accuracy Score on Testing Data : 0.8333333333333334
```