```
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 google.colab import drive
drive.mount("/content/drive")
```

```
Mounted at /content/drive
```

Data collection and processing

```
#loading the datraset to pandas dataframe
loan_dataset = pd.read_csv("/content/train_u6lujuX_CVtuZ9i (1) (4).csv")
```

```
type(loan_dataset)
```

```
pandas.core.frame.DataFrame
def __init__(data=None, index: Axes | None=None, columns: Axes | None=None, dtype: Dtype |
None=None, copy: bool | None=None) -> None
```

```
/usr/local/lib/python3.12/dist-packages/pandas/core/frame.py
Two-dimensional, size-mutable, potentially heterogeneous tabular data.

Data structure also contains labeled axes (rows and columns).
Arithmetic operations align on both row and column labels. Can be
thought of as a dict-like container for Series objects. The primary
```

```
#print first five column of dataset
loan_dataset.head()
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Te
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360

Next steps: New interactive sheet

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

```
(614, 13)
```

```
loan_dataset.describe()
```

	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

```
# number of missing values in dataset
loan_dataset.isnull().sum()
```

	0
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()
```

```
loan_dataset.isnull().sum()
```

	0
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)
```

```
/tmp/ipython-input-3213138282.py:2: FutureWarning: Downcasting behavior in `replace` is deprecated and will be removed in a future
  loan_dataset.replace({'Loan_Status':{"N":0,'Y':1}},inplace=True)
```

```
# Dependant value counts
loan_dataset["Dependents"].value_counts()
```

Dependents	count
0	274
2	85
1	80
3+	41

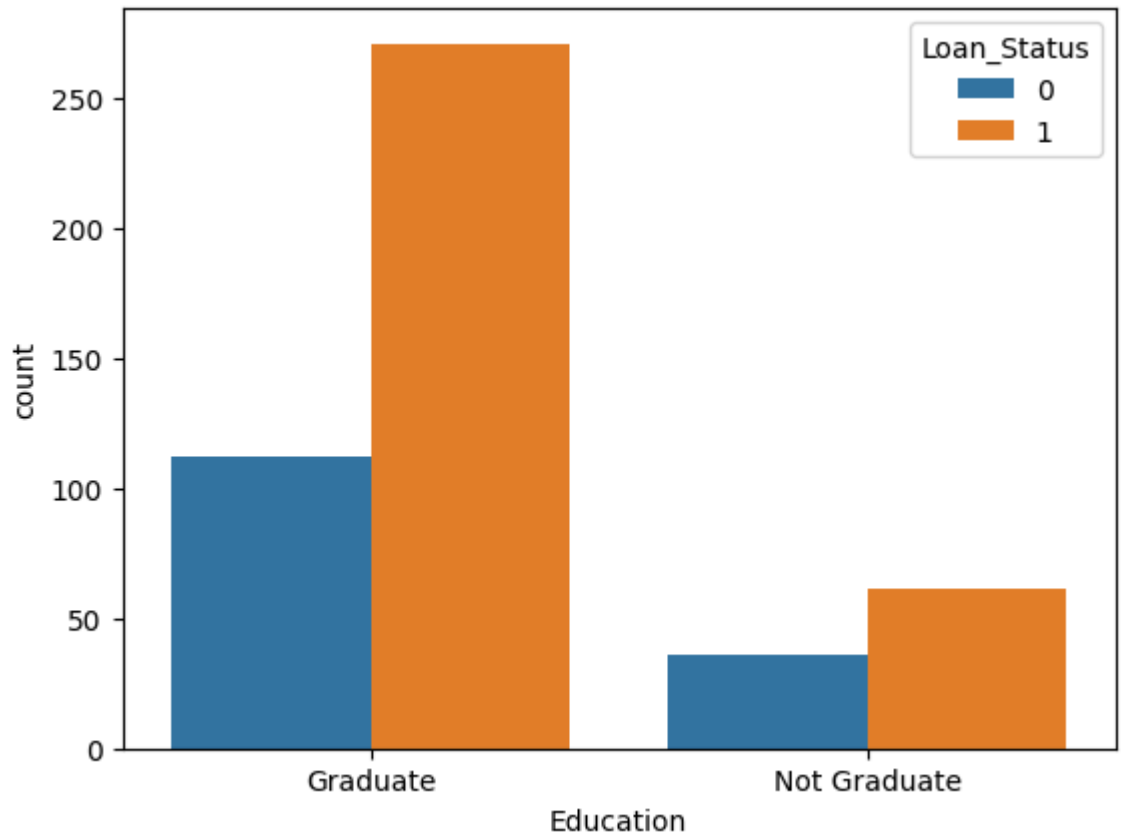
dtype: int64

```
# replacing 3+ to 4
loan_dataset.replace({"Dependents":{'3+' : '4'}},inplace=True)
```

Data Visualizations

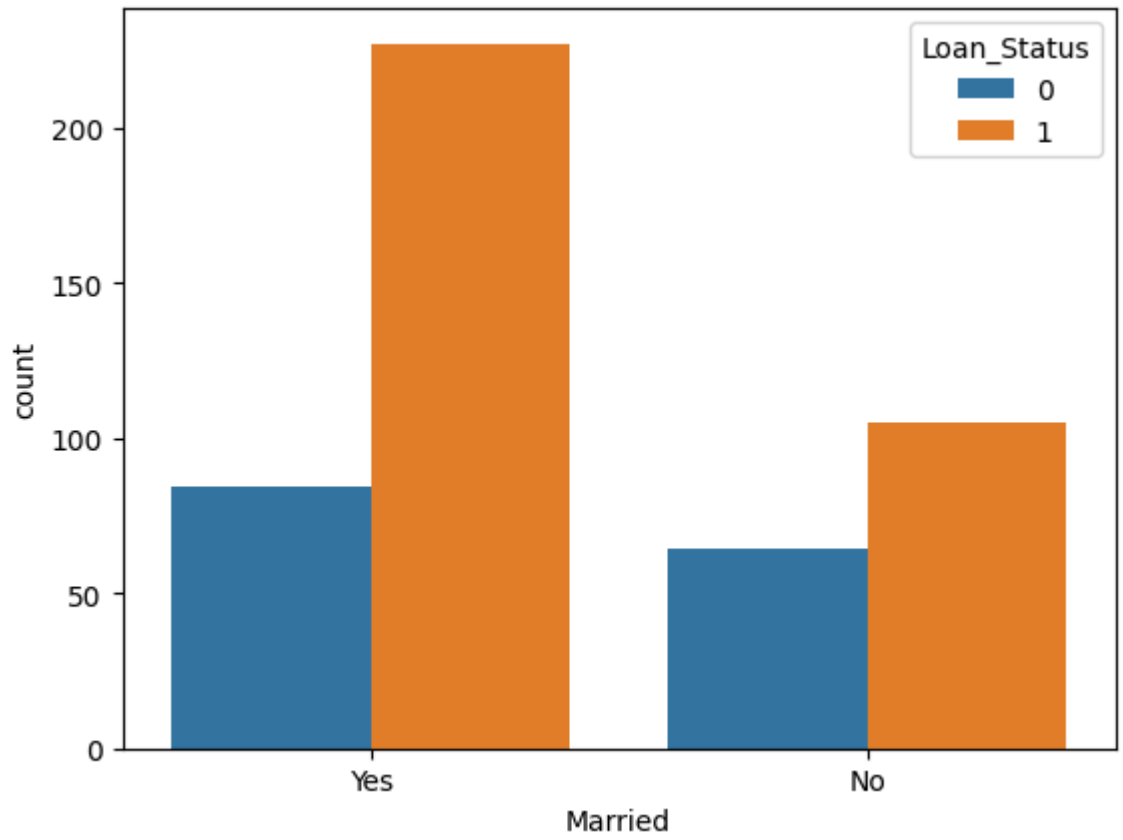
```
# education and loan status
sns.countplot(x='Education',hue='Loan_Status',data=loan_dataset)
```

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




```
# marital status and loan status
sns.countplot(x='Married',hue='Loan_Status',data=loan_dataset)
```

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




```
loan_dataset.head()
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Te
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360
5	LP001011	Male	Yes	2	Graduate	Yes	5417	4196.0	267.0	360

Next steps: New interactive sheet

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

```
/tmp/ipython-input-420738567.py:2: FutureWarning: Downcasting behavior in `replace` is deprecated and will be removed in a future
  loan_dataset.replace({'Married':{'Yes':1,'No':0},'Gender':{'Male':1,'Female':0},'Self_Employed':{'No':0,'Yes':1},'Property_Area
```

```
loan_dataset.head()
```

Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Ter
0.0	5849	0.0	NaN	360
0.0	4583	1508.0	128.0	360
1.0	3000	0.0	66.0	360
0.0	2583	2358.0	120.0	360
0.0	6000	0.0	141.0	360

Next steps: New interactive sheet

```
# separating the data and label
x=loan_dataset.drop(columns=['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
```

Train Test Split

```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.1,stratify=y,random_state=5121)
```

```
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