Lovely Professional University

Phagwara, Panjab

School of Computer Applications

Project Topic

Prediction of Loan Approval by using Machine Learning with Detailed Analysis, Graphs/Figures, and Reports

Submitted To:- Sumit Jaswal (30272)

Anurag Mittal(12209102)
Sukhdeep Kaur(12209123)
Ekhlakh Ahmad(12209166)

```
import pandas as pd
df2 = pd.read_csv("ProjectPy2.csv")
df2.head(50)
```

```
#no of missing values
df2.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
```

```
#type of data
type(df2)
```

```
pandas.core.frame.DataFrame
```

```
# no of rwos and coloums
df2.shape
```

```
(614, 13)
```

```
# statistical measurment
df2.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

```
# drop a missing values
df2 = df2.dropna()
```

```
df2.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
```

In [9]:
```
df2['Dependents'].value_counts()
```

Out[9]:
```
0     274
2      85
1      80
3+     41
Name: Dependents, dtype: int64
```

In [10]:
```
df2 = df2.replace(to_replace='3+' , value = 4)
```

In [11]:
```
df2['Dependents'].value_counts()
```

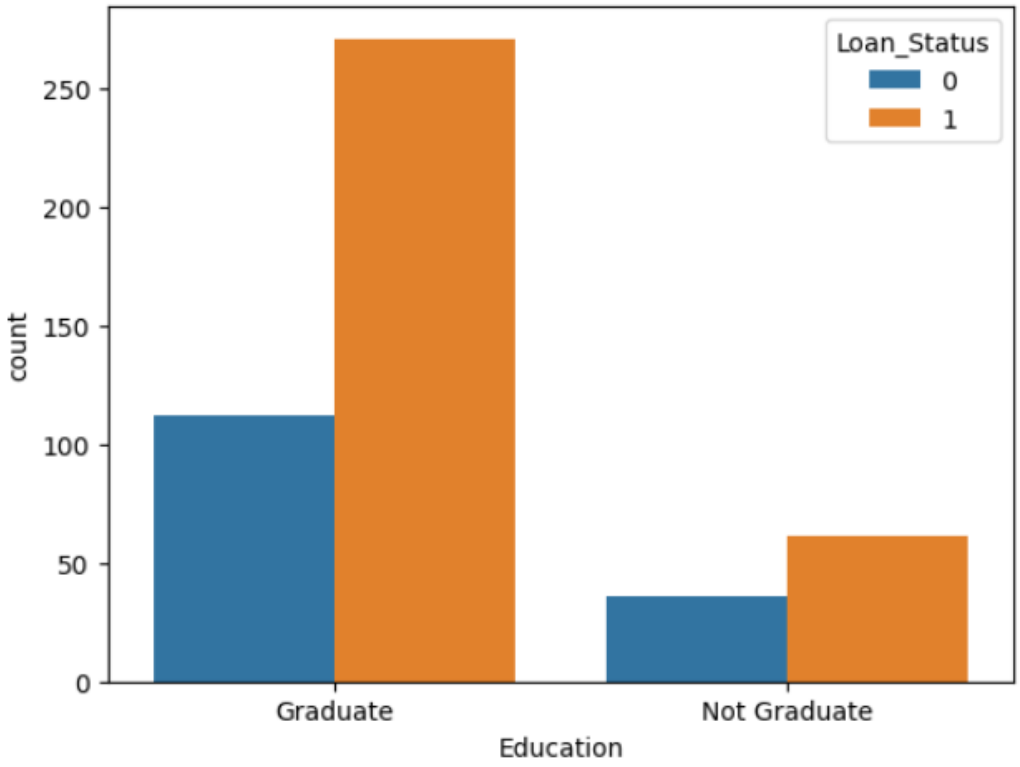
Out[11]:
```
0    274
2     85
1     80
4     41
Name: Dependents, dtype: int64
```

In [12]:
```
df2.replace({"Loan_Status":{'N':0 ,'Y':1}},inplace=True)
```

In [14]:
```
import seaborn as sns
sns.countplot(x='Education',hue='Loan_Status',data=df2)
```

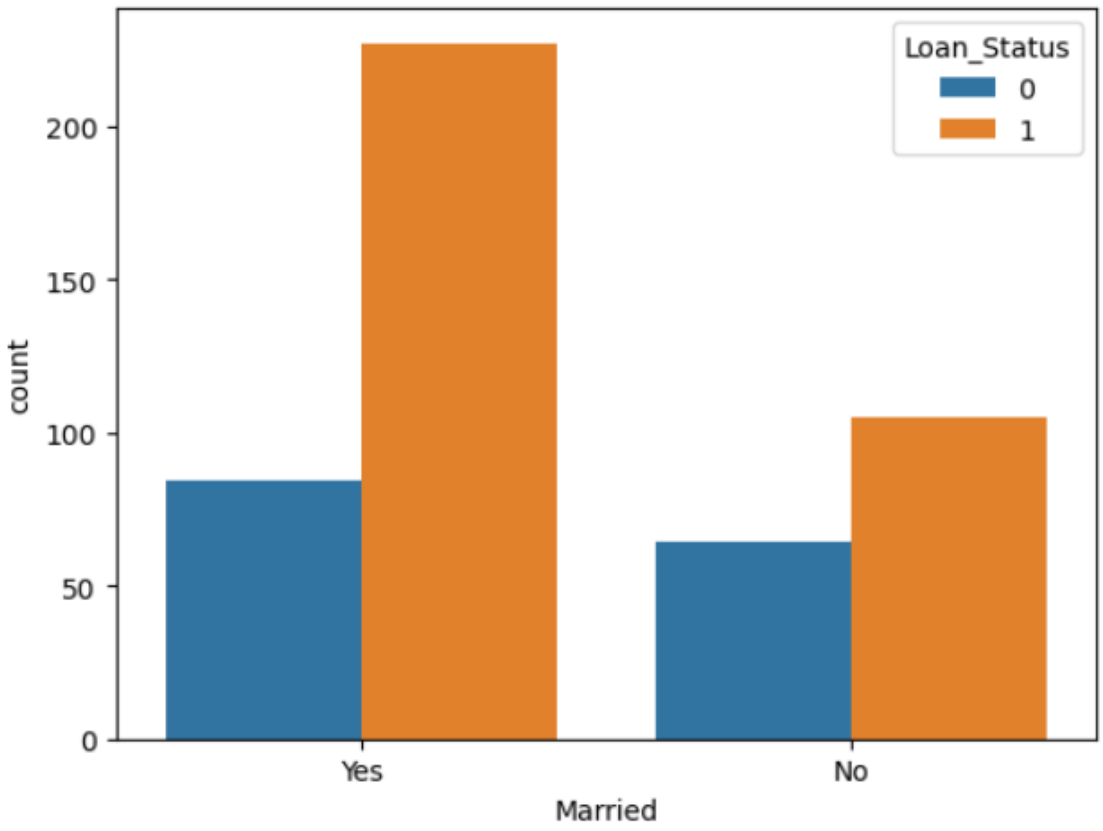
Out[14]:
```
<Axes: xlabel='Education', ylabel='count'>
```

Out[14]: <Axes: xlabel='Education', ylabel='count'>



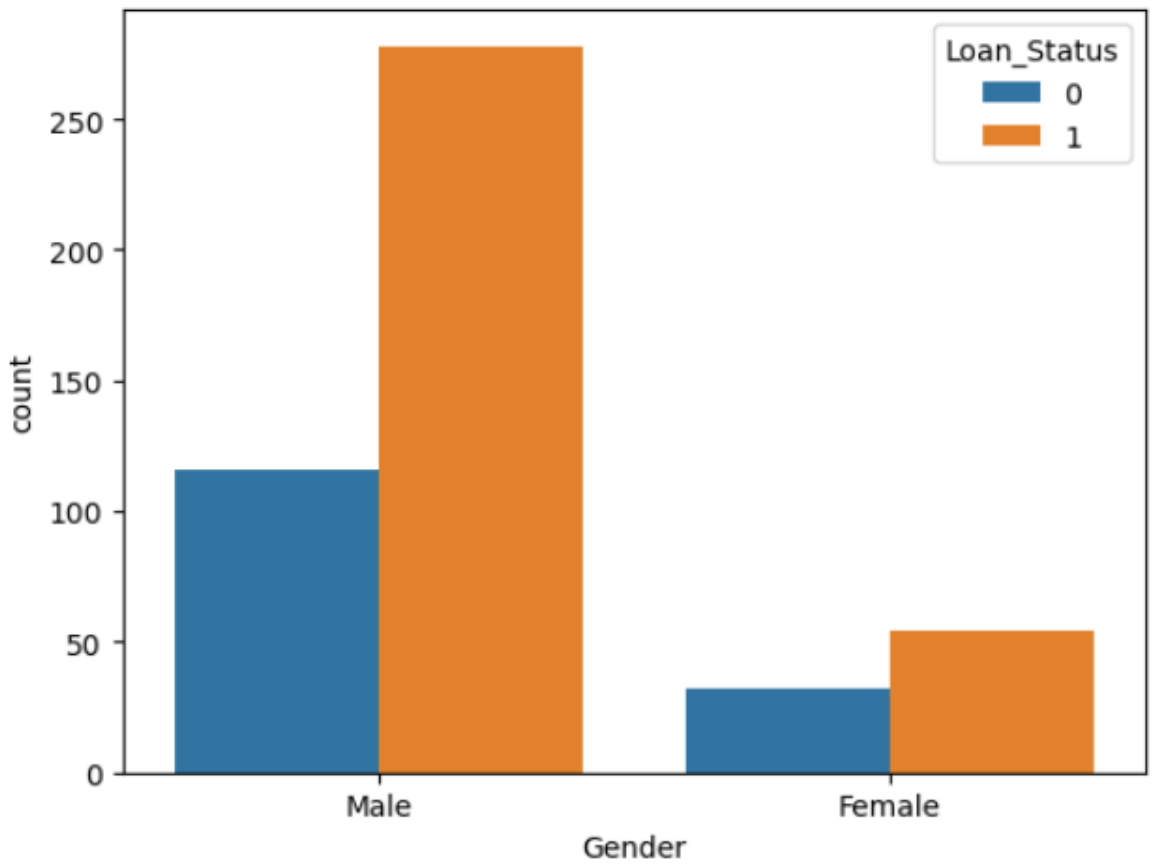
In [15]:
```
sns.countplot(x="Married",hue="Loan_Status",data=df2)
```

Out[15]: <Axes: xlabel='Married', ylabel='count'>



[16]:
```
sns.countplot(x="Gender",hue="Loan_Status",data=df2)
```

[16]:
```
<Axes: xlabel='Gender', ylabel='count'>
```



In [18]:
```
# convert categrical colums to numerical values
df2.replace({"Married":{'Yes':1,'No':0},"Gender":{'Male':1,'Female':0}},inplace=True)
```

In [19]:
```
df2.head(10)
```

Out[19]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_
1	LP001003	1	1	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	
2	LP001005	1	1	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	
3	LP001006	1	1	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	
4	LP001008	1	0	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	

```
In [20]: df2.replace({"Self_Employed":{'No':0,'Yes':1}},inplace=True)
```

```
In [21]: df2.head(20)
```

Out[21]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_
1	LP001003	1	1	1	Graduate	0	4583	1508.0	128.0	360.0	1.0	Rural	
2	LP001005	1	1	0	Graduate	1	3000	0.0	66.0	360.0	1.0	Urban	
3	LP001006	1	1	0	Not Graduate	0	2583	2358.0	120.0	360.0	1.0	Urban	
4	LP001008	1	0	0	Graduate	0	6000	0.0	141.0	360.0	1.0	Urban	
5	LP001011	1	1	2	Graduate	1	5417	4196.0	267.0	360.0	1.0	Urban	
6	LP001013	1	1	0	Not Graduate	0	2333	1516.0	95.0	360.0	1.0	Urban	

```
In [22]: df2.replace({"Property_Area":{'Rural':0,'Urban':2,'Semiurban':1}},inplace=True)
```

```
In [23]: df2.replace({"Education":{'Graduate':1,'Not Graduate':0}},inplace=True)
```

```
In [24]: df2.head(10)
```

Out[24]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_
1	LP001003	1	1	1	1	0	4583	1508.0	128.0	360.0	1.0	0	
2	LP001005	1	1	0	1	1	3000	0.0	66.0	360.0	1.0	2	
3	LP001006	1	1	0	0	0	2583	2358.0	120.0	360.0	1.0	2	
4	LP001008	1	0	0	1	0	6000	0.0	141.0	360.0	1.0	2	
5	LP001011	1	1	2	1	1	5417	4196.0	267.0	360.0	1.0	2	

```
In [25]: # Separating the data and lable
X = df2.drop(columns=['Loan_ID','Loan_Status'],axis=1)
Y = df2['Loan_Status']
```

```
In [26]: 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
from sklearn.model_selection import 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)
```

```
#Supprot Vector Machine
from sklearn import svm
from sklearn.metrics import accuracy_score
classifier = svm.SVC(kernel = 'linear')
```

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

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

```
X_train_prediction = classifier.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction, Y_train)
```

```
print("Accuracy on data",training_data_accuracy)
```

```
Accuracy on data 0.7986111111111112
```

```
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
```