Loan Status Prediction

April 24, 2025

1 Loan Approval Status Prediction

[1]:
```
# Importing the dependencies
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import svm
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import warnings
warnings.filterwarnings('ignore')
```

Data Collection and processing

[2]:
```
# Loading the dataset into a pandas dataframe
loan_dataset = pd.read_csv('dataset/loan_dataset.csv')
```

[3]:
```
# Printing the first 5 rows
loan_dataset.head()
```

[3]:

	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			
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0			
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0			

```
2             1.0         Urban           Y
3             1.0         Urban           Y
4             1.0         Urban           Y
```

[4]:
```
# Checking the shape of the dataset
loan_dataset.shape
```

[4]:
```
(614, 13)
```

[5]:
```
# Total value counts of the labels
loan_dataset['Loan_Status'].value_counts()
```

[5]:
```
Loan_Status
Y    422
N    192
Name: count, dtype: int64
```

Stastical Measures

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

[6]:
```
       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
```

Dealing with the Missing Values

[7]:
```
# Checking for null values
loan_dataset.isnull().sum()
```

[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
```

[8]:
```
# Dropping the missing values
loan_dataset = loan_dataset.dropna()
```

[9]:
```
# Checking for null values
loan_dataset.isnull().sum()
```

[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
```

[10]:
```
# Label Encoding the dataset
loan_dataset.replace({"Loan_Status":{'N':0,'Y':1}},inplace=True)
```

[11]:
```
# Checking the replaced values
loan_dataset['Loan_Status'].value_counts()
```

[11]:
```
Loan_Status
1    332
0    148
Name: count, dtype: int64
```

Now we can see that all the yes is replaced by 1 and all the no is replaced by 0

2 Dependent Column Values

loan_dataset['Dependents'].value_counts()

[12]:
```
# Replacing the Value og 3+ to 4
loan_dataset = loan_dataset.replace(to_replace='3+',value=4)
```

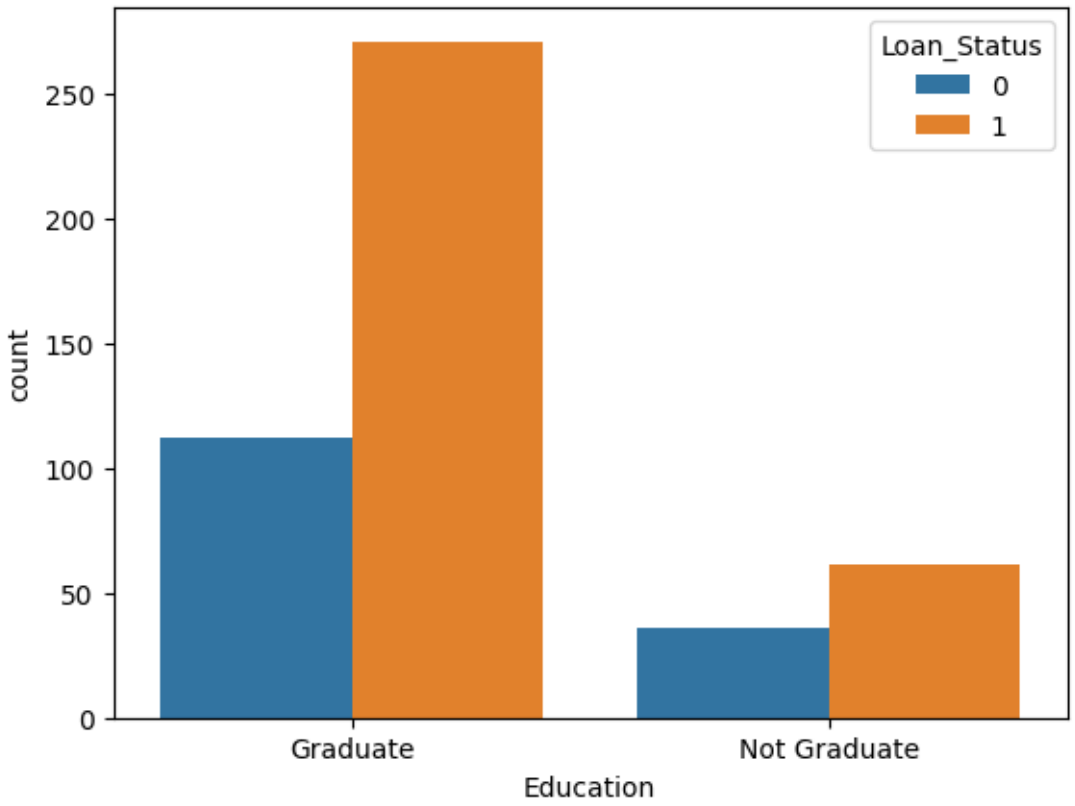
[13]:
```
# Dependent Column Values
loan_dataset['Dependents'].value_counts()
```

[13]:
```
Dependents
0    274
2     85
1     80
4     41
Name: count, dtype: int64
```

Data Visualization

[14]:
```
# Education and Loan Status
sns.countplot(x='Education',hue='Loan_Status',data=loan_dataset)
```

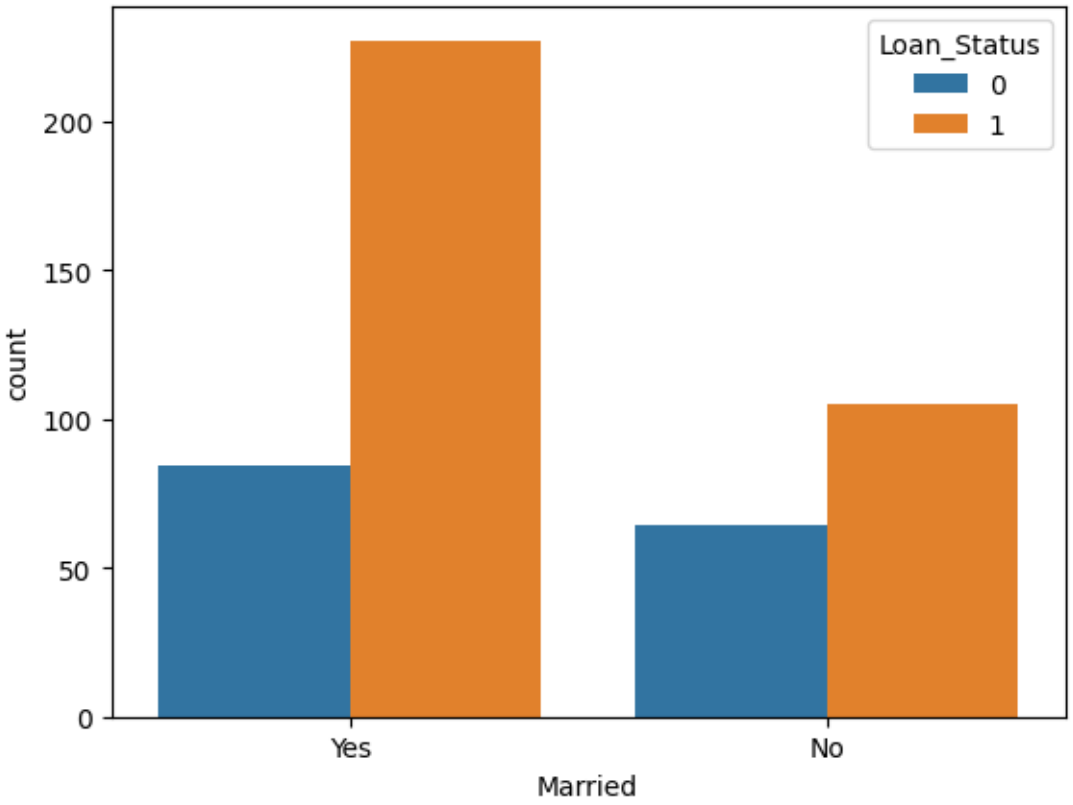
[14]:
```
<Axes: xlabel='Education', ylabel='count'>
```



This means loan is approved if the person is graduated

[15]:
```
# Maritial Status and Loan Status
sns.countplot(x='Married',hue='Loan_Status',data=loan_dataset)
```

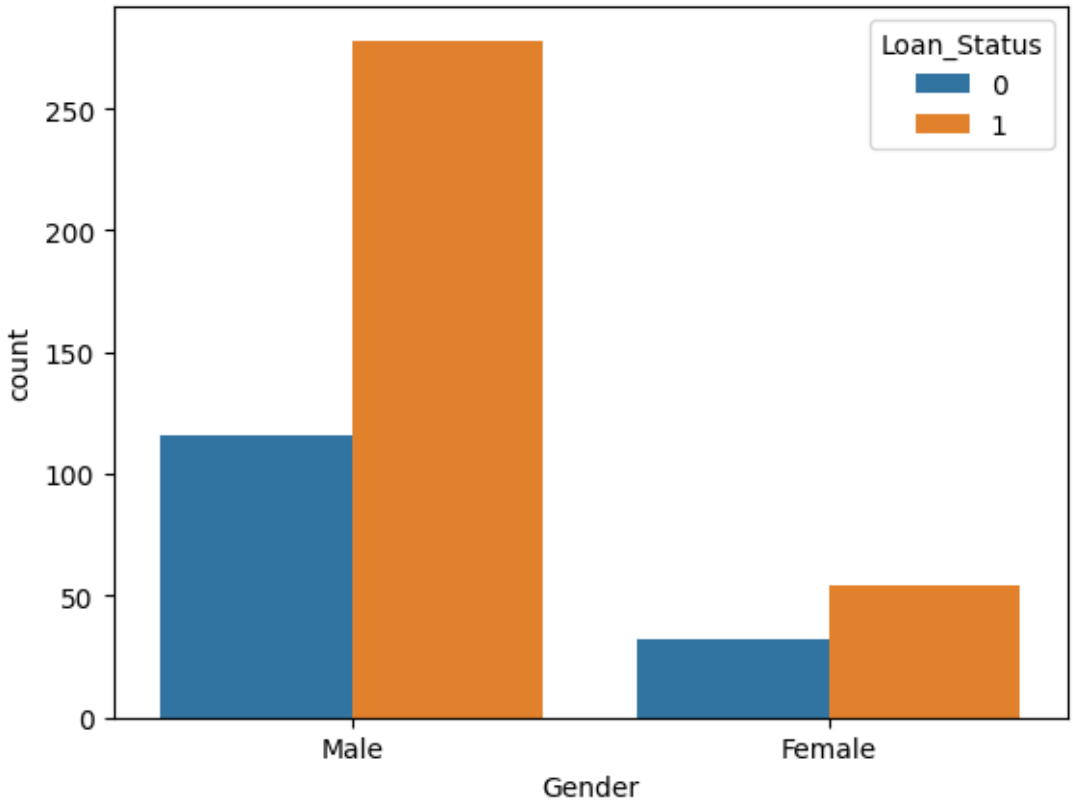
[15]:
```
<Axes: xlabel='Married', ylabel='count'>
```



From this we can say that a married person has a higher chance for loan approval

[16]:
```
# Checking the same for gender
sns.countplot(x='Gender',hue='Loan_Status',data=loan_dataset)
```

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



From this we can say that the male have higher chances of getting a loan

Converting all the text categorical cols to numerical values

[17]:
```
loan_dataset = 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}})
```

[18]:
```
loan_dataset.head()
```

[18]:

	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
```

Seprating the data and label

[19]:
```
X = loan_dataset.drop(columns=['Loan_ID','Loan_Status'],axis = 1)
y = loan_dataset['Loan_Status']
```

[20]:
```
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
```

Splitting the data into train and test

[21]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.
↪1,stratify=y,random_state=2)
```

[22]:
```
# Checking the splitted data shape
print(X.shape,X_train.shape,X_test.shape)
```

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

Training the Model

[23]:
```
classifier = svm.SVC(kernel='linear')
```

[24]:
```
classifier.fit(X_train,y_train)
```

[24]:
```
SVC(kernel='linear')
```

Model Evaluation

[25]:
```
training_data_predicted = classifier.predict(X_train)
training_data_accuracy = accuracy_score(training_data_predicted,y_train)

print("The Accuracy for Training data is : ",training_data_accuracy)
```

```
The Accuracy for Training data is :  0.7986111111111112
```

[26]:
```
test_data_predicted = classifier.predict(X_test)
test_data_accuracy = accuracy_score(test_data_predicted,y_test)

print("The Accuracy for Training data is : ",test_data_accuracy)
```

```
The Accuracy for Training data is :  0.8333333333333334
```

Making a Predictive System

[27]:
```
# Input data (11 features)
input_data = (1, 0, 0, 1, 0, 5849, 0, 0, 360, 1, 2)

# Convert to NumPy array
input_data_numpy_array = np.asarray(input_data)

# Reshape to (1, 11) for 1 sample with 11 features
input_data_reshaped = input_data_numpy_array.reshape(1, -1)

# Make prediction
prediction = classifier.predict(input_data_reshaped)

# Output result
if prediction[0] == 1:  # Assuming prediction is numeric (0 or 1), not a string
    print("The Loan Can be Approved")
else:
    print("The Loan Can't be Approved")
```

```
The Loan Can be Approved
```