uh8xtvssa

February 19, 2025

[52]:
```
# Importing the Dependencies
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn import svm
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

[51]:
```
# importing data
from google.colab import files
uploaded = files.upload()
```

```
<IPython.core.display.HTML object>
```

```
Saving dataset.csv to dataset (1).csv
```

[ ]:
```

```

[3]:
```
# Loading the dataset into a dataframe
loan_dataset = pd.read_csv('dataset.csv')
```

[4]:
```
# Display the first five rows
loan_dataset.head()
```

[4]:

	Loan_ID	Gender	Married	Dependents	…	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	LP001002	Male	No	0	…	360.0	1.0	Urban	Y
1	LP001003	Male	Yes	1	…	360.0	1.0	Rural	N
2	LP001005	Male	Yes	0	…	360.0	1.0	Urban	Y
3	LP001006	Male	Yes	0	…	360.0	1.0	Urban	Y
4	LP001008	Male	No	0	…	360.0	1.0	Urban	Y

[5 rows x 13 columns]

[5]:
```
# Number of rows & columns
loan_dataset.shape
```

[5]:
```
(614, 13)
```

[6]:
```
# Statistical measures for our dataset
loan_dataset.describe()
```

[6]:
```
       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
```

[7]:
```
# Number of missing 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
```

[9]:
```
# Dropping the missing values
loan_dataset = loan_dataset.dropna()
```

[10]:
```
# Checking for missing values
loan_dataset.isnull().sum()
```

[10]:
```
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
```

[11]:
```
# Label encoding
loan_dataset.replace({"Loan_Status":{'N':0,'Y':1}},inplace=True)
```

```
<ipython-input-11-3cd225bb69a8>:2: FutureWarning: Downcasting behavior in
`replace` is deprecated and will be removed in a future version. To retain the
old behavior, explicitly call `result.infer_objects(copy=False)`. To opt-in to
the future behavior, set `pd.set_option('future.no_silent_downcasting', True)`
  loan_dataset.replace({"Loan_Status":{'N':0,'Y':1}},inplace=True)
```

[12]:
```
# print the first five rows of the dataframe
loan_dataset.head()
```

[12]:
```
    Loan_ID Gender Married Dependents  … Loan_Amount_Term Credit_History
Property_Area  Loan_Status
1  LP001003   Male     Yes          1  …            360.0            1.0
Rural            0
2  LP001005   Male     Yes          0  …            360.0            1.0
Urban            1
3  LP001006   Male     Yes          0  …            360.0            1.0
Urban            1
4  LP001008   Male      No          0  …            360.0            1.0
Urban            1
5  LP001011   Male     Yes          2  …            360.0            1.0
Urban            1

[5 rows x 13 columns]
```

[15]:
```
# Dependent column values
loan_dataset['Dependents'].value_counts()
```

[15]:
```
Dependents
0     274
2      85
1      80
3+     41
Name: count, dtype: int64
```

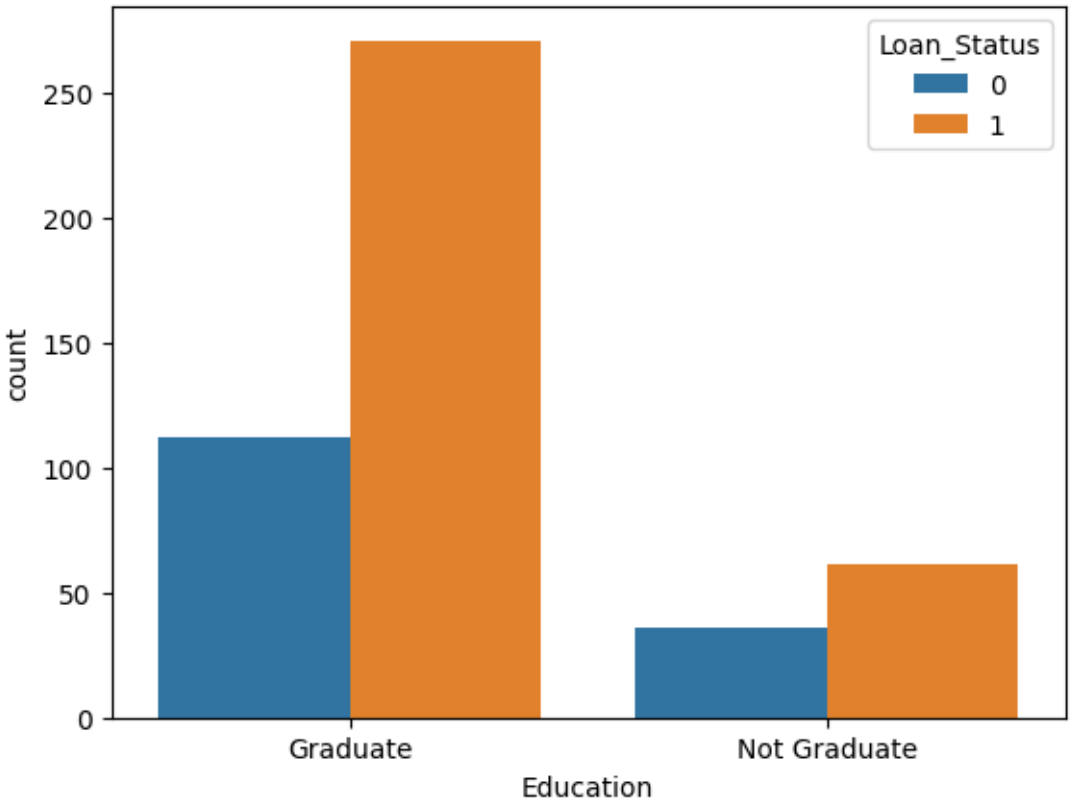
[16]:
```
# Replacing 3+ with a 4
loan_dataset = loan_dataset.replace(to_replace='3+', value=4)
```

[17]:
```
# Dependent values
loan_dataset['Dependents'].value_counts()
```

[17]:
```
Dependents
0    274
2     85
1     80
4     41
Name: count, dtype: int64
```

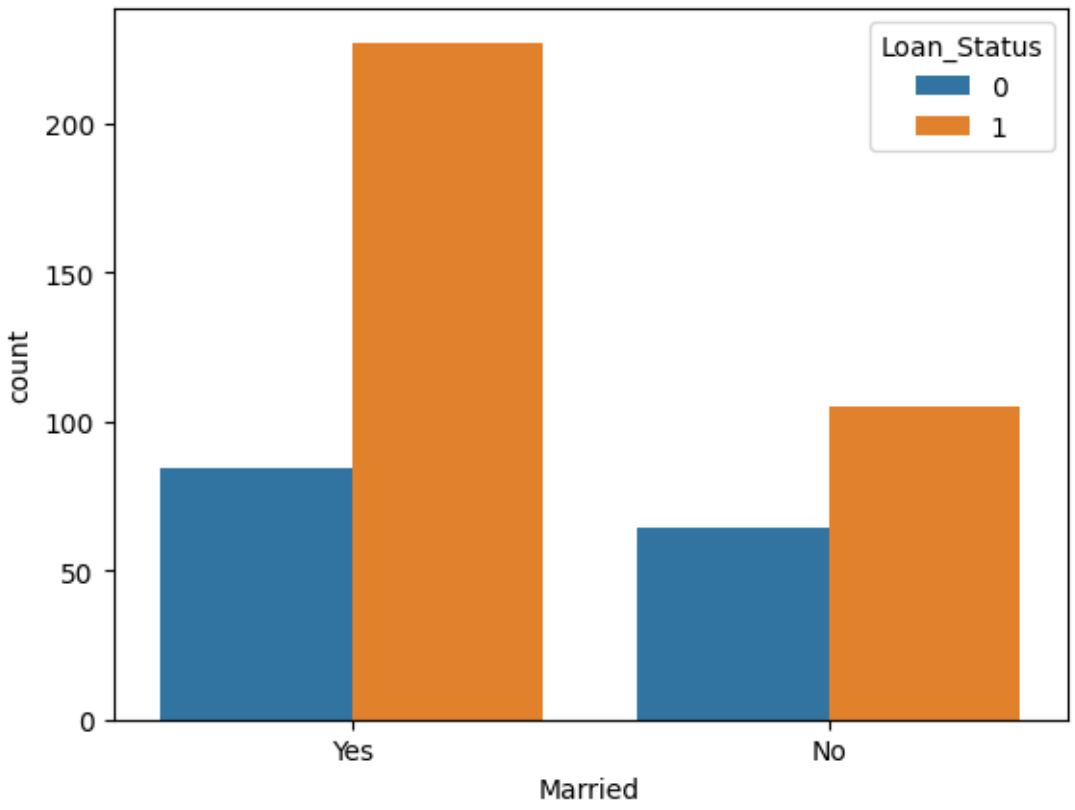
[18]:
```
# Visualize education & loan status data
sns.countplot(x='Education',hue='Loan_Status',data=loan_dataset)
```

[18]:
```
<Axes: xlabel='Education', ylabel='count'>
```



[20]:
```
# Visualize marital status & loan status data
sns.countplot(x='Married',hue='Loan_Status',data=loan_dataset)
```

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



[44]:
```
# Convert categorical columns to numerical values
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}})

# Ensure all columns are numeric
loan_dataset = loan_dataset.apply(pd.to_numeric, errors='coerce')
```

[45]:
```
# Separate the data & label
X = loan_dataset.drop(columns=['Loan_ID','Loan_Status'],axis=1)
Y = loan_dataset['Loan_Status']
```

[46]:
```
print(X)
print(Y)
```

```
     Gender  Married  Dependents  …  Loan_Amount_Term  Credit_History
Property_Area
```

```
1          1        1           1  …              360.0             1.0
0
2          1        1           0  …              360.0             1.0
2
3          1        1           0  …              360.0             1.0
2
4          1        0           0  …              360.0             1.0
2
5          1        1           2  …              360.0             1.0
2
..       …        …         …  …                …               …
…
609        0        0           0  …              360.0             1.0
0
610        1        1           4  …              180.0             1.0
0
611        1        1           1  …              360.0             1.0
2
612        1        1           2  …              360.0             1.0
2
613        0        0           0  …              360.0             0.0
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
```

[47]:
```
# Splitting the data into training & testing data
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.1,␣
 ↪stratify=Y, random_state=2)
```

[48]:
```
# Checking the shape
print(X.shape, X_train.shape, X_test.shape)
```

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

[49]:
```
# Trainig the model
classifier=svm.SVC(kernel='linear')
```

[50]:
```
# Training the Support Vector Machine model
classifier.fit(X_train,Y_train)
```

[50]:
```
SVC(kernel='linear')
```

[53]:
```
# Model evaluation
# acuuracy_score on training data
X_train_prediction = classifier.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction, Y_train)
```

[54]:
```
print('Accuracy on training data:', training_data_accuracy)
```

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

[55]:
```
# Accuracy data on training data
X_test_prediction = classifier.predict(X_test)
test_data_accuracy = accuracy_score(X_test_prediction, Y_test)
```

[56]:
```
print('Accuracy on test data:', test_data_accuracy)
```

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

[57]:
```
# Making a prediction system
input_data = (1,1,2,1,0,4000,0,1,0,0,1)
```