In [1]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```python
credit_df = pd.read_csv('CreditRisk.csv')
```

In [3]:
```python
credit_df.shape
```

Out[3]:
```
(614, 13)
```

In [4]:
```python
credit_df.head()
```

Out[4]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount
0	LP001002	Male	No	0	Graduate	No	5849	0.0	0
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141

In [5]:
```python
credit_df.tail()
```

Out[5]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount
609	LP002978	Female	No	0	Graduate	No	2900	0.0	71
610	LP002979	Male	Yes	3+	Graduate	No	4106	0.0	40
611	LP002983	Male	Yes	1	Graduate	No	8072	240.0	253
612	LP002984	Male	Yes	2	Graduate	No	7583	0.0	187
613	LP002990	Female	No	0	Graduate	Yes	4583	0.0	133

In [6]:
```python
credit_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Loan_ID            614 non-null    object 
 1   Gender             601 non-null    object 
 2   Married            611 non-null    object 
 3   Dependents         599 non-null    object 
 4   Education          614 non-null    object 
 5   Self_Employed      582 non-null    object 
 6   ApplicantIncome    614 non-null    int64  
 7   CoapplicantIncome  614 non-null    float64
 8   LoanAmount         614 non-null    int64  
 9   Loan_Amount_Term   600 non-null    float64
 10  Credit_History     564 non-null    float64
 11  Property_Area      614 non-null    object 
 12  Loan_Status        614 non-null    int64  
dtypes: float64(3), int64(3), object(7)
memory usage: 62.5+ KB
```

In [7]:
```python
credit_df.describe()
```

Out[7]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Loan_Status
count	614.000000	614.000000	614.000000	600.00000	564.000000	614.000000
mean	5403.459283	1621.245798	141.166124	342.00000	0.842199	0.687296
std	6109.041673	2926.248369	88.340630	65.12041	0.364878	0.463973
min	150.000000	0.000000	0.000000	12.00000	0.000000	0.000000
25%	2877.500000	0.000000	98.000000	360.00000	1.000000	0.000000
50%	3812.500000	1188.500000	125.000000	360.00000	1.000000	1.000000
75%	5795.000000	2297.250000	164.750000	360.00000	1.000000	1.000000
max	81000.000000	41667.000000	700.000000	480.00000	1.000000	1.000000

In [3]:
```python
credit_df.Loan_Status.value_counts()
```

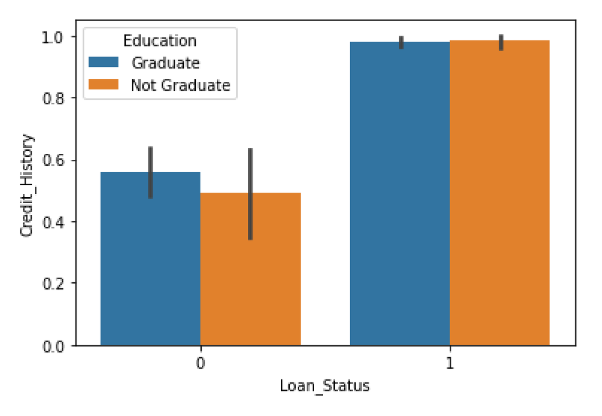
Out[3]:
```
1    422
0    192
Name: Loan_Status, dtype: int64
```

In [4]:
```python
credit_df.groupby(['Education', 'Loan_Status']).Education.count()
```

Out[4]:
```
Education     Loan_Status
Graduate      0              140
              1              340
Not Graduate  0               52
              1               82
Name: Education, dtype: int64
```

In [10]:
```python
sns.barplot(y = 'Credit_History', x = 'Loan_Status', hue='Education', data = credit_df)
```

Out[10]:
```
<AxesSubplot:xlabel='Loan_Status', ylabel='Credit_History'>
```



Fill Null Values

In [5]:
```python
100 * credit_df.isnull().sum() / credit_df.shape[0]
```

Out[5]:
```
Loan_ID              0.000000
Gender               2.117264
Married              0.488599
Dependents           2.442997
Education            0.000000
Self_Employed        5.211726
ApplicantIncome      0.000000
CoapplicantIncome    0.000000
LoanAmount           0.000000
Loan_Amount_Term     2.280130
Credit_History       8.143322
Property_Area        0.000000
Loan_Status          0.000000
dtype: float64
```

In [6]:
```python
object_columns = credit_df.select_dtypes(include=['object']).columns
numeric_columns = credit_df.select_dtypes(exclude=['object']).columns
```

In [13]:
```python
#credit_df.columns[credit_df.dtypes == object]
#credit_df.columns[credit_df.dtypes == object]
```

In [7]:
```python
for column in object_columns:
    majority = credit_df[column].value_counts().iloc[0]
    credit_df[column].fillna(majority, inplace=True)
```

In [8]:
```python
for column in numeric_columns:
    mean = credit_df[column].mean()
    credit_df[column].fillna(mean, inplace=True)
```

In [16]:
```python
# Impute
```

In [9]:
```python
credit_df.head()
```

Out[9]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount
0	LP001002	Male	No	0	Graduate	No	5849	0.0	0
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141

In [10]:
```python
credit_df.drop('Loan_ID', axis=1, inplace=True)
```

In [11]:
```python
object_columns = credit_df.select_dtypes(include=['object']).columns
```

In [12]:
```python
credit_df.head()
```

Out[12]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_
0	Male	No	0	Graduate	No	5849	0.0	0	
1	Male	Yes	1	Graduate	No	4583	1508.0	128	
2	Male	Yes	0	Graduate	Yes	3000	0.0	66	
3	Male	Yes	0	Not Graduate	No	2583	2358.0	120	
4	Male	No	0	Graduate	No	6000	0.0	141	

Categorical Columns

In [13]:
```python
credit_df[object_columns].Property_Area
```

Out[13]:
```
0          Urban
1          Rural
2          Urban
3          Urban
4          Urban
         ...   
609        Rural
610        Rural
611        Urban
612        Urban
613    Semiurban
Name: Property_Area, Length: 614, dtype: object
```

In [14]:
```python
credit_df[object_columns].Property_Area.head()
```

Out[14]:
```
0    Urban
1    Rural
2    Urban
3    Urban
4    Urban
Name: Property_Area, dtype: object
```

In [24]:
```python
credit_df_dummy = pd.get_dummies(credit_df, columns=object_columns)
```

In [25]:
```python
# Sklearn - LabelEncoding
# Sklearn - LabelBinarize
# Sklearn - OneHotEncoding
```

In [26]:
```python
credit_df_dummy.shape
```

Out[26]:
```
(614, 25)
```

Model

In [28]:
```python
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

In [31]:
```python
X = credit_df_dummy.drop('Loan_Status', axis=1)
y = credit_df_dummy.Loan_Status
train_x, test_x, train_y, test_y = train_test_split(X, y, test_size=0.3, random_state=42)
```

In [32]:
```python
train_x.shape, test_x.shape
```

Out[32]:
```
((429, 24), (185, 24))
```

Decision Tree

In [54]:
```python
dt_model = DecisionTreeClassifier(max_depth=14)
```

In [55]:
```python
dt_model.fit(train_x, train_y)
```

Out[55]:
```
DecisionTreeClassifier(max_depth=14)
```

In [56]:
```python
train_y_hat = dt_model.predict(train_x)
test_y_hat = dt_model.predict(test_x)
```

In [41]:
```python
print('-'*20, 'Train', '-'*20)
print(classification_report(train_y, train_y_hat))
print('-'*20, 'Test', '-'*20)
print(classification_report(test_y, test_y_hat))
```

```
-------------------- Train --------------------
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       127
           1       1.00      1.00      1.00       302

    accuracy                           1.00       429
   macro avg       1.00      1.00      1.00       429
weighted avg       1.00      1.00      1.00       429

-------------------- Test --------------------
              precision    recall  f1-score   support

           0       0.53      0.48      0.50        65
           1       0.73      0.78      0.75       120

    accuracy                           0.67       185
   macro avg       0.63      0.63      0.63       185
weighted avg       0.66      0.67      0.67       185
```

In [57]:
```python
print('-'*20, 'Train', '-'*20)
print(classification_report(train_y, train_y_hat))
print('-'*20, 'Test', '-'*20)
print(classification_report(test_y, test_y_hat))
```

```
-------------------- Train --------------------
              precision    recall  f1-score   support

           0       0.99      1.00      1.00       127
           1       1.00      1.00      1.00       302

    accuracy                           1.00       429
   macro avg       1.00      1.00      1.00       429
weighted avg       1.00      1.00      1.00       429

-------------------- Test --------------------
              precision    recall  f1-score   support

           0       0.58      0.54      0.56        65
           1       0.76      0.79      0.78       120

    accuracy                           0.70       185
   macro avg       0.67      0.67      0.67       185
weighted avg       0.70      0.70      0.70       185
```

SVM

In [109]:
```python
svm_model = SVC(kernel='rbf', gamma=0.00001, C=1000)
```

In [110]:
```python
svm_model.fit(train_x, train_y)
```

Out[110]:
```
SVC(C=1000, gamma=1e-05)
```

In [111]:
```python
train_y_hat = svm_model.predict(train_x)
test_y_hat = svm_model.predict(test_x)
```

In [112]:
```python
print('-'*20, 'Train', '-'*20)
print(classification_report(train_y, train_y_hat))
print('-'*20, 'Test', '-'*20)
print(classification_report(test_y, test_y_hat))
```

```
-------------------- Train --------------------
              precision    recall  f1-score   support

           0       0.95      0.95      0.95       127
           1       0.98      0.98      0.98       302

    accuracy                           0.97       429
   macro avg       0.96      0.96      0.96       429
weighted avg       0.97      0.97      0.97       429

-------------------- Test --------------------
              precision    recall  f1-score   support

           0       0.36      0.18      0.24        65
           1       0.65      0.82      0.73       120

    accuracy                           0.60       185
   macro avg       0.51      0.50      0.49       185
weighted avg       0.55      0.60      0.56       185
```

In [113]:
```python
confusion_matrix(test_y, test_y_hat)
```

Out[113]:
```
array([[12, 53],
       [21, 99]], dtype=int64)
```

In [ ]:
```python

```