# Table of Contents



# Importing the Dependencies

In [3]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.metrics import accuracy_score
```

Data Collection and Processing

In [4]:
```
loan = pd.read_csv('loan.xls')
```

In [5]:
```
loan.head()
```

Out[5]:

| | Loan_ID | Gender | Married | Dependents | Education | Self_Employed | ApplicantIncome | CoapplicantIncome | LoanAmount | Loan_Amount_Term |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | LP001002 | Male | No | 0 | Graduate | No | 5849 | 0.0 | NaN | 360.0 |
| 1 | LP001003 | Male | Yes | 1 | Graduate | No | 4583 | 1508.0 | 128.0 | 360.0 |
| 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 |

In [6]:
```
loan.shape
```

Out[6]:
```
(614, 13)
```

In [7]:
```
loan.isnull().sum()
```

Out[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
```

In [8]:
```
loan=loan.dropna()
```

In [9]:
```
loan.isnull().sum()
```

Out[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
```

In [10]:
```
loan.shape
```

Out[10]:
```
(480, 13)
```

In [11]:
```
del loan['Loan_ID']
```

In [12]:
```
loan.shape
```

Out[12]:
```
(480, 12)
```

In [13]:
```
# Check Class Distribution

loan['Loan_Status'].value_counts(normalize=True)
```

Out[13]:
```
Y    0.691667
N    0.308333
Name: Loan_Status, dtype: float64
```

In [14]:
```
# Check Data Types

loan.dtypes
```

Out[14]:
```
Gender                object
Married               object
Dependents            object
Education             object
Self_Employed         object
ApplicantIncome        int64
CoapplicantIncome    float64
LoanAmount           float64
Loan_Amount_Term     float64
Credit_History       float64
Property_Area         object
Loan_Status           object
dtype: object
```

In [15]:
```
# Descriptive Statistic

loan.describe()
```

Out[15]:

| | ApplicantIncome | CoapplicantIncome | LoanAmount | Loan_Amount_Term | Credit_History |
|---|---|---|---|---|---|
| **count** | 480.000000 | 480.000000 | 480.000000 | 480.000000 | 480.000000 |
| **mean** | 5364.231250 | 1581.093583 | 144.735417 | 342.050000 | 0.854167 |
| **std** | 5668.251251 | 2617.692267 | 80.508164 | 65.212401 | 0.353307 |
| **min** | 150.000000 | 0.000000 | 9.000000 | 36.000000 | 0.000000 |
| **25%** | 2898.750000 | 0.000000 | 100.000000 | 360.000000 | 1.000000 |
| **50%** | 3859.000000 | 1084.500000 | 128.000000 | 360.000000 | 1.000000 |
| **75%** | 5852.500000 | 2253.250000 | 170.000000 | 360.000000 | 1.000000 |
| **max** | 81000.000000 | 33837.000000 | 600.000000 | 480.000000 | 1.000000 |

In [16]:
```
# Label encoding

loan.replace({"Loan_Status":{'N':0,'Y':1}},inplace=True)
```

In [17]:
```
loan.head(5)
```

Out[17]:

| | Gender | Married | Dependents | Education | Self_Employed | ApplicantIncome | CoapplicantIncome | LoanAmount | Loan_Amount_Term | Credit_His |
|---|---|---|---|---|---|---|---|---|---|---|
| **1** | Male | Yes | 1 | Graduate | No | 4583 | 1508.0 | 128.0 | 360.0 | |
| **2** | Male | Yes | 0 | Graduate | Yes | 3000 | 0.0 | 66.0 | 360.0 | |
| **3** | Male | Yes | 0 | Not Graduate | No | 2583 | 2358.0 | 120.0 | 360.0 | |
| **4** | Male | No | 0 | Graduate | No | 6000 | 0.0 | 141.0 | 360.0 | |
| **5** | Male | Yes | 2 | Graduate | Yes | 5417 | 4196.0 | 267.0 | 360.0 | |

In [18]:
```
# Dependent column values

loan['Dependents'].value_counts()
```

Out[18]:
```
0     274
2      85
1      80
3+     41
Name: Dependents, dtype: int64
```

In [19]:
```
# replacing the value of 3+ to 4

loan = loan.replace(to_replace='3+', value=4)
```

In [21]:
```
# dependent values

loan['Dependents'].value_counts()
```

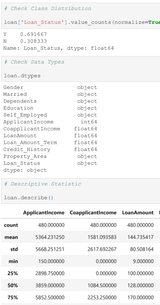
Out[21]:
```
0    274
2     85
1     80
4     41
Name: Dependents, dtype: int64
```

Data Visualization

In [22]:
```
# education & Loan Status

sns.countplot(x='Education',hue='Loan_Status',data=loan)
```

Out[22]:
```
<AxesSubplot:xlabel='Education', ylabel='count'>
```




In [23]:
```
# marital status & Loan Status

sns.countplot(x='Married',hue='Loan_Status',data=loan)
```

Out[23]:
```
<AxesSubplot:xlabel='Married', ylabel='count'>
```


In [24]:
```
# convert categorical columns to numerical values

loan.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 Gradua
```

In [25]:
```
loan.head()
```

Out[25]:

| | Gender | Married | Dependents | Education | Self_Employed | ApplicantIncome | CoapplicantIncome | LoanAmount | Loan_Amount_Term | Credit_His |
|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 1 | 1 | 1 | 1 | 0 | 4583 | 1508.0 | 128.0 | 360.0 | |
| **2** | 1 | 1 | 0 | 1 | 1 | 3000 | 0.0 | 66.0 | 360.0 | |
| **3** | 1 | 1 | 0 | 0 | 0 | 2583 | 2358.0 | 120.0 | 360.0 | |
| **4** | 1 | 0 | 0 | 1 | 0 | 6000 | 0.0 | 141.0 | 360.0 | |
| **5** | 1 | 1 | 2 | 1 | 1 | 5417 | 4196.0 | 267.0 | 360.0 | |

In [27]:
```
# separating the data and label

X = loan.drop(columns='Loan_Status',axis=1)
Y = loan['Loan_Status']
```

In [29]:
```
X
```

Out[29]:

| | Gender | Married | Dependents | Education | Self_Employed | ApplicantIncome | CoapplicantIncome | LoanAmount | Loan_Amount_Term | Credit_H |
|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 1 | 1 | 1 | 1 | 0 | 4583 | 1508.0 | 128.0 | 360.0 | |
| **2** | 1 | 1 | 0 | 1 | 1 | 3000 | 0.0 | 66.0 | 360.0 | |
| **3** | 1 | 1 | 0 | 0 | 0 | 2583 | 2358.0 | 120.0 | 360.0 | |
| **4** | 1 | 0 | 0 | 1 | 0 | 6000 | 0.0 | 141.0 | 360.0 | |
| **5** | 1 | 1 | 2 | 1 | 1 | 5417 | 4196.0 | 267.0 | 360.0 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **609** | 0 | 0 | 0 | 1 | 0 | 2900 | 0.0 | 71.0 | 360.0 | |
| **610** | 1 | 1 | 4 | 1 | 0 | 4106 | 0.0 | 40.0 | 180.0 | |
| **611** | 1 | 1 | 1 | 1 | 0 | 8072 | 240.0 | 253.0 | 360.0 | |
| **612** | 1 | 1 | 2 | 1 | 0 | 7583 | 0.0 | 187.0 | 360.0 | |
| **613** | 0 | 0 | 0 | 1 | 1 | 4583 | 0.0 | 133.0 | 360.0 | |

480 rows × 11 columns

In [30]:
```
Y
```

Out[30]:
```
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

In [31]:
```
X_train, X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.1,stratify=Y,random_state=2)
```

In [32]:
```
print(X.shape, X_train.shape, X_test.shape)
```

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

Training the model:

Support Vector Machine Model

In [33]:
```
classifier = svm.SVC(kernel='linear')
```

In [34]:
```
#training the support Vector Machine model

classifier.fit(X_train,Y_train)
```

Out[34]:
```
SVC(kernel='linear')
```

Model Evaluation

In [35]:
```
# accuracy score on training data


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

In [36]:
```
print('Accuracy on training data : ', training_data_accuracy)
```

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

In [37]:
```
# accuracy score on training data

X_test_prediction = classifier.predict(X_test)
test_data_accuracy = accuracy_score(X_test_prediction,Y_test)
```

In [38]:
```
print('Accuracy on test data : ', test_data_accuracy)
```

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

Making a Prediction System

In [ ]:
```

```